Global Information Assurance Certification Paper

GIAC Training & Certification

Level Two Firewalls, Perimeter Protection, and VPN's
GCFW Practical Assignment
Version 1.5d

SANS – Darling Harbour 2001

Norrie Bennie

GIAC Training & Certification

Table of Contents

GIAC Training & Certification

Level Two Firewalls, Perimeter Protection, and VPN's
GCFW Practical Assignment

Assignment 1 - Security Architecture (25 Points)

Define security architecture for GIAC Enterprises, a growing Internet startup that expects to earn $200 million per year in online sales of fortune cookie sayings, and which has just completed a merger/acquisition. Your architecture must specify filtering routers, firewalls, VPNs to partners, secure remote access, and internal firewalls. Be explicit about the brand and version of each perimeter technologies to implement your security architecture.

You must consider and define access for:

- *Customers (the companies that purchase bulk online fortunes);*
- *Suppliers (the authors of fortune cookie sayings that connect to supply fortunes);*
- *Partners (the international partners that translate and resell fortunes).*

Assignment 2 - Security Policy (25 Points)

Based on the security architecture that you defined in Assignment 1, provide a security policy for AT LEAST the following three components:

Border Router
Primary Firewall
VPN

You may also wish to include one or more internal firewallls used to implement defence in depth or to separate business functions.

By 'security policy' we mean the specific ACLs, firewall ruleset, IPSec policy, etc. (as appropriate) for the specific component used in your architecture. For each component, be sure to consider internal business operations, customers, suppliers and partners. Keep in mind you are an E-Business with customers, suppliers, and partners - you MAY NOT simply block everything!

(Special note VPNs: since Ipsec VPNs are still a bit flaky when it comes to implementation, that component will be graded more loosely than the border router and primary firewalls. However, be sure to define whether split-horizon is implemented, key exchange parameters, the choice of AH or ESP and why, PPP-based VPNs are also fully acceptable as long as they are well defined.)

For each security policy, write a tutorial on how to implement each ACL, rule, or policy measure on your specific component. Please use screen shots, network traffic traces, firewall log information, and/or URLs to find further information as appropriate. Be certain to include the following:

1.The service or protocol addressed by the ACL or rule, and the reason these services might be considered vulnerability.
2.Any relevant information about the behaviour of the service or protocol on the network.
3.The syntax of the ACL, filter, rule, etc.
4.A description of each of the parts of the filter.
5.An explanation of how to apply the filter.
6.If the filter is order-dependent, list any rules that should precede and/or follow this filter, and why this order is important. (Note: instead of explaining order dependencies for each individual rule, you may wish to create a separate section of your practical that describes the order in which ALL of the rules should be applied, and why.)
7.Explain how to test the ACL/filter/rule.

Be certain to point out any tips, tricks, or "gotchas".

Assignment 3 - Audit Your Security Architecture (25 points)

You have been assigned to provide technical support for a comprehensive information systems audit for GIAC Enterprises. You are required to audit the Primary Firewall described in Assignments 1 and 2. Your assignment is to:

1.Plan the assessment. Describe the technical approach you would recommend to assess your perimeter. Be certain to include considerations such as what shift or day you would do the assessment. Estimate costs and level of effort. Identify risks and considerations.
2.Implement the assessment. Validate that the Primary Firewall is actually implementing the security policy. Be certain to state exactly how you would do this, including the tools and commands used. Include screen shots in your report if possible.
3.Conduct a perimeter analysis. Based on your assessment (and referring to data from your assessment), analyse the perimeter defence and make recommendations for improvements or alternate architectures. Diagrams are strongly recommended for this part of the assignment.

Note: DO NOT simply submit the output of nmap or a similar tool here. It is fine to use any assessment tool you choose, but annotate the output.

Assignment 4 - Design Under Fire (25 Points)

The purpose of this exercise is to help you think about threats to your network and therefore develop a more robust design. Keep in mind that the next certification group will be attacking your architecture!

Select a network design from an previously posted GCFW practical (http://www.sans.org/giactc/gcfw.htm) and paste the graphic into your submission. Be certain to list the URL of the practical you are using. Design the following three attacks against the architecture:

1. *An attack against the firewall itself. Research vulnerabilites that have been found for the type of firewall chosen for the design. Choose an attack and explain the results of running that attack against the firewall.*
2. *A denial of service attack. Subject the design to a theoretical attach from 50 compromised cable modem/DSL systems using TCP SYN, UDP and ICMP floods. Describe the countermeasures that can be put into place to mitigate the attack you choose.*
3. *An attack plan to compromise an internal system through the perimeter system. Select a target, explain your reasons for choosing that target, and describe the process to compromise the target.*

Note: this is the second time this assignment has been used. The first time, a number of students came up with magical "hand-waving" attacks. You must supply documentation (preferably a URL) for any vulnerability you use in your attack, and the exploit code that you use to accomplish the attack. The purpose of this exercise is for the student to clearly demonstrate they understand that firewall and perimeter systems are not magic "silver bullets" immune to all attacks.

Assignment 1 - Security Architecture

Introduction

For the system architecture I will try to explain my decisions for deciding on the layout of the network and it's components. I will also describe what types of product I have chosen for the components.

The Network Design

The diagram as shown in Figure 1 represents the security architecture I have decided upon.

The Border Router

The task of the border router is to block the most basic attacks. It is common to use static packet filters here. Our border router has two interfaces, one is directly connected to the Internet, and the second is connected to our external firewall.

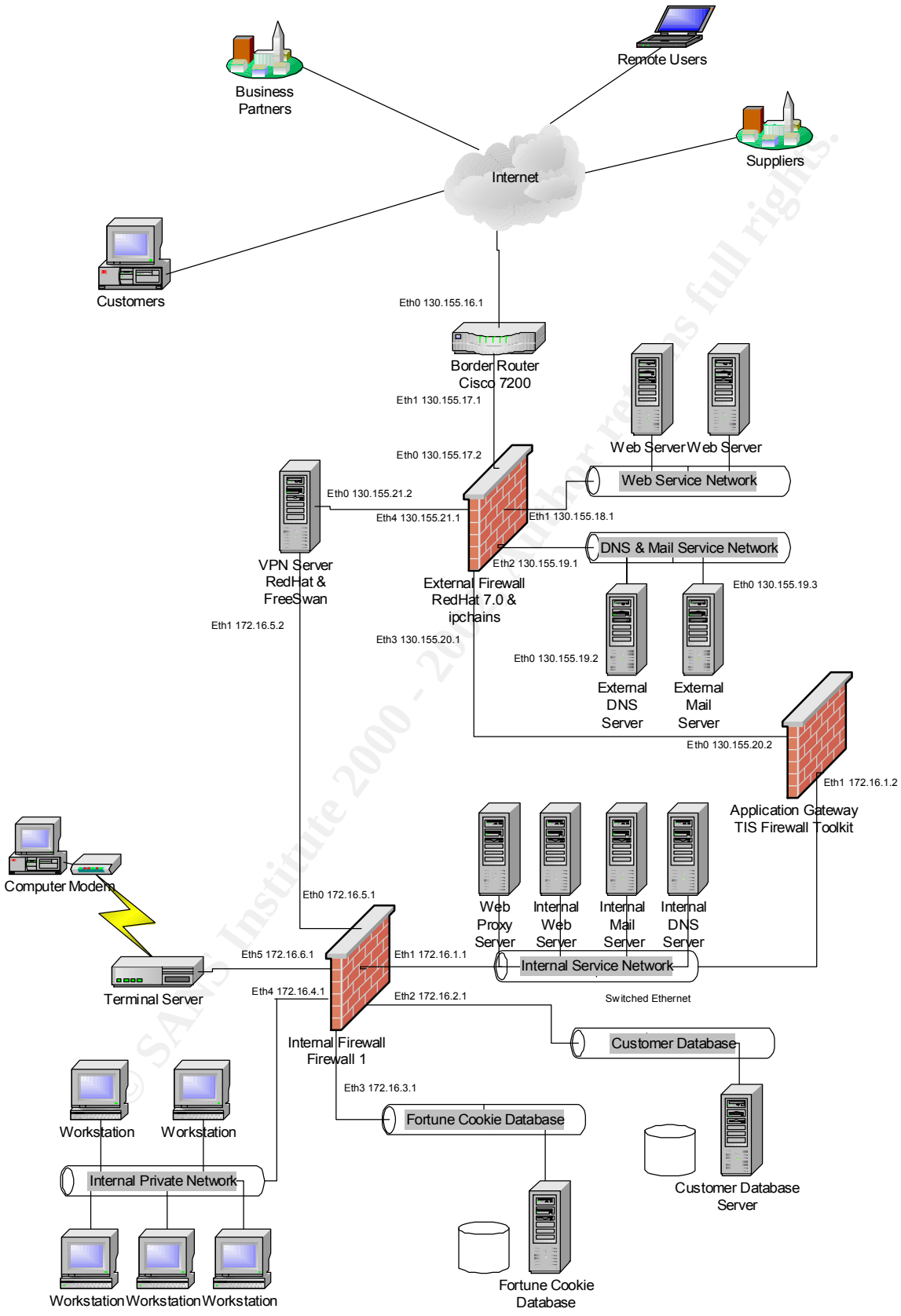
The main types of IP traffic we want to stop at the border router include:
- Blocking Private Address Space,
- Anti-Spoofing,
- Controlling ICMP traffic, and
- Blocking Source Routing.

Why do we want a border router, why not just have the firewall to handle it? Well, the border router is used to block most common traffic. This is done at a coarse level. This leaves more complicated filtering of packets at the firewall for more precise filtering of packets.

Border routers are most commonly hardware solutions. Firewalls are more commonly software solutions - a host running an operating system and firewall software. If a firewall is the first line of defence, then if it compromised - which a software or host based solution is more prone to, especially if the rest of the machine apart from the firewall software is not be hardened correctly, then network is vulnerable. Also being a hardware solution means that it is less likely to have the same number of vulnerabilties as a software solution. A hardware solution is much faster, so performance may be degraded if software solution is in place as the initial point of contact.

For the border router I have chosen CISCO 7200. This is a hardware solution and is faster than a software solution. Deciding what type of CISCO product depends on the connection speed and therefore the interfaces we require in it. I am assuming that we have a reasonably fast connection to the Internet so am assuming we have two ethernet interfaces in it.

Figure 1



Many border routers do static packet filtering - that is just looking at the packet and seeing if it is valid, and filtering it based on where its from and where its going. These do not look at whether the packet is for an established connection or not - ie. They do not keep track of the state of the connection. CISCO provides statefull filtering of packets. For an e-commerce site I think this is a good solution. The CISCO also provides CBAC (- formerly known as Firewall Feature Set FFS). This allows some application layer based filtering as well. However if this is used you need to consider this in selecting a router, as a higher end model is required as it uses a lot of CPU and can degrade performance.

The External Firewall

The second line of defence is our external firewall. For this it was my intention to use a RedHat 7.0 machine running iptables (iptables replaces previous ipchains firewall and provides statefull filtering) - the reason for the selection is that it is the most accessible firewall I can acquire to do the assignment. However due to problems I had compiling the kernel-2.4 which I have not sorted out in time, I will be using ipchains. Probably a commercial firewall would be a better selection, such as Checkpoint Firewall 1 as it is proven system, while iptables is still in it's infancy.

The external firewall has five interfaces on it. Each of the interfaces is assigned an IP address from the public address range - the first interface is connected to the border router. Off the second interface are the web servers. I have separated the web servers from others. This to make packet sniffing more difficult if the other public servers are compromised, as the web servers are located on different physical connection.
Off the third interface are the other services available to the public such as domain name system (DNS) and Mail.

The fourth interface is connected to an application gateway. This allows us to access internal services from the screened service network, and also allows the internal network access to the outside world.

The fifth interface is connected to the VPN server. This allows us to filter what traffic goes to the VPN - why do this? - the VPN is a host running IPSec and so if there any vulnerabilities on the server to ports/services other than those used by IPSec, these can be blocked. If the VPN was a hardware solution then we could possibly bypass the external firewall. However the firewall gives us that extra bit of protection - we can stop traffic that otherwise might have hit the VPN server.

The Virtual Private Network (VPN) Server

The VPN is used to provide secure sessions and connections to our business partners (I am treating the companies that GIAC has had mergers or acquisitions as business partners as well.) The VPN also allows us to provide secure connections to remote users how may be connected via an ISP or other connection to the Internet. (many IPSec documents refer to these types of users as 'Road Warriors' [refs]). The VPN is also

performing Network Address Translation (NAT) with our private IP address range. This allows business partners to get access to internal services and databases.

For the VPN implementation I have chosen a redhat linux machine running freewan 1.8 (redhat's IPSec) with X.509 certificate patches - this allows us to use windows 2000 IPSec to talk to the VPN server. Otherwise we would be limiting ourselves to linux machines for use as remote user machines.

This particular VPN has two interfaces - one with an IP address in the public address range and the other with a private address range. We have public IP address for the external interface to allow remote users and business partners to locate the VPN in order to connect to it.

There are a couple of possibilities for the VPN. With the application gateway in place I could have set the VPN up to not perform NAT, and have public IP addresses on the internal interface, and make the business partners connect via the application gateway. This has the benefit that our business partners do not see our internal address range. However, it means that the end of the secure connection would be in the public view, which may lead to a compromise situation. If these machines on this network were compromised, then it may mean that traffic going to the business partners could be sniffed, and also lead to invalid packets being sent to the partner causing them to be compromised. Also traffic from our internal network would have to be send to the public network unencrypted before they reached the VPN interface. Since we are treating the business partners and remote users in the same manner, having the VPN translate to internal address space seems to be the better solution. The traffic from the VPN in either case would be filtered. In this case it is being filtered via the internal firewall. If we wanted to treat the remote users and business partners separately then we could use two VPN's.

IPSec is a security suite which provides a mechanism for establishing secure communications over the internet, providing authentication and encryption. It allows for hosts-to-host, host-to-network (or security gateway) and network-to-network (security gateway to security gateway) connections. This makes it an attractive method for implementing a Virtual Private Network, or VPN tunnel between networks.

The process of setting up a VPN connection with IPSec starts with the hosts or security gateways at each end of the secure communication being authenticated, for example via LDAP (Lightweight Directory Access Protocol). Once the authentication is done the keys and encryption algorithms are negotiated, after which the secure connection is setup and the data payload is exchanged - with an ongoing key exchange as data is passed.

IPSec uses the following IP protocols:
- Authentication Header (AH),
- Encapsulating Security Payload (ESP) and
- Internet Key Exchange (IKE).

In IPSec the authentication of the ends of the connection is done by one of two methods. The first of these done by exchanging secrets known as "preshared secrets" which is some identifier each end already knows. The second type of authentication is through the use of public keys such as certificates.

Once authentication is done, the two ends use the IKE protocol (a hybrid of ISAKMP & Oakley Protocol and using the same port as ISAKMP) to do the key exchange. In the key exchange the type of security protocol (ESP or AH) and the encryption algorithm (such as Triple DES, BlowFish, etc) for the payload are passed and negotiated. To know what to send, each end of the connection has a reference to the other. These details, are known as a Security Association. This allows for one or more IPSec connections on each gateway.

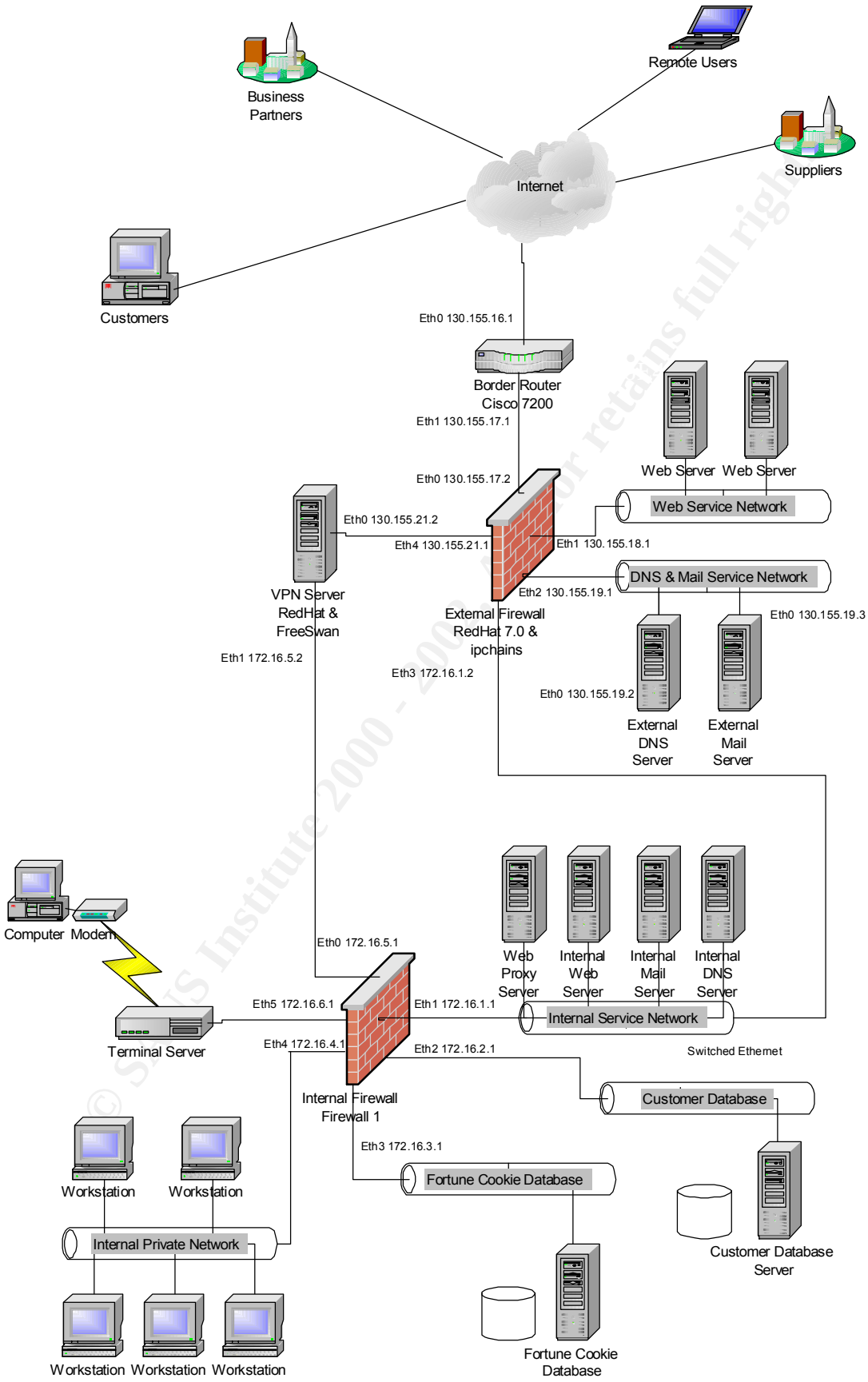
The Authentication Header (AH) is used for assuring that the data is coming from a legitimate source. It covers the IP header, however the data remains unprotected. The Encapsulating Security Payload (ESP) on the other hand can be used to ensure authenticity as well as encrypt the packet/data contents.

There are two modes of connections IPSec - transport mode, and tunnel mode. Transport mode can be used between two hosts. In transport mode, only the data is encrypted or authenticated. The IP headers of the packet are left in their original state. Tunnel mode is required when a security gateway is involved. In tunnel mode, both the IP headers and the data are encrypted or authenticated.

As a result of transport mode not encapsulating the IP headers, address translation cannot be used. In this instance anyway we have to use tunnel mode as we are establishing a gateway between our network and the business partners and remote users. We will be using the ESP protocol for payload encryption and using 3des encryption algorithm for each connection between business partners and remote users.

The Application Gateway

Why an application gateway? In my first go at doing the network architecture I thought I could just have the external firewall connect to the internal firewall, with a set of internal services on the network in between, as seen in the Figure 2. This seemed ok to me at first, but at closer inspection I discovered a problem. It was ok if a client from the internal network wanted to get to an external network services, such as the web, as address translation could be done at the external firewall to get to the service - we could allow internal clients to have knowledge of the external/public network. Address translation from the internal network clients would work as the client IP address and port could be kept record of by the firewall and the external service to connect to is known.



However there is a problem when something on the external network needs access to internal services, for example access to the fortune cookie and customer databases. The machines in the private address space had no problem getting to it. However what about the web servers - they need to be able write into the database customer details and retrieve fortune cookie sayings from it. The only problem is we don't let any of the public machines have any knowledge of the internal machines (as we are using split DNS this only allows external machines or any machines outside on the Internet knowledge of public IP addresses). The web server doesn't know where to go to get the data, as we don't have the destination address of the internal machine running the database. We could tell it the router address but the router still doesn't know about which machine to connect to. The solution is to provide a public IP address to the web server so that it can connect to the database. However we don't want the database to be on the public network - so we need a proxy server - one with two interfaces - one on the public network and one the private network. The proxy will look as though it is the database server, but in actual fact will be a middle man between the public network and the real database. We can then allow the web to connect securely with the database. The same goes to the mail relay and web proxy as well.

The alternative to the application gateway is port forwarding on a firewall, so in actual fact my original design would work ok if port forwarding were enabled. Port forwarding works by listening to requests to the firewall based on port service and also by port service and IP destination. It forwards a packet to the internal host after rewriting the IP header - however IP forwarding has limited application knowledge - a proxy gateway has application layer knowledge and provides a more granular filtering, so an application gateway is a higher level of security.

In essence the difference between a Proxy or Application Gateway and Port Forwarding or redirection - the Proxy/Application Gateway - proxy running for each service we want - we talk to proxy, proxy then talks to internal service (or external depending on which direction traffic is going). With port forwarding we are just rewriting the headers and passing the packets on, so in general application gateway is at the application layer, while IP forwarding is mostly at the IP layer.

The Application Gateway will be running a syslog daemon, which all other servers will write their logs to. The application gateway itself will be a linux machine running ipchains for the filtering rules and masquerading the internal network clients, and will be running the Trusted Information Systems (TIS) Firewall Toolkit (FWTK) version 2.1. There are several application gateway software products available. TIS also produce commercial Application Gateway software called Gauntlet. This has several more proxy services available than FWTK. Depending on what Database servers are running depends on what product you should choose. Gauntlet supports Oracle, as well as Sybase and Microsoft SQL. The FWTK on the other hand does not support Oracle at this stage. We will assume that our Database servers are running Sybase SQL.

The Internal Firewall

The internal firewall could be any number of various firewall products. The general rule of thumb is to use a firewall which is a different product or from a different vendor than the other firewalls in the network, especially if one is protecting the other. Since I have used a RedHat machine as the external firewall, I will use checkpoint firewall-1 with latest patches for the internal firewall.

The internal firewall is used to segment off our private internal network and databases, and filter the unencrypted packets from the VPN. This is required since no filtering on the actual VPN packets has been done yet, as all previous packets through the router and external firewall have been encrypted or tunnelled through IPsec. The internal firewall also filters packets from our dialup network, and limits packets to and from our internal service network where our internal DNS and Mail server are located.

A brief description of what is filtered with the internal firewall. Since VPN traffic contains information from business partners and suppliers, as well as remote users, the VPN network is stopped from accessing the customer detail database, but is allowed access to the fortune cookie database. Because remote users may require access to the private internal network, limit access is available. The VPN network has access to the internal DNS server. The internal private network has access to all internal services.

The Dialup Server

The Dialup server is a terminal annex connected to a collection of modems, with an authentication hose providing the IP addresses which are statically allocated based on user accounts.

The Domain Name Service (DNS) Server Architecture

We will be using split DNS for name resolution. This allows us to advertise our screened service networks to the world, while still maintaining our own internal DNS for looking up machines on our internal network without these internal machines being looked up from the world.

The internal DNS will have all the names of the internal machines and their IP addresses as well as having access to the external DNS server to obtain DNS information about the Internet.

While split DNS can be performed with BIND versions 8.2.x and greater on a single host, it is better to separate the external and internal DNS into two separate machines. The reason for this is that if the external DNS is compromised, than names and IP addresses of internal machines would be available to whoever compromised it, allowing them to map our internal network and get names or our internal services. For example they could pick up the mail server through MX records and also possible get information about the

machines operating systems from the HINFO records if they exist. This enables them to direct an attack more accurately at vulnerable services if they exist.

A point to make is not to always associate the internal network with private IP address ranges. In this case it is the same, however, it is possible to have a spit DNS for networks with registered public IP address ranges, but not advertise machines in certain ranges on the external DNS.

The internal DNS will be a forwarder only, as will the DNS running on the Application Gateway – it will forward on requests to the external DNS which will be recursive. The external DNS however will only allow clients in our public service network to use it as a recursive server. It will not allow recursive queries for clients on the Internet. A recursive DNS is where the DNS server will perform searches for hosts. A forwarder doesn't do the searching, but lets the DNS server it forwards it onto do the search for it. Recursive DNS's are more prone to DNS cache poisoning, that is why the internal DNS's are just forwarders and the external DNS will only do recursive lookups for machines on our network.

The Mail Server Architecture

The mail server on the screened services network is used as a mail relay to forward all mail to the internal mail server. The external mail server will rewrite mail addresses and headers so that user's e-mail appears as user@giac.com rather than user@internalhost.giac.com. - rewriting headers so that mail appears to be originating from mail server rather than from the internal mail server. This is to stop hackers/people outside finding information out about our internal network from the mail headers.

Assignment 2 - Security Policy

Introduction

In the following sections I describe the security policy used for the border router, the external firewall, the VPN server and the Application Gateway. A point to note is that the VPN server and Application Gateway are also configured with firewall filtering rules.

The Border Router

Some points to note about CISCO routers:
The matching of access lists is done on a line by line basis - the first line that matches a pattern is the action that is taken. Therefore the order of items in the access lists is important.

CISCO created a list that looks at only the source IP address, as this is the most common field examined. As it only looks at source IP address, it is fast. Access lists that look at only source are known as standard access lists and have access list numbers from 1-99.

Access lists, which look at source, destination, protocol, tcp/udp port, icmp type and sequence, are known as extended access lists, and are numbered from 100-199.

A third type of access lists is a reflexive list; this keeps track of connection state. These types of access lists are referenced by labels. This provides statefull packet filtering as opposed to static packet filtering (or just pattern matching without worrying whether a connection has been established).

By default CISCO routers allow all traffic to pass through them if there are no access lists defined for an interface. However as soon as a rule or access-list has been added, the default is to allow added access-lists items and deny everything else. An implicit deny once an access-list is enforced.

Each interface has two directions in which to filter - inbound and outbound. There can only be one access-list applied per direction per interface. In general inbound filtering saves the CPU compared to outbound where the inbound list has already been processed. So it is important to point out, that if you apply a standard access list on an interface, you can't apply an extended access list on the same interface and have them both work. However you can place standard and extended lists inside reflexive lists.
We will be using extended access lists to define security at the border router.

Extended access lists can be associated to an interface in two ways - you define the access list directly under the interface after specifying the group should be applied, or you can define the access list and then apply it to the interface.

For example, to create the group after assigning it to the interface you would use the following:

```
interface ethernet 0
ip access-group 101 in

access-list 101 ....
...
...
```

The alternative is to define the list first and then assign it to the interface:

```
ip access-list extended 101
permit ....

interface ethernet 0
ip access-group 101 in      - apply list to interface
```

The syntax of extended access lists are as follows:

For IP/tcp/udp packets:

```
access-list <number> permit|deny  tcp|udp|ip
            host <source IP>|<source_network><netmask>
            host <destintation IP>|<destination_network><netmask>
            [[eq|gt|lt <port>]|[range <port1> <portn>]] [log]
```

For icmp packets:

```
access-list <number> permit|deny  icmp
            host <source IP>|<source_network><netmask>
            host <destintation IP>|<destination_network><netmask>
            <icmp message type> [<icmp message code>] [log]
```

There are several basic rules that should normally always be blocked at the router, these are defined in the SANS Top Ten security threats. [Brenton - Firewalls 101: Perimeter Protection with Firewalls]. The list is as follows:

1. Block spoofed addresses - packets coming from outside but have their source address defined as your internal or private address (RFC 1918 and localhost 127) addresses, as well as packet which are source routed, that is packets where a predefined route is described in the packet telling it which routes to take in order to get to the destination.
2. Login services which are vulnerable. These include telnet (tcp/23), ssh (22/tcp), ftp (21/tcp), NetBIOS (139/tcp) and the rlogin services ranging from port 512 to 514 on tcp.
3. Remote Procedure Call (RPC) and Network File System (NFS). RPC runs portmapper and rpcbind on port 111 tcp and udp respectively. NFS runs on ports 2049 tcp and udp, and also uses lockd which runs on port 4045 tcp and udp.
4. NetBIOS in windows NT, which uses ports 135 (tcp and udp), 137 (udp), 138 (udp) and port 139 (udp). Windows 2000 also uses these ports as well as port 445 (udp and tcp).
5. X Windows which runs on port 6000 to port 6255 over tcp.

6. Naming services - DNS should only be allowed to DNS servers, it uses port 53 with udp for DNS queries and tcp for DNS zone transfers, all other traffic going to port 53 destined for any other machine should be blocked. LDAP ports 389 udp and tcp.
7. Mail - SMTP (port 25 over tcp) should be blocked to all machines except mail relays. POP which uses ports 109 and 110 on tcp and IMAP which uses port 143 over tcp should also be blocked from all other machines.
8. Web - only HTTP (port 80 over tcp) and SSL (port 443 over tcp) should be allowed to the web servers. Also some common high end ports used for web should be considered for being blocked , such as ports 8000, 8080 and 8888, etc over tcp.
9. "Small services" - those below port 20 over tcp and udp should be blocked. Also the time service (port 37 over tcp and udp) should be blocked.
10. Miscellaneous traffic, such as TFTP (69/udp), finger (79/tcp), NNTP (119/tcp), NTP (123/tcp), LPD(515/tcp), syslog(514/udp), SNMP (ports 161 and 162 over tcp and udp), BGP (179/tcp) and socks (1080/tcp) should be blocked.
11. ICMP traffic should also be managed. Incoming echo requests should be blocked, outgoing echo replies, time exceeded and destination unreachable messages (except for 'packet too big' messages (type 3 code 4) should be blocked.

Before we consider the packets to filter, we should consider the security of the router itself, an we should enable encryption of the password for the router to stop it being visible in clear text.

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

To log all traffic, we need to tell the router which machine to log packets to. In the case we are going to make it the Application Gateway, so we specify the following, where 130.155.20.2 is the IP address of the Application Gateway.

```
logging 130.155.20.2
```

In regard to the first item on the SANS top ten list, we want to stop source routing of packets, this is one of the easiest items to implement and is done via the following command:

```
no ip source-route
```

Before continuing with the rest of the first option, we will look at the ninth item on the list, stopping the small services. In a similar manner to stopping source routing this is relatively easy, and consists of the following commands:

```
no service tcp-small-servers
no service udp-small-servers
```

Other similar services which should be disabled on the border router are the finger service, bootp server, http server, the stopping of broadcasts being routed and the snmp management running on the router. These items are solved with the following lines respectively.

```
no service finger
no ip bootp server
no ip http server
no ip directed-broadcast
no snmp
```

Let us now associate the access list to the interface before we create the list, and give the external interface an ip address.

```
interface ethernet 0
ip address 130.155.16.1
ip access-group 101 in
```

Getting back to the first item, the next thing we are going to do is block the private IP address ranges defined in RFC 1918. We should never see any traffic/packets arriving on our external interface on our router coming with a source address the private IP address range. If we see this it could be someone trying to spoof your internal private address range. This could be used to do a DOS attack. The blocking of incoming packets like these is known as ingress filtering. The opposite of this is the blocking of outgoing packets, which is known as egress filtering.

```
access-list 101 deny ip 192.168.0.0 0.0.255.255 any log
access-list 101 deny ip 172.16.0.0 0.15.255.255 any log
access-list 101 deny ip 10.0.0.0 0.255.255.255 any log
```

We now want to block packets which appear to be coming from local host (127.0.0.1), broadcast and multicast addresses.

```
access-list 101 deny ip 127.0.0.0 0.255.255.255 any log
access-list 101 deny ip 255.0.0.0 0.255.255.255 any log
access-list 101 deny ip 224.0.0.0 31.255.255.255 any log
```

We also want to deny packets which appear to have no source IP address.

```
access-list 101 deny ip host 0.0.0.0 any log
```

We next want to block any packets appearing to come from our own IP address range coming from the Internet on our external interface. This is known as anti-spoofing.

```
access-list 101 deny ip 130.155.16.0 0.0.255.255 any log
access-list 101 deny ip 130.155.17.0 0.0.255.255 any log
access-list 101 deny ip 130.155.18.0 0.0.255.255 any log
access-list 101 deny ip 130.155.19.0 0.0.255.255 any log
access-list 101 deny ip 130.155.20.0 0.0.255.255 any log
access-list 101 deny ip 130.155.21.0 0.0.255.255 any log
```

Finally for the first item in the SANS top ten, we want to deny packets on our external router interface which appear to come from the external router interface itself.

```
access-list 101 deny ip host 130.155.16.1 any log
```

For the second item we want to block login services, we have the following:

```
access-list 101 deny tcp any any range ftp telnet log
access-list 101 deny tcp any any range exec lpd log
access-list 101 deny tcp any any eq 139 log
```

Note in the above we are specifying the ranges of the ports, but instead of numbers are using the well known names associated with the ports, so in actual fact, the first line could have been written as:

```
access-list 101 deny tcp any any range 21 23 log
```

To block RPC and NFS the following ACL rules are used. It is important to block RPC and NFS as these are some of the most vulnerable services that people used to use to gain access to users machines. By querying the filesystems mounted on a machine, users used to be able to mount the file system and write to the disk.

```
access-list 101 deny udp any any eq sunrpc log
access-list 101 deny tcp any any eq sunrpc log
access-list 101 deny udp any any eq 2049 log
access-list 101 deny tcp any any eq 2049 log
access-list 101 deny udp any any eq 4045 log
access-list 101 deny tcp any any eq 4045 log
```

We want to block the NetBIOS on Windows NT as it is used for WINS resolution and windows authentication services. It is also used for Windows filesharing, which would mean that people would be able to connect or attempt to connect to a windows share from outside the network. The ACLs for blocking NetBIOS is as follows.

```
access-list 101 deny udp any any eq 135 log
access-list 101 deny tcp any any eq 135 log
access-list 101 deny udp any any range 137 139 log
access-list 101 deny tcp any any eq 139 log
access-list 101 deny udp any any eq 445 log
access-list 101 deny tcp any any eq 445 log
```

In relation to the fifth item on the list, blocking X windows ports, we issue the following rule. It is important to block X windows ports especially when hosts are misconfigured. For example in a case where a host may be setup to allow any remote machine to establish an X connection.

```
access-list 101 deny tcp any any range 6000 6255 log
```

For naming services, it is important not to allow just any host to connect to the DNS server. The DNS server does name resolution for the network, and if a malicious user can connect to the server and perform zone transfers, not only would they be able to get the list of all hosts that the DNS server maintains IP to name resolution, but the DNS server can be given invalid DNS information, known as 'cache poisoning'. If the DNS has been poisoned when a host tries to connect to a service it may be redirected to another false host, providing the service, or be connected to a host that does not have the service at all. This leads to a denial of service attack. We will allow DNS queries to be made to the

server from anywhere, but only allow authorised machines with which to zone transfers. We want to allow LDAP only to the VPN server so that it can do authentication for IPSec connections.

```
access-list 101 permit udp any host 130.155.19.2 eq 53
access-list 101 deny udp any any eq 53 log
access-list 101 permit tcp host $specific_external_dns_ip host 130.155.19.2 eq
53
access-list 101 deny tcp any any eq 35 log
access-list 101 permit tcp any host 130.155.21.2 eq 389
access-list 101 deny tcp any any eq 389 log
access-list 101 permit udp any host 130.155.21.2 eq 389
access-list 101 deny udp any any eq 389 log
```

For mail services, we only want to let SMTP port 25 open to the public mail server, all other packets going to port 25 destined to anywhere else should be blocked, and any other packets going to the mail server should be blocked. Since users will only be reading their mail from behind the border router, we will not allow POP or IMAP

```
access-list 101 permit tcp any host 130.155.19.3 eq 25
access-list 101 deny tcp any any eq 25 log
access-list 101 deny tcp any any eq 109 log
access-list 101 deny tcp any any eq 110 log
access-list 101 deny tcp any any eq 143 log
```

For the screened service network we will allow web connections to ports 80 and 443 over tcp. All other ports will be blocked. We block the higher order web ports as these are commonly used for running web servers which are unauthorised.

```
access-list 101 permit tcp any 130.155.18.0 0.0.255.255 eq 80
access-list 101 deny tcp any any eq 80 log
access-list 101 permit tcp any 130.155.18.0 0.0.255.255 eq 443
access-list 101 deny tcp any any eq 443 log
access-list 101 deny tcp any any eq 8000 log
access-list 101 deny tcp any any eq 8080 log
access-list 101 deny tcp any any eq 8888 log
```

For the ninth security risk, the small services have already been discussed, however we need to add the ACL for the time service. It will look as follows.

```
access-list 101 deny tcp any any eq 37 log
access-list 101 deny udp any any eq 37 log
```

For the miscellaneous services that we should block the ACLs follow. Finger is important to block as it allow people to see the username of the users connected to a machine and what terminal connection they are on. SNMP is used for network management, it is relatively unsecure in terms of password protection, and the default passwords or community names are well known. If someone does not configure their community name, then it means that someone could get control of the machine, or at least the machine settings such as the time to live for the machine, and so on. People could modify these values, and cause denial of service attacks as well. Note that the access lists for syslog and lpd are included in the range which was specified for the rlogin services.

```
access-list 101 deny udp any any eq 69 log
access-list 101 deny tcp any any eq 79 log
access-list 101 deny tcp any any eq 119 log
access-list 101 deny tcp any any eq 123 log
access-list 101 deny tcp any any range 161 162 log
access-list 101 deny udp any any range 161 162 log
access-list 101 deny tcp any any eq 179 log
access-list 101 deny tcp any any eq 1080 log
```

We now need to look at the icmp packets. We want to stop incoming echo requests, which are used by ping and traceroute. If we allow this, then we open ourselves for attacks like the 'ping o'death' in which fragmented icmp echo requests ended up building into a bigger than expected packet when they were reassembled. You also want to deny icmp redirect requests which can be used to modify routing tables on hosts and cause denial of service attacks as well. Other types of attack with icmp are the smurf attack, where an icmp echo request is sent to a broadcast address, and the tribe flood network attack in which multiple compromised hosts, or spoofed host packets send icmp echo request. Both of these cause a denial of service attack. The ICMP protocol is associated with a protocol number of 3. But has several codes associated with it. A selection of these codes are as follows:

0 - echo reply
1 - host unreachable
3 - destination-unreachable
4 - source-quench
5 - redirect
8 - echo-request
11 - time exceeded
12 - parameter problem.
13 - administrivily prohibited

From this list, we want to allow codes 0, 1, 3, 4, 11, 12 and 13 in (as these are responses to requests we send out), but block 8 and 5.

```
access-list 101 deny icmp any any 3 5 log
access-list 101 deny icmp any any 3 8 log
```

It is important to note that the access-list 101 rules are used for packets coming in, so most of the packets which have ports destined for ports below 1024 are usually clients from outside accessing server services that we provide, rather than being responses to our internal clients. This access list is only for the external interface of the router, it does not consider the internal interface of the router. For filtering what traffic goes out of the network, we need to look at specifying an outbound interface access list.

Finally only allow permit packets for which a connection has been established on the unprivileged ports.

```
access-list 101 permit tcp any 130.155.16.0 0.0.255.255 gt 1023 established
access-list 101 permit tcp any 130.155.17.0 0.0.255.255 gt 1023 established
access-list 101 permit tcp any 130.155.18.0 0.0.255.255 gt 1023 established
```

```
access-list 101 permit tcp any 130.155.19.0 0.0.255.255 gt 1023 established
access-list 101 permit tcp any 130.155.20.0 0.0.255.255 gt 1023 established
access-list 101 permit tcp any 130.155.21.0 0.0.255.255 gt 1023 established
```

We now need to define the access list that we wish to apply on the external interface going out. Determining whether to have an output filter enabled, or an input filter on the internal interface to do the filtering to the outside world when there are only two interfaces in the router depends on the requirements. Here anyway I will be listing the output filter for the external interface. Why filter outgoing requests - it stops us from being a site which if people can get into, they cannot perform or find it difficult to perform attacks on other sites from ours. If we deny and log such activity occurring, it could be a sign that we have been compromised, other than being a malicious employee.

Let us first associate a the interface to the outbound access list. In general the outbound list is much smaller as we tend to deny less packets going out than going in.

```
interface ehternet 0
ip access-group 102 out    ! we would normally put this up the top when we
                           ! associated the in access-group 101 to the !
                           interface
```

In a similar manner in which we blocked incoming private addresses, we also want to block them outgoing.

```
access-list 102 deny ip 192.168.0.0 0.0.255.255 any log
access-list 102 deny ip 172.16.0.0 0.15.255.255 any log
access-list 102 deny ip 10.0.0.0 0.255.255.255 any log
access-list 102 deny ip 127.0.0.0 0.255.255.255 any log
access-list 102 deny ip 255.0.0.0 0.255.255.255 any log
access-list 102 deny ip 224.0.0.0 7.255.255.255 any log
access-list 102 deny ip host 0.0.0.0 any log
```

We not only want to deny packets with the source address as private, but we also want to deny destination as private.

```
access-list 102 deny ip any 192.168.0.0 0.0.255.255 log
access-list 102 deny ip any 172.16.0.0 0.15.255.255 log
access-list 102 deny ip any 10.0.0.0 0.255.255.255 log
access-list 102 deny ip any 255.0.0.0 0.255.255.255 log
access-list 102 deny ip any host 0.0.0.0 log
```

We need to control name services as well, we only want our name server to do a zone transfer with someone it knows, and allow LDAP to our VPN server

```
access-list 102 permit tcp host 130.155.18.2 host $specific_external_dns_ip  eq
53
access-list 102 deny tcp host 130.155.18.2 any eq 53 log
access-list 102 permit tcp host 130.155.21.2 any eq 389
access-list 102 deny tcp any any eq 389 log
```

Stop icmp messages echo reply, time exceeded and unreachable messages.

```
access-list 102 deny icmp any any 3 0 log
```

```
access-list 102 deny icmp any any 3 1 log
access-list 102 deny icmp any any 3 3 log
access-list 102 deny icmp any any 3 11 log
```

Allow everything else out.

```
access-list 102 permit ip any any
access-list 102 permit icmp any any
```

To test the router acl's software such as nmap can be used. This will allow you to spoof ip addresses, and to scan udp and tcp ports. For instance, to scan tcp ports on the DNS and Mail service network, we could use the following:

```
nmap -v 130.155.19.0
```

To scan udp ports open on the Mail and DNS service network, use the following nmap command:

```
nmap -sU -v 130.155.19.0
```

These commands will produce a report of how many hosts are on the network and what ports are open on the hosts.

The External Firewall

For the external firewall we were planning on using iptables for redhat linux. Iptables is similar to ipchains, however while ipchains is a static packet filter, iptables is a statefull packet filter. However, as already stated due to problems building a 2.4 kernel, we are using ipchains.

To add rules to the firewall, you run the *ipchains* command specifying the rule options afterwards. You must run the *ipchains* command for every rule you want to add. Due to the rule database being lost everytime the external firewall is rebooted, rules for the iptables database are best placed in a script which gets run at boot time.

The syntax of ipchains is as follows - taken from the output of *'ipchains -h'*.

```
ipchains 1.3.9, 17-Mar-1999

Usage: ipchains -[ADC] chain rule-specification [options]
       ipchains -[RI] chain rulenum rule-specification [options]
       ipchains -D chain rulenum [options]
       ipchains -[LFZNX] [chain] [options]
       ipchains -P chain target [options]
       ipchains -M [ -L | -S ] [options]
       ipchains -h [icmp] (print this help information, or ICMP list)

Commands:
Either long or short options are allowed.
  --add     -A chain                Append to chain
  --delete  -D chain                Delete matching rule from chain
  --delete  -D chain rulenum        Delete rule rulenum (1 = first) from chain
  --insert  -I chain [rulenum]      Insert in chain as rulenum (default 1=first)
  --replace -R chain rulenum        Replace rule rulenum (1 = first) in chain
```

```
  --list    -L [chain]               List the rules in a chain or all chains
  --flush   -F [chain]               Delete all rules in  chain or all chains
  --zero    -Z [chain]               Zero counters in chain or all chains
  --check   -C chain                 Test this packet on chain
  --new     -N chain                 Create a new user-defined chain
  --delete-chain
            -X chain                 Delete a user-defined chain
  --policy  -P chain target          Change policy on chain to target
  --masquerade -M -L                 List current masqerading connections
  --set     -M -S tcp tcpfin udp     Set masquerading timeout values

Options:
  --bidirectional -b                 insert two rules: one with -s & -d reversed
  --proto      -p [!] proto          protocol: by number or name, eg. `tcp'
  --source     -s [!] address[/mask] [!] [port[:port]]
                                     source specification
  --source-port [!] [port[:port]]
                                     source port specification
  --destination -d [!] address[/mask] [!] [port[:port]]
                                     destination specification
  --destination-port [!] [port[:port]]
                                     destination port specification
  --icmp-type [!] typename           specify ICMP type
  --interface  -i [!] name[+] network interface name ([+] for wildcard)
  --jump       -j target [port]      target for rule ([port] for REDIRECT)
  --mark       -m [+-]mark           number to mark on matching packet
  --numeric    -n                    numeric output of addresses and ports
  --log -l                           turn on kernel logging for matching packets
  --output     -o [maxsize]          output matching packet to netlink device
  --TOS -t and xor                   and/xor masks for TOS field
  --verbose    -v                    verbose mode
  --exact      -x                    expand numbers (display exact values)
[!] --fragment -f                    match second or further fragments only
[!] --syn      -y                    match TCP packets only when SYN set
[!] --version  -V                    print package version.
```

I will explain the most commonly used options of *ipchains* below. For further information, see the ipchains man page, or the 'ipchains-HOWTO' and the 'Firewall-HOWTO'.

There are 3 chains by default - input, output and forward. The input chain is used for filtering packets coming into an interface. The output chain is used for filtering packets going out of the interface. The forward chain is used when packets are to be forwarded to a different IP address and/or port.

When adding a rule, you can either append it to the end of the rule database using the *-A* option, or you can insert it using the *-I* option. When inserting you can also specify the line or rule number at which to insert the rule. In the following example we are going to block all traffic.

```
ipchains -A input -s 0.0.0.0/0 -d 0.0.0.0/0 -j DENY
```

To remove a rule, you use the *-D* option. To remove a rule you can either specify the rule number, or you specify the whole rule. So for the above adding example, if we wanted to remove it, and it was the only rule in the database, we could use the following commands:

```
ipchains -D input 1                     - delete 1st rule or
ipchains -D input                       - delete rule at top of rule list or
ipchains -D input -s 0.0.0.0/0 -d 0.0.0.0/0 -j DENY  - delete specific rule
```

To replace a rule, the *-R* option can be used. For example:

```
ipchains -R input 1 -s 0.0.0.0/0 -d 0.0.0.0/0 -j ACCEPT
```

Ipchains, like ipchains allows users to specify their own chains if they wish, this is done using the *-N* option, to delete a user specified chain, use the *-X* option, to rename a chain use the *-E* option. For example:

```
ipchains -N mychain              - create a chain called mychain
ipchains -X mychain              - delete a chain called mychain
ipchains -E mychain mynewchain   - rename a chain from mychain to mynewchain
```

To clear all the rules from a chain or all chains use the *-F* option. Don't specify a chain if you want to delete all database rules. For example:

```
ipchains -F input                - delete rules from the input chain
ipchains -F                      - delete rules from all chains
```

To list all the rules for a specific chain or all chains use the *-L* option. For example:

```
ipchains -L input                - list all the rules for the input chain
ipchains -L                      - list all rules on all chains.
```

To define default policies for each chain, use the *-P* option. Default policies determine the default action that ipchains takes when a packet arrives that there is no rule for. The main policy options are *ACCEPT, DENY or REJECT*. The difference between *DENY* and *REJECT* is that packets matched with *DENY* get dropped as if they never occurred, whereas packets matched with *REJECT* get a message sent back to the host sending the packet. For example:

```
ipchains -P input DENY           - by default DENY packets on the input chain
ipchains -P output ACCEPT        - by default accept packets on the output
chain
```

To define an interface to which you want to filter use the *-i* option followed by the name of the interface for the chain. For example:

```
ipchains -A input -i eth0 -s 0.0.0.0/0 -d 0.0.0.0/0 -j ACCEPT
ipchains -A output -i eth0 -s and -d 0.0.0.0/0 -j ACCEPT
```

To define source IP address or source network from where a packet you wish to filter comes from, use the *-s* option with the address. Similarly to filter based on destination address use the *-d* option with the address of the host or network. The wildcard *0.0.0.0/0* or *0/0* can be used for any address. For example:

```
ipchains -A INPUT -i eth0 -s 10.0.0.0/24 -d 0.0.0.0/0 -j ACCEPT
                                         - source is a network
ipchains -A INPUT -i eth0 -s 10.0.0.1 -d 0.0.0.0/0 -j ACCEPT
                                         - source is a host
```

For more granular filtering, source port(s) and destination port(s) can be specified using the by specifying the port after the source or destination respectively. When specifying ports, you can specify ranges by separating the lower range port and the higher range port with a colon ':'. For example:

```
ipchains -A input -i eth0 -s 0.0.0.0/0 80 -d 10.0.0.0/24 1024:65535 \
       -j ACCEPT
```

Packets can also be filtered by protocol as well. A protocol is specified by the *-p* option followed by either *tcp, udp* or *icmp* or the protocol number. The list of protocols and their associated numbers can be found at URL. Protocol specification can also be combined with source and destination ports as well. For example

```
ipchains -A input -i eth0 -p udp -s 10.0.0.0/24 1024:65535 \
       -p udp -d 10.0.1.3 53 -j ACCEPT
```

To define what action to take when a packet filter matches, the *-j* option is used. This can be seen in the previous examples.

The bang symbol *'!'* can be used to negate the expression. It is often used with protocol, IP addresses and port specification. For example to specify all ports but port 80, you could do the following:

```
ipchains -A input -i eth0 -p tcp  -s 0.0.0.0/0 ! 80 -d 0.0.0.0/0 -j DENY
```

The *-f* option can be used for matching fragments of packets. The *-n* option is used to tell ipchains not to resolve IP addresses to host names via DNS, but to leave the IP addresses in the output. To log a packet filter match, use the *-l* option. For example a rule to log may look like:

```
ipchains -A input -i eth0 -p tcp  -s 0.0.0.0/0 ! 80 -d 0.0.0.0/0 -j DENY -l
```

Another parameter that can be specified is *-y*. This refers to SYN. Usually this parameter is prefixed with the bang option, ie. *'! -y'* which says NOT a SYN packet. When do we use this?. We use the *'! -y'* option when we are replying to a service request, such as a DNS zone transfer. If we go back to the 3 way hand shake of tcp, a client requests a connection with SYN, the server replies with a SYN ACK and the client replies. Since we are the server, we never want to reply with just SYN. If we see just a SYN it means that the service port is requesting to establish a connection - something it should not normally do. So in summary services always reply, they never ask to establish a connection, that is why *'! -y'* (no SYN allowed) is used. It is also only used with tcp connections.

Let us now examine our business needs to find out our security policy for what we want to block at the firewall and what we want to let through.

Customers need access to our web servers so that they can purchase online fortune cookies. Because we are doing credit card transactions, we will want to use secure web. However for general information regarding GIAC Enterprises we want to let anyone get

information about the company profile and services it provides. So we will need to look at what ports and protocols must be let through to our network. Secure HTTP uses tcp on port 443, and general HTTP uses tcp port 80, so we want to allow this on our external firewall.
For customers to be able to access our web servers they will also need to be able to resolve IP addresses related to giac.com, thus we will need to enable DNS traffic to be allow through. This is udp port 53.

Customers may also want to send us e-mail, as well as suppliers and business partners, so we need to allow SMTP traffic through, so we need tcp port 25.

For our business partners, we want to have a VPN to them. The VPN uses protocols 51 (AH) or 50 (ESP) and use udp port 500 (ISAKMP) for transferring security key information between them, and port 389 (LDAP) for sending authentication information, so we need these ports open to our VPN server.

To keep all machines synchronised, we will use NTP, port 123 over udp. This will be between machines in our service networks, with the Application Gateway being the NTP server for the site. NTP will only be allowed from outside to the Application Gateway and only from a specific NTP site, which we trust. Similarly users may want network news, so we will have the Application Gateway as an NNTP proxy. We will allow outgoing finger, whois, telnet and ftp from the Application Gateway. Robert Zielger also recommends some the auth service in his book on Linux Firewalls, which may be needed for some services to work. All machines will log errors and messages to the Application Gateway which will be running a syslog daemon, so port 514 on udp will be open between the application gateway and other hosts in the service networks.

When the firewall is rebooted, it is necessary to load the firewall rules - because of this it is easier to put these rules in a script which starts up when the machine restarts. I will define variables for certain IP addresses that are used and for certain ports as it makes it easier to change in the future.

```
#IP address of external interface of VPN server
VPN_EXT_IP   = "130.155.21.2"
#IP address of public DNS server
DNS_EXT_IP   = "130.155.19.2"
#IP address of public Mail server
MAIL_EXT_IP  = "130.155.19.3"
#Service network and mask
SERVICE_NET  = "130.155.19.0/24"
#Web server network and netmask
WEB_NET      = "130.155.18.0/24"
#Ip address of external interface of Application Gateway
APPGW_EXT_IP = "130.155.20.2"
#Port used for database
DB_PORT      = "1512"
#Application Gateway network
APPGW_NET    = "130.155.20.0/24"
#The external IP address of the Firewall
FW_EXT_IP      = "130.155.17.2"
```

The first thing to do is enable the firewall to forward traffic:

```
#Turn on IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

The next thing is to clean out any previous rules that may exist and set the default policies for each of the chains. We by default deny any packet coming in, and reject all outgoing or forwarded packets. One thing to note is that when rules are flushed, the default policy of ipchains is to let everything through, that is why we are setting defaults to block all packets.

```
#Flush all chains
ipchains -F
#Set default policy for chains
ipchains -P input   DENY
ipchains -P output  ACCEPT
ipchains -P forward REJECT
```

We want to do the same packet filtering that has been done on the router for the blocking the basic traffic. The following lines will block the private addresses, and broadcast addresses. As well as blocking traffic which appears to be coming from inside (anti-spoofing).

```
#Block packets coming in on our external interface eth0 which
#have our external interfaces IP address as the source and log it.
ipchains -A input -i eth0 -s $FW_EXT_IP -j DENY -l
#Block packets with source or destination address of private
#networks coming in on all the interfaces
ipchains -A input -s 10.0.0.0/8 -j DENY
ipchains -A input -d 10.0.0./8 -j DENY
ipchains -A input -s 172.16.0.0/12 -j DENY
ipchains -A input -d 172.16.0.0/12 -j DENY
ipchains -A input -s 192.168.0.0/16 -j DENY
ipchains -A input -d 192.168.0.0/16 -j DENY
#Block packets going out with source or destination as a private
#address and log it.
ipchains -A output -s 10.0.0.0/8 -j DENY -l
ipchains -A output -d 10.0.0.0/8 -j DENY -l
ipchains -A output -s 172.16.0.0/12 -j DENY -l
ipchains -A output -d 172.16.0.0/12 -j DENY -l
ipchains -A output -s 192.168.0.0/16 -j DENY -l
ipchains -A output -d 192.168.0.0/16 -j DENY -l
#Block class E reserved packets from coming in or going out
ipchains -A input -s 240.0.0.0/5 -j DENY -l
ipchains -A output -s 240.0.0.0/5 -j DENY -l
```

Spoofing can also be reduce through the following setting on the firewall machine:

```
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
       echo 0 > $f
done
```

To block our firewall from SYN attacks and from source routed packets we can do the following commands:

```
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

With this when our firewall gets a connection establishment it sends back a SYN cookie to the requesting client, and clears the SYN from it's connection table. This stops our

machine from being attack from the SYN denial of service which fills up the connection table with SYN requests.

```
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
       echo 0 > $f
done
```

This command is used to turn off source routing.

For icmp control of messages - the last point which the SANS top ten mentions, we can use the following commands:

```
#Allow icmp from border router, but deny all other icmp echo requests from the Internet.
ipchains -A input -i eth0 -p icmp --icmp-type echo-request -s 130.155.17.1 -j ACCEPT
ipchains -A input -i eth0 -p icmp --icmp-type echo-request -s 0.0.0.0/0 -j DENY -l
```

Allow icmp between hosts on our service networks

```
ipchains -A input -i eth1 -p icmp -s 0.0.0.0/0 -d 0.0.0.0/0 -j ACCEPT
ipchains -A input -i eth2 -p icmp -s 0.0.0.0/0 -d 0.0.0.0/0 -j ACCEPT
ipchains -A input -i eth3 -p icmp -s 0.0.0.0/0 -d 0.0.0.0/0 -j ACCEPT
ipchains -A input -i eth4 -p icmp -s 0.0.0.0/0 -d 0.0.0.0/0 -j ACCEPT
```

Block outgoing icmp unreachable, time-exceeded and echo replies to all but the border router.

```
ipchains -A output -i eth0 -p icmp -s 0.0.0.0/0 -d 130.155.17.1 -j ACCEPT
ipchains -A output -i eth0 -p icmp --icmp-type  echo-reply -d 0.0.0.0/0 -j DENY -l
ipchains -A output -i eth0 -p icmp --icmp-type  destination-unreachable  -d 0.0.0.0/0 -j
DENY -l
ipchains -A output -i eth0 -p icmp --icmp-type  time-exceeded -d 0.0.0.0/0 -j DENY -l
```

The VPN Network

In the next few rules, we want to allow access to our VPN server for hosts or security gateways to establish IPSec connections:

```
#Allow access to VPN server from Internet on external interface eth0:
# Allow LDAP through for authentication
ipchains -A input  -i eth0 -s 0.0.0.0/0 389 -d $VPN_EXT_IP 389 -j ACCEPT
ipchains -A output -i eth0 ! -y -p tcp -s $VPN_EXT_IP 389 -d 0.0.0.0/0 389 -j ACCEPT
ipchains -A output -i eth0 -p udp -s $VPN_EXT_IP 389 -d 0.0.0.0/0 389 -j ACCEPT

# Allow ISAKMP/IKE for Key exchange
ipchains -A input  -i eth0 -p udp -s 0.0.0.0/0 500 -d $VPN_EXT_IP 500 -j ACCEPT
ipchains -A output -i eth0 -p udp -s $VPN_EXT_IP 500 -d 0.0.0.0/0 500 -j ACCEPT
# Allow ESP protocol
ipchains -A input  -i eth0 -p 50 -s 0.0.0.0/0 -d $VPN_EXT_IP -j ACCEPT
ipchains -A output -i eth0 -p 50 -s $VPN_EXT_IP -d 0.0.0.0/0 -j ACCEPT
#We must now do the same one the internal interface connected to the VPN - eth4
#Allow LDAP through for authentication
ipchains -A input  -i eth4 -s $VPN_EXT_IP 389 -d 0.0.0.0/0 389 -j ACCEPT
ipchains -A output -i eth4 -s 0.0.0.0/0 389 -d $VPN_EXT_IP 389 -j ACCEPT
# Allow ISAKMP/IKE for Key exchange
ipchains -A input  -i eth4 -p udp -s $VPN_EXT_IP 500 -d 0.0.0.0/0 500 -j ACCEPT
ipchains -A output -i eth4 -p udp -s 0.0.0.0/0 500 -d $VPN_EXT_IP 500 -j ACCEPT
# Allow ESP protocol
ipchains -A input  -i eth4 -p 50 -s $VPN_EXT_IP -d 0.0.0.0/0 -j ACCEPT
ipchains -A output -i eth4 -p 50 -s 0.0.0.0/0 -d $VPN_EXT_IP -j ACCEPT
```

We also want the VPN server to be able to DNS lookups from the external DNS server. The next rules do this.

```
#Allow VPN server to do DNS lookups
ipchains -A input -i eth4 -p udp -s $VPN_EXT_IP 1024:65535 -d $DNS_EXT_IP -j ACCEPT
ipchains -A output -i eth4 -p udp -s $DNS_EXT_IP 53 -d $VPN_EXT_IP 1024:65535 -j ACCEPT
```

We want to allow the VPN server to synchronise it's time with the NTP server which is the application gateway.

```
#Allow VPN server to sync with NTP server.
ipchains -A input -i eth4 -p udp -s $VPN_EXT_IP 1024:65535 -d $APPGW_EXT_IP 123 -j ACCEPT
ipchains -A output -i eth4 -p udp -s $APPGW_EXT_IP 123 -d $VPN_EXT_IP 1024:65535 -j
ACCEPT
```

Allow the VPN server to write it's log information to the log server, which is the application gateway.

```
#Allow VPN server to write to syslog server
ipchains -A input -i eth4 -p udp -s $VPN_EXT_IP 1024:65535 -d $APPGW_EXT_IP 514 -j ACCEPT
ipchains -A output -i eth4 -p udp -s $APPGW_EXT_IP 514 -d $VPN_EXT_IP 1024:65535 -j
ACCEPT
```

We have allowed all that we want to allow, so all other traffic should be denied going to the VPN server.

```
#Block all other packets to VPN server
ipchains -A input -i eth0 -s 0.0.0.0/0 -d $VPN_EXT_IP -j DENY -l
#Block all other packets from VPN to external firewall
ipchains -A input -i eth4 -s 0.0.0.0/0 -d 0.0.0.0/0 -j DENY -l
```

The Web Service Network

We'll now look at the web services network and see what rules we must have. We want HTTP and SSL access to the web from the Internet.

```
#Allow access to the web servers from Internet:
#For the external interface eth0 connected to the Internet
#Allow for HTTP
ipchains -A input  -i eth0 -p tcp -s 0.0.0.0/0 1024:65535 -d $WEB_NET 80 -j ACCEPT
ipchains -A output -i eth0 ! -y -p tcp -s $WEB_NET 80 -d 0.0.0.0/0 1024:65535 -j ACCEPT
#Allow for HTTPS/SSL
ipchains -A input  -i eth0 -p tcp -s 0.0.0.0/0 1024:65535 -d $WEB_NET 443 -j ACCEPT
ipchains -A output -i eth0 ! -y -p tcp -s $WEB_NET 443 -d 0.0.0.0/0 1024:65535 -j ACCEPT

#For the internal interface eth1 connected to the web network
#Allow for HTTP
ipchains -A input  -i eth1 ! -y -p tcp -s $WEB_NET 80 -d 0.0.0.0/0 1024:65535 -j ACCEPT
ipchains -A output -i eth1 -p tcp -s 0.0.0.0/0 1024:65535 -d $WEB_NET 80 -j ACCEPT
#Allow for HTTPS/SSL
ipchains -A input  -i eth1 ! -y -p tcp -s $WEB_NET 443 -d 0.0.0.0/0 1024:65535 -j ACCEPT
ipchains -A output -i eth1 -p tcp -s 0.0.0.0/0 1024:65535 -d $WEB_NET 443 -j ACCEPT
```

We want to allow the web servers to synchronise their time with the NTP server which is the application gateway.

```
#Allow Web servers to sync with NTP server.
ipchains -A input -i eth1 -p udp -s $WEB_NET 1024:65535 -d $APPGW_EXT_IP 123 -j ACCEPT
ipchains -A output -i eth1 -p udp -s $APPGW_EXT_IP 123 -d $WEB_NET 1024:65535 -j ACCEPT
```

Allow the web servers to write their log info to the log server which is the application gateway.

```
#Allow Web servers to write to syslog server
ipchains -A input -i eth1 -p udp -s $WEB_NET 1024:65535 -d $APPGW_EXT_IP 514 -j ACCEPT
ipchains -A output -i eth1 -p udp -s $APPGW_EXT_IP 514 -d $WEB_NET 1024:65535 -j ACCEPT
```

We want the web servers to be able to communicate with the database proxy running on the Application Gateway.

```
#Allow Web servers to write talk to Database Proxy
ipchains -A input -i eth1 -s $WEB_NET 1024:65535 -d $APPGW_EXT_IP $DB_PORT -j ACCEPT
ipchains -A output -i eth1 -s $APPGW_EXT_IP $DB_PORT -d $WEB_NET 1024:65535 -j ACCEPT
```

We have allowed all the traffic we want to the web servers, so we will deny everything else.

```
#Deny everything else going and coming from web servers.
ipchains -A input -i eth1 -s 0.0.0.0/0 -d 0.0.0.0/0 -j DENY -l
ipchains -A output -i eth1 -s 0.0.0.0/0 -d 0.0.0.0/0 -j DENY -l
#Block other traffic on eth0
ipchains -A input -i eth0 -s 0.0.0.0/0 -d $WEB_NET -j DENY -l
```

The External DNS and Mail services Network

We will now look at the Mail server and DNS server on the service network. We want to allow mail to go to the mail server only, and dns queries and zone transfers to the DNS server only.

```
#Allow access to Mail server from Internet:
#For external interface eth0:
#Allow for SMTP for other mail servers
ipchains -A input  -i eth0 -p tcp -s 0.0.0.0/0 25 -d $MAIL_EXT_IP 25 -j ACCEPT
ipchains -A output -i eth0 ! -y -p tcp -s $MAIL_EXT_IP 25 -d 0.0.0.0/0 25 -j ACCEPT
#For internal interface eth2:
#Allow for SMTP for other mail servers
ipchains -A input  -i eth2 ! -y -p tcp -s $MAIL_EXT_IP 25 -d 0.0.0.0/0 25 -j ACCEPT
ipchains -A output -i eth2 -p tcp -s 0.0.0.0/0 25 -d $MAIL_EXT_IP 25 -j ACCEPT

#Allow access to DNS server for client queries:
#For external interface eth0:
#Allow clients from any to query DNS server
ipchains -A input -i eth0 -p udp -s 0.0.0.0/0 1024:65535 -d $DNS_EXT_IP 53 -j ACCEPT
ipchains -A output -i eth0 -p udp -s $DNS_EXT_IP 53 -d 0.0.0.0/0 1024:65535 -j ACCEPT
#For internal interface eth2:
#Allow DNS to answer queries from clients
ipchains -A input -i eth2 -p udp -s $DNS_EXT_IP 53 -d 0.0.0.0/0 1024:65535 -j ACCEPT
ipchains -A output -i eth2 -p udp -s 0.0.0.0/0 1024:65535 -d $DNS_EXT_IP 53 -j ACCEPT

#Allow web servers to query our DNS server:
#For internal interface eth1:
#Allow web servers to do DNS queries to our DNS server
ipchains -A input -i eth1 -p udp -s $WEB_NET 1024:65535 -d $DNS_EXT_IP 53 -j ACCEPT
ipchains -A output -i eth1 -p udp -s $DNS_EXT_IP 53 -d $WEB_NET 1024:65535 -j ACCEPT
```

We don't need to add another reciprocal rule for the interface eth2 for the DNS server to accept DNS queries, as the rules previously done for the DNS already take care of it. The next rules allow the Mail relay to talk to the mail relay on the application gateway.

```
#Allow mail server to relay to mail proxy
#Allow external Mail server to relay mail to proxy server
ipchains -A input -i eth2  -p tcp -s $MAIL_EXT_IP 25 -d $APPGW_EXT_IP 25 -j ACCEPT
```

```
ipchains -A output -i eth2 -p tcp -s $APPGW_EXT_IP 25 -d $MAIL_EXT_IP 25 -j ACCEPT
```

We now want to allow the DNS and Mail servers to be able to sync with the time server and write logs to the syslog daemon.

```
#Allow Mail and DNS servers to sync with time server
ipchains -A input -i eth2 -p udp -s $SERVICE_NET 1024:65535 -d $APPGW_EXT_IP 123 -j
ACCEPT
ipchains -A output -i eth2 -p udp -s $APPGW_EXT_IP 123 -d $SERVICE_NET 1024:65535 -j
ACCEPT
#Allow Mail and DNS servers to write to syslog server
ipchains -A input -i eth2 -p udp -s $SERVICE_NET 1024:65535 -d $APPGW_EXT_IP 514 -j
ACCEPT
ipchains -A output -i eth2 -p udp -s $APPGW_EXT_IP 514 -d $SERVICE_NET 1024:65535 -j
ACCEPT
```

Allow DNS server on proxy to forward DNS queries:

```
#for external DNS to talk to DNS server on proxy for DNS forwarding
ipchains -A input -i eth2 -p udp -s $DNS_EXT_IP 53 -d $APPGW_EXT_IP 53 -j ACCEPT
ipchains -A output -i eth2 -p udp -s $APPGW_EXT_IP 53 -d $DNS_EXT_IP 53 -j ACCEPT
```

Allow DNS server to talk to a specific external DNS server which we trust:

```
#Allow DNS Server to talk to specific Internet DNS server for zone transfers.
ipchains -A input -i eth0 -p tcp -s $SPECIFIC_EXT_DNS_IP 53 -d $DNS_EXT_IP 53 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $DNS_EXT_IP 53 -d $SPECIFIC_EXT_DNS_IP 53 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $SPECIFIC_EXT_DNS_IP 53 -d $DNS_EXT_IP 53 -j ACCEPT
ipchains -A input -i eth2 -p tcp -s $DNS_EXT_IP 53 -d $SPECIFIC_EXT_DNS_IP 53 -j ACCEPT
```

Deny all other traffic to the service network:

```
#Deny everything else going and coming from service network
ipchains -A input -i eth2 -s 0.0.0.0/0 -d 0.0.0.0/0 -j DENY -l
ipchains -A output -i eth2 -s 0.0.0.0/0 -d 0.0.0.0/0 -j DENY -l
#Block other traffic on eth0
ipchains -A input -i eth0 -s 0.0.0.0/0 -d $SERVICE_NET -j DENY -l
```

The Application Gateway Network

With the application gateway we want to allow the web servers to talk to the database proxy the application gateway is running.

```
#For the internal interface eth3:
#Allow web to talk to DB proxy
ipchains -A input -i eth3 -s $APPGW_EXT_IP $DB_PORT -d $WEB_NET 1024:65535 -j ACCEPT
ipchains -A output -i eth3 -s $WEB_NET 1024:65535 -d $APPGW_EXT_IP $DB_PORT -j ACCEPT
```

We want to allow the external Mail server to relay mail to the application gateway which is running a mail relay to the internal mail server.

```
#Allow mail proxy to talk to external mail server
ipchains -A input -i eth3 -p tcp -s $APPGW_EXT_IP 25 -d $MAIL_EXT_IP 25 -j ACCEPT
ipchains -A output -i eth3 -p tcp -s $MAIL_EXT_IP 25 -d $APPGW_EXT_IP 25 -j ACCEPT
```

We want to be able to do DNS queries to the DNS server, as well as do DNS forwarding to the external DNS server.

```
#Allow DNS lookups between DNS on proxy and external DNS
#For external DNS to DNS proxy client queries
ipchains -A input -i eth3 -p udp -s $APPGW_EXT_IP 1024:65535 -d $DNS_EXT_IP 53 -j ACCEPT
ipchains -A output eth3 -p udp -s $DNS_EXT_IP 53 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
```

we don't need to do the reverse for interface eth2 as it accepts client queries from external clients, however we want to have our proxy dns be a forwarder to the external dns, we don't want the proxy dns to be recursive (leads to DNS poisoning attacks).

```
#for external DNS to talk to DNS server on proxy for DNS forwarding
ipchains -A input -i eth2 -p udp -s $DNS_EXT_IP 53 -d $APPGW_EXT_IP 53 -j ACCEPT
ipchains -A output -i eth2 -p udp -s $APPGW_EXT_IP 53 -d $DNS_EXT_IP 53 -j ACCEPT
ipchains -A input -i eth3 -p udp -s $APPGW_EXT_IP 53 -d $DNS_EXT_IP 53 -j ACCEPT
ipchains -A output -i eth3 -p udp -s $DNS_EXT_IP 53 -d $DNS_EXT_IP 53 -j ACCEPT
```

We want to control internal client request for our proxy server - basically most of the traffic coming from the proxy server will be proxy services for clients on our internal network - apart from the auth service.

```
#Web services - client only Real audio + QuickTime can be sent through http
# so we won't be adding entries for the specific ports that could be used for it.
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 80 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 80 -j ACCEPT
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 443 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 443 -j ACCEPT
ipchains -A input -i eth3  -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 80 -j ACCEPT
ipchains -A output -i eth3 -p tcp -s 0.0.0.0/0 80 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A input -i eth3 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 443 -j ACCEPT
ipchains -A output -i eth3 -p tcp -s 0.0.0.0/0 443 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT


#FTP services - client only
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 20 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 20 -j ACCEPT
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 21 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 21 -j ACCEPT
ipchains -A input -i eth3 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 20 -j ACCEPT
ipchains -A output -i eth3 -p tcp -s 0.0.0.0/0 20 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A input -i eth3 -p tcp -s $APPGW_EXT_IP 1024:65535 -s 0.0.0.0/0 21 -j ACCEPT
ipchains -A output -i eth3 -p tcp -s 0.0.0.0/0 21 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT

#Telnet Services - client only
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 23 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 23 -j ACCEPT
ipchains -A input -i eth3 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 23 -j ACCEPT
ipchains -A output -i eth3 -p tcp -s 0.0.0.0/0 23 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT

#Auth Ident Service (recommended from Linux Firewalls)
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 1024:65535 -d $APPGW_EXT_IP 113 -j ACCEPT
ipchains -A output -i eth0 ! -y -p tcp -s $APPGW_EXT_IP 113 -d 0.0.0.0/0 1024:65535 -j
ACCEPT
ipchains -A input -i eth3 ! -y -p tcp -s $APPGW_EXT_IP 113 -d 0.0.0.0/0 1024:65535 -j
ACCEPT
ipchains -A output -i eth3 -p tcp -s 0.0.0.0/0 1024:65535 -d $APPGW_EXT_IP 113 -j ACCEPT

#Auth Ident Service as a client
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 113 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP -d 0.0.0.0/0 113 -j ACCEPT
ipchains -A input -p -i eth3 tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 113 -j ACCEPT
ipchains -A output -i eth3 -p tcp -s 0.0.0.0/0 113 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT

#NNTP Service - client only
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 119 -d $APPGW_EXT_IP 1024:65535 - j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 119 -j ACCEPT
ipchains -A input -i eth3 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 119 -j ACCEPT
ipchains -A output -i eth3 -p tcp -s 0.0.0.0/0 119 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT

#SSH Service - client only
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 22 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 22 -j ACCEPT
ipchains -A input -i eth3 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 22 -j ACCEPT
```

```
ipchains -A output -i eth3 -p tcp -s 0.0.0.0/0 22 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT

#WHOIS Service - client only
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 43 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 43 -j ACCEPT
ipchains -A input -i eth3 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 43 -j ACCEPT
ipchains -A output -i eth3 -p tcp -s 0.0.0.0/0 43 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT

#Finger Service - client only
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 79 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 79 -j ACCEPT
ipchains -A input -i eth3 -p tcp -s $APPGW_EXT_IP 1024:65535 -d 0.0.0.0/0 79 -j ACCEPT
ipchains -A output -i eth3 -p tcp -s 0.0.0.0/0 79 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
```

We will have the Application gateway as the NTP server for service networks as well as for machines on our internal private address space. We won't allow NTP service requests from the Internet to the application gateway, we will however allow NTP requests out for the external NTP server which the application gateway synchronises with.

```
#NTP as Client
ipchains -A input -i eth0 -p udp -s $NTP_SERVER 119 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p udp -s $APPGW_EXT_IP 1024:65535 -d $NTP_SERVER 123 -j
ACCEPT
ipchains -A output -i eth3 -p udp -s $NTP_SERVER 119 -d $APPGW_EXT_IP 1024:65535 -j
ACCEPT
ipchains -A input -i eth3 -p udp -s $APPGW_EXT_IP 1024:65535 -d $NTP_SERVER 123 -j ACCEPT

#NTP as Server
#NTP for VPN server
ipchains -A input -i eth4 -p udp -s $VPN_EXT_IP 1024:65535 -d $APPGW_EXT_IP 123 -j ACCEPT
ipchains -A output -i eth4 -p udp -s $APPGW_EXT_IP 123 -d $VPN_EXT_IP 1024:65535 -j
ACCEPT
ipchains -A input -i eth3 -p udp -s $APPGW_EXT_IP 123 -d $VPN_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth3 -p udp -s $VPN_EXT_IP 1024:65535 -d $APPGW_EXT_IP 123 -j
ACCEPT

#NTP for WEB servers
ipchains -A input -i eth1 -p udp -s $WEB_NET_IP 1024:65535 -d $APPGW_EXT_IP 123 -j ACCEPT
ipchains -A output -i eth1 -p udp -s $APPGW_EXT_IP 123 -d $WEB_NET_IP 1024:65536 -j
ACCEPT
ipchains -A input -i eth3 -p udp -s $APPGW_EXT_IP 123 -d $WEB_NET_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth3 -p udp -s $WEB_NET_IP 1024:65535 -d $APPGW_EXT_IP 123 -j
ACCEPT

#NTP for Mail + DNS screened service net
ipchains -A input -i eth2 -p udp -s $SERVICE _NET 1024:65535 -d $APPGW_EXT_IP 123 -j
ACCEPT
ipchains -A output -i eth2 -p udp -s $APPGW_EXT_IP 123 -d $SERVICE_NET 1024:65535 -j
ACCEPT
ipchains -A input -i eth3 -p udp -s $APPGW_EXT_IP 123 -d $SERVICE_NET 1024:65535 -j
ACCEPT
ipchains -A output -i eth3 -p udp -s $SERVICE_NET 1024:65535 -d $APPGW_EXT_IP 123 -j
ACCEPT
```

The application gateway is also running as the syslog daemon, so we must allow connections to our syslog server from our service networks, but not from outside.

```
#Allow syslog from The VPN network
ipchains -A input -i eth3 -p udp -s $APPGW_EXT_IP 514 -d $VPN_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth3 -p udp -s $VPN_EXT_IP 1024:65535 -d $APPGW_EXT_IP 514 -j
ACCEPT

#Allow syslog from Web network
ipchains -A input -i eth3 -p udp -s $APPGW_EXT_IP 514 -d $WEB_NET 1024:65535 -j ACCEPT
ipchains -A output -i eth3 -p udp -s $WEB_NET 1024:65535 -d $APPGW_EXT_IP 514 -j ACCEPT

#Allow syslog from DNS and Mail service network
```

```
ipchains -A input -i eth3 -p udp -s $APPGW_EXT_IP 514 -d $SERVICE_NET 1024:65535 -j
ACCEPT
ipchains -A output -i eth3 -p udp -s $SERVICE_NET 1024:65535 -d $APPGW_EXT_IP 514 -j
ACCEPT
```

Deny everything else to or from the Application Gateway network:

```
#deny all other packets to GW net
ipchains -A input -i eth0 -s 0.0.0.0/0 -d $APPGW_NET -j DENY -l
# deny all other packets outgoing from GW net
ipchains -A input -i eth3 -s 0.0.0.0/0 -d 0.0.0.0/0 -j DENY -l
```

To test the external firewall rules, port scanners such as nmap and hping2 can be used to scan open ports on the networks, to make sure we have not left anything open. This testing is very similar to the testing of the border router. A detailed description of how to test the firewall is described in Assignment 3 - Audit Your Security Policy.

The VPN Server

The VPN server consists of the configuration of the IPSec software, and the configuration of the IP filters that it uses. The filtering is done with ipchains. We will first look at the configuration of the IPSec software and then the filtering. For more information on the VPN setup in terms of filtering rules, see the VPN-HOWTO and the VPN-Masquerade-HOWTO.

The IPSec Software Configuration - Freeswan

In setting up the VPN server we will look at the configuration of IPSec for establishing a secure connection with business partners, as well as for secure connections with remote users (the 'Road Warriors'). For security gateways between networks, Freeswan is compatible with most setups (we assume that it is) for the business partners associated with GIAC Enterprises. For remote users with laptops, they will either be using a linux or Microsoft Windows. In general however, most users have Microsoft Windows on their laptops. We will assume that users are using Microsoft Windows 2000. By default Freeswan appears to enable split-horizon. Split Horizon in IPSec terms is whether when the VPN connection is established, all traffic no matter whether local or remote, goes via the VPN tunnel, or whether local traffic goes locally and only remote traffic goes via the VPN tunnel. This Split-Horizon should not be confused with router split-horizon which is whether routing information relating to neighbours is passed on to other routers or not.

There are two ways to identify either end of a connection when establishing a link - manual keying or automatic keying. In manual keying the two ends share a key - this key will be used for the rest of the session for identifying each end. In automatic keying, the two ends negotiate the keys for identifying each other, and will periodically negotiate new keys during the session. Automatic keying provides better security as if a key is

found out by an intruder, it will only be of use for a small amount of time as the keys periodically change. We will be using automatic keying, which is the most commonly used. For more information see http://www.freeswan.org.

In automatic keying, authentication between machines is either the 'preshared secrets' or the use of public keys. We will assume that we will be using certificates for public keys, for use with our remote users. Windows 2000 uses X.509 certificates, for Freeswan to support this, additional patches are required. Also, by default, Windows 2000 only supports DES encryption - the implementors of Freeswan however, along with other implementations of IPSec consider DES to be a low encryption mechanism and so do not support it. So in order for Windows 2000 and Freeswan to interoperate, Windows 2000 machines need to get the high encryption patch so they can do 3DES. By default if no other encryption is specified Freeswan uses 3DES. For more information see http://www.strongsec.com/freeswan.

The Freeswan implementation of IPSec refers to two components which you should be aware of:

- KLIPS which is the kernel part of IPSec. It implements the AH and ESP protocols and also does the packet handling in the kernel.
- Pluto which is a daemon process which implements IKE and connection negotiation with other systems.

For installation of Freeswan, see the Freeswan installation documentation which can be found at http://www.freeswan.org.

The configuration file for Freeswan is /etc/ipsec.conf. It consists of several sections, two of which are used by all connections. The first of these is the '*config setup*' section which describes the configuration of the local machine, the second is the '*conn %default*' which describes the default connection details for all connections. After these, all other sections use '*conn <connection name>*', where *<connection name>* is a label to identify the connection configuration details for each of the IPSec connections which are setup on the machine.

The basic configuration for the config section is as follows:

```
config setup
       interfaces=%defaultroute
       klipsdebug=none
       plutodebug=none
       plutoload=%search
       plutostart=%search
       uniqueids=yes
```

The first line of the config setup specifies the interface the Ipsec gateway will use to talk to other Ipsec connections. *'%defaultroute'* will use the default gateway defined in your routing table, you can specify specific interfaces if you have more than one that you will be using as the IPSec gateway out. The *klipsdebug* and *plutodebug* options are used for debugging purposes and in most cases, unless you are troubleshooting, will be set to

'none'. The *plutoload* and *plutostart* options can be used to list the connections to be loaded into memory when pluto starts, and list of connections to startup when pluto starts respectively. When set to *'%search'* pluto will load and startup any connection with *'auto=add'* and *'auto=start'* respectively. For IPSec tunnels which are permanently up, the *'auto=start'* should be used - this will tell Freeswan to startup the connection, and is useful if a connection is broken as it will be restarted automatically. For those connections where the connection must be initiated by the other end, such as for remote users, then the *'auto=load'* option should be used. The *uniqueid* is used to control whether two or more connections with the same subnet on the other end of the IPSec connection are allowed. Usually there will only be one tunnel to the other end. By specifying *'yes'* it means that if the connection is broken with the other end and a new connection is established, the old one is torn down.

The basic configuration for the *conn %default* section is:

```
conn %default
       keyingtries=0
       authby=rsasig
       syslog=daemon.error
```

The *conn %default* options are used by all other connections if they are not specifically listed in the other connections. The *keyingtries* is used to specify how many times we should try to connect or reconnect a connection - by specifying zero this means to try forever. The *authby* is used to specify how to authenticate other IPSec gateways, RSA signatures is one method, shared secrets is another. We will be using RSA signatures with our connections. The shared secrets and private keys of the VPN server are kept in the */etc/ipsecrets.conf* file. The *syslog* specifies what error level we should be specifying for messages.

Lets look at the other connection configurations:

```
conn giac-babble

       keyexchange=ike
       ikelifetime=480m
       auth=esp
       type=tunnel
       esp=3des-md5-96
                    (implicit values that are not normally necessary to define)
       leftid=vpn@giac.com
       leftrsasigkey=0x03010001B9D41654B7B38DFF617292F63B058C294B306BD7AB4E
       CC7A249C187C0F60202E96892D62EF0549F3D497631B6501B03EABC1CDC35C0C4858
       6BC22C986D3509E90FB197AE99E99B9224D7B7BB1A754B2E0CC62D19B6672DB5AD01
       B27B56A93C98F8CDBD01A6B41B10B29BAFAD5BD65426680E6AF96F19CDE410FDD0FC
       4B6D1331
       leftsubnet=172.16.5.0/24
              (our internal network)
       leftfirewall=yes
       left=130.155.20.2
```

```
        (the external interface of the VPN/IPSec server)
        (could also use %defaultroute)
leftnexthop=130.155.20.1
        (the internal interface of our external firewall)
rightid=vpn@babble.com
rightrsasigkey=0x03010001DE04BCAEFAE21AD45B6F40FD06FD39E7EE15BC54D13
0AB9AF34DF80B0E9794B036105F45A25B2F69AE31CC03E12CBC275ACD68173AD3FB5
9A0E5452A60BAB4C65648A447B3B553D94B6DE0AA471A5C51D833D808554BF639D34
F073C8649961EAF9ADD2B617B8F331C51DF1A03D0450E11E348040BE02AD6A9D9ABE
A40DCD295
rightnexthop=140.155.10.1
        (the internal interface of the router/firewall at the other end)
right=140.155.10.2
        (the external interface of the VPN at the remote/other end)
rightsubnet=10.0.1.0
        (the network at the other end)
auto=start
```

Here our connection is with our business partner 'Babble' our international partner who translates fortune cookie sayings to other languages ☺. I have included some options (shown in italics above) which you do not normally have to specify as they are implicit by default. The *type* option specifies the IPSec mode - transport or tunnel. The *keyexchange* specifies the protocol to exchange keys with, *'ike'* is default, others would be *iaskmp*. They *ikelifetime* is used to specify how often each end should re-negotiate keys. The *auth* option is used to specify the protocol to use, either *'ah'* for Authentication Header, or *'esp'* for Encapsulated Security Payload. Whichever auth method is chosen, the related encryption algorithm can be specified, which in this case is triple DES with MD5.
The *leftid* and *rightid* options are used as identifiers for the RSA keys of the machines at each end of the connection. This is usually the fully qualified hostname or address of the machine with an at *'@'* symbol infront of it which tells Freeswan not to resolve it. The *leftrsasigkey* and *rightrsasigkey* options specify the RSA public keys for each of the machines at the end of the connection. In future the implementors of Freeswan hope to use secure DNS to obtain the key information, in which case the value for each of these would be *'%dns'*.
The *leftsubnet* and *rightsubnet* options identify the subnets at either end of the connection. In the case where you want the subnet to be hidden, you can specify *leftsubnet* without any value and don't specify *leftfirewall*.
The *left* and *right* options identify the IP address of the external interface of each of the IPSec servers - we could have also used *'left=%defaultroute'* in the case where the default route was the IP of the interface we wanted packets sent via. The *nextlefthop* and *nextrighthop* identify the IP address of the gateway or router that the IPSec gateway will send the packets to in order to get to the rest of the world. While this information exists in the normal routing table, it must be specified as the IPSec kernel (KILPS) bypasses this when sending packets. In our case the *nextlefthop* is the IP address of the internal interface of the firewall that the VPN server connects to. The *auto* option is set to *'start'* as we want the link established when the IPSec server starts up. In general, most Freeswan documentation uses the *left* side for our local security gateway, and the *right* side refers to the remote end of the connection.

Now we will have a look at the configuration of the connection for the remote user:

```
conn roadwarrior1
       leftid=@vpn.giac.com
       leftrsasigkey=0x03010001B9D41654B7B38DFF617292F63B058C294B306BD7AB4E
       CC7A249C187C0F60202E96892D62EF0549F3D497631B6501B03EABC1CDC35C0C4858
       6BC22C986D3509E90FB197AE99E99B9224D7B7BB1A754B2E0CC62D19B6672DB5AD01
       B27B56A93C98F8CDBD01A6B41B10B29BAFAD5BD65426680E6AF96F19CDE410FDD0FC
       4B6D1331
       leftsubnet=172.16.5.0/24
                     (our internal network)
       left=130.155.20.2
       leftnexthop=130.155.20.1
                     (the internal interface of our external firewall)
       rightid=@roadwarrior1.giac.com
       rightrsasigkey=0x03010001DE04BCAEFAE21AD45B6F40FD06FD39E7EE15BC54D13
       0AB9AF34DF80B0E9794B036105F45A25B2F69AE31CC03E12CBC275ACD68173AD3FB5
       9A0E5452A60BAB4C65648A447B3B553D94B6DE0AA471A5C51D833D808554BF639D34
       F073C8649961EAF9ADD2B617B8F331C51DF1A03D0450E11E348040BE02AD6A9D9ABE
       A40DCD295
       right=%any
       keyingtries=1
       auto=load
```

Here we have the *right* option set to *'%any'* as we don't know what the IP address of the remote machine will be. For a similar reason the *rightnexthop* and *rightsubnet* are not listed. The number of retries of the connection in the *keyingtries* is set to *'1'* - this is because once the connection to the remote user is terminated, we do not want to keep trying to re-establish the connection. We have *auto* set to *'load'* since we don't want to establish the link at startup.

With the X.509 patches and fswcert installed, the remote user configuration would look like the following, as fswcert replaces the *leftid*, *leftrsasigkey* and *rightid* and *rightrsasigkey* option pairs.

```
conn roadwarrior2
       leftcert=giaccert.pem
       leftsubnet=172.16.5.0/24
              (our internal network)
       left=130.155.20.2
       leftnexthop=130.155.20.1
              (the internal interface of our external firewall)
       rightcert=roadwarrior2cert.pem
       right=%any
       keyingtries=1
       auto=load
```

In the above example, certificate files are used to authenticate each end. The certification files *giaccert.pem* and *roadwarrior2cert.pem* are kept in the */etc/ipsec.d/* directory.

In order to generate the certificates, *openssl* can be installed. The steps to creating the certificate are provided in the paper by Oscar Delgado.

The first step is to create a new certificate authority for which we can sign certificates. In order to make it easier to use openssl, install the openssl-perl module which provides the script CA.pl which is a front end to openssl. We do this via:

```
# CA.pl -newca
```

This will either ask as for a filename or if we don't specify anything create a new certificate. When a new certificate is created you will be asked to fill in details about the certificate. For example, the following is the output of running the command:

```
[root@vpn /root]# /usr/share/ssl/misc/CA.pl -newca
CA certificate filename (or enter to create)

Making CA certificate ...
Using configuration from /usr/share/ssl/openssl.cnf
Generating a 1024 bit RSA private key
............++++++
.....................................++++++
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:NSW
Locality Name (eg, city) []:North Ryde
Organization Name (eg, company) [Internet Widgits Pty Ltd]:GIAC Enterprises
Organizational Unit Name (eg, section) []:SysAdministration
Common Name (eg, your name or your server's hostname) []:vpn.giac.com
Email Address []:vpnuser@giac.com
[root@vpn /root]#
```

Once running this, the a directory called *demoCA* is created. Under this is the certificate *demoCA/cacert.pem* and the private key in *demoCA/private/cakey.pem*. The initial release of Windows 2000 has some problems with the certificates that openssl generates as openssl allows the date the certificate is valid for to be outside the valid time period. To fix this, we must change the date of the certificate. To do this we use the following command:

```
# openssl x509 -in demoCA/cacert.pem -days 1024 -out demoCA/newcacert.pem -
signkey demoCA/private/cakey.pem
```

This command takes the original certificate and modifies the valid dates and creates a new certificate file. This new file is the certificate we will use to sign certificates that windows 2000 machines will accept. We can remove the cacert.pem file and move the newcacert.pem into it's place.

We now want to generate a certificate for a windows 2000 machine, to do this we use the following command:

```
# CA.pl -newreq
```

This generates a new certificate request, which contains the private key. The file generated is *newreq.pem*. During this process you will be asked the same questions you were asked in generating the certificate authority as well as some additional attributes.

We now want to sign this request with our certificate. This is done with the following command:

```
# CA.pl -sign
```

Once we sign the request, a new certificate is generated. The new certificate is called *newcert.pem*.

We now need to combine the *newreq.pem* and *newcert.pem* files into a PKCS12 file format, which the windows 2000 machine can use. To do this we do the following:

```
# openssl pkcs12 -export -in newcert.pem -inkey newreq.pem -certfile
demoCA/cacert.pem -out newpkcs.p12
```

This step is not necessary generating certificates with linux machines, however the *CA.pl -newreq* and *CA.pl -sign* commands must be done for each machine you want to put a certificate on.

Below is a transcript of generating a new certificate request and signing it, then converting it into a .p12 file.

```
[root@vpn /root]# /usr/share/ssl/misc/CA.pl -newreq
Using configuration from /usr/share/ssl/openssl.cnf
Generating a 1024 bit RSA private key
....++++++
..............++++++
writing new private key to 'newreq.pem'
Enter PEM pass phrase:
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
Verify failure
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:NSW
Locality Name (eg, city) []:North Ryde
Organization Name (eg, company) [Internet Widgits Pty Ltd]:GIAC Enterprises
Organizational Unit Name (eg, section) []:cookie crunchers
Common Name (eg, your name or your server's hostname) []:vpn.giac.com
Email Address []:vpnmaster@giac.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:challenge
An optional company name []:GIAC Ent Pty Ltd
Request (and private key) is in newreq.pem
```

```
[root@vpn /root]# /usr/share/ssl/misc/CA.pl -sign
Using configuration from /usr/share/ssl/openssl.cnf
Enter PEM pass phrase:
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'AU'
stateOrProvinceName   :PRINTABLE:'NSW'
localityName          :PRINTABLE:'North Ryde'
organizationName      :PRINTABLE:'GIAC Enterprises'
organizationalUnitName:PRINTABLE:'cookie crunchers'
commonName            :PRINTABLE:'vpn.giac.com'
emailAddress          :IA5STRING:'vpnmaster@giac.com'
Certificate is to be certified until Apr  9 08:28:31 2002 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Signed certificate is in newcert.pem

[root@vpn /root]# openssl pkcs12 -export -in newcert.pem -inkey newreq.pem -certfile
demoCA/cacert.pem -out win2000.p12
Enter PEM pass phrase:
Enter Export Password:
Verifying password - Enter Export Password:
[root@vpn /root]#
```

When running Freeswan, you can check to see if IPSec tunnels are up by running the command *ipsec look*. This will show the list of tunnels, their encryption methods and your routing table. The tunnel information shows the IP address of the destination security gateway, along with details about the incoming and outgoing connections for the tunnel. The routing table entry will show *ipsec0* for the first Ipsec tunnel. You should se an output similar to the following:

```
vpn.giac.com Wed Apr 11 14:34:23 EST 2001
------------------------
172.16.5.0/24 -> 10.0.1.0/24 => tun0x200@10.0.0.1 esp@0x200@10.0.0.1
------------------------
tun0x200@10.0.0.1   IPv4_Encapsulation: dir=out   172.16.5.1 -> 10.0.0.1
esp0x203@172.16.5.1 3DES-MD5-96_Encryption: dir=in iv=0xc2ca5ba42ffbb6   seq=0
esp0x203@10.0.0.1   3DES-MD5-96_Encryption: dir=out iv=0xc2ca5ba42ffbb6   seq=0
Destination     Gateway         Genmask         Flags  MMS Window irtt Iface
10.0.0.1         0.0.0.0        255.255.255.0   U     1500 0       0   eth1
10.0.1.0        10.0.0.1        255.255.255.0   UG    1404 0       0   ipsec0
```

Other tests which you can use are *ping* and *traceroute* to see if you can reach hosts at the other end of the VPN tunnel. For example, the following traceroute can be used:

```
traceroute -i eth1 -f 20 10.0.0.2
```

Here *eth1* is on our private network, and *10.0.0.2* is a machine on the other sides network. By specifying the private or internal network interface, we cause the trace to go over the IPsec tunnel.

Other tests to test the tunnel is working is to try telneting to the remote network, or some other application which can be used to test a service at the remote network on the other side of the VPN.

The IPSec filters

Usually IPSec is running on a machine which is also a gateway, and as such, usually requires filters. Our IPSec or VPN server is running ipchains as it's packet filter. The rules for which are below, these are just a subset of the rules needed.

```
#Turn on IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
#accept incoming IPsec requests
ipchains -A input -i eth0 -p udp -d 130.155.21.2 500 -j ACCEPT
ipchains -A input -i eth0 -p 50 -d 130.155.21.2 -j ACCEPT
ipchains -A input -i eth0 -d 130.155.21.2 389 -j ACCEPT
#allow outgoing IPSec requests
ipchains -A output -i eth0 -p udp -s 130.155.21.2 500 -j ACCEPT
ipchains -A output -i eth0 -p 50 -s 130.155.21.2 -j ACCEPT
ipchains -A output -i eth0 -s 130.155.21.2 389 -j ACCEPT
#Accept internal traffic and masquerade it.
ipchains -A input -i eth1 -s 172.16.0.0/16 -j ACCEPT
ipchains -A output -i eth1 -d 172.16.0.0/16 -j ACCEPT
ipchains -A forward -i eth0 -s 172.16.0.0/16 -j MASQ
```

Testing of the filtering rules is much the same as the border router and external firewall testing.

The Application Gateway

The application Gateway is running ipchains and has proxies for applications. We could be running TIS firewall toolkit or TIS gauntlet. TIS Gauntlet has more proxy services available for it, but TIS can still be used via it's plug-gw for those services without proper proxy applications, or we could do forwarding with masquerading for those services that don't have proxies. We don't actually need to run either TIS application, we could run our own proxies - such as squid for WEB and smap for mail without having TIS installed. We can use port redirection (not the same as forwarding) where when we receive traffic destined for a specific port we redirect it to a different local port with our proxy application listening at the new local port - this allows the proxy to take over and send out on the real port/well known port).

The TIS Firewall Toolkit and Gauntlet are similar in configuration, although the underlying code for the proxies is meant to be significantly different. The documentation for the Firewall Toolkit can be found at http://www.tis.com and http://www.fwtk.org. The documentation for the Firewall Toolkit does not give details on how to configure your firewall performing routing and packet filtering, but assumes that this will be done. The documentation does however describe how to configure the toolkit.

We will first look at the rules for the packet filtering, which will be done by ipchains, we will then examine the configuration of the Firewall Toolkit.

The Filtering Rules

In implementing the packet filtering rules, we should consider that the machine is also hosting the proxy applications the external address space and private address space hosts communicate through. The configuration of the ipchains filters are similar to the external firewall rules and the VPN server. The first thing is to enable IP forwarding:

```
#Turn on IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Now lets look at those packets which we will be expecting. Most of these will be those services that the proxy is acting as a client for.

```
#Rule for HTTP between Internet and client running on Application Gateway
ipchains -A input -i eth0 -p tcp -s any 80 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d any 80 -j ACCEPT
#Rule for HTTPS between Internet and client running on Application Gateway
ipchains -A input -i eth0 -p tcp -s any 443 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d any 443 -j ACCEPT
#Rules for FTP control data between Internet and client running on Application Gateway
ipchains -A input -i eth0 -p tcp -s any 20 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d any 20 -j ACCEPT
#Rules for FTP data packets between Internet and client running on Application Gateway
ipchains -A input -i eth0 -p tcp -s any 21 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d any 21 -j ACCEPT
#Rules for Telnet data between Internet and client running on Application Gateway
ipchains -A input -i eth0 -p tcp -s any 23 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d any 23 -j ACCEPT
#Rules for SSH data between Internet and client running on Application Gateway
ipchains -A input -i eth0 -p tcp -s any 22 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d any 22 -j ACCEPT
#Rules for Finger data between Internet and client running on Application Gateway
ipchains -A input -i eth0 -p tcp -s any 79 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d any 79 -j ACCEPT
#Rules for WHOIS data between Internet and client running on Application Gateway
ipchains -A input -i eth0 -p tcp -s any 43 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d any 43 -j ACCEPT
#Rules for NNTP data between Internet and client running on Application Gateway
ipchains -A input -i eth0 -p tcp -s $NEWS_SERVER 119 -d $APPGW_EXT_IP 1024:65535 -j
ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d $NEWS_SERVER 119 -j
ACCEPT
#Rules for AUTH data between Internet and client running on Application Gateway
ipchains -A input -i eth0 -p tcp -s any 113 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -s any 113 -j ACCEPT
#Rules for AUTH data between Internet and server running on Application Gateway
ipchains -A input -i eth0 -p tcp -s any 1024:65535 -d $APPGW_EXT_IP 113 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 113 -d any 1024:65535 -j ACCEPT
#Rules for DNS data between External DNS and client on Application Gateway
ipchains -A input -i eth0 -p udp -s $DNS_EXT_IP 53 -d $APPGW_EXT_IP 1024:65535 -j ACCEPT
ipchains -A output -i eth0 -p udp -s $APP_GW_UP 1024:65535 -d $DNS_EXT_IP 53 -j ACCEPT
#Rules for DNS data between External DNS and DNS service running on Application Gateway
ipchains -A input -i eth0 -p udp -s $DNS_EXT_IP 53 -d $APPGW_EXT_IP 53 -j ACCEPT
ipchains -A output -i eth0 -p udp -s $APPGW_EXT_IP 53 -d $DNS_EXT_IP 53 -j ACCEPT
#Rules for NTP data between Internet and client on Application Gateway
ipchains -A input -i eth0 -p tcp -s $TIME_SERVER 123 -d $APPGW_EXT_IP 1024:65535 -j
ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 1024:65535 -d $TIME_SERVER 123 -j
ACCEPT
#Rules for NTP data between VPN server and service running on Application Gateway
ipchains -A input -i eth0 -p tcp -s $VPN_EXT_IP 1024:65535 -d $APPGW_EXT_IP 123 -j ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 123 -d $VPN_EXT_IP 1024:65535 -j
ACCEPT
#Rules for NTP data between WEB Servers and service running on Application Gateway
ipchains -A input -i eth0 -p tcp -s $WEB_NET 1024:65535 -d $APPGW_EXT_IP 123 -j ACCEPT
ipchains -A output -i eth0 -p tcp -d $APP_GWP_IP 123 -d $WEB_NET 1024:65535 -j ACCEPT
#Rules for NTP data between Service Network and NTP service running on Application
Gateway
```

```
ipchains -A input -i eth0 -p tcp -s $SERVICE_NET 1024:65535 -d $APPGW_EXT_IP 123 -j
ACCEPT
ipchains -A output -i eth0 -p tcp -s $APPGW_EXT_IP 123 -d $SERVICE_NET 1024:65535 -j
ACCEPT
#Rules for Syslog data from VPN server
ipchains -A input -i eth0 -p udp -s $VPN_EXT_IP 1024:65535 -d $APPGW_EXT_IP 514 -j ACCEPT
ipchains -A output -i eth0 -p udp -s $APPGW_EXT_IP 514 -d $VPN_EXT_IP 1024:65535 -j
ACCEPT
#Rules for Syslog data from Web Servers
ipchains -A input -i eth0 -p udp -s $WEB_NET 1024:65535 -d $APPGW_EXT_IP 514 -j ACCEPT
ipchains -A output -i eth0 -p udp -s $APPGW_EXT_IP 514 -d $WEB_NET 1024:65535 -j ACCEPT
#Rules for Syslog data from Mail and DNS servers
ipchains -A input -i eth0 -p udp -s $SERVICE_NET 1024:65535 -d $APPGW_EXT_IP 514 -j
ACCEPT
ipchains -A output -i eth0 -p udp -s $APPGW_EXT_IP 514 -d $SERVICE_NET 1024:65535 -j
ACCEPT
#Rules for Syslog data from External Firewall
ipchains -A input -i eth0 -p udp -s 130.155.20.1 1024:65535 -d $APPGW_EXT_IP 514 -j
ACCEPT
ipchains -A output -i eth0 -p udp -s $APPGW_EXT_IP 514 -d 130.155.20.1 1024:65535 -j
ACCEPT
#Rules for Web servers to talk to database proxy
ipchains -A input -i eth0 -s $WEB_NET 1024:65535 -d $APPGW_EXT_IP $DB_PORT -j ACCEPT
ipchains -A output -i eth0 -s $APPGW_EXT_IP $DB_PORT -d $WEB_NET 1024:65535 -j ACCEPT

#Accept internal traffic and masquerade it.
ipchains -A input -i eth1 -s 172.16.0.0/16 -j ACCEPT
ipchains -A output -i eth1 -d 172.16.0.0/16 -j ACCEPT
ipchains -A forward -i eth0 -s 172.16.0.0/16 -j MASQ
```

The TIS Firewall Toolkit

The TIS firewall toolkit can be used as an Application Gateway, it does this by providing proxy services for common applications. For example, it provides a Web proxy, an FTP proxy, a telnet proxy and many others. However there are still numerous applications that do not have proxy programs created to plug into the TIS firewall toolkit.
For those services that proxies exist for, we need to modify the */etc/inetd.conf* file to specify what program will be listening on the port. We may also need to modify the */etc/services* if we wish to modify the ports that services will listen on. A program called *'netacl'* is used for tcp services and acts as a wrapper for the services. The *netacl* program determines what service should be used for proxying. A point to note is that proxy services may not always be running on the well known ports. For example, for ftp we would have the following lines in the */etc/inetd.conf* file:

```
#Specify all ftp traffic to goto netacl
ftp    stream tcp    nowait root   /usr/local/etc/netacl      in.ftpd
#ftp-gw is the proxy service running for ftp.
ftp-gw stream tcp    nowait root   /usr/local/etc/ftp-gw      ftp-gw
```

In the above example, ftp may be running at port 20, but ftp-gw could be running at some other port.

For those services for which there may not be a specific proxy application, a generic proxy, called plug-gw can be used for redirecting traffic.

A shared configuration file is used for configuring the proxy services - the */usr/local/etc/netperm-table* file. In this file, depending on where packets are coming

from we can specify actions to take - whether to forward the information to the proxy service, or whether to take some other action. This file is also used to specify the rules for particular proxy services themselves, such as what hosts are permitted to use the service and what error message files to display for different actions - whether a deny text, welcome text, etc. (See TIS Firewall Toolkit - Configuration and Administration). The TIS Firewall Toolkit provides a mechanism to authenticate against users accounts as well as host addresses, this information is stored in a database on the machine and is access via the *'authsrv'* daemon. This can be specified in the *netperm-table* file, however it is not necessary if you only want to use host addresses for authentication. It can be specified by a service by putting *'-auth'* after the *permit-hosts* item.

An example from the *netperm-table* file for the ftp gateway proxy is below, taken from the TIS documentation and modified according to the security architecture for GIAC Enterprises. The same format is used by all the proxy services.

```
#Specify what error messages to be displayed
ftp-gw:      denial-msg    /usr/local/etc/ftp-deny.txt
ftp-gw:      welcome-msg   /usr/local/etc/ftp-welcome.txt
ftp-gw:      help-msg      /usr/local/etc/ftp-help.txt
#Specify which hosts can use the service
ftp-gw:      permit-hosts 172.16.0.0/16 -log { retr stor }
ftp-gw:      timeout 3600

#Specify what action to take depending on host address
#If internal clients - allow full ftp access. (will use ftp-gw)
netacl.in-ftpd      hosts  172.16.0.0/16 -exec /usr/etc/in.ftpd
#Deny all other hosts ftp access
netacl.in-ftpd      hosts  *          -exec /bin/cat /usr/local/etc/noftp.txt
```

The last two lines of the file say that if we receive a connection request for ftp from the hosts on our private internal LAN then we should run the ftp proxy program to allow the user to ftp. For all other hosts, which do not match the internal private IP address, display the error message saying that they cannot ftp.

For the other services, the netperm-table would have similar configurations. For all other services that we will be providing, but proxies do not exist for, then port forwarding could be performed, or they could be redirected to the *plug-gw* proxy which is used for services for which no proxies exist. For example, for NNTP we might have the following entry in the *netperm-table* file:

```
#NetNews
plug-gw:    timeout 3600
plug-gw:     port nntp 172.16.1.1 -plug-to 130.155.20.2 -port nntp
plug-gw:     port nntp 130.15.520.2 -plug-to 172.16.1.1 -port nntp
```

The above basically is doing the same task as what port forwarding would do, if we weren't using the TIS firewall Toolkit.

For more information on configuring the TIS firewall toolkit, have a look at the 'Firewall-HOWTO'.

To test the application gateway, a client from the internal network should try and connect to a service on the Internet. For the filter rules, as with the external firewall nmap can be used to ensure that there are no holes which an intruder can get through.

Assignment 3 - Audit Your Security Architecture

Introduction

The purpose of the security audit is to determine how secure our organisation is, both from criminal intentions and also from accidental occurrences (for example a file system has read/write permission's for everyone and a user accidentally deletes files, eg 'rm -r *.*'). It involves looking at all aspects of security from the physical security - such as who is allowed in, where are computers kept, etc, to the security of files on systems. This includes aspects of data security such as backups (how often they occur, what is backed up, where backups are stored), permission's on file systems as well as network security including what machines are visible to others and what access machines in the network have in relation to one another and the outside world. Not only does a security audit tell us where we are vulnerable, but it can also be used to point out areas which we might be blocking off but shouldn't, for example we might be blocking packets, which result in services not being accessible when they should be. For instance we may be able to get access to all services from inside our network, but have we tried accessing services from outside - such as if we were a business partner or customer, or a remote user.

The area of security auditing we will be limiting ourself to is in the area of accessibility of services that we provide to the Internet community, as well as business partners, suppliers and remote users, in terms of the external firewall access lists and the security of the firewall host itself.

The result of a security audit is to state the vulnerability of our organisation and what the risks of these vulnerablities are, and to specify a recommendation as to what can be done to rectify or reduce the vulnerability (in terms of reducing a vulnerability for example a solution might be to implement an IDS to recognise symptoms rather than wait until the attack has occurred in instances where the vulnerability may not be able to be remedied).

Planning the Assessment

When planning a security audit there are several important requirements that should be considered. One of the first of these, which a lot of technically focused people don't want to really know about is to inform management and get them to sign off to say that is ok for you and your security team to perform the audit which will involve attacking the site which may cause disruption to services for both ourselves and customers - it is important to cover yourself.
Other considerations involve when or what time the security audit of the system should take place. For instance is it going to happen on a weekday or weekend, at what time, and how many times are you planning to do this in order to complete the audit (for example the audit might be in staged processes).
In considering the time frame, the best days and times to perform the audit are when the system is either not in use or has minimal usage. There are several reasons for this, firstly an audit is performed in office hours, we may cause disruption to users if we cause a service or services to be lost, for example if we manage to crash a machine. Secondly, if

the audit is performed outside office hours and peak times, it is much easier to detect our own intrusion, as there is likely to be less traffic and logs will be easier to read, as there will be a less log entries regarding other traffic which might get logged during peak times, and the log entries will be closed together and not spread throughout the file. During non peak times it is also easier to detect traffic, especially if you are using a sniffer, generally because there will be less unwanted traffic to filter. There are arguments for and against having an audit done in off peak times, some may argue that a good reason to audit during weekdays and office hours is that since we may be looking at attack patterns, we may pick up some malicious user activity which we may not have picked up otherwise. Also if we did the audit during peak times, we would be able to see how the firewall handles heavy loads if we were to bombard it with some attacks - we could stress test the network.
Another consideration of the assessment is the cost and effort required to complete the audit. It must be shown to management that it is worthwhile (a point which one would think is self evident, but in many cases in organisations must be proven to management). In considering costs you must weight up the risks - should the cost of an audit and analysis outweigh the costs and risks associated with an attacker successfully penetrating your security and gaining access to valuable or confidential data.
In general the costs of the security audit will ultimately involve the cost of man-hours, in both implementing the audit and analysing the data. Other costs may be the cost in time repairing a system is if is brought down, which is the cost os service availability - for example, how long are we prepared to have our web site down which is our core business. The cost of the audit will also involve the cost of hiring or purchasing equipment if necessary in order to perform the audit - for instance the cost of purchasing a specialised, or commercial network sniffer.

The risks involved in the audit include such things as disruptions to services, which we have mentioned earlier which may also effect customers. These could be caused by denial of services or a degradation in network performance which may occur by trying to perform an attack. Other risks associated with the audit, is that we may not pick up some of the vulnerabilities of our system, this could be related to the tools we are using which may not pick up a vulnerability, or could be related to the tests we perform, maybe we didn't check something.

In planning the technical details associated with the security audit, we will look at the security of the firewall itself, and the security of the filtering rules it provides.
In looking at the firewall itself, we want to ensure that there are no unnecessary services running on it, for example we don't wan xdm services running which allow remote X sessions, similarly we don't want to be running as a NIS client and we don't want to allow automounting of remote files systems either or be sharing filessystems. We also want all the file permission's set correctly, for example the files in the /etc/ directory. For those services that are necessary we don't want them running are root if they don't need to be.

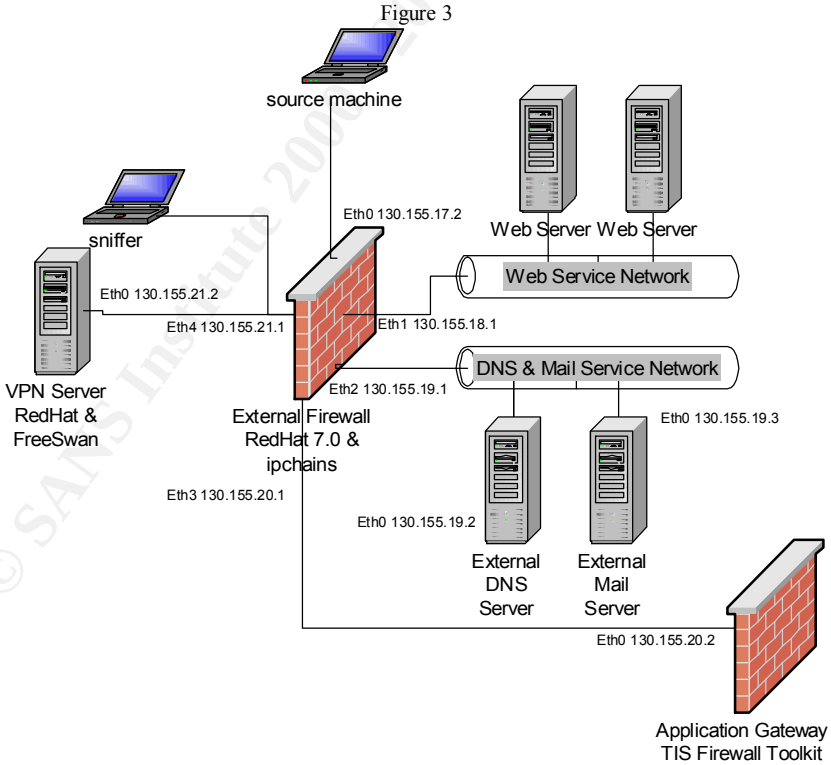
In testing the filtering rules, the test on the firewall should not only test that the required packets get blocked coming in, but also the appropriate packets are allowed to get through to the appropriate destination (we don't want to block packets which should be

allowed, causing services not to work). We also need to test the firewall for traffic going out of our network, and make sure we are filtering out invalid packets, such as the private IP address range source address or destination address, so that our site isn't used to break into other sites.

To implement the audit we will be placing the intruder/source machine on one side of the firewall, and have another machine running a sniffer on the destination network. We have a sniffer to see what gets through the firewall, which may not be being logged. This will also tell us whether the firewall is logging everything it should, or whether what is being logged is correct. The placement of the source and sniffer machines will be done for each interface on the firewall and each destination network – see Figure 3. For example to test external traffic to the VPN, the source machine will be on the external interface 'eth0' of the firewall, and the sniffer will be on the same network as the VPN server is on. To test the traffic from the VPN server to the DNS server, we will have the source machine on the VPN network interface side of the firewall and the sniffer on the external services network that the DNS is on.

We not only want to test the outside world coming in, but also the inside network (the publicly visible services) behind the firewall - testing the inside network talking to other parts of the inside network. For example we don't want DNS zone transfers from the VPN server to the DNS allowed, only DNS queries. Similarly we don't want HTTP access to be available to the VPN from the service networks, etc.

Figure 3



To test the actual packet filters for general services, we will actually have some of the service clients run on our source machines. For example we will run telnet, ftp, dns, finger, etc to see if we can connect. These will test for valid packet types. We will also use the firewall logs and network sniffer to see if tests have been successful or partially

successful. For other packet testing, we will use tools for crafting packets and see what happens. For some tests IP spoofing will need to be done, such as in the cases where the filters define explicit hosts.

Implementing The Assessment

To check the firewalls own security and integrity we can run port scanners, such as nmap to examine what services are running and what ports are open on the machine, and identify those which should not be there. A simple process listing using 'ps' will also help show or identify some services.
To show services running use *'ps -ef'*. This will give us a list of processes running on the machine.

To look at the ports we can run nmap locally on the firewall machine or run it from the source machine. Either way the name command is similar.

nmap 127.0.0.1 on the firewall, or *nmap <ipaddress of firewall>* on the source machine. There is a *'-v'* option which can be used to produce a more verbose output. The output of nmap will produce a list of the tcp ports on the machine. To list udp ports, there is a *-sU* option. The list will either say open or filtered (filtered is when the ports are going through the firewall).

```
[root@firewall /root]# nmap 127.0.0.1

Starting nmap V. 2.54BETA20 ( www.insecure.org/nmap/ )
Interesting ports on firewall.giac.com (127.0.0.1):
(The 1530 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     filtered    ssh
23/tcp     filtered    telnet
25/tcp     filtered    smtp
79/tcp     filtered    finger
111/tcp    filtered    sunrpc
113/tcp    filtered    auth
513/tcp    filtered    login
514/tcp    filtered    shell
515/tcp    filtered    printer
587/tcp    open        submission
1024/tcp   open        kdm
6000/tcp   filtered    X11


Nmap run completed -- 1 IP address (1 host up) scanned in 3 seconds
[root@firewall /root]#
```

The output from the -sU option of nmap results in:

```
[root@firewall /root]# nmap -sU 127.0.0.1

Starting nmap V. 2.54BETA20 ( www.insecure.org/nmap/ )
Interesting ports on firewall.giac.com (127.0.0.1):
(The 1449 ports scanned but not shown below are in state: closed)
Port       State       Service
```

```
111/udp    filtered    sunrpc
789/udp    open        unknown
1024/udp   open        unknown
1025/udp   open        blackjack


Nmap run completed -- 1 IP address (1 host up) scanned in 6 seconds
[root@firewall /root]#
```

In the output of nmap we don't really want to see any *'open'* ports. So from the output we see there is a problem. Also we can see that xdm is running on the machine, which we don't want to be there. (I was running xdm to capture output of commands). There is also smtp, and a number of other services we don't want - we need to turn these services off. There is also an option for nmap to return who is running the service, the *'-I'* option uses reverse ident to find this out.

For a non Unix firewall such as windows NT, you can user *TLIST* from the windows NT resource kit or task manager to list the services and to see what software is running. You can also look at the registry entries for services using the registry editor. The nmap command should be done on the source machine in this case.

For the filtering rules we need to check the traffic on all ethernet interfaces in both directions. We will require several setups with the source and sniffer machines, which results in the following test configurations:

Test 1	outside network to VPN network
Test 2	outside network to Web network
Test 3	outside network to service network
Test 4	outside network to Application Gateway network
Test 5	VPN network to Web network
Test 6	VPN network to service network
Test 7	VPN network to Application Gateway network
Test 8	VPN network to outside network
Test 9	Web network to VPN network
Test 10	Web network to service network
Test 11	Web network to Application Gateway network
Test 12	Web network to outside network
Test 13	Service network to VPN network
Test 14	Service network to Web network
Test 15	Service network to Application Gateway network
Test 16	Service network to outside network
Test 17	Application Gateway network to VPN network
Test 18	Application Gateway network to Web network
Test 19	Application Gateway network to service network
Test 20	Application Gateway network to outside network

The basic tests will be the same for each configuration we are testing from. So I will only include one group of testing to stop this from being a long winded repeating section. We

will test for packets coming from the outside network to the inside network. We will use the SANS top ten as the basic guide for testing normal or valid packets formats.

To test for rules where data must come from a specific address we may need to spoof our IP address. The first item in the SANS top ten is to block private IP address ranges and is one such case. To test the private IP address we need a tool which will allow us to spoof our source address, Nmap or Hping2 among many others will allow this. In this instance I am choosing nmap. For the sniffing tool, tcpdump or a commercial product like Network General sniffer could be used. I will be using tcpdump for the tests here.

For the brief syntax of nmap, here is the output of just running nmap without any arguments:

```
Nmap V. 2.54BETA20 Usage: nmap [Scan Type(s)] [Options] <host or net list>
Some Common Scan Types ('*' options require root privileges)
  -sT TCP connect() port scan (default)
* -sS TCP SYN stealth port scan (best all-around TCP scan)
* -sU UDP port scan
  -sP ping scan (Find any reachable machines)
* -sF,-sX,-sN Stealth FIN, Xmas, or Null scan (experts only)
  -sR/-I RPC/Identd scan (use with other scan types)
Some Common Options (none are required, most can be combined):
* -O Use TCP/IP fingerprinting to guess remote operating system
  -p <range> ports to scan.  Example range: '1-1024,1080,6666,31337'
  -F Only scans ports listed in nmap-services
  -v Verbose. Its use is recommended.  Use twice for greater effect.
  -P0 Don't ping hosts (needed to scan www.microsoft.com and others)
* -Ddecoy_host1,decoy2[,...] Hide scan using many decoys
  -T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane> General timing policy
  -n/-R Never do DNS resolution/Always resolve [default: sometimes resolve]
  -oN/-oX/-oG <logfile> Output normal/XML/grepable scan logs to <logfile>
  -iL <inputfile> Get targets from file; Use '-' for stdin
* -S <your_IP>/-e <devicename> Specify source address or network interface
  --interactive Go into interactive mode (then press h for help)
Example: nmap -v -sS -O www.my.com 192.168.0.0/16 '192.88-90.*.*'
SEE THE MAN PAGE FOR MANY MORE OPTIONS, DESCRIPTIONS, AND EXAMPLES
```

To perform the test of private IP address ranges being blocked, we will use the following command:

```
nmap -S 10.0.0.1 130.155.18.2
```

Here we are specifying to spoof a private IP address 10.0.0.1 that we are sending as the source address. We are sending it to a machine in our service network. Without specifying any scan type, nmap uses tcp connections, so this command is trying to scan tcp ports on host 130.155.18.2. Alternatively we could have specified the broadcast address for this network, as there may be specific rules in our firewall for this host which may block this, but not for other hosts. So in actual fact we should run this test for every host in the network that we are looking at and do so for each private IP address range. This test should be done also for each network.

For the other private IP address ranges, we would do the following:

```
nmap -S 198.168.0.1 130.155.18.2
nmap -S 172.16.0.1 130.155.18.2
```

For the localhost address and for broadcast and multicast addresses and for a host with no source address the following nmap commands would be used respectively:

```
nmap -S 127.0.0.1 130.155.18.2
nmap -S 255.0.0.0 130.155.18.2
nmap -S 244.0.0.0 130.155.18.2
nmap -S 0.0.0.0 130.155.18.2
```

To test for packets with our network addresses or the address of the external firewall interface we would use the following tests respectively:

```
nmap -S 130.155.18.2 130.155.19.2
nmap -S 130.155.19.1 130.155.18.2
nmap -S 130.155.20.1 130.155.18.2
nmap -S 130.155.21.1 130.155.18.2
nmap -S 130.155.17.2 130.155.18.2
```
- tests for external IP address of firewall

For the results of these tests we should not see any traffic using tcpdump to the host 130.155.18.2. Our logs should show us that packets have been denied. Also nmap will tell us that it has failed in trying to scan ports on 130.155.18.2 and 130.155.19.2. Here are the results of the log for trying to spoof the source address as 10.0.0.1:

```
Apr 13 14:35:21 firewall kernel: Packet log: input DENY eth0 PROTO=1 10.0.0.1:8
130.155.18.2:0 L=28 S=0x00 I=34479 F=0x0000 T=48 (#7)
Apr 13 14:35:21 firewall kernel: Packet log: input DENY eth0 PROTO=6
10.0.0.1:50061 127.0.0.1:80 L=40 S=0x00 I=6960 F=0x0000 T=42 (#7)
Apr 13 14:35:27 firewall kernel: Packet log: input DENY eth0 PROTO=1 10.0.0.1:8
130.155.18.2:0 L=28 S=0x00 I=10884 F=0x0000 T=48 (#7)
Apr 13 14:35:27 firewall kernel: Packet log: input DENY eth0 PROTO=6
10.0.0.1:50062 127.0.0.1:80 L=40 S=0x00 I=22508 F=0x0000 T=42 (#7)
Apr 13 14:35:33 firewall kernel: Packet log: input DENY eth0 PROTO=1 10.0.0.1:8
130.155.18.2:0 L=28 S=0x00 I=49332 F=0x0000 T=48 (#7)
Apr 13 14:35:33 firewall kernel: Packet log: input DENY eth0 PROTO=6
10.0.0.1:50063 127.0.0.1:80 L=40 S=0x00 I=7717 F=0x0000 T=42 (#7)
Apr 13 14:35:39 firewall kernel: Packet log: input DENY eth0 PROTO=1 10.0.0.1:8
130.155.18.2:0 L=28 S=0x00 I=19237 F=0x0000 T=48 (#7)
Apr 13 14:35:39 firewall kernel: Packet log: input DENY eth0 PROTO=6
10.0.0.1:50064 127.0.0.1:80 L=40 S=0x00 I=11547 F=0x0000 T=42 (#7)
Apr 13 14:35:45 firewall kernel: Packet log: input DENY eth0 PROTO=1 10.0.0.1:8
130.155.18.2:0 L=28 S=0x00 I=2591 F=0x0000 T=48 (#7)
```

For the second item in the SANS top ten, we are testing for telnet, ssh, ftp, NetBIOS and the rlogin services.

To test telnet we will use telnet, for ssh and ftp we will use ssh and ftp. While these steps seems pretty basic it is a useful one to use, if you have no other tool - what better tool to use to test than the application itself to test whether you get through the firewall or are blocked. We can also use nmap to test these items as well.

To test telnet, try telneting to a host on each network.

```
telnet 130.155.18.2
telnet 130.155.19.2
telnet 130.155.20.2
telnet 130.155.21.2
```

We could also test using nmap via the following commands:

```
nmap -p 23 130.155.18.2
nmap -p 23 130.155.19.2
nmap -p 23 130.155.20.2
nmap -p 23 130.155.21.2
```

Here we are telling nmap to use port 23 (telnet) followed by the host to attempt to connect to. To test all for all hosts on each of the networks, we can specify the network address itself:

```
nmap -p 23 130.155.18.0
nmap -p 23 130.155.19.0
nmap -p 23 130.155.20.0
nmap -p 23 130.155.21.0
```

To test for ftp, we can use the following ftp commands:

```
ftp 130.155.18.2
ftp 130.155.19.2
ftp 130.155.20.2
ftp 130.155.21.2
```

Using the nmap equivalent but do scan the network, we would use the following nmap commands:

```
nmap -p 21 130.155.18.0
nmap -p 21 130.155.19.0
nmap -p 21 130.155.20.0
nmap -p 21 130.155.21.0
```

A useful aspect of nmap is that it allows us to specify networks and port numbers. When specifying port numbers, it allows us to put multiple ports, ranges and multiple ranges as well. Therefore we could have written the following for checking telnet and ftp when trying to scan for hosts on the 130.155.18.0 network.

```
nmap -p 21,23 130.155.18.0
```

If we had have used *'-p 21-23*' we would have been able to scan for ftp, ssh and telnet in the one go:

```
nmap -p 21-23 130.155.18.0
```

With this information about nmap, we can actually test for all of the elements in the second item of the SANS top ten. We would use the following commands to test these elements. By default nmap uses tcp so we do not need to specify it.

```
nmap -p 21-23,139,512-514 130.155.18.0
nmap -p 21-23,139,512-514 130.155.19.0
nmap -p 21-23,139,512-514 130.155.20.0
nmap -p 21-23,139,512-514 130.155.21.0
```

As with the private ip addresses and other checks in the first item of the top ten, the logs should show that an attempt was made, but was denied.

The test with telnet will result with telnet timing out. For example:

```
#telnet 130.155.18.2
Trying 130.155.18.2...
```

The logs from ipchains show the following output:

```
Apr 13 14:06:10 firewall login[1167]: FAILED LOGIN SESSION FROM 130.155.17.3
FOR , Error in service module
Apr 13 14:06:27 firewall kernel: Packet log: input DENY eth0 PROTO=6
130.155.17.3:1032 130.155.18.2:23 L=60 S=0x00 I=136 F=0x4000 T=64 SYN (#6)
Apr 13 14:06:30 firewall kernel: Packet log: input DENY eth0 PROTO=6
130.155.17.3:1032 130.155.18.2:23 L=60 S=0x00 I=137 F=0x4000 T=64 SYN (#6)
Apr 13 14:06:36 firewall kernel: Packet log: input DENY eth0 PROTO=6
130.155.17.3:1032 130.155.18.2:23 L=60 S=0x00 I=138 F=0x4000 T=64 SYN (#6)
Apr 13 14:06:48 firewall kernel: Packet log: input DENY eth0 PROTO=6
130.155.17.3:1032 130.155.18.2:23 L=60 S=0x00 I=139 F=0x4000 T=64 SYN (#6)
Apr 13 14:07:12 firewall kernel: Packet log: input DENY eth0 PROTO=6
130.155.17.3:1032 130.155.18.2:23 L=60 S=0x00 I=140 F=0x4000 T=64 SYN (#6)
Apr 13 14:08:00 firewall kernel: Packet log: input DENY eth0 PROTO=6
130.155.17.3:1032 130.155.18.2:23 L=60 S=0x00 I=141 F=0x4000 T=64 SYN (#6)
```

The result of the nmap scan on port 23 shows the following log entries.

```
Apr 13 14:11:04 firewall kernel: Packet log: input DENY eth4 PROTO=6
130.155.17.3:1033 130.155.18.2:23 L=60 S=0x00 I=144 F=0x4000 T=64 SYN (#6)
Apr 13 14:11:05 firewall kernel: Packet log: input DENY eth4 PROTO=6
130.155.17.3:1034 130.155.18.2:23 L=60 S=0x00 I=145 F=0x4000 T=64 SYN (#6)
Apr 13 14:11:05 firewall kernel: Packet log: input DENY eth4 PROTO=6
130.155.17.3:1035 130.155.18.2:23 L=60 S=0x00 I=146 F=0x4000 T=64 SYN (#6)
Apr 13 14:11:05 firewall kernel: Packet log: input DENY eth4 PROTO=6
130.155.17.3:1036 130.155.18.2:23 L=60 S=0x00 I=147 F=0x4000 T=64 SYN (#6)
Apr 13 14:11:06 firewall kernel: Packet log: input DENY eth4 PROTO=6
130.155.17.3:1037 130.155.18.2:23 L=60 S=0x00 I=148 F=0x4000 T=64 SYN (#6)
Apr 13 14:11:06 firewall kernel: Packet log: input DENY eth4 PROTO=6
130.155.17.3:1038 130.155.18.2:23 L=60 S=0x00 I=149 F=0x4000 T=64 SYN (#6)
```

Note that login session message is the only difference between telnet and nmap when trying to connect to the telnet port, and the source port of the client.

For the third SANS top ten item RPC and NFS and lockd are tested. These both tcp and udp ports. To test these we will use the following nmap commands:

```
nmap -sT -p 111,2049,4045 130.155.18.0
nmap -sT -p 111,2049,4045 130.155.19.0
nmap -sT -p 111,2049,4045 130.155.20.0
nmap -sT -p 111,2049,4045 130.155.21.0
```

The above will test for RPC portmapper (tcp 111), NFS (tcp 2049) and lockd (tcp 4045). The -sT is specifying to use tcp connection (which we could have left out as nmap tests this by default).

```
nmap -sU -p 111,2049,4045 130.155.18.0
nmap -sU -p 111,2049,4045 130.155.19.0
nmap -sU -p 111,2049,4045 130.155.20.0
nmap -sU -p 111,2049,4045 130.155.21.0
```

The above will test for RPC rpcbind (udp 111), NFS (udp 2049) and lockd (udp 4045). We are specifying to use udp with the -sU option. The result of these commands will show deny actions similar to the logs for the private IP and telnet tests

The fourth item in the SANS top ten is NetBIOS in windows NT. There are two options for testing this. The first is to use nmap, the second is to have the source machine be a windows machine or a linux machine with samba installed on it, and attempt to browse the windows network or map a network drive. If you use the later and are on a windows machine you will get a message saying that there is a TCP/IP problem. The nmap commands to test these ports are as follows:

```
nmap -sT -p 135,445 130.155.18.0
nmap -sT -p 135,445 130.155.19.0
nmap -sT -p 135,445 130.155.20.0
nmap -sT -p 135,445 130.155.21.0
nmap -sU -p 135,137-139,445 130.155.18.0
nmap -sU -p 135,137-139,445 130.155.19.0
nmap -sU -p 135,137-139,445 130.155.20.0
nmap -sU -p 135,137-139,445 130.155.21.0
```

The fifth SANS top ten item is testing for X windows which runs on port 6000 to port 6255 over tcp. The nmap commands to test this are:

```
nmap -p 6000-6255 130.155.18.0
nmap -p 6000-6255 130.155.19.0
nmap -p 6000-6255 130.155.20.0
nmap -p 6000-6255 130.155.21.0
```

The sixth SANS top ten is testing DNS and LDAP. Unlike the other items so far described which are blocked for traffic from the outside world, DNS and LDAP are open for some hosts on the inside network. (please not that when referring to inside I am talking about those hosts which are visible to the public but behind the firewall, I am not discussing the hosts within our private address space). We would expect to see denied messages for DNS queries to all networks apart from the host 130.155.19.2, which is the external DNS server for our network. DNS queries use udp to port 53. We would however expect to see zone transfer queries to 130.155.19.2 to be denied, unless of course we had a specific host we allowed externally to do zone transfers with (zone transfers are tcp port 53), in which case we could test by spoofing the IP address of the external machine. We would also expect to see LDAP denied for all hosts apart from 130.155.21.2, which is our VPN server. This is left open for other IPSec machines to authenticate with it. To test DNS zone transfers we could have our source machine

running a DNS server and then try to do a zone transfer. nmap will be used to do the filtering tests.

The nmap commands we will use are as follows.

```
nmap -sU -p 53 130.155.18.0          - scan networks for port 53 using udp
nmap -sU -p 53 130.155.19.0
nmap -sU -p 53 130.155.20.0
nmap -sU -p 53 130.155.21.0
```

Here we are scanning for DNS queries. nmap should respond by saying that there is one host in network 130.155.19.0, which is 130.155.19.2, which is open to DNS queries.

```
nmap -sT -p 53 130.155.18.0          - scan networks for port 53 using tcp
nmap -sT -p 53 130.155.19.0
nmap -sT -p 53 130.155.20.0
nmap -sT -p 53 130.155.21.0
```

Here we are scanning for DNS zone transfers. nmap will report back that there are no open ports.

```
nmap -sU -p 53 130.155.19.2          - scan DNS host for DNS queries
nmap -sT -p 53 130.155.19.2          - scan DNS host for zone transfer
```

In the above two nmap commands the first succeeds but the second fails.
Another test for zone transfers is the following, using the *host* command.

```
host -l giac.com
```

This will download the complete zone data for giac.com by trying to do a zone transfer. However this will be blocked by the firewall unless we spoof the address on an external DNS server we trust. The *'nslookup'* command can also be used to test zone transfers.

To test for LDAP, we will use the following nmap commands. Like the DNS tests, we expect that LDAP will be denied for all hosts but the VPN server 130.155.21.2.

```
nmap -sT -p 389 130.155.18.0
nmap -sT -p 389 130.155.19.0
nmap -sT -p 389 130.155.20.0
nmap -sT -p 389 130.155.21.0
nmap -sU -p 389 130.155.18.0
nmap -sU -p 389 130.155.19.0
nmap -sU -p 389 130.155.20.0
nmap -sU -p 389 130.155.21.0
```

As expected nmap will return a successful on the 130.155.21.0 network. But will fail on all others.

The seventh SANS top ten item is mail, port 25 for SMTP, and ports 109 and 110 for POP, and port 143 for IMAP. We will expect that POP and IMAP will fail for all tests,

but SMTP will work to host 130.155.19.3 (the external Mail server). Each of these services only use tcp.

The nmap commands for testing these are as follows:

```
nmap -p 25,109-110,143 130.155.18.0
nmap -p 25,109-110,143 130.155.19.0
nmap -p 25,109-110,143 130.155.20.0
nmap -p 25,109-110,143 130.155.21.0
```

One thing to note is that we should probably not be complacent with just looking at tcp for services, we should also look at udp. We may have specified in the firewall rules to open the port, but not specify tcp or udp only, thus both might be open.

The eighth SANS top ten are the web services. Here we want to block HTTP, Secure HTTP (SSL) as well as high end ports used to provide web services (ports 8000,8080 and 8888). However we want to let HTTP and SSL to our web service network (130.155.18.0). The following nmap commands will test for these.

```
nmap -p 80,443,8000,8080,8888 130.155.18.0
nmap -p 80,443,8000,8080,8888 130.155.19.0
nmap -p 80,443,8000,8080,8888 130.155.20.0
nmap -p 80,443,8000,8080,8888 130.155.21.0
```

In the above, nmap will report back saying that there are several services for ports 80 and 443 which are active for the 130.155.18.0 network, but will report a failure for the ports 8000,8080 and 8888. All the other scans to other networks will report failures.

The ninth item in the SANS top ten is testing the small services, which are all the ports below port 20 running udp and tcp. To test this, we will using the following nmap commands:

```
nmap -sT -p 1-20 130.155.18.0      - these first four test tcp ports
nmap -sT -p 1-20 130.155.19.0
nmap -sT -p 1-20 130.155.20.0
nmap -sT -p 1-20 130.155.21.0
nmap -sU -p 1-20 130.155.18.0      - the following four test udp ports
nmap -sU -p 1-20 130.155.19.0
nmap -sU -p 1-20 130.155.20.0
nmap -sU -p 1-20 130.155.21.0
```

All of these will tests will fail and will be seen in the logs.

The tenth item on the SANS top ten is the miscellaneous traffic such as TFTP (69/udp), finger (79/tcp), NNTP (119/tcp), NTP (123/tcp), LPD (515/tcp), syslog (514/tcp), SNMP (ports 161 & 162 over tcp and udp), BGP (179/tcp) and socks (1080/tcp).
We expect from the scans and logs that all services will be denied except NTP and NNTP to the machine which is the Application Gateway. However to test these we will need to spoof the NTP server and NNTP server which the Application Gateway communicates with. If we do not specify a specific source address, then all of the scans will fail.

Here are the nmap commands to test this tenth item on the SANS top ten.

```
nmap -sU -p 69,161-162 130.155.18.0      - test udp ports
nmap -sU -p 69,161-162 130.155.19.0
nmap -sU -p 69,161-162 130.155.20.0
nmap -sU -p 69,161-162 130.155.21.0

nmap -sT -p 79,119,123,514-515,161-162,179,1080 130.155.18.0      - test tcp ports
nmap -sT -p 79,119,123,514-515,161-162,179,1080 130.155.19.0
nmap -sT -p 79,119,123,514-515,161-162,179,1080 130.155.20.0
nmap -sT -p 79,119,123,514-515,161-162,179,1080 130.155.21.0
```

These tests will fail and will be written to the logs. To test if NTP and NNTP is allowed we need to spoof the IP address of the NTP server and NNTP server that we talk to with the Application Gateway.

```
nmap -sT -p 123 130.155.20.2      - test NTP
nmap -sT -p 119 130.155.20.2      - test NNTP
```

With these tests nmap will report successfully scanning the host. Testing the above will also work for the inside networks as well as for NTP as the Application Gateway is the time server for the network.

The final test from the SANS top ten is managing ICMP traffic. Incoming echo requests should be blocked from the outside world and outgoing icmp echo replies, time exceeded and destination unreachable should be blocked.

To test ICMP echo requests, we can use the ping command, or nmap with the -sP option which tells nmap to use ICMP echo - like ping. To ping a network, we can use the -b (broadcast option) of ping. Note that within our network we probably want icmp messages available so in testing one inside network to another, these tests will show the ping gets through.

```
ping -b 130.155.18.0
ping -b 130.155.19.0
ping -b 130.155.20.0
ping -b 130.155.21.0
```

Alternatively the using the nmap command:

```
nmap -sP 130.155.18.0
nmap -sP 130.155.19.0
nmap -sP 130.155.20.0
nmap -sP 130.155.21.0
```

The next step is in testing that our firewall is blocking outgoing echo replies, time exceeded, and destination unreachable messages. Note this test needs to be done from the inside network as we are testing packets going out or alternatively we can test it by letting icmp requests in from outside, but stopping the responses - this would mean modifying

the access lists, but would also mean that the network would be vulnerable for a certain period of time.

What about scans which are done by crafting packets. One nmap options allows someone to send ACK packets. This can be used to tell whether the firewall is a statefull or static packet filter. This can be done by specifying the -sA option which tells nmap to use ACK packets. We will use the following nmap command to test our firewall:

```
nmap -sA 130.155.18.2
```

Our firewall does not block all of the ACK packets, and so we can tell that our firewall is only a static packet filter, which is what we expected.

Analysing the Assessment

The results of the analysis shows that our firewall has several problems. Firstly the integrity testing of the actual firewall itself shows that there are several services running on the machine that should not be. These services should be shutdown.
In terms of the filtering tests of the firewall, the results show that the firewall is letting packets through which we don't really want. The tests which showed this were the ones relating to sending crafted packets. Our firewall allows ACK packets through without. This shows that our firewall does static packet filtering, not statefull - this was expected as I am using ipchains, but my original intention was to use iptables. This would have denied several of the crafted packets and the tcp ACK scan of the network.

Conclusion

This audit was just looking at the primary firewall, so did not cover such items as the vulnerability of the other hosts or other firewalls being used, such as the Web, Mail and DNS servers, or the VPN server and Application Gateway and internal firewall respectively. For a full audit I would cover such items. Such an audit would probably find the VPN setup with both remote users and business partners and suppliers on the one network a security risk, in which case I would provide two VPN servers, one for business partners and suppliers, and another for the remote users. The internal firewall could then treat dialup and remote users the same, and allow access to the customer database, which is probably closer to meeting business requirements. In this way the business partners and suppliers VPN network could be totally blocked from the internal network and therefore the customer DB. See Figure 4.

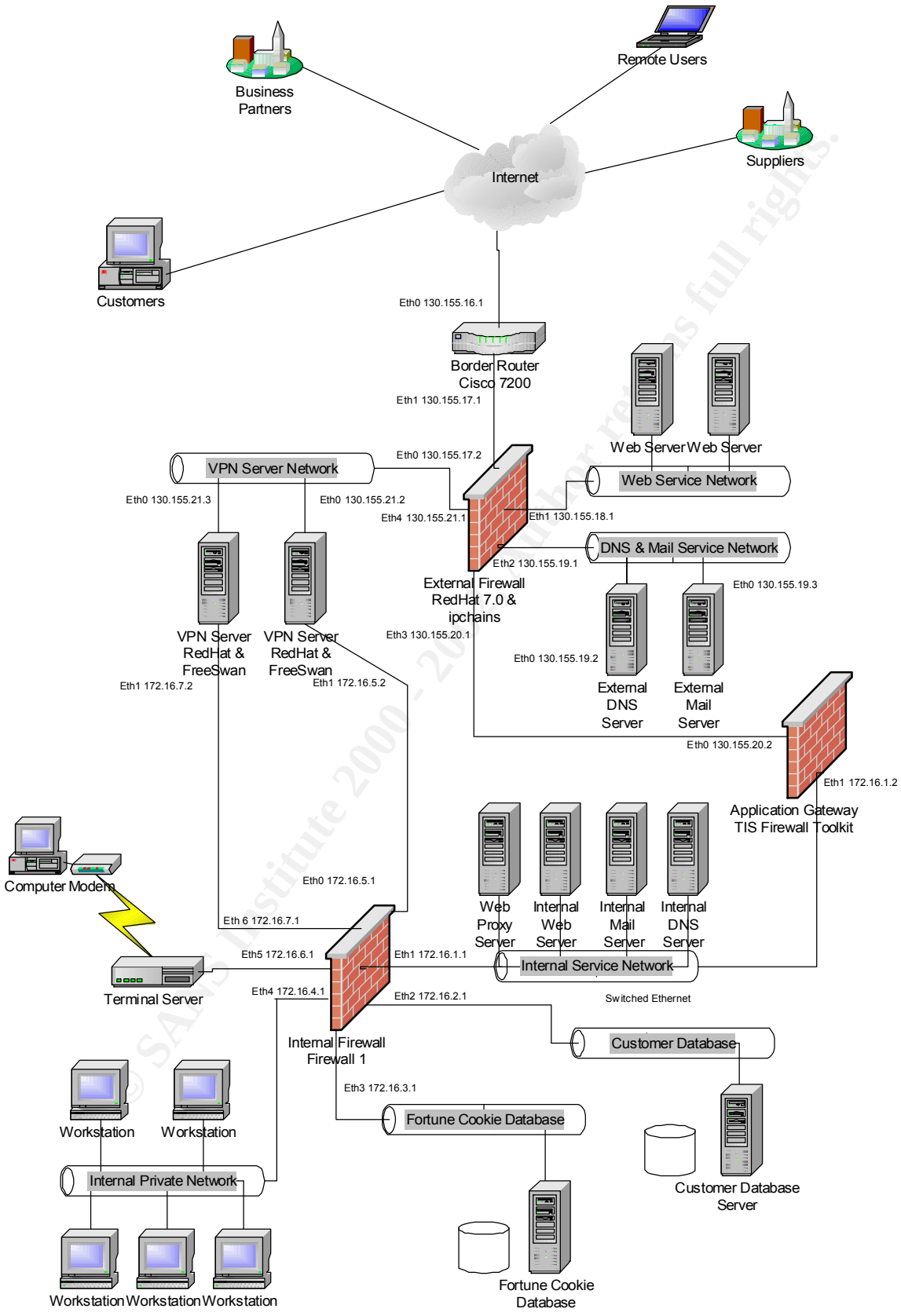
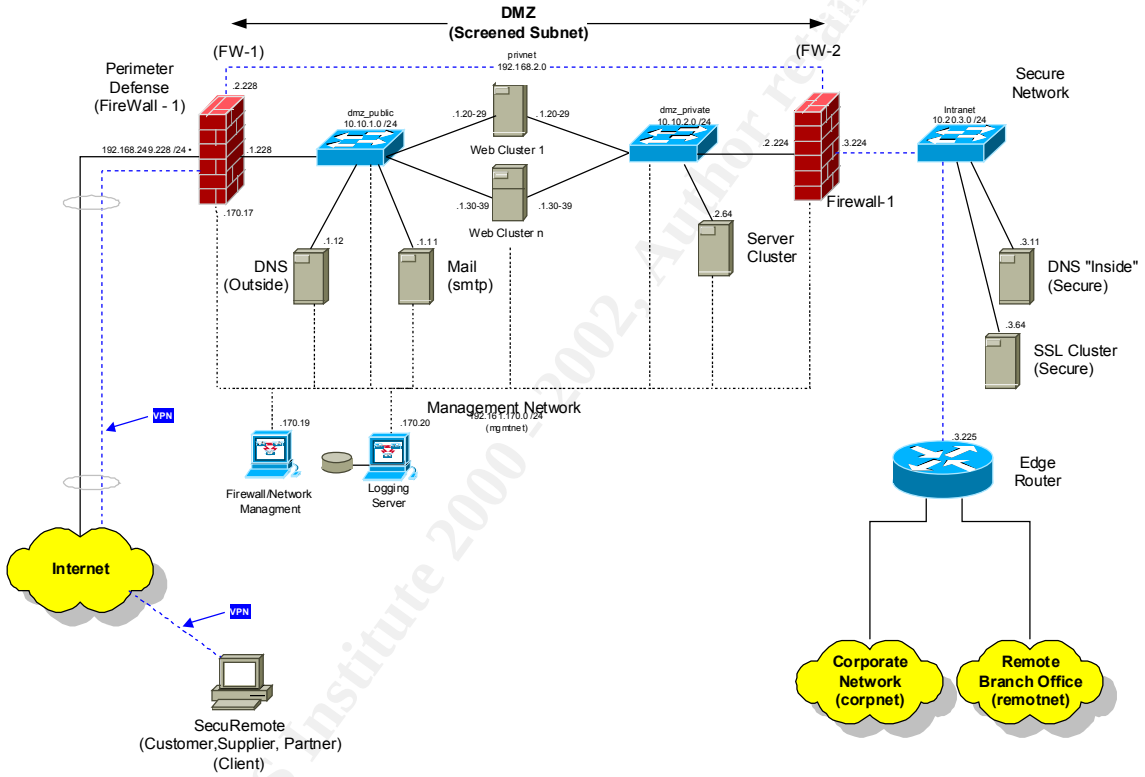


Figure 4

Assignment 4 - Design Under Fire

Introduction

For the design under fire, I have chosen Keith Wilcox's design, which can be found at http://www.sans.org/y2k/practical/Keith_Wilcox.doc. Both his primary firewall and internal firewall are running Checkpoint Firewall-1 with Syn defender enabled to block SYN flood attacks. His external firewall is running on a Sun Netra 125 machine and the internal firewall is running Windows NT. The internal firewall is also being used as the VPN server with Checkpoint VPN-1. The diagram of his network is shown below.



The Firewall Attack

In planning the firewall attack, searches for Checkpoint Firewall-1 reveal several recent vulnerabilities. These vulnerabilities can be found at such sites as http://xforce.iss.net as well as Checkpoints own web site http://www.checkpoint.com/tech_support. These vulnerabilities include the following:

- One-way Connection Enforcement Bypass
- Improper stderr Handling for RSH/REXEC
- FTP Connection Enforcement Bypass
- Retransmission of Encapsulated Packets
- FWA1 Authentication Mechanism Hole
- OPSEC Authentication Spoof
- S/Key Password Authentication Brute Force Vulnerability
- GetKey Buffer Overflow.
- SMTP Security Server DoS
- Fast Mode Vulnerability
- IP Fragment DoS Vulnerability
- ACK DoS Attack
- Passive FTP Vulnerability

A description of each of these attacks follows.

In the One-way Connection Enforcement Bypass an authorised connection can be made through the firewall which would normally be blocked. By using TCP fragmented connection requests or by closing and reopening one-way TCP connections in conjunction with multi-connection protocols the directional check of the packets can be bypassed.

In the improper handling of the stderr for RSH/REXEC an attacker can establish a connection with an RSH or REXEC client. This is done by sending malformed or specially formatted sdterr connection requests to a client. The attacking machine id made to appear to be an RSH or REXEC server. This vulnerability is also found in VPN-1 as well as Firewall-1.

In the FTP Connection Enforcement Bypass an attacker can establish connections to machines which are accessible from an internal ftp server. An attacker tries to connect to the ftp server in a passive mode, and advertises that he has a small maximum segment size. As a result the ftp server replies are split. The attacker can then use the ftp-bounce attack to bounce connections from the ftp server to the machines which are accessible to the ftp server.

In the retransmission of encapsulated packets an attacker can pass packets through the firewall without being an FWZ client, bypassing the anti-spoofing checks if the retransmission option is enabled. By sending specially encapsulated FWZ packets, an attacker can effectively spoof through the firewall.

In the FWA1 Authentication Mechanism Hole an attacker can compromise the authentication mechanism. In FWA1 authentication the server sends a random number and a hash of a random number plus a shared secret key to the client. The client is required to send a random number and has of the random number sent by the server Xor's with the clients random number plus the secret key. By sending a zero as the client's random number, an Xor of the server's number and zero will equal the server's number and result in the same hash value. This attack can be potentially used to cause a denial of service by flooding the authentication mechanism with successful authentications.

In the OPSEC authentication spoof, an attacker can authenticate to any OPSEC channel and gain full access to the services allowed through the channel. In the OPSEC authentication a client requests a connection, the server sends a random number, hash and key to the client, the client is expected to verify the key, and then send a different random number and their own hash of the key which the server then verifies. However the server does not check that the random number is different from its - by replaying back to the server the information it sent us we can authenticate, and in fact are effectively bypass the authentication service.

In the S/Key password authentication brute force vulnerability an attacker can gain intermodule access through VPN-1 using S/Key authentication. In S/Key authentication, the client is sent an index number *'n'*, the client authenticates by sending the secrete key hashed *'n'* number of times. An attacker can brute force the authentication by trying all possible secret keys if they know the range of secret key values.

In the GetKey buffer overflow an attacker can stop the firewall daemon leaving whatever policy in the firewall at the time enabled. Due to insufficient bound checking within the GetKey procedure within the intermodule communication protocol, a attacker can send specially crafted packets causing a buffer overflow. By leaving the policy enforcement operational, the attacker may be able to use the policy rules to their advantage.

In the SMTP Security Server DoS, an attacker can send a stream of invalid SMTP commands to the SMTP security server, raising the load of the CPU on the firewall and disabling SMTP mail to get through.

With the fast mode vulnerability, if an attacker knows the address of a protected host, or can discover it, an unauthorised connection to the host can be made using a series of specifically malformed TCP packet fragments.

In the IP fragmented DoS vulnerability, a stream of large IP fragments can be used to cause the code that Firewall-1 uses to log the fragment events to consume most of the CPU cycles. Firewall-1 reassembles packets before they are passed through to identify and audit attacks such as the ping of death, and if the policy allows the packets are then refragmented and forwarded. A denial of service tool, called *'jolt2'* can be used to send large IP fragments.

To resolve the attacks described so far, Checkpoint has release service packs which can be applied to fix the vulnerabilities. These can be obtained from the checkpoint web site.

In the ACK DoS attack, TCP ACK packet can be passed through the firewall without seeing a TCP SYN packet first for the connection. This is a common problem with many firewalls. A work around is available which will check that a TCP ACK packet is only allowed in if a connection has already been established for it. This involves modifying the INSPECT scripts. This information can be obtained from the checkpoint web site.

In the passive FTP vulnerability, an FTP server's PASV port number can be associated with the port number of a service which has a known security vulnerability. This enables the client to exploit the vulnerability of the service to which the port was connected. A work around for this problem is available and can be found on the checkpoint web site.

There are not many attacks in those above that can cause the actual firewall to be haltered or crashed, most attacks are aimed at beyond the firewall and gain access to internal machines on the network behind the firewall. The above attacks are aimed at the firewall software itself, however since the firewall is running on a host, attacks against the operating system that it the firewall is running on should be looked at.

The attack on Keith's primary firewall is made easier by the fact that he does not have a border router or screening router to block or filter packets to the firewall. So the security of the network not only relies on the integrity of the firewall rules and software, but also on the security of the host it is running on.

From the checkpoint Firewall-1 vulnerabilities, the IP Fragmentation DoS attacks can be used to bring the firewall down or put it in an unusable state. The TCP ACK DoS may possibly be used against the external firewall if it is aimed at a service port running on the firewall host. A scan of the external firewall with a port scanner may reveal ports which are open. Because Keith is using Firewall-1 on the internal firewall as well, this means that the above vulnerabilites may also be present. In general for an attack against this particular firewall, a denial of service attack is probably the best choice to use.

Denial of Service Attack

There are several forms of denial of service attacks. There are some denial of service attacks aimed at effecting network performance, these types of attacks use up the network bandwidth. In these attacks a packet storm is created, the aim of the attack is to create as many packets as possible. With too many packets, the network grinds to a halt or an unusable state or is otherwise painfully slow. These type of attacks may are often not directed to a single host, but rather many hosts on the network.

Other forms of denial of service attacks are aimed at specifically bringing down a service or host. These attacks are generally aimed at a specific host. In these attacks, packets are sent to exploit a specific vulnerability (usually in it's TCP/IP stack) that either effect the CPU usage or bring the machine down. An example of an attack which was used to bring

machines down was the ping of death. This is where a ping packet is fragmented, but when the fragments are reassembled at the host, the packet is bigger than it is allowed, this causes the IP stack to fall over as it can't handle this, resulting in many cases for the machine to crash. An example of making a machine unusable while not bringing it down is the SYN attack. A SYN tcp packet is normally sent as part of the 3 way tcp handshake, used to establish a connection. The TCP SYN attack is often known as a half-open connection, as to complete the connection, the client requesting the service is meant to send a SYN ACK back to the server, which the SYN attack does not do. By sending many SYN packets we are trying to establish multiple connections. Because the server is waiting for the acknowledgment response it doesn't close the connection until after a certain time out period, so as a result the machine reaches its limit of the maximum number of connections, and therefore can no longer accept requests, when there are a number os SYN requests at once.

Other denial of service attacks don't actually try to use up resources or crashes or disable a machine, but instead aim to confuse or inject incorrect information into the network. For instance, one form of attack is to poison the DNS server's cache. By providing incorrect DNS information, when hosts go to look up an IP address of a machine supplying a service, instead of getting the right IP for a machine, they either get the IP address of another machine which may not be running the service. This in effect causes a denial of service as the user is being redirected to the wrong service or one that doesn't exist. Common attacks with DNS poisoning have been to redirect web host resolution, so instead of going to ww.microsoft.com, the IP address returned is really for some other web server, for instance www.gambling.com. A hacker can use this form of denial of service to their own advantage, by actually pointing client machines to a fake server the hacker has setup, which may allow them to get more information about a victims network. For example if a mail relay server was setup, a hacker could capture e-mail being sent by users.

Other denial of service methods apart from the ping of death and TCP SYN attack already mentioned include the UDP flood, Smurf, ICMP echo flood, the TCP ACK flood, the ICMP flood and DNS flood among many others which exist.

In a Smurf attack, we spoof the victim's machine and use it as the source IP address. We then send echo request packets to a broadcast address. By having the destination address as a broadcast address, all machine on the destination network try to respond to echo request by sending echo replies to the source address, which is the IP address of the victim which we have spoofed. This causes the victim's machine to be overwhelmed with ICMP echo reply packets.

Different to DNS poisoning in where clients are effected by being given incorrect name resolution information, is a DNS flood. There are two forms of DNS flood, one aimed at the DNS server itself, and the other aimed at a victim's machine. In the first case where the DNS server is aimed at a large number of DNS queries are sent to the DNS server. This attack is aimed at making the DNS server unusable. In the second case aimed at the victim's machine, DNS queries are sent to DNS servers with a spoofed address of the

victim. The DNS servers respond to the victim's machine with DNS query responses. This attack usually causes bandwidth denial of service.

In a UDP flood, two common services on a host are utilised, the chargen (character generator) service and the echo (echo character) service. UDP packets are sent with the source address being spoofed IP address of one victim machine and the source port being the character generator service, to the echo character service of another machine. Another packet is sent to a second victim's machine with the same service and ports. As a result when a packet is sent from the character generator port, the echo service responds with the character back to the character generator, which then sends the character back to the echo, causing an infinite loop of packets. This type of flood can be done to a single machine by specifying the source and destination IP address to be the same. This type of attack is usually aimed at causing bandwidth denial of service.
The ICMP Echo flood is also known as the Ping flood, and is basically a distributed denial of service attack where a host is flooded with icmp echo requests. In looking up information about the ICMP echo flood and ICMP flood, there were several documents saying that they were essentially the same. However, the Network ICE web site distinguishes between the two. Basically an ICMP flood is a large number of icmp control messages (except the icmp echo messages) sent in a short period of time.

Given that we have many compromised machines which we have control of, the logical type of denial of service attack is a distributed denial of service attack. In a distributed denial of service attack, as it's name suggests, comprises of a number of distributed machines running an attacking on a host or network, rather than just a single machine running an attack.

In a distributed denial of service attack a client daemon is installed on the remote machines which have been compromised. A master daemon is installed on another machine. The master daemon is then used to control the client services on the compromised machines. With this in mind, the master directs or coordinates the client machines to perform an attack and send packets to the victims machine all at once, causing a denial of service attack on the victims machine. An example of this is the Tribal Flood Network (TFN). Other similar attack tools include Trinoo, TFN2K, Stacheldraft, mstream and shaft to name just a few. Each of these tools acts in a similar manner with a master service directing all the client services. The main differences between them is the type of protocols and packets they use for the attacks, and how the clients and masters communicate.

The Trinoo distributed denial of service attack tool, can only be used for generating UDP floods. The clients and master communicate via tcp and udp.

The tribal flood network on the other hand has an array of options for attacks. It allows for UDP floods, TCP SYN floods, ICMP Echo floods and Smurf attacks. The clients and masters communicate via icmp echo reply packets. For more information on the tribe flood network see the cert web pages, http://www.cert.org, or x-force http://xforce.iss.net.

For a detailed description including packet values and what each mean between client and master see http://xforce.iss.net/alerts/advice40.php.

TFN2K is one of the first distributed denial of service tools ported to windows, it is a port of Tribal Flood Network, but has additional features. The master and client can communicate over encrypted channels with either TCP/UDP or ICMP. It has the same attack capabilities as TFN as well as being able to combine all the attack methods into a single attack.

Stacheldraht is the combination of Trinoo and TFN. It has clients, masters and agents. Clients are used to control agents. Masters and clients communicate with blowfish encrypted icmp echo replies. Masters also communicate with agents specifying a command sequence number. The master and agent keep a list of command numbers and commands. Agents can even be directed to upgrade themselves. Masters and agents communicate with ICMP echo replies. The attack capabilities are the same as TFN.

Mstream is used for generating TCP ACK attacks on victim's machines. It makes the victims CPU get consumed by processing ACK packets which are flooded to it.

Shaft is similar to TFN and Trinoo. Like Stacheldraht it uses more than just a client and master. It uses agents as well. It uses tcp to communicate with clients, and the clients use udp to communicate with the agents. It has the same attack methods as TFN2K.

For the distributed denial of service attack, I will use TFN2K because this allows a combination of attack methods to be used in the single attack. I will have the master process running on one of the compromised machines, as a trace of icmp echo reply packets could be used to detect which machine I am using to command the attack. If I was using my own machine, then the trace would lead to me. Running the master on the compromised remote machine means that it is much harder to trace me. The TFN2K tool plus the others mentioned, along with many others can be downloaded from the PacketStorm web site - http://packetstorm.secufiy.com.

Each of the compromised machines will be using spoofed source addresses as well for the TCP SYN flood and ICMP flood messages to make the trace even harder. Because the router may block internally spoofed addresses the spoofed addresses may not be from the internal network, but from other addresses on the Internet.

How do you mitigate an attack such as the Tribal Flood Network. Lets look at each type of attack separately.

To mitigate the UDP flood attack, the character generator and echo character ports should be blocked at the firewall or border router. All unused udp services should be disabled on hosts. The firewall should also block all UDP ports less than 900 apart from those UDP ports required. This limits other similar UDP attacks.

As spoofing is usually involved in UDP denial of service attacks, spoofing filters should be put in place at the firewall and/or border router. For further information see http://www.cert.org/advisories/CA-1996-01.html.

To mitigate SYN floods, there are some operating system patches or features to reduce the threat. Some operating systems, such as Linux use SYN cookies. These are sent back to the client instead of the SYN ACK. This allows the SYN queue to be cleared to that the queue will never fill up. Other operating systems uses cached SYN's which reduces or eliminates the affects of a SYN attack. Other vendors have patches for the operating system. More information is available from http://www.cert.org/advisories/CA-1996-21.html.

You can detect a syn flood by running *netstat -s* on most Linux machines to get statistics and look at the SYN-RECIEVED. If there are lots, then this indicates it may be a SYN attack. As with the UDP flood, spoofing is usually used, so the same spoofing remedies should be used.

On linux machines, SYN floods can be disabled via enabling SYN cookies:

```
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

To enable ip spoofing protection on linux machines:

```
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
      echo 0 > $f
done
```

To mitigate smurfing, directed broadcasts can be stopped at the firewall and/or router. To stop at the router if you are using CISCO, you would use *no ip directed-broadcast*. Ping and traceroute can be blocked at the router, and ICMP traffic should be filtered.

For ICMP flood, we cannot stop everything, but can put an intrusion detection system in - by looking at the number of icmp messages going to a machine or coming from a source or sources, you can determine if an ICMP flood is happening. Filters can also be setup on firewalls or routers to limit the number of ICMP packets coming in.

A paper from the CISCO web site http://www.cisco.com entitled 'Distributed Denial of Services (DDos) News Flash' provides a list of preventative measures against the Tribal Flood Network type tools for CISCO routers:

- Use *ip verify unicast reverse-path* command. This will check if the route to the source IP address does actually come from the same interface it arrived from and if not drop the packet.
- Filter the private IP address ranges as per RFC 1918
- Apply ingress and egress filtering as per RFC 2267
- Rate limit IMCP packets, using the *rate-limit* command.
- Rate limit SYN packets using the *rate-limit* command.

Because Keith's firewall has Syn defender enabled a TCP SYN attack would not work, however, and ICMP flood or UDP flood could be directed at the DNS or WEB servers behind the firewall. Similarly a TCP ACK attack could be used at the DNS, WEB servers.

To reduce the affects of a distributed denial of service attack, a screening or border router such as a CISCO router could be placed between the Internet and the external firewall. This would reduce the load of the firewall, and we would be able to put rate limits on the traffic such as those mentioned above.

Attack Plan to Compromise Internal System

There are several attack plans one could use to compromise the internal systems. An attack on an internal system however is generally not a simple matter as attacking a single machine. In reality multiple machine must be compromised.

The first step in the attack would be to try and do some reconnaissance and gather information about the network. This would involve doing network scans. This will provide us with information such as which hosts exist, as well as providing us with information such as what host is the DNS server, what host is the web server.

To map the network we could use nmap or some other similar tool. We will try several types of scans in case the router or firewall is blocking certain types of packets - such as icmp echo requests. To find the DNS server we can run nmap specifying the port 53 and using a udp scan. To find the web servers we would do a tcp scan at port 80 for the network. By using a tool such as nmap we can also determine whether the router or firewall is actually static packet filtering or statefull, by the results of the scans, and can determine some of the router rules.

Why focus on DNS and Web servers. These are some of the most vulnerable in recent times. For example, one attack on DNS, among many others is the BIND NXT attack. This allows the attacker to gain root access on the machine. Other Web vulnerabilites in the past include being able to read files on the web server which aren't part of the document directory used - this allowed people to read the password file on the machine. Web servers are also prone to misconfiguration sometimes.

Once finding the DNS and web servers we should run a reverse ident scan to see what the processes are running as. This will tell us for example is the web is running as *root* or *nobody*. If it's running as root, then it makes a good target to compromise. We should also try and do some operating system finger printing to find out what type of system we are dealing with. We can obtain this information by running nmap with the *'-O'* option. For the version of DNS being run, we can use the *'dig'* command. Similarly to find out information about the web server we can telnet to port 80 and type in some html which will return back to us the version of the web service.

We will initially centre our attack on the DNS server. Assuming we have run the scan of the network and found the DNS server and have run the operating system fingerprinting and found the DNS server to be a Unix variant, we now want to find the version of BIND it is running. In most cases in many organisation, the DNS service is usually BIND. Lets run dig to find the BIND version we dealing with:

```
dig @giac.com version.bind txt chaos
```

We'll assume that the version information we have received back from this command has known vulnerabilities. Apart from the BIND NXT vulnerability, there are two other recent vulnerabilities in BIND, which when used in conjunction can be used to compromise the host that the DNS is running on. The two vulnerabilities referred to are the cert vulnerabilities VU#325431 and VU#196945. These relate to the vulnerability of BIND responding to DNS queries in which it discloses environment variables and the buffer overflow problem in the transaction signature handling (TSIG) code.

We begin the attack on the DNS server, by sending a TCP SYN packet to port 53 to establish a connection. Tcp port 53 is normally used for zone transfers or for DNS queries which are to large for udp to be used. The firewall rules that Keith has provided shows that TCP port 53 is open to anyone – if this port was closed, this attack may not work and another BIND vulnerability such as the BIND NXT exploit could be used. The DNS server sends us back the SYN ACK, and we finish the three way handshake with the ACK. We now want to test if the server has the has the vulnerability which discloses environment information. We do this by sending a packet requesting an inverse query to udp port 53. If information is returned, then we know that the TSIG vulnerability also exists. We now send a UDP packet which contains shell code executable (such as /bin/sh) in it, and other contents in it to cause the TSIG buffer overflow. Due to the buffer overflow, we have now attached /bin/sh to tcp port 53, which runs as the user which was running bind. We can now send shell script commands to port 53 via tcp packets which containing the commands – for example we could not telnet to port 53 and start typing shell commands. At this point we can now download toolkits to enable us to become root on the machine.

We have now compromised the DNS server. We may wish to now download the contents of the /etc/password and /etc/shadow files, or the DNS records containing the names and IP addresses of machines. The MX and HINFO records will be useful in finding the mail server and information about machines in the network. We may also wish to add our own DNS entries into the name server.

We may also wish to install a sniffer that picks up traffic on the local ethernet. This may be useful as we might be able to determine other sites to use as backdoor's. For example, if we can see that a certain external site has sent a lot of name resolution requests, then maybe that site is a business partner. If we broke into that site, maybe there is a way from there into here. If we were able to see packets using protocol 50 or 51 then we could assume that there is a VPN service running and possibly another way into the network – in Keith's network we would not be able to see these packets as they bypass the wire the DNS server is on, however we would still be able to examine which external sites are

using the DNS the most (assuming the external site had a DNS server which did recursive DNS lookups). Another thing we can do now is to use a network scanner to see if we can scan the internal network and find out what ports are open, and what hosts may exist, etc.

Examples of two toolkits to exploit the TSIG vulnerability are the *'erkms'* toolkit and the *'li0n'* worm. For more information of the above attack and vulnerability see http://www.cert.org/incident_notes/IN-2001-03.html and for information on *'li0n'* or *'lion'* as it is also known as, see http://www.sans.org/y2k/lion.htm.

The next host or hosts we could attack are the web servers, but this would only gain us access to more machines in the demilitarised zone (DMZ). Compromising the web servers might be useful if we wanted to get more information that would be useful for getting customer information or fortune cookie sayings.

Other attacks we could do is to use the Firewall-1 vulnerabilities discussed in the section on attacking the firewall to get access to machines or cause a denial of service. For example, maybe the results of a network scan show there is an FTP server located behind the firewall. We could use the FTP Connection Enforcement Bypass vulnerability to do ftp-bounces to machines that are accessible to the FTP server.

However, my next host to attack will be the internal DNS server. If we can obtain access on this machine, then we may have free reign on the internal network depending on if the edge router allows us. In either case, this server will provide us with more information about the network from the DNS records and from running a network scanner if we can get root access. If we are unable to get root access to the machine, we could at least perform a denial of service on the machine by causing a DNS flood, which would knock out the corporate and remote user networks from communicating at least. If the machines on the internal networks were on different segments which did not propagate broadcasts to find out host IP addresses, then this denial of service attack would have more of an impact. For example could use the DNS flood tool called *'DDNSF'* which is written by Extirpater to do this.

We will use the BIND NXT vulnerability to compromise the internal name server, assuming that the internal DNS server is vulnerable. With this exploit, we send a DNS query to the internal DNS server. The DNS server will reply to the external DNS server. We construct a reply with a NXT record around 6500 characters long, which will overflow the internal DNS server. By placing certain executable code in the buffer overflow we can obtain the privilege level of the user running the internal DNS. The source code for this exploit can be obtained from http://wwww.rootshell.com and http://www.antionline.com/cgi-bin/anticode/anticode.pl?dir=nameserver-exploits, as well as http://www.now.net/security/exploits.

We have now gained access to the internal DNS server, and can install our toolkits to become root. We can now obtain the DNS information of all the internal clients and decide on which ones to try to compromise.

For information about protecting DNS servers, see http://www.securityportal.com and search for a paper titled 'Hardening the BIND DNS server' by Sean Boran.

Conclusion

There are vulnerabilities being found all the time for different firewall products and for services such as DNS and WEB servers which are just a few among a large number of services people have available. We cannot rely on just one line of defence for our network systems, but must employ a number of defences and auditing tools to keep us aware of what is happening within our network, and even then there may be those new vulnerabilities that were are unable to detect. Firewalls and routers are not the be-all and end-all of network security but are just a few of the items that are recommended in trying keeping our network secure.

References

- Chris Brenton: SANS Firewall 101 Course Notes:

  2.1 TCP/IP for Firewalls and Intrusion Detection – Chris Brenton
  2.2 Firewall 101: Perimeter Protection with Firewalls – Chris Brenton
  2.3 Advanced Perimeter Protection and Defence – Chris Brenton
  2.4 VPNs and Remote Access – Chris Brenton

- Robert L. Ziegler: "Linux Firewalls",1999, New Riders
- Stephen Northcutt and Judy Novak: "Network Intrusion Detection An Analyst's Handbook" (2nd Ed), 2001. New Riders.
- Elizabeth D. Zwicky, Simon Cooper and D. Brent Chapman: "Building Internet Firewalls", (2nd Ed). 2000. O'Reilly.
- Matt Curtin and Marcus J Ranum: "Internet Firewalls: Frequently Asked Questions", 2000 Rev 10.0.
- Rusty Russell: "Linux netfilter HOWTO"
- Rusty Russell: "Linux iptables HOWTO"
- Paul Russell: "ipchains HOWTO"
- Various: "Firewall-HOWTO"
- Unknown: "FreeS/WAN How-to". http://www.freeswan.org
- Oscar Delgado: "Windows 2000, Linux FreeS/WAN and PGPNet interoperability using x509v3 certificates". http://www.strongsec.com/freeswan
- Various: "IP Security Protocol (ipsec)". Links to IPSec drafts and RFCs http://www.ieft.cnri.reston.va.us/html.charters/ipsec-charter.html
- Jean-Francois Nadeau: "IPSec Practical Configurations for Linux Freeswan 1.x", 4-3-2001. http://jixen.tripod.com
- John D. Hardin: "Linux VPN Masquerade HOWTO"
- Mathew D. Wilson: "VPN HOWTO"
- Unknown: "TIS Internet Firewall Toolkit Overview". http://www.tis.com
- Unknown: "FWTK FAQ – Fixes for FWTK 2.1". http://www.fwtk.org
- Unknown: Guantlet 6.0. NAI. http://www.nai.com
- Various: Cert Incident Notices. http://www.cert.org
- Various: xforce vulnerability lists. http://xforce.iss.net
- Various: "Help Defeat Denial of Service Attacks: Step-byStep", SANS. http://www.sans.org
- Various: "Consensus Roadmap for Defeating Distributed Denial os Service Attacks". SANS Global Incident Analysis Centre. http://www.sans.org
- David Dittrich: "An Analysis of the "Shaft" Distributed Denial of Service Tool", SANS Global Incident Analysis Centre. http://www.sans.org
- Unknown: "Denial of Service Attacks". CERT Coordination Centre. http://www.cert.org
- Unknown: "Denial os Service Attacks using Nameserver", CERT Incident Note IN-2000-04. http://www.cert.org

- Unknown: "Denial of Service Attack using the TFN2K and Stacheldraht Programs". IIS Security Alert – Feb 9, 2000. http://xforce.iis.net
- Unknown: "Remote Vulnerabilities in BIND versions 4 and 8". IIS Security Alert – Jan 29, 2001. http://xforce.iis.net
- Unknown: "Multiple Vulnerabilities on all platforms and versions of Check Point Firewall-1". IIS Security Alert. Sep 27, 2000. http://xforce.iis.net
- Rich Jankowski: "Scanning and Defending Networks with Nmap". 20-2-200. Linux Security. http://www.linuxsecurity.com
- Sean Boran: "Hardening the BIND DNS Server". Security Portal. http://www.securityportal.com
- Unknown: "Denial of Service Developments", Jan 3, 2000. CERT Advisory CA-2000-01. http://www.cert.org
- Unknown: "UDP Port Denial-of-Service Attack", Sep 24, 1997. CERT Advisory CA-1996-01. http://www.cert.org
- Unknown: "TCP SYN Flooding and IP Spoofing Attacks", Sep 19, 1996. CERT Advisory CA-1996-21. http://www.cert.org
- Unknown: "IP Spoofing Attacks and Hijacked Terminal Connections", Jan 23, 1995. CERT Advisory CA-1995-01. http://www.cert.org
- Various: "CERT Security Improvement Modules", (various papers) CERT. http://www.cert.org/security_improvement
- Unknown: "IP Denial-of-Service Attacks", Dec 16, 1997. CERT Advisory CA-1997-28.
- Unknown: "CERT Summary CS-2000-01". CERT. http://www.cert.org/summaries
- Unknown: "Distributed Denial of Service Tools", Jan 15, 2001. CERT Incident Note IN-99-07. http://www.cert.org
- Unknown: "Exploitation of BIND Vulnerabilities", Mar 30, 2001. CERT Incident Note IN-2001-03. http://www.cert.org
- Fyodor: "Top 50 Security Tools". http://www.insecure.org
- Unknown: "CERT/CC Advisories 1988-2001". CERT. http://www.cert.org
- Unknown: "Tribe Flood Network Denial of Service Tool" IIS Security. http://xforece.iis.net
- Unknown: Distributed Attack Tools – http://packetstorm.securify.com
- Unknown: "Distributed Denial os Service (DDoS) News Flash", Feb 9, 2000. CISCO Systems. http://www.cisco.com
- Kurt Seifried: "Denial of Service (DoS) FAQ" Mar 5, 2000. Security Portal. http://www.securityportal.com
- Unknown: "ICMP Flood". Security Alert. Net ICE. http://advice.networkice.com/Advise/Intrustions
- DNS Vulnerabilities – Lace Spitzner: http://www.enteract.com/~lspitz/intrustion.html