GIAC Certified Firewall Analyst
Practical Assignment
Version 2.0

Ronald Young

March 4, 2004

ABSTRACT 4

DEFINITIONS 5

INTERNAL NETWORK IP NUMBER ASSIGNMENTS 5

RUNNING THE SIMULATION 5

ASSIGNMENT I: SECURITY ARCHITECTURE 8

Business Operations 8
Business Operations for the General Public 8
General Operations for Customers, Suppliers, and Partners 8
Specific Business Operations for Customers 9
Specific Business Operations for Suppliers 9
Specific Business Operations for Partners 10
Specific Business Operations for Employees Using the Internal Network 10
Specific Business Operations for Mobile Sales Force and Teleworkers 11
Technical Description and Requirements 11
General Considerations 11
Employee Desktop/Laptop Systems 11
Mobile Sales Force and Teleworkers 12
Customers, Suppliers, and Partners 13
Technical Description and Requirements for Server Systems 13
Use of Virtual Servers 17
West Site Virtual Server Platforms vs1, vs2, and vs3 18
East Site Virtual Server Platforms: vs4 and vs5 18
West site router: router1 (perimeter screening router / External IDS sensor 19
West Site Router: router2 (interior router / IDS) 21
East Site Router: router3 (perimeter screening router / external IDS sensor) 23
Network Time (NTP) servers: ntp1.example.com and ntp2.example.com 25
Domain Name System (DNS) server: ns1.example.com and ns2.example.com 25
Web server: www.example.com 26
External E-mail Server: mail.example.com 27
Network Management Hosts: netmgmt1 and netmgmt2 28
Network Information Logging Hosts: loghost1 and loghost2 28
Internal Web Server: intweb1 29
Partners Web Server: srvweb1 29
Data Base Server: dbsrv1 30
Network File Servers: samba1 and samba2 30
SSH gateways: gw1a.example.com and gw2a.example.com 31
VPN gateways: gw1b.example.com and gw2b.example.com 31
Internal DNS Servers: intns1 and intns2 32
Internal E-mail servers: intmail1 and intmail2 32
Web Proxy Servers: gw1.example.com and gw2.example.com 33
Network Infrastructure Components: switches 33

ASSIGNMENT 2: SECURITY POLICY AND TUTORIAL 34

Common Rules for All GIAC Firewalls 34
Border Router and Primary Firewall (router1, west site) 38
Firewall Policy (router2 -- west site internal) 43
Border Router and Primary Firewall (router3, east site) 46
VPN Configuration for west-east site tunnel 51
GIAC Enterprises VPN Installation and Configuration Tutorial 54

ASSIGNMENT 3: VERIFY THE FIREWALL POLICY 79

Validation Plan 79
Conducting the validation 80

IMPORTANT NOTE ABOUT THE "NMAP" PORT SCANNER .... 80\\
INGRESS FILTERING TESTS.. 81
Test Sequence 1: Use default iptables policies only *81*
Test Sequence 2: No rc.common-fw rules, NAT and Filtering are active....................... *84*
Test Sequence 3: Full Configuration.. *90*
EGRESS FILTERING TESTS 94
EVALUATION OF RESULTS AND RECOMMENDATIONS.. 96

ASSIGNMENT 4: DESIGN UNDER FIRE 97

DISTRIBUTED DENIAL OF SERVICE ATTACK AND COUNTERMEASURES.. 98
ATTACK PLAN TO COMPROMISE AN INTERNAL SYSTEM 101
COUNTERMEASURES FOR COMPROMISED INTERNAL SYSTEM 103

APPENDIX 1: FULL VNUMLPARSER.PL CONFIGURATION FILE FOR TEST NETWORK SIMULATION.. 104

APPENDIX 2: REVISED NMAP PATCH.. 110

REFERENCES.. 112

ABSTRACT

GIAC Enterprises is a small but rapidly expanding e-commerce company specializing in selling fortune cookie sayings. GIAC currently has approximately 45 employees in two separate locations. Company management has requested a revised information technology strategy that will allow the opening of new locations or staff increases on short notice. In addition to short implementation timelines, minimization of equipment costs and changes to the existing corporate infrastructure must also be considered.

By using open source technologies, all of these considerations can be accomplished with little cost. In addition, there are several open source projects that provide advance technologies. For example, the User Mode Linux project allows the Linux kernel to run as a user mode process, creating a self contained virtual environment with its own separate security context. UML will be used by GIAC Enterprises as an essential component in its network security architecture. This study also provides details on how to use UML to provide an isolated full-featured network test environment.

This study is composed of four parts:

- GIAC network and security architecture. This section describes the structure of the proposed network and access conditions.
- Security policy and tutorial. The complete security policy for the primary firewall, VPN, and border routers is presented in the second part. A tutorial regarding the configuration and implementation of the Virtual Private Network (VPN) gateways are also included.
- Verification of the primary firewall policy.
- Design under Fire. The network design chosen for analysis was presented by Lesa Ludwig's GCFW Practical of October 2003. Her presentation is available at the GIAC website: http://www.giac.org/practical/GCFW/Lesa_Ludwig_GCFW.pdf

Throughout the rest of this document, frequent use is made of information presented at the SANS Los Angeles 2003 Training Track 2. This track was presented by Chris Brenton and consisted of six days of material: TCP/IP, Packet Filters, Firewalls, Defense in Depth, VPNs, and Network Design & Assessment. This material is used without further reference. Additionally, portions of the SANS San Diego 2004 Training Track 4, Hacker Techniques, Exploits & Incident Handling, presented by Ed Skoudis was use as a reference for exploits used in Part 4 of this document (Design Under Fire).

For the purposes of this study, it is assumed that GIAC Enterprises has obtained the domain "example.com" for use on the Internet.

Definitions

Throughout this document, references are made to several terms, particularly in the security policy tables. Commonly used terms are:

- “Any External” - refers to any valid external (routable) Internet address,
- “aaa.bbb.ccc.###” refers to the ip number block assigned to the west site of GIAC Enterprises. It contains routable addresses. Note: for the internal test environment, the IP number 10.0.3.1 is used for the west site Internet connection and numbers 10.0.3.32-10.0.3.47 are used as the public addresses for external west site services.
- “eee.fff.ggg.###” refers to the ip number block assigned to the east site of GIAC Enterprises. It contains routable addresses. Note: for the internal test environment, the IP number 10.0.3.100 is used for the east site Internet connection and numbers 10.0.3.80-10.0.3.95 are used as the public addresses for external east site services.
- Host system (in network simulation terms only): this is the process that initiated the vnumlparser.pl network simulation script. Whenever a reference is made “to entering a command on the host system”, this means that the command is to be entered into the shell of this process.

Note: fully qualified domain names (FQDN) are used to identify hosts (i.e. gw1b.example.com) that are reachable from the Internet (either directly or via NAT). Hosts that are not directly reachable from the Internet are referred to by a simple host name (intns1).

Internal Network IP Number Assignments:

The network design for GIAC Enterprises uses non-routable IP numbers internally. A numbering convention has been adopted to simplify understanding of the network topology. All addresses are contained in the 10.0.0.0/8 network. West site addresses are all in 10.1.0.0/16, east site addresses are all in 10.2.0.0/16. Public service networks are 10.x.1.0/24. Network management IP addresses are 10.x.10.0/24. The private service networks are 10.x.20.0/24. Internal corporate networks are 10.x.30.0/24. VPN addresses are 10.x.40.0/24. The restricted network between the internal web and database servers is 10.1.50.0/24. For simulation purposes, network 10.0.3.0/24 is used to represent the external Internet. Throughout the rest of this document, all references to 10.0.3.xxx/24 should be treated as “real” routable IP addresses.

Running a Simulated Network Environment

One of the tools used in developing and testing the network described in this document was “Virtual Network User Mode Linux”, developed by The Virtual Network User Mode Linux (VNUML) project. Vnuml allows the development of a complete network simulated on a single Linux host. The project’s home page is located at http://jungla.dit.upm.es/~vnuml. The configuration file for this project’s virtual network can be found in Appendix 1. Note: this does not configure the entire GIAC network, only the portions of interest for perimeter security. For example, all of the virtual machines in this document are configured as separate machines. This was done for convenience, vnuml could handle the full configuration, but doing so would be extra effort that wouldn't affect anything.

While vnuml does an excellent job at providing a robust simulated network environment, there are several things that must be done manually to provide routing into and out of the network. A set of static routes must be added on the host system and two simulated routers for the "external" IP addresses present on the simulated network. Otherwise a routing protocol infrastructure would have to be installed. Such an infrastructure is beyond the scope of this study since it would normally be provided by the ISP assigning the IP numbers.

The routes needed on the host system are:

```
route add -host 10.0.3.33 gw 10.0.3.1
route add -host 10.0.3.34 gw 10.0.3.1
route add -host 10.0.3.35 gw 10.0.3.1
route add -host 10.0.3.36 gw 10.0.3.1
route add -host 10.0.3.37 gw 10.0.3.1
route add -host 10.0.3.38 gw 10.0.3.1
route add -host 10.0.3.39 gw 10.0.3.1
route add -host 10.0.3.81 gw 10.0.3.100
route add -host 10.0.3.82 gw 10.0.3.100
route add -host 10.0.3.83 gw 10.0.3.100
route add -host 10.0.3.84 gw 10.0.3.100
route add -host 10.0.3.85 gw 10.0.3.100
route add -host 10.0.3.86 gw 10.0.3.100
route add -host 10.0.3.87 gw 10.0.3.100
```

The routes needed on the west primary router (router1) are:

```
route add -host 10.0.3.81 gw 10.0.3.100
route add -host 10.0.3.82 gw 10.0.3.100
route add -host 10.0.3.83 gw 10.0.3.100
route add -host 10.0.3.84 gw 10.0.3.100
route add -host 10.0.3.85 gw 10.0.3.100
route add -host 10.0.3.86 gw 10.0.3.100
route add -host 10.0.3.87 gw 10.0.3.100
```

The routes needed on the east primary router (router3) are:

```
route add -host 10.0.3.33 gw 10.0.3.1
route add -host 10.0.3.34 gw 10.0.3.1
route add -host 10.0.3.35 gw 10.0.3.1
route add -host 10.0.3.36 gw 10.0.3.1
route add -host 10.0.3.37 gw 10.0.3.1
route add -host 10.0.3.38 gw 10.0.3.1
route add -host 10.0.3.39 gw 10.0.3.1
```

Note: There is a one line program bug in the vnumlparser.pl (version 1.3.1) utility that causes duplicate maintenance interface IP numbers to be assigned to hosts in networks with more than 16 hosts. The line 2392 should be changed from: `"my $byte4 = ($seed << 2 ) & 63;"` to "`my $byte4 = ($seed & 63) << 2;".`

Also, since the purpose of this study is network security and not system configuration, it is not necessary to fully configure all of the services outlined in the security policy. It is only necessary to respond to connection attempts on specific ports. An easy way to accomplish this is to use the “nc” (netcat) utility in listen mode. For example, to make the public web server (www1) accept incoming http and https connections, simply run the following commands in a shell on www1:

```
# nc -l -p 80 &
# nc -l -p 443 &
```

The external name server (ns1) will respond to DNS packets by listening to both the udp and tcp ports (from a shell on ns1):

```
# nc -l -u -p 53 &
# nc -l -p 53 &
```

ASSIGNMENT 1:
Security Architecture

This section describes in detail the security requirements and methods of access for each class of users of the GIAC Enterprise's corporate network. The first part describes the different user classes and their associated business operational requirements. The second part of this section describes the security architecture components of the network and their access policies.

Business Operations for the General Public

There are two possible methods for use by the general public in contacting GIAC Enterprises. For Internet users, a web-page that can send requests for information or comments to the sales staff is available. For non-Internet inquiries, an advertised phone number and postal mail address will be available. Both the phone number and mail address is also directed to the sales staff. Only general inquires will be accepted by phone or mail, all others will be directed to use the Internet.

The web-site for use by the general public will only serve static content from a dedicated server. The general public may access most of the content of this web-site using a normal, non-secure HTTP connection. If they send an information request to the sales department, this message will be sent using a secured connection back the web server. Except for the sending of information requests to the sales department and maintaining the server and its content, no other access to/from other services, internal systems, or networks will be allowed by this server.

If a member of the general public desires to establish a business relationship (as a customer, supplier, or partner) the sales staff will be responsible for creating and maintaining the records necessary to accomplish this. After vetting the new prospect and agreeing to establish this relationship, the prospect will become an account. The sales staff will request the generation of the authentication tokens necessary for the new account to access the GIAC corporate network resources. A member of the technical support team will be available to assist the new account in the acquisition, installation and configuration of the equipment necessary to connect to the GIAC corporate network.

General Business Operations for Customers, Suppliers, and Partners

While there are distinct requirements for each user group: customers, suppliers, and partners, they all have some in common. These common requirements are:

1. The most important requirement for GIAC Enterprises is the protection of their critical business assets. Due to the nature of GIAC Enterprises' business, unauthorized release or destruction of their assets could potentially ruin the company.
2. Routine business communications between GIAC Enterprises and their

customers, suppliers, partners, and the general public normally take place electronically via the Internet. This will require that the person attempting to contact GIAC Enterprises have a network connection. For our purposes, all of these Internet links will be considered insecure and subject to interception.

3. Access to GIAC Enterprises resources require authentication tokens that uniquely identify the person and system connecting to those resources. These tokens also control the user's access rights to content and resources.
4. Except for electronic mail, time synchronization, and public web site access, secure communication methods will be required to use GIAC Enterprises resources. The specific methods and products used will vary and are outlined in the technical descriptions below. Usually an encrypted connection to a server located on the GIAC network will be used.
5. Outgoing communications by users located inside the GIAC corporate network must be routed through the GIAC network gateway/proxies. This access will normally be done using normal connections to a few well-known services. All access other than to the following services will be prevented by the firewall. The permitted TCP services are: 25 (SMTP), 22 (SSH), 80 (HTTP), and 443 (HTTPS). The permitted UDP services are 123 (NTP), 53 (DNS) and ports in the range of 5xxx (OpenVPN virtual private networks), currently only port 5000 is used.
6. All network usage (including any attempted accesses) will be logged in a format suitable for auditing. This information will also be used to generate invoices for content downloads and payment verification for content uploads.

These requirements insure that the critical business assets of GIAC Enterprises are reasonably protected from unauthorized disclosure, modification or destruction. The primary method of protection is to control access to the critical resources. These requirements are also used to mitigate the effects if an unauthorized access does happen. It is necessary to capture sufficient information to identify the scope of the release or modification as well as aiding in the identification of the person(s) responsible. Since it may be impossible to obtain this information after the fact, it must be captured and stored in a secure manner as it is generated.

Specific Business Operations for Customers

In addition to the general business operations requirements described above, there is an added requirement of being able to securely accept credit and debit cards for payment. Without this ability, customers without an established history with GIAC could possibly be required to have funds deposited on account prior to an order being processed. A prepayment requirement would, therefore, place an added burden on both the customer and GIAC Enterprises.

Specific Business Operations for Suppliers

In addition to the general business operations requirements described above, suppliers have the added requirement of “write-only” data uploads. They must be able to upload a batch of new fortune cookie sayings to a temporary staging area in a secure manner. Once a batch has been uploaded, GIAC technical staff will process and insert the batch into the production content collection. One of the key steps in this processing is to validate the format of each saying before allowing it to be added to the collection. Additionally, suppliers must not be able to read or otherwise download any other information from the GIAC network or data collections.

A key concept of reliable and secure information processing is to restrict the authorized access to company assets to the minimal level necessary to accomplish the legitimate business uses. Content suppliers generally have no legitimate business need to read information from the existing collection. Verification of the uploaded sayings prior to adding them to the production collection is a generally accepted method to help prevent the introduction of improperly formatted data.

Specific Business Operations for Partners

The business operation requirements for partners can be thought of as a hybrid of the ones for suppliers and customers. The specific details depend on the partnership agreement. If the partner is only involved with translating and reselling content, then they could be treated as a customer with preferred pricing and distribution rights. If there is an agreement that allows GIAC to include the partner's translated sayings in the production collection, then a special supplier relationship exists. Because of these possibilities, the requirements for partners are the combination of the requirements for both suppliers and customers. The reasons that justify these requirements are also the same as those for customers and suppliers

Specific Business Operations for Employees Using the Internal Network

In addition to the business operations required to provide services to GIAC Enterprises' accounts, routine business computer uses must also be supported in a secure manner. These include, file storage on central servers, document printing, electronic mail, “terminal” sessions and web access. Employee access to the network is provided by 10/100mb CAT5 wired network connections to each system on the internal corporate network. Wireless network access is planned but is presently not activated in the corporate offices. When wireless access is activated, the initial deployment will be to central meeting locations like conference rooms.

The current standard computer environment provided to most GIAC Enterprises employees is an IBM-PC compatible desktop or laptop system running Microsoft Windows XP. Computer capabilities vary but a realistic minimum configuration is a system with at least a 1.0Ghz Pentium CPU, 256mb RAM, and 40GB hard-drive. While GIAC Enterprises strives to keep the operating systems on employee systems updated with relevant vendor fixes, this cannot be guaranteed. The technical descriptions below describe the software packages and configuration settings used for employee computer systems.

Specific Business Operations for Mobile Sales Force and Teleworkers

For our purposes, the only significant distinction between employees of the mobile sales force and teleworkers is in their method of connection to the Internet. Mobile sales will normally connect to the GIAC internal network via a dial-in connection. These dial-in connections will be to a national ISP like AOL or Earthlink. Teleworkers will normally connect via a high-speed “always-on” device similar to a cable-modem, DSL phone line, or dedicated frame-relay lease line. The dial-in connections will normally be assigned a transient network address. The high-speed connections may either have a transient or static IP addresses. The mobile sales force and teleworkers require the same level of access to content and other resources as their counterparts located on the internal GIAC network.

There is another requirement for mobile and teleworkers in addition to those for employees located on the GIAC internal network. Since employees connect directly to the internal GIAC network, the security perimeter expands to include the remote system and any intermediate insecure network links. Since the level of access on remote systems is identical to those directly connected to the internal network, the same justifications apply to remote systems.

Technical Description and Requirements General Considerations

A major component of GIAC Enterprises' corporate philosophy is to embrace the promise of open source. To this end, preference is given to Linux and other open source products running on commodity hardware platforms instead of proprietary solutions. A proprietary solution will only be chosen over an open source one, if a compelling business case can be made. Therefore, Linux running on commodity hardware platforms is the preferred choice for GIAC Enterprises' network infrastructure and server operations. Currently, the use of Microsoft Windows is preferred for employee desktop systems. At the current time, it is not cost effective due to training costs to transition employees over to Linux. However, this is expected to change within the next 2-3 years, which coincides with the next upgrade cycle for employee systems.

Technical Description and Requirements for Employee Desktop/Laptop Systems

Unless otherwise justified by business needs, the following products and configuration settings are required to be used whenever an Employee system is connected to the GIAC internal network.

> Operating System: Microsoft Windows XP with latest service pack installed (web reference: http://www.microsoft.com/windowsXP). Internet Explorer and Outlook/Outlook Express must not be installed (if present, they should be removed).

Web Browser: Mozilla Firebird 0.7 (http://mozilla.org/products/firebird).
Email Client: Mozilla Thunderbird 0.4
(http://mozilla.org/products/thunderbird).

Terminal: PuTTY 0.53b
(http://www.chiark.greenend.org.uk/~sgtatham/putty).
PuTTY provides a secure terminal connection by using the "secure shell" (SSH) protocol. In addition to the terminal connections, PuTTY can also be used to securely transfer otherwise vulnerable traffic to the GIAC network by use of "SSH tunnels". SSH tunnels allow plain-text data like email to be sent over insecure intermediate network links. The data is encrypted before it is transmitted and then decrypted on the receiving host.

File Transfers: WinSCP 3.3.0 (http://winscp.sourceforge.net). WinSCP uses SSH to securely transfer files between the employee's computer and the GIAC file servers.

File Compression/Archiver: WinZip 8.0 (http://www.winzip.com). WinZip is used to compress user files.

Personal Firewall: ZoneAlarm Pro 4 (http://zonelabs.com). The intent of using a personal firewall on employee systems is to notify them of unexpected activity that has made it through the main defenses. Because of this secondary protection role, ZoneAlarm Pro was chosen instead of a centrally managed firewall like sygate (http://www.sygate.com). NOTE: It is part of the employee's employment agreement to make a reasonable effort to maintain network and computer security, so it is their responsibility to make sure that any changes that they may make to their personal firewall does not compromise network/system security. If they are unsure, it is their responsibility to contact the technical staff for assistance.

Time Synchronization client: AboutTime4.8 (http://www.arachnoid.com). While Windows XP has time synchronization built-in to its control panel, the vendor supplied version does not provide sufficient capabilities for our needs. AboutTime allows us to support selective time protocols and more granular configuration settings.

Technical Description and Requirements for Mobile Sales Force and Teleworkers

The requirements for mobile sales force and teleworker employees are the same as those for employees directly connected to the internal GIAC network with two additions. The first added requirement is for systems that connect to the network via an "always-on" connection: a hardware device capable of filtering network traffic must be installed between the computer system and the modem/router for the network connection. This device must be able to provide network address translation (NAT) and selective blocking of network traffic based on ports and protocols. An example of such a device is the Linksys BFSR41 Cable/DSL

Router with 4-Port switch (reference web page: http://www.linksys.com/products/product.asp?grid=34&scid=29&prid=561). However, the Linksys WRT54G Wireless-G Broadband Router (reference web page: http://www.linksys.com/products/product.asp?grid=33&scid=35&pid=577) is the preferred device.

While both of the BFSR41 and WRT54G devices provide NAT and filtering capabilities, the WRT54G is preferred because it provides additional capabilities and is based on Linux firmware. Linksys provides a freely redistributable development environment that the GIAC technical staff can use to customize the router's firmware.

The router hardware requirement has been relaxed for the mobile sales force connecting to the GIAC network by a dial-in phone line. Requiring the use of a separate hardware device, would require workers to transport an additional piece of equipment, thereby placing an added burden on them for a relative modest increase in security.

The second additional requirement for mobile sales force and teleworkers is that all traffic must be encrypted and directed towards the GIAC Enterprises network. Any non-GIAC destined traffic will be forwarded to the correct host by proxy servers located on the GIAC network.

Technical Description and Requirements for Customers, Suppliers, and Partners

The technical requirements for customers, suppliers, and partners are identical to those defined for GIAC Enterprises' employees. However, the use of specific products and/or configuration settings will be waived provided that the customer, supplier, or partners certifies that their own computer security policies meet or exceed those of GIAC Enterprises.

Technical Description and Requirements for Server Systems

The basic component building blocks used for the GIAC Enterprises server and network infrastructure are purchased as configured systems and components from national vendors. The principle vendors selected for the network infrastructure components are Dell (servers) and Hewlett-Packard (switches). Dell offers a wide range of systems in a 1U form factor. The preferred operating system for the server platforms is Red Hat's Fedora Core-1 Linux distribution (http://fedora.redhat.com).

The corporate computer resources of GIAC Enterprises consist of two main sites, “east” and “west”. The east site is located in GIAC Enterprises' east coast regional sales offices. The west site is located at corporate headquarters and is used to provide normal production service. This approach was chosen for two reasons. The first reason is to allow for additional sites to be added with minimal changes to the existing network infrastructure. Secondly, this design allows the east site to provide limited business continuity (disaster recovery) services in case of a failure of the main Internet connection or servers.

The general network architecture also adheres to commonly held “best practices”. A screening router protects the internal networks from the Internet. At each site, there are four internal networks: a public service network (DMZ), an internal services network (SRV), the internal employee network (INT), and a secure management network (NET). For more information regarding industry best practices, enter the search terms: network architecture best practices” into a search engine (such as Google or Yahoo!). There have been many books published on the subject as well. The book “Linux Firewalls, 2nd edition, by Robert Ziegler, New Riders Publishing, 2002, pg 214-232, describes some of these practices in detail. The addition of a secure network management segment is based on its successful use by a GIAC technical staff member at a prior place of employment (unnamed in accordance to their security policy).

The exposure of GIAC Enterprises' computer resources to the Internet will limited to a few well know fully qualified domain names (and associated IP numbers). As part of the network infrastructure, we have arranged network connectivity with a national ISP for both the east and west sites. Each site has a small block of assigned IP addresses, aaa.bbb.ccc.32/28 (16 hosts – west) and eee.fff.ggg.80/28 (16 hosts – east).

West site (aaa.bbb.ccc.32/28)		***East site (eee.fff.ggg.80/28)***	
host	***DNS FQDN***	***host***	***DNS FQDN***
32	Reserved (this host)	80	Reserved (this host)
33	ntp1.example.com	81	ntp2.example.com
34	ns1.example.com	82	ns2.example.com
35	www1.example.com	83	www2.example.com
36	gwa1.example.com (SSH)	84	gwa2.example.com (SSH)
37	gwb1.example.com (OpenVPN)	85	gwb2.example.com (OpenVPN)
38	Mail1.example.com	86	mail2.example.com
39	gw1.example.com	87	gw2.example.com
40-46	Reserved for future use	88-94	Reserved for future use
47	Reserved (broadcast)	95	Reserved (broadcast)

Table 1: External Host Assignments

With the exception of the above hosts, all other GIAC Enterprises systems are not directly accessible from the Internet. All internal systems will be configured using non-routable IP addresses and the perimeter security policies will prohibit the entry and exit of non-routable addresses from the GIAC network.
There is an additional DNS related issue with how the GIAC network is designed. There are multiple servers with unique host names and IP numbers providing the same service. For example: www.example.com refers to the currently active public web server, either www1.example.com or www2.example.com, and mail.example.com refers to the currently active public e-mail server, either mail1.example.com or mail2.example.com.

The approach taken to address this problem is to include all of the public names in the domain tables for example.com. A CNAME (canonical name) record points to the currently active server for each service. The TTL (time-to-live) of the name server entries is set to be only slightly longer than the length of a normal session. This minimizes the length of interruption of service if one of the name servers should be unavailable.

In the future, depending on growth, it may be desirable to change the DNS architecture to one based on “load balancing”. Using a load balancing approach allows the DNS servers to dynamically change the generic name to point to the specific server with the least “load” to improve performance. An example of a load balancing name server is lbnamed, written by Roland Schemers and currently maintained by Rob Riepel both of Stanford University. Lbnamed’s home page is: http://www.stanford.edu/~riepel/lbnamed.

Public Name (CNAME)	***West Site (normal)***	***East Site (backup)***
ntp.example.com	ntp1.example.com	ntp2.example.com
www.example.com	www1.example.com	www2.example.com
mail.example.com	mail1.example.com	mail2.example.com

Table 2: External Host Alias Names

Figure 1: GIAC Enterprises West Site Internet Connection and network

```
West Site Network Assignments:

 10.1.1.xxx/24 -- general service network (NATed from Internet)
10.1.10.xxx/24 -- network management net
10.1.20.xxx/24 -- private service network
10.1.30.xxx/24 -- internal corporate network
10.1.40.xxx/24 -- west site VPN
10.1.50.xxx/24 -- Web-Database point-to-point server connection

West Site Static NAT Assignments:

aaa.bbb.ccc.41 -> 10.1.1.10  ntp1.example.com
aaa.bbb.ccc.42 -> 10.1.1.11  ns1.example.com
aaa.bbb.ccc.43 -> 10.1.1.12  www1.example.com  (www.example.com)
aaa.bbb.ccc.44 -> 10.1.20.32 gwa1.example.com  (gwa.example.com - SSH gw)
aaa.bbb.ccc.45 -> 10.1.20.33 gwb1.example.com  (gwb.example.com - OpenVPN gw)
aaa.bbb.ccc.46 -> 10.1.1.41  mail1.example.com (mail.example.com)
aaa.bbb.ccc.47 -> 10.1.20.31 gw1.example.com
```

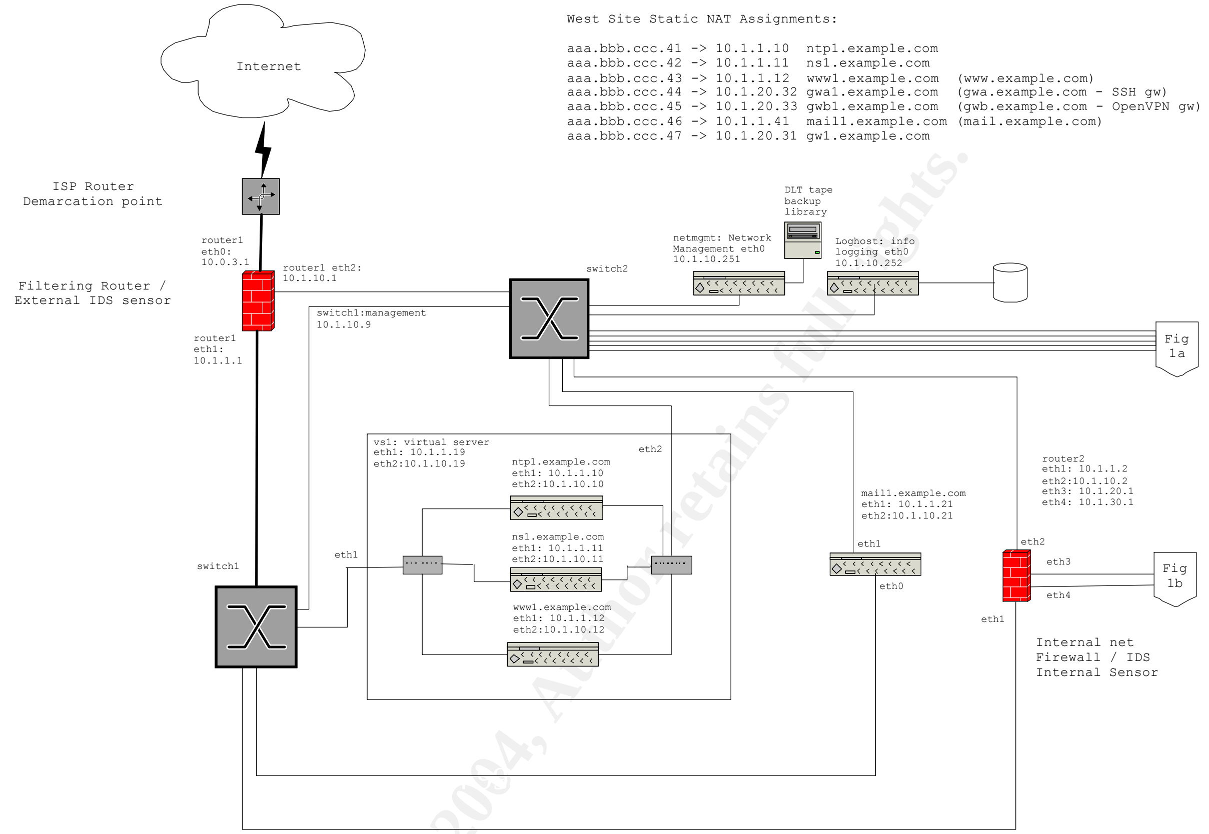


Figure 1: GIAC Enterprises West Site Internet Connection and network (continued)

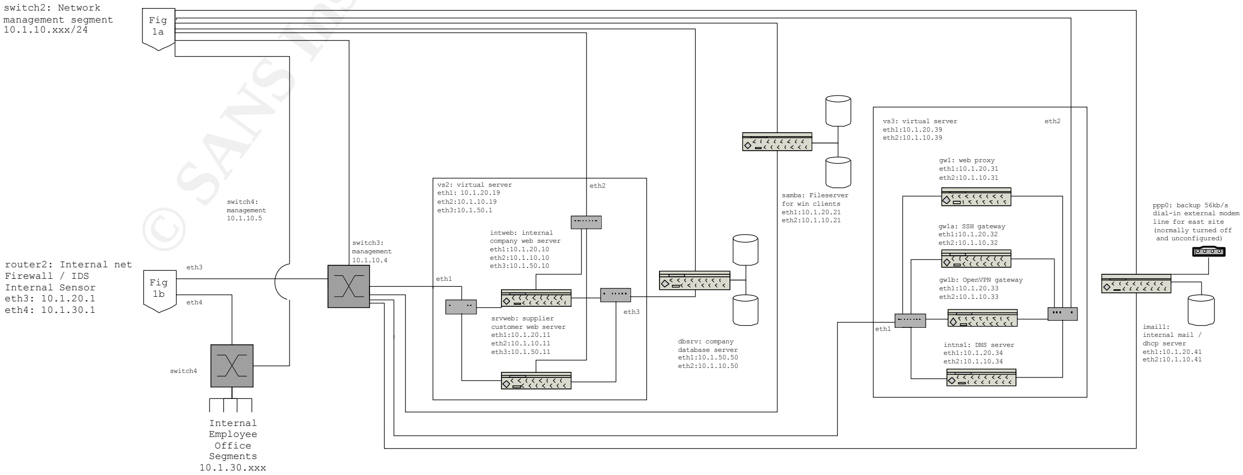
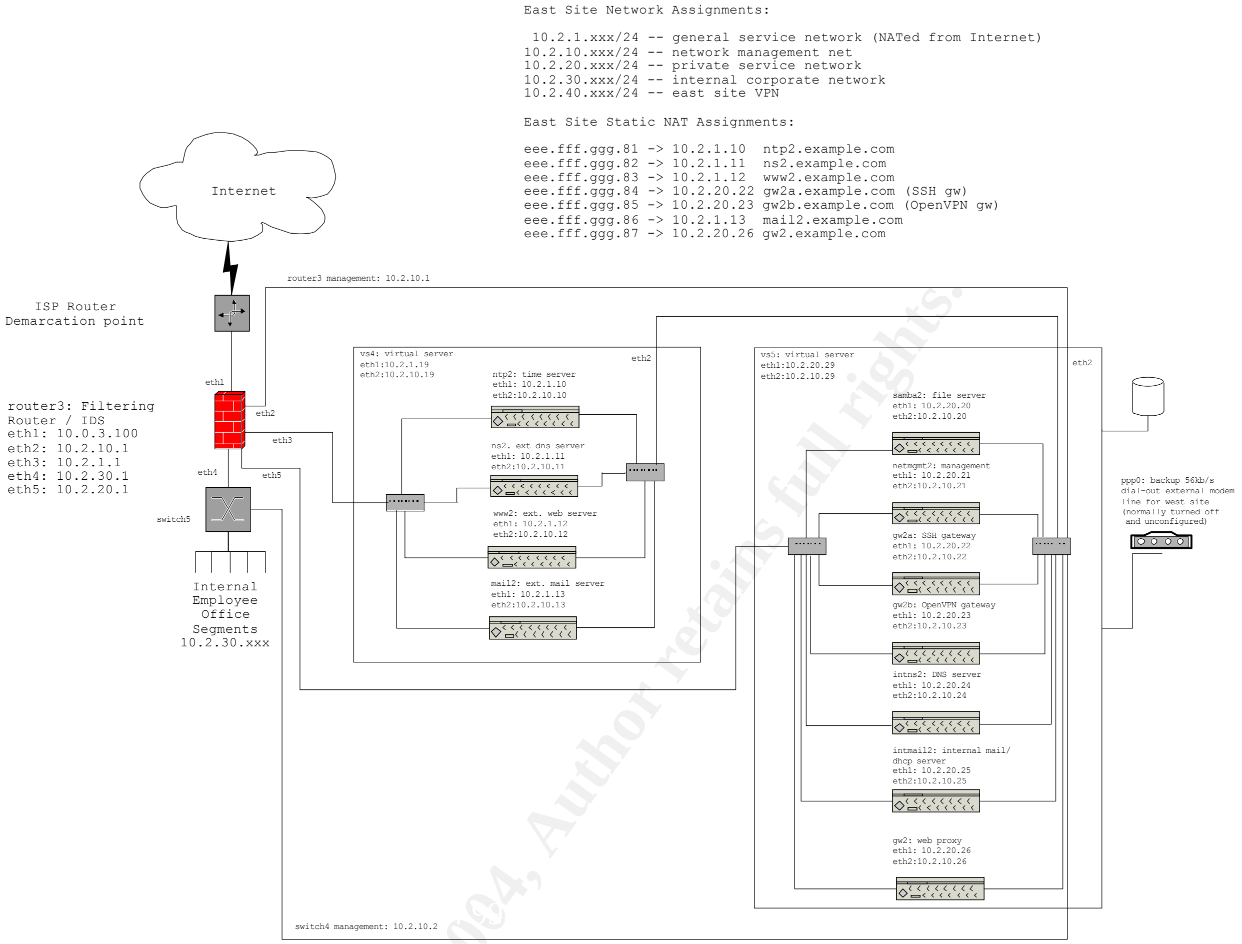


Figure 2: GIAC Enterprises East Site Internet Connection and network



```
East Site Network Assignments:

 10.2.1.xxx/24 -- general service network (NATed from Internet)
10.2.10.xxx/24 -- network management net
10.2.20.xxx/24 -- private service network
10.2.30.xxx/24 -- internal corporate network
10.2.40.xxx/24 -- east site VPN

East Site Static NAT Assignments:

eee.fff.ggg.81 -> 10.2.1.10  ntp2.example.com
eee.fff.ggg.82 -> 10.2.1.11  ns2.example.com
eee.fff.ggg.83 -> 10.2.1.12  www2.example.com
eee.fff.ggg.84 -> 10.2.20.22 gw2a.example.com (SSH gw)
eee.fff.ggg.85 -> 10.2.20.23 gw2b.example.com (OpenVPN gw)
eee.fff.ggg.86 -> 10.2.1.13  mail2.example.com
eee.fff.ggg.87 -> 10.2.20.26 gw2.example.com
```

Note: only security related configuration information is presented in this section. Normal configuration settings for common applications are beyond the scope of this study and can be obtained by referring to the specific application documentation.

Use of Virtual Servers

Depending on the type and usage requirements, a virtual server may be used to host a service instead on an actual hardware platform. Linux has a capability called “User Mode Linux” that allows a kernel to run as a normal user process. Each virtual machine has a separate kernel, network identity, security context, and disk storage system. Since the virtual machines are still under the control of the host system, the host may also enforce additional security mechanisms unknown to them.

The UML package consists of two main parts, a “wrapper” that converts the Linux kernel into a user mode program and a kernel patch called “skas” (Single Kernel Address Space). The skas patch is not required, but applying it to the host system's kernel provides significant performance improvements. It also hides some of the visible UML changes from the virtual machines. The home page for

UML is http://user-mode-linux.sourceforge.net.

By using virtual servers, equipment, space and support costs are minimized. Each virtual server platform is described in more detail below.

West Site Virtual Server Platforms vs1, vs2, and vs3

There are three systems located at the west site that are used to host virtual machines. They are called vs1, vs2, and vs3. All three systems run the Fedora Core-1 Linux distribution. The hardware platform for each is configured based on the type and number of virtual servers loaded onto it. The vs1 server hosts the small public services for the west site (ntp, ns, and www). It is a Dell PowerEdge 1650 dual Xeon 1.2GHz, 512mb ram, two 10/100 Ethernet NICs, and 40 GB local disk.

The vs2 server is also a Dell PowerEdge 1650 dual Xeon 1.2GHz, 512mb ram, three 10/100 Ethernet NICs, and 40 GB local disk. The vs2 server hosts the internal corporate and partner web servers. It is a separate system from vs1 for security purposes.

The vs3 server hosts the network security gateways and associated services (web proxy, SSH & VPN gateways, and internal DNS server). It is also a Dell PowerEdge 1650 dual Xeon 1.2GHz, 512mb ram, two 10/100 Ethernet NICs, and 40 GB local disk.

East Site Virtual Server Platforms: vs4 and vs5

The east site vs4 virtual server hardware configuration is identical to the west site vs1 virtual server. The justification and considerations are also the same, with a single exception. In addition to the ntp2, ns2, and www2 backup servers, vs4 also hosts the general mail backup server (mail2).

The east site server vs5 is a Linux system running the Fedora Core-1 distribution with a custom 2.4 kernel. The hardware platform is a Dell PowerEdge 2650 dual Xeon 2.4 GHz, 512mb ram, two 10/100 Ethernet NICs, and 120 GB local disk.

The justifications and considerations are the same as those for the vs3 corporate server, samba1 file server, and intmail1 internal mail systems located at the west site. To simplify physical and support issues, all of these services have been consolidated onto a single faster system.

There is also a 56kb modem connected to vs5.example.com. This modem is present solely as a temporary communication link between the west and east sites of the GIAC network in case the main Internet connection is unavailable. The modem is normally turned off and is not configured into the system. It must be manually configured by GIAC technical staff before use and should be tested periodically.

West site router: router1 (perimeter screening router / External IDS sensor)

The west site screening router "router1" runs the Fedora Core-1 Linux distribution. The hardware platform is a Dell PowerEdge 1650 dual Xeon 1.2GHz, 512mb ram, three 10/100 Ethernet NICs, and 40 GB local disk. The purpose of this screening router is to protect GIAC Enterprise's computer resources from unauthorized access or modification from hosts on the Internet.

A Linux system was chosen instead of a commercial solution like a Cisco router and firewall combination for several reasons: performance, price, and flexibility. The limiting factor for performance of the screening router is the amount of bandwidth available on the Internet interface. Since the Internet connection is currently a single T1 (1.54mb/s), the specified system is more than capable of handling the network load. The specified Linux system costs more than a minimal commercial solution, but provides significant room for growth. Also, by using the Linux system, we are able to configure the system with capabilities that may be additional cost otherwise. By using open source, we are able address security issues as they are discovered in a timely manner without having to wait for a vendor response.

One of the key functions that all of the routers must provide is the filtering out invalid packets from the data stream entering and leaving the GIAC network. The book "Linux Firewalls, Robert L. Ziegler, 2nd edition, New Riders Publishing, 2001, page 36-37" contains a very complete listing of the types of invalid packets that should be filtered out of the data stream.

Filtering requirements to handle Source address spoofing and Illegal addresses (extracted with modifications from Linux Firewalls 2nd edition by Robert Ziegler)

- **Your IP address.** In general, the IP address for the router itself should be filtered out of the network traffic, but since the router does not have IP addresses assigned to its Internet visible interfaces, this is done be default.
- **Your LAN Addresses.** Since the GIAC network architecture only provides a single connection to the Internet from each location, the GIAC assigned network addresses should never appear as a source address on incoming packets. Note: while there is an emergency backup dial-up connection capability between the internal GIAC mail servers, this link is strictly point-to-point, no general Internet traffic will be routed over the link. Since no routable traffic will be sent over this link, our LAN addresses cannot appear as a source address for an incoming packet. Therefore, any incoming packets with our LAN addresses as its source should be considered illegal and dropped.
- **Class A, B, and C private IP addresses.** IANA has defined several network address blocks for private use. Because these addresses are private, they are not routable on the Internet and should not appear as a source address on inbound traffic. The address blocks that are defined as private are: 10.0.0.0 – 10.255.255.255 (class A), 172.16.0.0. - 172.31.255.255 (class B), and 192.168.0.0 – 192.168.255.255 (class C). For filtering purposes, we consider

unassigned and reserved IANA address blocks as private and will therefore drop inbound packets using them as a source address.

- **Class D multicast IP addresses.** Network address blocks 224.0.0.0 – 239.255.255.255 have been allocated for use as destinations for multi-point audio and video broadcasting. Since GIAC Enterprises does not broadcast any multi-point content, these address blocks should not appear as a source address.
- **Class E reserved IP addresses.** Network block 240.0.0.0 – 247.255.255.255 (some sources define the upper bound to be 255.255.255.255) is reserved for future and experimental use by IANA. They should not appear as source address on any valid traffic that GIAC Enterprises could reasonably expect to receive.
- **Loopback Interface addresses.** Network block 127.0.0.0/8 has been defined by IANA for use as “loopback” interface addresses. A loopback interface is a mechanism that allows computer systems to declare internal network interfaces for use by applications located on the same computer. All loopback traffic is internal to a single computer and therefore should not appear on inbound Internet packets.
- **Malformed broadcast addresses.** The special broadcast address 0.0.0.0 is used by DHCP clients and servers as part of the Dynamic host configuration protocol. It should not appear anywhere else as a source address.
- **Class A network 0 addresses.** Since the special broadcast address 0.0.0.0 described above is defined as network block 0.0.0.0/8, all addresses in the range 0.0.0.0 – 0.255.255.255 are also invalid as source addresses.
- **Link Local network addresses.** Addresses in the range 169.254.0.0 – 169.254.255.255 are assigned by DHCP clients when they are unable to obtain a valid address from a DHCP server. Valid packets should not contain a source address from this block, since it indicates a failure of the GIAC Enterprise's DHCP server.
- **TEST-NET addresses.** Network block 192.0.2.0 – 192.0.2.255 is reserved for test networks and is unused by GIAC Enterprises.

In addition to filtering packets based on IP source addresses, it is also the responsibility of router1 to provide network address translation (NAT) for the west site hosts. NAT provides a conversion mapping between internal hosts and their external IP addresses.

To the extent possible, all GIAC servers should have only a single protocol port visible from the Internet. A notable exception is for web servers that provide both normal and secure HTTP services. Limiting the number of ports will make the downloading of malicious software, like rootkits or spam relays, more difficult if a server is compromised.

Table 3 contains the publicly available network connections allowed by router1 to users located outside of the west site network.

Rule #	*Source IP*	*Source Port*	*Destination IP*	*Destination Port*	*Action*
1	Any External	Any	aaa.bbb.ccc.41	123/UDP	Allow
2	Any External	Any	aaa.bbb.ccc.42	53/UDP	Allow
3	Any External	Any	aaa.bbb.ccc.43	80/TCP	Allow
4	Any External	Any	aaa.bbb.ccc.43	443/TCP	Allow
5	Any External	Any	aaa.bbb.ccc.44	22/TCP	Allow
6	eee.fff.ggg.85	Any	aaa.bbb.ccc.45	5000/UDP	Allow
7	Any External	Any	aaa.bbb.ccc.46	25/TCP	Allow
8	aaa.bbb.ccc.47	Any	Any External	80/TCP	Established
9	aaa.bbb.ccc.47	Any	Any External	443/TCP	Established

Table 3:router1.example.com network connections

Access to the internal systems from outside the corporate firewall, is only possible by using VPN tunnels or SSH terminal sessions/tunnels. For example, an employee connecting from outside of the corporate network would need to use a SSH or VPN tunnel to connect to the email server to send or retrieve his messages.

West Site Router: router2 (interior router / IDS)

The west site interior router “router2” also runs the Fedora Core-1 Linux distribution. The hardware platform is a Dell PowerEdge 1350, 850Mhz Pentium III, 128mb ram, three 10/100 Ethernet NICs, and 40 GB local disk. The purpose of this router is to further isolate GIAC Enterprise's core computer resources from unauthorized access or modification from GIAC public network resources. Router2 also provides an added layer of protection for internal hosts from Internet in case the perimeter router (router1) fails.

This system will also function as an Intrusion Detection System (IDS) to monitor for unauthorized access attempts to the critical corporate resources for GIAC. The software package chosen for the IDS is the Snort open source project. Snort was originally written by Marty Roesch and is hosted at http://www.snort.org. There is detailed documentation contained in the Snort distribution as well as a new book. The book “Snort 2.0: Intrusion Detection” is written by Brian Caswell, Jay Beale, James Foster, Jeffrey Posluns, and others, 2003, Syngress Publishing Inc.

Table 4 contains the allowed network connections between the internal service and employee segments and the DMZ network & the Internet.

Rule #	***Source IP / Interface***	***Source Port***	***Destination IP / Interface***	***Destination Port***	***Action***
1	Any	Any	eth0 (input)	22/TCP	Allow
2	10.1.10.251	Any	10.1.10.1	22/TCP	Allow
3	10.1.10.1	Any	10.1.10.252	514/UDP	Allow
4	10.1.0.0/16	Any	10.1.10.252	514/UDP	Allow
5	10.1.30.0/24	Any	10.1.20.33	Any	Allow
6	10.1.20.33	Any	10.1.30.0/24	Any	Allow
7	10.1.20.33	Any	eee.fff.ggg.85	Any	Allow
8	eee.fff.ggg.85	Any	10.1.20.33	Any	Allow
9	10.1.10.251	Any	eee.fff.ggg.85	Any	Allow
10	10.1.30.0/24	Any	10.1.20.21	139,445/TCP	Allow
11	10.1.30.0/24	Any	10.1.20.21	137,138/UDP	Allow
12	10.1.20.31	Any	Any External	80,443/TCP	Allow
13	Any External	Any	10.1.20.32	22	Allow
14	10.1.30.0/24	Any	10.1.20.32	22	Allow
15	10.1.20.0/24	Any	10.1.1.10	123/UDP	Allow
16	10.1.30.0/24	Any	10.1.1.10	123/UDP	Allow
17	10.1.20.34	Any	10.1.1.11	53/TCP,UDP	Allow
18	10.1.1.21	Any	10.1.20.41	25/TCP	Allow
19	10.1.20.41	Any	10.1.1.21	25/TCP	Allow
20	Any	Any	Any	Any	Deny

Table 4: router2 network connections

Rules 1 and 2 allows the direct connect console or anyone using the network management workstation to connect to the SSH port of the router. Rules 3 and 4 allows any internal system including router2 to send messages to the logging host. Rules 5 and 6 allows traffic to pass between any system on the corporate network segment and the VPN gateway. Rules 8 and 9 allows traffic to pass between the west and east VPN gateways.

Rules 10 and 11 allows traffic to pass between the corporate network and the west site file server.

Rule 12 allows the web proxy to send traffic requested by systems on the corporate network segment to external Internet web servers. Rules 13-14 allows anyone to connect to the west site SSH gateway. Rules 15 and 16 allows systems on the internal service and corporate networks to connect to NTP time server.

Rule 17 allows DNS requests from the internal DNS server to be forwarded to our external DNS server. Rules 18 and 19 allow e-mail messages to pass between our internal and external mail servers.

The final rule sets an explicit rule to drop all other packets.

East Site Router: router3 (perimeter screening router / external IDS sensor)

The east site filtering & interior router “router3” runs the Fedora Core-1 Linux distribution. The hardware platform is a Dell PowerEdge 1350, 850Mhz Pentium III, 128mb ram, three 10/100 Ethernet NICs, and 40 GB local disk.

The purpose of this router is to protect GIAC Enterprise's east site computer resources from unauthorized access or modification from hosts on the Internet. Since the critical corporate assets are located elsewhere, it’s an acceptable risk to have a single router managing resources for the east site.

This system will also function as an Intrusion Detection System (IDS) to monitor for unauthorized access attempts to the east site. As with the west site, the snort software package configured in IDS mode will be used.

Rule #	***Source IP / Interface***	***Source Port***	***Destination IP / Interface***	***Destination Port***	***Action***
1	Any	Any	eth0 (input)	22/TCP	Allow
2	10.2.10.21	Any	10.2.10.1	22/TCP	Allow
3	10.2.10.1	Any	10.1.10.21	514/UDP	Allow
4	10.2.0.0/16	Any	10.1.10.21	514/UDP	Allow
5	10.2.30.0/24	Any	10.2.20.23	Any	Allow
6	10.2.20.23	Any	10.2.30.0/24	Any	Allow
7	10.2.20.23	Any	aaa.bbb.ccc.45	Any	Allow
8	aaa.bbb.ccc.45	Any	10.2.20.23	Any	Allow
9	10.2.10.21	Any	aaa.bbb.ccc.45	Any	Allow
10	10.2.30.0/24	Any	10.2.20.20	139,445/TCP	Allow
11	10.2.30.0/24	Any	10.2.20.20	137,138/UDP	Allow
12	10.2.20.26	Any	eth1 (forward)	80,443/TCP	Allow

Rule #	***Source IP / Interface***	***Source Port***	***Destination IP / Interface***	***Destination Port***	***Action***
13	10.2.20.22	Any	Any External	22	Allow
14	10.2.30.0/24	Any	10.2.20.22	22	Allow
15	Any External	Any	10.2.20.22	22	Allow
16	Any	Any	10.2.1.10	123/UDP	Allow
17	10.2.1.10	Any	Any External	123/UDP	Allow
18	10.2.1.11	Any	Any External	53/TCP,UDP	Allow
19	Any External	Any	10.2.1.11	53/UDP	Allow
20	10.2.20.24	Any	10.2.1.11	53/TCP,UDP	Allow
21	Any External	Any	10.2.1.13	25/TCP	Allow
22	10.2.1.13	Any	10.2.20.25	25/TCP	Allow
23	10.2.20.25	Any	10.2.1.13	25/TCP	Allow
24	Any	Any	Any	Any	Deny

Table 5: router3 network connections

Rules 1 and 2 allows the direct connect console or anyone using the network management workstation to connect to the SSH port of the router. Rules 3 and 4 allows any internal system including router3 to send messages to the logging host. Rules 5 and 6 allows traffic to pass between any system on the corporate network segment and the VPN gateway. Rules 7 and 8 allows traffic to pass between the west and east VPN gateways. Rule 9 allows the east network management host to access the east VPN gateway.

Rules 10 and 11 allows traffic to pass between the corporate network and the east site file server.

Rule 12 allows the web proxy to send traffic requested by systems on the corporate network segment to external Internet web servers. Rules 13-15 allows anyone to connect to the east site SSH gateway. Rule 16 allows anyone to connect to the east site NTP time server. Rule 17 allows the NTP server to connect to any time server on the Internet.

Rule 18 allows DNS requests from the external DNS server to be forwarded to a DNS server on the Internet. Rule 19 allows any Internet host to issue udp DNS requests to our external DNS server. Rule 20 allows our internal DNS server to issue both udp and tcp requests to the external DNS server.

Rule 21 allows incoming e-mail to be sent to our external mail server. Rules 22 and 23 allow e-mail messages to pass between our internal and external mail servers.

The final rule sets an explicit rule to drop all other packets.

Network Time (NTP) servers: ntp1.example.com and ntp2.example.com
west: ntp1 (virtual, hosted on vs1), east: ntp2 (virtual, hosted on vs4)

The purpose of the network time server is to provide a reliable time base to which all of the other systems in the GIAC network can synchronize their system clocks. A synchronized time base is particularly useful when correlating logs from multiple systems.

The time server software used is the reference implementation provided by the Network Time Protocol (NTP) project (http://www.ntp.org). The software runs inside of a basic UML instance. The only network port available from the time server visible to the Internet will be 123/UDP. All connections will be read-only with the exception of the NTP servers at other GIAC sites and the external reference time source. The external source is a trusted remote site that has agreed to act as our reference source.

The network time servers are specifically placed and configured so that employees and partners not directly connected to the GIAC internal network may use the same time source as the corporate resources. This means that the general public may also access this resource. Since the NTP software has a large user base, is actively maintained, and access is read-only, the security vulnerabilities are minimal.

External DNS server ns1.example.com and ns2.example.com
west: ns1 (virtual, hosted on vs1), east: ns2 (virtual hosted on vs4)

The purpose of the domain name system server is to provide Internet users a directory lookup of the names and IP numbers of the publicly available services on the GIAC network.

The DNS software used is the Berkeley Internet Name Domain (BIND) implementation provided by ISC (the Internet Software Consortium), home page: http://www.isc.org/products/BIND. The only network port available from the external DNS servers visible to the Internet is 53/UDP. All connections will be read-only. While we could enable zone transfers between the GIAC sites, doing so would open an unnecessary attack vector. Since the public DNS entries are static and small in number, any changes can be done manually on each of the name servers. In all probability, the public name server entries will only change if the IP numbers assigned to GIAC change.

The root file-system for the BIND instance is the basic GIAC standardized UML & busybox file-system with the BIND software package (static-linked) installed and configured. This file-system is created and configured on the network management station and downloaded to the virtual server (vs1.example.com) over the maintenance network.

The DNS servers are specifically placed and configured such that general Internet users may use them. While there has been several severe security vulnerabilities discovered with BIND, they are fixed quickly. Because of this history and BIND's monolithic design, it is important that the GIAC technical staff be aware of new vulnerabilities as they are announced, so corrective action may be taken.

Only information for the hosts found in Table 1 (External Host Assignments) will be available from the ns1 name server. In addition to the individual “A” (address) records for each host, a “CNAME” (canonical name) record that points the generic service name to the appropriate server, and the reverse pointer “PTR” records will also be present. Recursive DNS requests will be rejected.

Information regarding the configuration of BIND and the syntax of the host data can be found on the project's home page or in the documentation and examples included in the software distribution.

Public Web server: www.example.com.
west: www1 (virtual, hosted on vs1), east: www2 (virtual, hosted on vs4)

The public web servers are some of the most highly visible systems on the GIAC network. Because of this, they should be considered high-risk candidates for compromise attempts by malicious Internet users. To counter these risks, the type of content available on the web servers as well as the selection of their software components is critical.

The type of content available from the public web server will be limited to static HTML pages and one HTML “post” type application. The application will accept a simple text message (no attachments allowed) and forward it to the Sales staff's information request mailbox.

The web server software chosen is the Apache Software Foundation's HTTP server version 1.3.29 (home page: http://httpd.apache.org). Apache is the number one HTTP server in use on the Internet and has a very good security track record.

Care must be taken with handling data submitted by untrusted users. Any text sent by the user must be considered “tainted” and validated by the mail submission processor before being sent to the sales staff's information mailbox. There are several ways this validation may be accomplished. The chosen method is to use the PHP version 4.3.4 preprocessor developed by the PHP Project (home page: http://www.php.net). PHP was chosen for two main reasons. First, it is able to incorporate high-level language type constructs (variables, control structures, database access, etc.) into HTML pages using special HTML tags. The second reason is that it can be incorporated transparently into the Apache web server as a compiled module (mod_php). By configuring the Apache server to automatically process every HTML page with PHP, input data can be processed without having to use a separate CGI (common gateway interface) program. Not having to use a separate CGI program reduces the avenues of attack towards the web server. Also, Apache will be configured to obfuscate its version and PHP will have the ?phpinfo feature disabled. These two configuration changes are not intended as an effective security measure, only to help discourage “script kiddies”.

GIAC Corporate policy is to use secure methods for the submission of informational requests from the public web site. Most, if not all, modern browsers support a standard method of secure communication called “Secure Sockets Layer” (SSL). SSL uses strong public key encryption with encryption key certificates. Both Apache (by the mod_ssl module) and PHP provide support for a common SSL implementation called OpenSSL. OpenSSL is developed and maintained by the OpenSSL project (home page: http://www.openssl.org). GIAC Enterprises has obtained a signed SSL certificate from Verisign Inc. (home page: http://www.verisign.com).

External E-mail Server: mail.example.com.
west: mail1 (dedicated host), east: mail2 (virtual, hosted on vs4)

The west site electronic mail server “mail1” is a Linux system running the Fedora Core-1 distribution. The hardware platform is a Dell PowerEdge 1350, 850Mhz Pentium III, 128mb ram, two 10/100 Ethernet NICs, and 40 GB local disk. The purpose of this server is to isolate and protect GIAC Enterprise's internal e-mail system when accepting messages from users on the Internet.

This isolation affects both inbound and outbound messages. Inbound messages are scanned for known virus signatures as well as “spam” content. Any message that contains non-text characters or more than a single zip file attachment will be rejected with an error message. Outbound messages are also “sanitized” by deleting header information that could be used to map the internal GIAC network. GIAC corporate email addresses are rewritten into a standard external form to hide any user specific login and host information.

The configuration and selection of the software that performs the scanning of incoming messages is based on the Freshmeat Tutorial “Configuring an Open Source Mail Gateway” by David Handermann. This tutorial was posted May 31, 2003 on the Freshmeat web site. A direct link to this tutorial is: http://freshmeat.net/articles/view/857.

The software packages required to implement Handermann's mail gateway on GIAC's external e-mail server are:

- AMaViS (A Mail Virus Scanner), home page: http://www.amavis.org. AMaViS is a Perl script which provides a framework to interface an email server and an anti-virus scanner. There are several different versions of AMaViS available. The version used by the email gateway is AMaViS-new which supports the use of SpamAssassin.

- SpamAssassin, by the SpamAssassin development team, home page: http://www.spamassassin.org. SpamAssassin uses several different methods to analyze the contents of an email message to determine if it is spam or not. It is written in Perl and uses MIMEDefang to assist in processing message attachments.
- MIMEDefang by Roaring Penguin Software Inc., home page: http://www.roaringpenguin.com/products/mimedefang. This Perl module is used to process and optionally convert email MIME attachments.
- Clam AntiVirus, by the Clam AntiVirus development team, home page: http://www.clamav.net.
- Postfix email system, by Wietse Venema, home page: http://www.postfix.org.

Note: Perl is a widely-used scripting language written by Larry Wall. The first version was released in 1987. The current version is 5.8.2. More information on Perl can be found at http://www.perl.org.

Network Management Hosts
west: netmgmt1 (dedicated host), east: netmgmt2 (virtual, hosted on vs5)

The west site network management server netmgmt is a Linux system running the Fedora Core-1 distribution with a custom 2.4 kernel. The hardware platform is a Dell PowerEdge 1650 dual Xeon 1.2GHz, 512mb ram, two 10/100 Ethernet NICs, and 40 GB local disk. The purpose of this system is to provide a fast, reliable, and secure platform to allow GIAC technical staff to manage the corporate network resources. This includes backing up the network servers and building the file-systems for the user mode Linux servers (i.e. ntp1, ns1, gw1b, etc).

To provide for disaster recovery, selected data is compressed and sent via the VPN gateways to the east site network management system as an easy accessible online backup. The system running the main database (dbsrv1) uses this method in addition to tape backups. The database is dumped periodically and stored on the remote network management system, in case of a major failure of the west site network. To minimize the amount of network bandwidth required to copy this information, the open source “rsync” file transfer utility written by Andrew Tridgell, home page: http://rsync.samba.org.

Network Information Logging Hosts
west: loghost1 (dedicated host), east: loghost2 (virtual, hosted on vs5)

The west site network information logging host “loghost1” is a Linux system running the Fedora Core-1 distribution with a custom 2.4 kernel. The hardware platform is a Dell PowerEdge 1350, 850Mhz Pentium III, 128mb ram, a single10/100 Ethernet NIC, and 160 GB local disk.
The purpose of this host is capture information produced by the other systems located on the GIAC network. This information may include system and application messages as well as audit trails from the web servers used by internal GIAC and partner users. The audit trails from intweb1 and srvweb1 are also sent via rsync to the east site network management server.

Internal Web Server
west: intweb1 (virtual, hosted on vs2), east: none

The purpose of the internal web server is to provide web services to the employees of GIAC Enterprises. Web applications control employee read and write access to the fortune cookie database. A transaction report (audit trail) for each access to the database will be sent to the logging host.

The web server software chosen is the Apache Software Foundation's HTTP server version 1.3.29 (home page: http://httpd.apache.org). Apache is the number one HTTP server in use on the Internet and has a very good security track record.

Care must be taken with handling any data submitted (even by trusted users). Any text sent by the user must be considered “tainted” and validated by the web application before being processed.

The PHP version 4.3.4 preprocessor developed by the PHP Project (home page: http://www.php.net) will also be available for application development. A self-signed SSL certificate will also be used. Except for the above changes, the same justifications as described for the general purpose web server (www1.example.com) apply.

Because of the small amount of resources required to run the internal web server, it can easily run inside a virtual machine. The root file-system for the internal web server is the basic GIAC standardized UML & busybox file-system with the apache web server, mod_php, and mod_ssl components installed. The GIAC signed SSL certificate and the HTML document contents are also included in this file-system.

Partners Web Server
west: srvweb1 (virtual, hosted on vs2), east: none

The purpose of srvweb1 is to provide web services to external suppliers and partners of GIAC Enterprises. Web applications control all read and write access to the fortune cookie database. The type of content available from the unsecured web port (80/TCP) will be limited to a static HTML page which loads the SSL enabled application page on port 443/TCP.

The web server software chosen is the Apache Software Foundation's HTTP server version 1.3.29 (home page: http://httpd.apache.org). Apache is the number one HTTP server in use on the Internet and has a very good security track record.
Care must be taken with handling any data submitted (even by trusted users). Any text sent by the user must be considered “tainted” and validated by the web application before being processed.

The PHP version 4.3.4 preprocessor developed by the PHP Project (home page: http://www.php.net) will also be available for application development. A SSL certificate signed by Verisign Inc. will be used for secure communications. Note: this certificate is different from the one used by the general web server www1. Except for the above changes, the same justifications as described for the general purpose web server (www1.example.com) apply.

Because of the small amount of resources required to run the intweb1 web server, it can easily run inside a UML virtual machine. The root file-system for the intweb1 instance is the basic GIAC standardized UML & busybox file-system with the apache web server, mod_php, and mod_ssl components installed. The GIAC signed SSL certificate and the HTML document contents are also included in this file-system. This file-system is created and configured on the network management station and downloaded to the virtual server (vs2.example.com) over the maintenance network.

Data Base Server
west: dbsrv1 (dedicated host), east: none

The west site database server dbsrv1 is a Linux system running the Fedora Core-1 distribution with a custom 2.4 kernel. The hardware platform is a Dell PowerEdge 1650 dual Xeon 1.2GHz, 512mb ram, two 10/100 Ethernet NICs, and 380 GB local RAID disk.

The purpose of this system is to provide a fast, reliable and secure platform to process database queries for GIAC employees and partners connecting via secure Web servers.

Because of the sensitive nature of the data stored on this server, there will be no connections allowed to dbsrv1 except for those necessary to issue database queries and to maintain the server.

Network File Server
west: samba1 (dedicated host), east: samba2 (virtual, hosted on vs5)

The west site file server (samba1) is a Linux system running the Fedora Core-1 distribution with a custom 2.4 kernel. The hardware platform is a Dell PowerEdge 1650 dual Xeon 1.2GHz, 512mb ram, two 10/100 Ethernet NICs, and 320 GB local RAID disk. The purpose of this system is to provide a fast, reliable and secure platform to store work related files for GIAC employees.

To provide file service for employee desktop/laptop systems running Microsoft Windows, the open source Samba SMB/CIFS file and print server software has been installed. Samba is developed and maintained by the Samba Team, home page: http://samba.org. Chapter 15 (Securing Samba) of the Samba documentation (written by Andrew Tridgell and John H. Terpstra), home page:

http://de.samba.org/samba/docs/man/securing-samba.html contains several configuration recommendations for securing the Samba file server.

SSH gateway: gw1a.example.com and gw2a.example.com
west: gw1a (virtual, hosted on vs3), east: gw2a (virtual, hosted on vs5)

The purpose of the SSH gateway is provide employees access to the internal GIAC corporate network from remote sites on the Internet. SSH provides a secure data stream between the SSH the gateway and the employee's computer system. The employee uses an SSH capable client (i.e. Putty, winscp3, etc) to connect to the SSH gateway server.

The SSH server software used is the OpenSSH portable reference implementation provided by OpenBSD Project, home page: http://www.openssh.org. The software runs inside of a basic UML instance. The only network port available from this UML instance visible to the Internet is 22/TCP.

The root file-system for the SSH instance is the basic GIAC standardized UML & busybox file-system with the OpenSSH server package (static-linked) installed and configured. This file-system is created and configured on the network management station and downloaded to the virtual server (vs3.example.com) over the maintenance network.

The SSH server is specifically placed and configured such that general Internet users may use it. While there has been several security vulnerabilities discovered with OpenSSH, they are fixed quickly. Because of the critical nature of the SSH gateway, it is important that the GIAC technical staff be aware of new vulnerabilities as they are announced, so corrective action may be taken.

VPN gateway: gw1b.example.com and gw2b.example.com
west: gw1b (virtual, hosted on vs3), east: gw2b (virtual, hosted on vs5)

The purpose of the VPN gateway is to allow remote sites and teleworker employees secure access to the internal GIAC corporate network. VPN connections are used for long-term connections, (i.e. remote sites and teleworker's homes). For transient use, the SSH server should be used instead. The OpenVPN package was chosen over other VPN solutions like IPSEC and PPTP for two main reasons: OpenVPN does not require operating system modifications and there are versions available for both Window and various UNIX/Linux platforms.

The OpenVPN software runs inside of a UML instance. Since OpenVPN uses normal user-mode TCP/UDP sockets, we can use normal firewall and routing methods to control the visibility of the VPN gateway to the Internet. The root file-system used for the VPN gateway instance is the basic GIAC standardized UML & busybox file-system with the OpenVPN package (static-linked) installed and configured. This file-system is created and configured on the network management station and downloaded to the virtual server (vs3.example.com) over the maintenance network.

Because of the critical nature of the VPN gateway, it's important that the GIAC technical staff become aware of any program vulnerabilities as they are announced, so corrective action may be taken.

Internal DNS Servers
west: intns1 (virtual, hosted on vs3), east: intns2 (virtual, hosted on vs5)

The purpose of the internal domain name system servers are to provide GIAC corporate users a directory lookup service of the names and IP numbers of all of the GIAC network systems as well as any host on the Internet.

The DNS software used is the Berkeley Internet Name Domain (BIND) implementation provided by ISC (the Internet Software Consortium), home page: http://www.isc.org/products/BIND. The software runs inside of a basic UML instance.

Note: Name server requests received by intns1 that require lookups from external name servers are forwarded to ns1.example.com for resolution.

Internal E-mail servers
west: intmail1 (dedicated host), east: intmail2 (virtual, hosted on vs5)

The west site electronic mail server "intmail1" is a Linux system running the Fedora Core-1 distribution. The hardware platform is a Dell PowerEdge 1350 850Mhz Pentium III, 128mb ram, two 10/100 Ethernet NICs, and 40 GB local disk. The purpose of this server is to send and receive e-mail messages for GIAC employees.

While the public e-mail servers provide both spam and virus checking on all e-mail messages, the internal server will still perform the virus checking. This is to mitigate the risk of a virus infection that may originate inside of the company firewall. For example, an employee may accidentally use an infected floppy disk to transport data between their home and office computers. If the internal network did not do this additional checking, systems on the company network may also become infected.

There is also a 56kb modem connected to intmail1.example.com, this modem is intended solely as a temporary backup communication link between the west and east sites of the GIAC network. For example, assume that the Internet connection to the east site becomes unavailable because of a cable cut. The modem could be used to send internal e-mail and data while the cable is repaired.

The modem is normally turned off and is not configured into the system. The modem must be manually configured by GIAC technical staff before use and during periodic testing.

Web Proxy Servers: gw1.example.com and gw2.example.com
west: gw1 (virtual, hosted on vs3), east: gw2 (virtual, hosted on vs5)

The purpose of the web proxy gateway is to provide employees protected access to web sites on the Internet. The web proxy provides an application level gateway for internal users. The employee uses a regular web browser (i.e. Mozilla Firebird) that has been configured to use gw1.example.com as a proxy for http and https traffic. Internal users will not be able to connect directly to the Internet.

The web proxy uses the Apache HTTP server and the mod_security (http://www.mod_security.org) module to provide proxy services. The configuration for the proxy software is beyond the scope of this study.

The root file-system for the gw1 instance is the basic GIAC standardized UML & busybox file-system with the web proxy software (static-linked) installed and configured. This file-system is created and configured on the network management station and downloaded to the virtual server (vs3.example.com) over the maintenance network.

The web proxy server is specifically placed and configured such that internal users must send their web traffic through the proxy. The proxy is also placed on the internal network segment so it is protected from direct attack from the Internet.

Network Infrastructure Components: switches

There are numerous network switches placed throughout the GIAC corporate network. These switches control the flow of data between the various network segments. GIAC Enterprises has standardized on the Hewlett-Packard “HP Procurve 2524” switches. For a modest cost (< $1000 retail list price) the 2524 switch with 10/100 transceiver, provides management capability as well as supporting SSH connections for monitoring and configuration.

For the west and east site, there are four 2524 switches in the network. Switch1 is used to direct traffic for the general service network at the west site. Switch2 controls the west site network management segment. Switch3 controls the west site internal service network. Switch4 controls the west site employee office segments.

Since the sales offices are physically smaller and have less resource needs, only a single switch is needed to control the employee office segments.

Assignment 2
Security Policy and Tutorial

The following section provides a detailed implementation and description for each of the devices used to protect the perimeter of GIAC Enterprises computer network. The described devices include the border router and primary firewalls for the west & east sites as well as the VPN configuration for the secure network tunnel between them. A detailed tutorial describing the installation, configuration and validation of the west-east VPN tunnel is also included.

Common Rules for All GIAC Firewalls

All of the GIAC Firewalls are based on IBM-PC compatible hardware running the Red Hat Fedora Core-1 Linux distribution. The firewall capability of the operating system is based on the iptables (also known as netfilter) architecture. In addition, GIAC has developed a set of firewall configuration settings that must be included in the security policy for each firewall.

Here is a brief description of iptables and its configuration syntax, this information is based on the iptables manual page (i.e. man iptables on Linux). Please refer to the man page for more information.

Iptables provides a set of "chains" whose links contain firewall rules. Each packet that is processed by the firewall is compared against one or more chain links. If a packet matches a link, then the link's target (i.e. action) controls what happens to the packet. The common targets used by the firewalls are:

- ACCEPT: let the packet through the system,
- DROP: do not let the packet through,
- LOG: log the packet information to the SYSLOG port. NOTE: This target will not terminate the chain.
- <chain_name>: if a user defined chain name is specified as a target, it will be used to further classify the packet. NOTE: jumps to user defined chains are non-terminating by default.

There are three predefined chains: INPUT, FORWARD, and OUTPUT. Each of these chains classifies packets based on their source and destination:

- INPUT: a packet has been sent directly to this firewall's IP address.
- FORWARD: a packet is being sent through this system from one interface to another. This chain will handle the filtering of packets between network segments.
- OUTPUT: a packet is leaving this firewall.

A predefined chain may have a default policy applied to it by use of the "`iptables -P <chain> <policy>`" command.

Custom "user defined" chains can be created by using the "`iptables -N <chain>`" command.

Actions are added to a chain by use of "`iptables -A <chain> <options>`" command. The "-A" argument tells the system to append the action to the end of the specified chain.

Some of the commonly used filter options are:

- -s <ip> - specifies a source IP address or network. Format is dotted quad notation and CIDR network value separated by a slash. IP addresses or networks can be negated by placing a exclamation point at the front of the value (i.e. !10.0.0.0/8 - means everything except network 10).
- -d <ip> - specifies a destination IP address or network. Format is the same as "-s".
- -i <int> - input interface.
- -o <int> - output interface.
- -j <action> or -j <chain> - specify the action or alternate chain to be used if the packet matches this rule.
- -p <proto> - specify the protocol to match.
- -m <module> - specify an extension module that provides expanded matching options (i.e. -m state allows manipulation of information necessary for stateful packet inspection).
- --dport <port> - specifies a destination port for tcp and udp packets.
- --sport <port> - specifies a source port for tcp and udp packets.

For more information regarding construction of firewall and nat rules, please refer to chapters 6, 7, and Appendix B in "Linux Firewalls, 2nd Edition.", by Robert Ziegler, New Riders Publishing. This reference source was extensively used in creating the following firewall scripts. The following script (rc.west-fw) fully implements the security policies contained in tables W-1 and W-2.

rc.common-fw file listing

```
#
# rc.common-fw -- rules that are common to all GIAC
# Enterprises network firewalls.
#
# environment variables used in this script:
# OUR_NET -- the CIDR representation of our network (i.e.
#            10.2.0.0/16)
# IF_EXT -- name of the external interface (i.e. facing the
#           internet, i.e. eth1)
# QUENCH_NET -- the CIDR representation of the service network if
#                icmp source quench processing is desired or empty
#                if no processing is to be performed.
#

# Enable broadcast echo protection
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

# disable source routed packets
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
   echo 0 > $f
done
```

```
# enable TCP SYN Cookie protection
echo 1 > /proc/sys/net/ipv4/tcp_syncookies

# disable ICMP redirect acceptance
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
   echo 0 > $f
done

# don't send redirect messages
for f in /proc/sys/net/ipv4/conf/*/send_redirects; do
   echo 0 > $f
done

# drop spoofed packets coming in on an interface, if replied to
# would result in the packet going out on a different interface
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
   echo 1 > $f
done

# log packets with impossible addresses
for f in /proc/sys/net/ipv4/conf/*/log_martians; do
   echo 1 > $f
done

# Delete old iptables configuration
$IPTABLES --flush
$IPTABLES -t nat --flush
$IPTABLES -t mangle --flush

# Allow direct console connection (eth0)
$IPTABLES -A INPUT -i eth0 -j ACCEPT
$IPTABLES -A OUTPUT -o eth0 -j ACCEPT
$IPTABLES -A FORWARD -i eth0 -j ACCEPT

# Allow local loopback
$IPTABLES -A INPUT -i lo -j ACCEPT
$IPTABLES -A OUTPUT -o lo -j ACCEPT

# set default policies
$IPTABLES --policy INPUT DROP
$IPTABLES --policy OUTPUT DROP
$IPTABLES --policy FORWARD DROP
$IPTABLES -t nat --policy PREROUTING ACCEPT
$IPTABLES -t nat --policy OUTPUT ACCEPT
$IPTABLES -t nat --policy POSTROUTING ACCEPT

# delete any user defined chains
$IPTABLES --delete-chain
$IPTABLES -t nat --delete-chain
$IPTABLES -t mangle --delete-chain

# handle stealth scans and bogus flags
for t in INPUT FORWARD; do
   # all bits cleared
   $IPTABLES -A $t -p tcp --tcp-flags ALL NONE -j DROP
   # syn and fin both set
   $IPTABLES -A $t -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
   # syn and rst both set
   $IPTABLES -A $t -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
   # fin and rst  both set
   $IPTABLES -A $t -p tcp --tcp-flags FIN,RST FIN,RST -j DROP
   # fin is the only bit set, without expected ack
   $IPTABLES -A $t -p tcp --tcp-flags ACK,FIN FIN -j DROP
   # psh is the only bit set without expected ack
   $IPTABLES -A $t -p tcp --tcp-flags ACK,PSH PSH -j DROP
   # urg is the only bit set without expected ack
   $IPTABLES -A $t -p tcp --tcp-flags ACK,URG URG -j DROP
done

# allow reply packets for established connections originating
# from inside the perimeter
for t in INPUT OUTPUT FORWARD; do
   $IPTABLES -A $t -m state --state ESTABLISHED,RELATED -j ACCEPT
   $IPTABLES -A $t -m state --state INVALID -j LOG --log-prefix \
        "INVALID $t: "
   $IPTABLES -A $t -m state --state INVALID -j DROP
done

# define rules to identify spoofed packets
# claiming to be from us
$IPTABLES -A INPUT -s $OUR_NET -j DROP
$IPTABLES -A FORWARD -i $IF_EXT -s  $OUR_NET -j DROP
$IPTABLES -A OUTPUT -o $IF_EXT -s ! $OUR_NET -j DROP

# refuse bad broadcast packets
$IPTABLES -A INPUT -i $IF_EXT -d 0.0.0.0 -j DROP
$IPTABLES -A FORWARD -d 255.255.255.255 -j DROP

# icmp messages
for t in INPUT OUTPUT FORWARD; do
$IPTABLES -A $t --fragment -p icmp -j LOG --log-prefix \
     "Fragmented $t ICMP: "
$IPTABLES -A $t --fragment -p icmp -j DROP
$IPTABLES -A $t -p icmp --icmp-type parameter-problem \
     -j ACCEPT
$IPTABLES -A $t -p icmp --icmp-type destination-unreachable \
     -j ACCEPT
done

# handle source quench special, we only allow incoming
# source-quench messages directed to our service network
if [ "X$QUENCH_NET" != "X" ]; then
   $IPTABLES -A INPUT -p icmp --icmp-type source-quench \
        -d $QUENCH_NET -j ACCEPT
fi
$IPTABLES -A OUTPUT -p icmp --icmp-type source-quench -j ACCEPT
$IPTABLES -A FORWARD -p icmp --icmp-type source-quench -j ACCEPT

# filter out outgoing icmp replies that can be used to map the network
$IPTABLES -A OUTPUT -p icmp --icmp-type fragmentation-needed -j ACCEPT
$IPTABLES -A FORWARD -p icmp --icmp-type fragmentation-needed -j ACCEPT
$IPTABLES -A OUTPUT -p icmp --icmp-type destination-unreachable -j DROP
$IPTABLES -A FORWARD -p icmp --icmp-type destination-unreachable -j DROP

# handle internal traceroute/ping requests and responses
$IPTABLES -A INPUT -p icmp --icmp-type time-exceeded -j ACCEPT
$IPTABLES -A OUTPUT -p icmp --icmp-type echo-request -m state \
     --state NEW -j ACCEPT
$IPTABLES -A FORWARD -o $IF_EXT -p icmp --icmp-type echo-request \
123.        -s $OUR_NET -m state --state NEW -j ACCEPT
```

Lines 14-20 disable broadcast echoes and source-routed packets. Lines 21-39 configures the Linux kernel to protect itself from common attacks directed toward router1 itself. Lines 40-43 will clear the previous firewall configuration. Lines 44-50 allow the firewall to access its loopback and direct console interfaces.

Lines 51-54 reset all of the default policies for the firewall to drop packets by default. This approach is better than accepting everything and dropping on specific packets. By dropping packets by default, the possibility of leaving a vulnerability open is minimized. Lines 55-57 configure the chains necessary for the Network Address Translation function to accept packets for processing.

Lines 58-61 delete any user defined chains. We currently do not define any user chains, but we want to make sure that there are not any left from a previous configuration.

Lines 62-78 verify that the TCP packet header flags are valid. These checks defeat most of the attacks caused by nmap and other packet generators.

Lines 79-86 allow packets from established connections to be forwarded onto the corporate networks. This is first place in the firewall configuration that valid packets are actually processed. We place these rules here, since they will match most of the packets and rules are evaluated in order or appearance.

Lines 87-91 will drop any packets that have our internal IP numbers arriving from the Internet (i.e. spoofed packets).

Lines 92-94 will drop any broadcast packets that arrive from the Internet.

Lines 94-104 will process ICMP packets in a secure manner. Most of the incoming ICMP packets will be dropped, except for the replies to internal ping and traceroute requests. Lines 105-110 allows the configuration selection of incoming icmp source-quench processing (normally, systems on the service network will honor source-quench requests). Lines 111 and 112 allow outgoing source-quench requests. Lines 113-123 filter out icmp requests that could be used to map our internal network, while allowing outbound icmp requests through.

Security Policy for Border Router and Primary Firewall (router1, west site)

Router1 is a key part of the defense in depth strategy that protects the GIAC network. It serves the front line roll of filtering out malformed packets and controls the initial routing for packets destined inside of the corporate network. It also protects the service and internal networks from attacks originating from the Internet. If the security policy on router1 fails, secondary policies on the service network machines and the internal firewall (router2) also provide network defenses.

A Linux based solution was chosen for several reasons:

- it is normally cheaper than a comparable proprietary solution,
- adheres to GIAC's corporate policy of promoting open source,
- it is very flexible in meeting the current and anticipated future technical requirements for perimeter security.

Because the demarcation point of GIAC's Internet connection is behind the ISP's dedicated router, a separate hardware based border router is not necessary. The functions provided by a border router have been combined with those of the primary firewall. This is feasible if the firewall hardware has sufficient capacity to perform both functions. The Dell PowerEdge 1650 that hosts the firewall has more than enough capacity to handle the current and short term future increases in network capacity.

There are three interfaces on the router1, eth0: facing the internet, eth1: facing the GIAC internal network, and eth2: connected to the management network. The configuration for eth2 is done using the normal system management tools provided by Fedora Core-1:

Firewall Policy (router1 -- west site primary)

Router1 uses the Fedora Core-1 Linux distribution. The Linux kernel used is 2.4.22. Linux 2.4 contains built-in firewall capabilities by means of the iptables network control mechanism. Iptables normally loads its network configuration information during system boot.

rc.west1-fw file listing

```
#!/bin/bash
# rc.west1-fw script -- initialize the router1 firewall and NAT
# rule sets.
#

export IPTABLES="/sbin/iptables" # full path for iptables binary

export OUR_NET="10.1.0.0/16"     # used by rc.common-fw
export QUENCH_NET="10.1.1.0/24"  # used by rc.common-fw

export EXT_NTP="10.0.3.33"       # public address for time server
export EXT_NS="10.0.3.34"        # public address for ext. name server
export EXT_WWW="10.0.3.35"       # public address for ext. web server
export EXT_GWA="10.0.3.36"       # public address for SSH gateway
export EXT_GWB="10.0.3.37"       # public address for VPN gateway
export EXT_MAIL="10.0.3.38"      # public address for MAIL server
export EXT_GW="10.0.3.39"        # public address for web proxy
export EXT_OGWB="10.0.3.85"      # public addr. for opposite VPN gateway

export DMZ_NTP="10.1.1.10"       # private address for time server
export DMZ_NS="10.1.1.11"        # private address for ext. name server
export DMZ_WWW="10.1.1.12"       # private address for ext. web server
export DMZ_MAIL="10.1.1.21"      # private address for MAIL server
export SRV_INTWEB="10.1.20.10"   # private addr. for internal web server
export SRV_SRVWEB="10.1.20.11"   # private addr. for partners web server
export SRV_FS="10.1.20.21"       # private address for file server
export SRV_GW="10.1.20.31"       # private address for web proxy
export SRV_GWA="10.1.20.32"      # private address for SSH gateway
export SRV_GWB="10.1.20.33"      # private address for VPN gateway
export SRV_INS="10.1.20.34"      # private addr. for internal DNS server
export SRV_MAIL="10.1.20.41"     # private addr. for internal MAIL server

export NET_MGR="10.1.10.251"     # private address for network management
export NET_LOG="10.1.10.252"     # private address for network log host

export IF_EXT="eth1"             # external interface
export IF_NET="eth2"             # network management interface
export IF_DMZ="eth3"             # service network interface

#
# bring in the common firewall rules
#
. /etc/rc.common-fw

#
# this is the start of the actual filtering policies for our
# networks.
#

# NET: ssh connection from direct connection laptop (for testing)
$IPTABLES -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW \
     -j ACCEPT

# NET: ssh connection from west network management
$IPTABLES -A INPUT -i $IF_NET -p tcp -s $NET_MGR --dport 22 -m state \
     --state NEW -j ACCEPT

# NET: allow syslog messages from any internal host to log host
$IPTABLES -A OUTPUT -o $IF_NET -p udp -d $NET_LOG --dport 514 \
     -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i ! $IF_EXT -o $IF_NET -p udp -d $NET_LOG \
     --dport 514 -m state --state NEW -j ACCEPT

# VPN: outgoing vpn connection from west to east
$IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -s $SRV_GWB -d $EXT_OGWB \
     -m state --state NEW -j ACCEPT

# VPN: incoming vpn connection from east to west
$IPTABLES -A FORWARD -i $IF_EXT -o $IF_DMZ -s $EXT_OGWB -d $SRV_GWB \
     -m state --state NEW -j ACCEPT

# WWW: allow http connections to external web server from anywhere
$IPTABLES -A FORWARD -i $IF_EXT -o $IF_DMZ -p tcp -d $DMZ_WWW \
     -m multiport --dports 80,443 -m state --state NEW -j ACCEPT

# WWW: allow web proxy to connect to any http port
$IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -p tcp -s $SRV_GW \
     -m multiport --dports 80,443 -m state --state NEW -j ACCEPT

# SSH: outgoing ssh connections to the internet
$IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -p tcp -s $SRV_GWA \
     --dport 22 -m state --state NEW -j ACCEPT

# SSH: incoming ssh connections from the internet
$IPTABLES -A FORWARD -i $IF_EXT -o $IF_DMZ -p tcp -d $SRV_GWA \
     --dport 22 -m state --state NEW -j ACCEPT

# NTP: incoming, allow anyone to connect to our time server
$IPTABLES -A FORWARD -i $IF_EXT -o $IF_DMZ -p udp -d $DMZ_NTP \
     --dport 123 -m state --state NEW -j ACCEPT

# NTP: outgoing, only allow our time server to connect to the internet
$IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -p udp -s $DMZ_NTP \
     --dport 123 -m state --state NEW -j ACCEPT

# DNS: allow outgoing udp dns requests only from our public dns server
$IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -p udp -s $DMZ_NS \
     --dport 53 -m state --state NEW -j ACCEPT

# DNS: allow outgoing tcp dns requests only from our public dns server
$IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -p tcp -s $DMZ_NS \
     --dport 53 -m state --state NEW -j ACCEPT

# DNS: allow incoming udp dns requests from anyone
$IPTABLES -A FORWARD -i $IF_EXT -o $IF_DMZ -p udp -d $DMZ_NS \
     --dport 53 -m state --state NEW -j ACCEPT

# MAIL: no IMAP/POP traffic is allowed since it must travel through
# SSH/VPN tunnels only to the internal server (handled by routing on vs5)

# MAIL: anyone can connect to our external mail server
$IPTABLES -A FORWARD -i $IF_EXT -o $IF_DMZ -p tcp -d $DMZ_MAIL \
     --dport 25 -m state --state NEW -j ACCEPT

# MAIL: only our public mail server can connect to the internet
$IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -p tcp -s $DMZ_MAIL \
     --dport 25 -m state --state NEW -j ACCEPT

#
# NAT: convert our external services to/from their internal servers.
#

$IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_GWB -j DNAT \
     --to-destination $SRV_GWB
$IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $SRV_GWB -j SNAT \
     --to-source $EXT_GWB
$IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_GW -j DNAT \
     --to-destination $SRV_GW
$IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $SRV_GW -j SNAT \
     --to-source $EXT_GW
$IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_GWA -j DNAT \
     --to-destination $SRV_GWA
$IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $SRV_GWA -j SNAT \
     --to-source $EXT_GWB
$IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_NTP -j DNAT \
     --to-destination $DMZ_NTP
$IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $EXT_NTP -j SNAT \
     --to-source $DMZ_NTP
$IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_NS -j DNAT \
     --to-destination $DMZ_NS
$IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $DMZ_NS -j SNAT \
     --to-source $EXT_NS
$IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_WWW -j DNAT \
     --to-destination $DMZ_WWW
$IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $DMZ_WWW -j SNAT \
119.        --to-source $EXT_WWW
120.    $IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_MAIL -j DNAT \
121.        --to-destination $DMZ_MAIL
122.    $IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $DMZ_MAIL -j SNAT \
123.        --to-source $EXT_MAIL
124.    #
125.    # we are done...
126.    #
```

Lines 5-32 define the configuration parameters for this router. These parameters include internal and external host addresses for servers and interface names for network segments. Lines 33-36 include the common firewall configuration settings found in rc.common-fw (described above).

Lines 41-46 allow SSH connections to the router that come from the directly connected console or network management workstation. Lines 47-51 allows any internal host (including this router) to log messages to the network logging host.

Lines 52-57 allow VPN tunnel connections between the east and west sites.

Lines 58-60 allow incoming http and https connections to the public web server. Lines 61-63 allows the web proxy to access http and https content for internal users.

Lines 64-69 allow the SSH gateway to accept incoming connections and establish outgoing connections to the Internet.

Lines 70-72 allow incoming NTP connections to our time server. Lines 73-75 only allow outbound ntp connections from our time server.

Lines 76-81 allow outbound tcp and udp DNS requests from our external DNS server. Lines 82-84 allow incoming udp DNS requests directed towards our external DNS server.

Lines 87-92 allow incoming and outgoing e-mail traffic from our external mail server. Note: no IMAP/POP3 traffic is allowed, since it must be tunneled inside of SSH or VPN connections.

Lines 96-123 define the network address translation rules for our public services. Each service has two rules. The first rule provides translation of the destination addresses on incoming packets (DNAT). The second rule provides translation of the source addresses on outgoing packets (SNAT). Each of these rules are needed, without the DNAT rule, incoming packets would not be forwarded to the correct server. Without SNAT, internal host IP numbers would “leak” outside of our network and could allow unauthorized persons to map parts of our internal network.

The ordering of the rules is important. While an effort has been made to minimize dependences on rule order, keep in mind that rules are matched in order of occurrence. Rule matching is terminated once a matching rule has an ACCEPT or DROP action. This means that a more specific rule must appear before a more general rule for the same packet. Also, if possible, rules that occur frequently should appear before those that happen less often. For example, notice that the rules that allow packets from established connections to be forwarded are placed as the first checks after a packet has been determined to be valid (in rc.common-fw).

Firewall Policy (router2 -- west site internal)

Router2 uses the Fedora Core-1 Linux distribution. The Linux kernel used is 2.4.22. Linux 2.4 contains built-in firewall capabilities by means of the iptables network control mechanism. Iptables normally loads its network configuration information during system boot.

While the perimeter router router1 protects the public service network (DMZ) and provides address translation, it doesn't know anything about the internal network topology. The internal router router2 controls traffic between internal system and hosts located on the Internet. Router2 rule set must be compatible with those of router1.

rc.west2-fw file listing

```
#!/bin/bash
# rc.west2-fw script -- initialize the router2 firewall rule sets.
#

export IPTABLES="/sbin/iptables" # full path for iptables binary

export OUR_NET="10.1.20.0/24"     # used by rc.common-fw
export QUENCH_NET=""              # used by rc.common-fw

export EXT_NTP="10.0.3.33"        # public address for time server
export EXT_NS="10.0.3.34"         # public address for ext. name server
export EXT_WWW="10.0.3.35"        # public address for ext. web server
export EXT_GWA="10.0.3.36"        # public address for SSH gateway
export EXT_GWB="10.0.3.37"        # public address for VPN gateway
export EXT_MAIL="10.0.3.38"       # public address for MAIL server
export EXT_GW="10.0.3.39"         # public address for web proxy
export EXT_OGWB="10.0.3.85"       # public addr. for opposite VPN gateway

export DMZ_NTP="10.1.1.10"        # private address for time server
export DMZ_NS="10.1.1.11"         # private address for ext. name server
export DMZ_WWW="10.1.1.12"        # private address for ext. web server
export DMZ_MAIL="10.1.1.21"       # private address for MAIL server
export SRV_INTWEB="10.1.20.10"    # private addr. for internal web server
export SRV_SRVWEB="10.1.20.11"    # private addr. for suppliers web server
export SRV_FS="10.1.20.21"        # private address for file server
export SRV_GW="10.1.20.31"        # private address for web proxy
export SRV_GWA="10.1.20.32"       # private address for SSH gateway
export SRV_GWB="10.1.20.33"       # private address for VPN gateway
export SRV_INS="10.1.20.34"       # private addr. for internal DNS server
export SRV_MAIL="10.1.20.41"      # private addr. for internal MAIL server
export NET_MGR="10.1.10.251"     # private address for network management
export NET_LOG="10.1.10.252"     # private address for network log host

export IF_EXT="eth1"             # external interface
export IF_NET="eth2"             # network management interface
export IF_DMZ="eth3"             # "public" service network interface
export IF_SRV="eth3"             # service net interface(same as IF_DMZ)
export IF_INT="eth4"             # corporate network interface

#
# bring in the common firewall rules
#
. /etc/rc.common-fw

#
# this is the start of the actual filtering policies for our network.
#

# NET: ssh connection from direct connection laptop (for testing)
$IPTABLES -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW \
     -j ACCEPT

# NET: ssh connection from west network management
$IPTABLES -A INPUT -i $IF_NET -p tcp -s $NET_MGR --dport 22 -m state \
     --state NEW -j ACCEPT

# NET: allow syslog messages from any internal host to log host
$IPTABLES -A OUTPUT -o $IF_NET -p udp -d $NET_LOG --dport 514 \
     -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i ! $IF_EXT -o $IF_NET -p udp -d $NET_LOG \
     --dport 514 -m state --state NEW -j ACCEPT

# VPN: incoming corporate west net connection to west vpn gateway
$IPTABLES -A FORWARD -i $IF_INT -o $IF_DMZ -d $SRV_GWB \
     -m state --state NEW -j ACCEPT

# VPN: outgoing vpn traffic from west vpn gateway to west corporate net
$IPTABLES -A FORWARD -i $IF_DMZ -o $IF_INT -s $SRV_GWB \
     -m state --state NEW -j ACCEPT

# VPN: outgoing vpn connection from west to east
$IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -s $SRV_GWB -d $EXT_OGWB \
     -m state --state NEW -j ACCEPT

# VPN: incoming vpn connection from east to west
$IPTABLES -A FORWARD -i $IF_EXT -o IF_DMZ -s $EXT_OGWB -d $SRV_GWB \
     -m state --state NEW -j ACCEPT

# VPN: new connection from west net management to west vpn
$IPTABLES -A FORWARD -i $IF_NET -o $IF_SRV -s $NET_MGR -m state \
     --state NEW -j ACCEPT

# FS: only allow the corporate segment to connect to the file server
#     connections between the west VPN gateway and File server is handled
#     on the file server since they are on the same segment
$IPTABLES -A FORWARD -i $IF_INT -o $IF_SRV -p tcp -d $SRV_FS \
     -m multiport --dports 139,445 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $IF_INT -o $IF_SRV -p udp -d $SRV_FS \
     -m multiport --dports 137,138 -m state --state NEW -j ACCEPT
```

```
 74.   # WWW: allow http/https to int. corp. web server from corp. net
 75.   $IPTABLES -A FORWARD -i $IF_INT -o $IF_SRV -p tcp -d $SRV_INTWEB \
 76.        -m multiport --dports 80,443 -m state --state NEW -j ACCEPT

 77.   # WWW: allow http/https connections to partners web server from corp. net
 78.   $IPTABLES -A FORWARD -i $IF_INT -o $IF_SRV -p tcp -d $SRV_SRVWEB \
 79.        -m multiport --dports 80,443 -m state --state NEW -j ACCEPT

 80.   # WWW: allow web proxy to connect to any http port
 81.   $IPTABLES -A FORWARD -i $IF_SRV -o $IF_EXT -p tcp -s $SRV_GW \
 82.        -m multiport --dports 80,443 -m state --state NEW -j ACCEPT

 83.   # SSH: outgoing ssh connections to the internet
 84.   $IPTABLES -A FORWARD -i $IF_SRV -o $IF_EXT -p tcp -s $SRV_GWA \
 85.        --dport 22 -m state --state NEW -j ACCEPT

 86.   # SSH: incoming ssh connections from the internet
 87.   $IPTABLES -A FORWARD -i $IF_EXT -o $IF_SRV -p tcp -d $SRV_GWA \
 88.        --dport 22 -m state --state NEW -j ACCEPT

 89.   # SSH: incoming ssh connections from corporate segment to SSH gateway
 90.   $IPTABLES -A FORWARD -i $IF_INT -o $IF_SRV -p tcp -d $SRV_GWA \
 91.        --dport 22 -m state --state NEW -j ACCEPT

 92.   # NTP: allow our servers to connect to our ntp server
 93.   $IPTABLES -A FORWARD -i $IF_SRV -o $IF_EXT -p udp -d $DMZ_NTP \
 94.        --dport 123 -m state --state NEW -j ACCEPT

 95.   # NTP: allow our corporate systems to connect to our ntp server
 96.   $IPTABLES -A FORWARD -i $IF_INT -o $IF_EXT -p udp -d $DMZ_NTP \
 97.        --dport 123 -m state --state NEW -j ACCEPT

 98.   # DNS: allow outgoing udp dns requests only to our public DNS server
 99.   $IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -p udp -s $SRV_INS \
100.        -d $DMZ_NS --dport 53 -m state --state NEW -j ACCEPT

101.   # DNS: allow outgoing tcp dns requests only to our public DNS server
102.   $IPTABLES -A FORWARD -i $IF_SRV -o $IF_EXT -p tcp -s $SRV_INS \
103.        -d $DMZ_NS --dport 53 -m state --state NEW -j ACCEPT

104.   # MAIL: No IMAP/POP traffic is allowed since it must travel through
105.   # SSH/VPN tunnels only to the internal server (handled by routing on vs3)

106.   # MAIL: only our external mail server can connect to the internal server
107.   $IPTABLES -A FORWARD -i $IF_EXT -o $IF_DMZ -p tcp -s $DMZ_MAIL \
108.        -d $SRV_MAIL --dport 25 -m state --state NEW -j ACCEPT

109.   # MAIL: our internal mail server may only connect to our public mail one
110.   $IPTABLES -A FORWARD -i $IF_SRV -o $IF_EXT -p tcp -s $SRV_MAIL \
111.        -d $DMZ_MAIL --dport 25 -m state --state NEW -j ACCEPT

112.   #
113.   # we are done...
114.   #
```

Lines 5-33 define the configuration for this router's hosts and interfaces. Note: This configuration script is based on the same template as the border routers, references to the DMZ is not the public service network with router1, but the internal service network inside of router2. So the values for IF_DMZ and IF_SRV are set equal. This allows same rules to be written the same way regardless of which router is used.

Lines 33-36 include the common firewall configuration settings found in rc.common-fw (described above).

Lines 41-46 allow SSH connections to the router that come from the directly connected console or network management workstation. Lines 47-51 allows any internal hosts (including this router) to log messages to the network logging host.

Lines 52-57 allow VPN tunnel traffic to pass between the west site corporate network and the VPN gateway server. Lines 58-63 allow VPN traffic between the west and east VPN gateways. Lines 64-66 allow the west site network management system to connect to the west VPN gateway. This allows access to the east site network for management tasks.

Lines 67-73 allow incoming file server connections from the west corporate segment and the west VPN gateway system (tunneling east site traffic).

Lines 74-79 allow http and https connections to the internal corporate and partner's web servers. Note: a rule to allow traffic between the VPN gateway and the internal corporate and partner web servers is not necessary, since they are located on the same network segment, so that traffic will not appear on router2's interfaces. Access controls are also enforced by the web servers themselves. Lines 80-82 allows the web proxy to access http and https content for internal users.

Lines 83-88 allow the SSH gateway to accept incoming connections and establish outgoing connections to the Internet. Lines 89-91 allow users on the internal corporate network to connect to the SSH gateway.

Lines 92-94 allow our internal service network to connect to our NTP time server. Lines 95-97 allow systems on the corporate network to connect to our time server.

Lines 98-103 allow DNS queries to only be sent to the external DNS server from the internal networks.

Lines 104-111 allow incoming and outgoing e-mail traffic from our external mail server. Note: no IMAP/POP3 traffic is allowed, since it must be tunneled inside of SSH or VPN connections.

Security Policy for Border Router and Primary Firewall (router3, east site)

Router3 is a key part of the defense in depth strategy that protects the GIAC network. It serves the primary defense of the east site sales office and disaster recovery site. Because of the limited uses of the east site network, a separate internal router is not necessary. Since the critical company assets are located at the west site, a single router is an acceptable risk. Just like router1 at the west site, router3 has the front line role of filtering out malformed packets and controlling the initial routing for packets destined inside of the east site corporate network. It also protects the service and internal networks from attacks originating from the Internet.

The considerations and justifications for router3 are the same as those for the west site border router (router1). The same hardware configuration as router1 is used with an additional Ethernet interface. There are five interfaces on router3, eth0: direct connect console interface, eth1: facing the internet, eth2: connected to the management network, eth3 connected to the east site general service network, eth4 connected to the east site internal employee network, and eth5: connected to the internal service network.

Firewall Policy (router3, east site)

Router3 uses the Fedora Core-1 Linux distribution. The Linux kernel used is 2.4.22. Linux 2.4 contains built-in firewall capabilities by means of the iptables network control mechanism. The firewall rules are normally loaded during system boot.

The firewall configuration for router3 provides the same capabilities as both west site routers (router1 and router2). Its rule set contains rules from both of the west site routers.

rc.east-fw file listing

```
#!/bin/bash
# rc.east-fw script -- initialize the router3 firewall and NAT
# rule sets.
#

export IPTABLES="/sbin/iptables" # full path for iptables binary

export OUR_NET="10.2.0.0/24"     # used by rc.common-fw
export QUENCH_NET="10.2.1.0/24"  # used by rc.common-fw

export EXT_NTP="10.0.3.81"       # public address for time server
export EXT_NS="10.0.3.82"        # public address for ext. name server
export EXT_WWW="10.0.3.83"       # public address for ext. web server
export EXT_GWA="10.0.3.84"       # public address for SSH gateway
export EXT_GWB="10.0.3.85"       # public address for VPN gateway
export EXT_MAIL="10.0.3.86"      # public address for MAIL server
export EXT_GW="10.0.3.87"        # public address for web proxy
export EXT_OGWB="10.0.3.37"      # public addr. for opposite VPN gateway

export DMZ_NTP="10.2.1.10"       # private address for time server
export DMZ_NS="10.2.1.11"        # private address for ext. name server
export DMZ_WWW="10.2.1.12"       # private address for ext. web server
export DMZ_MAIL="10.2.1.13"      # private address for MAIL server
export SRV_FS="10.2.20.20"       # private address for file server
export SRV_GWA="10.2.20.22"      # private address for SSH gateway
export SRV_GWB="10.2.20.23"      # private address for VPN gateway
export SRV_INS="10.2.20.24"      # private addr. for internal DNS server
export SRV_MAIL="10.2.20.25"     # private addr. for internal MAIL server
export SRV_GW="10.2.20.26"       # private address for web proxy

export NET_MGR="10.2.20.21"      # private address for network management
export NET_LOG="10.2.20.21"      # private address for network log host

export IF_EXT="eth1"             # external interface
export IF_NET="eth2"             # network management interface
export IF_DMZ="eth3"             # service network interface
export IF_INT="eth4"             # internal corporate network
export IF_SRV="eth5"             # internal service/vpn network interface

#
# bring in the common firewall rules
#

. /etc/rc.common-fw

#
# this is the start of the actual filtering policies for this router
#

# NET: ssh connection from direct connection laptop (for testing)
$IPTABLES -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW \
     -j ACCEPT

# NET: ssh connection from east network management
$IPTABLES -A INPUT -i $IF_NET -p tcp -s $NET_MGR --dport 22 -m state \
     --state NEW -j ACCEPT

# NET: allow syslog messages from any internal host to log host
$IPTABLES -A OUTPUT -o $IF_NET -p udp -d $NET_LOG --dport 514 \
     -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i ! $IF_EXT -o $IF_SRV -p udp -d $NET_LOG \
     --dport 514 -m state --state NEW -j ACCEPT

# VPN: allow corporate to establish new connections on west vpn
$IPTABLES -A FORWARD -i $IF_INT -o $IF_SRV -d $SRV_GWB -m state \
     --state NEW -j ACCEPT

# VPN: allow west vpn new connections to our corporate segment
$IPTABLES -A FORWARD -i $IF_SRV -o $IF_INT -s $SRV_GWB -m state \
     --state NEW -j ACCEPT

# VPN: outgoing vpn connection from east to west
$IPTABLES -A FORWARD -i $IF_SRV -o $IF_EXT -p udp -s $SRV_GWB \
     -d $EXT_OGWB --dport 5000 -m state --state NEW -j ACCEPT

# VPN: incoming vpn connection from west to east
$IPTABLES -A FORWARD -i $IF_EXT -o $IF_SRV -p udp -s $EXT_OGWB \
     -d $SRV_GWB --dport 5000 -m state --state NEW -j ACCEPT

# VPN: new connection from east internal corporate net to west vpn
$IPTABLES -A FORWARD -i $IF_INT -o $IF_SRV -d $SRV_GWB -m state \
     --state NEW -j ACCEPT

# VPN: new connection from east net management to east vpn
$IPTABLES -A FORWARD -i $IF_NET -o $IF_SRV -s $NET_MGR -m state \
     --state NEW -j ACCEPT
```

```
# VPN: new connection from east vpn to east internal corporate net
$IPTABLES -A FORWARD -i $IF_SRV -o $IF_INT -s $SRV_GWB -m state \
     --state NEW -j ACCEPT

# FS: only allow the corporate segment to connect to the file server
#     connections between the west VPN gateway and File server is handled
#     on the file server since they are on the same segment
$IPTABLES -A FORWARD -i $IF_INT -o $IF_SRV -p tcp -d $SRV_FS \
     -m multiport --dports 139,445 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $IF_INT -o $IF_SRV -p udp -d $SRV_FS \
     -m multiport --dports 137,138 -m state --state NEW -j ACCEPT

# WWW: allow http connections to external web server from anywhere
$IPTABLES -A FORWARD -i $IF_EXT -o $IF_DMZ -p tcp -d $DMZ_WWW \
     -m multiport --dports 80,443 -m state --state NEW -j ACCEPT

# WWW: allow web proxy to connect to any http port
$IPTABLES -A FORWARD -i $IF_SRV -o $IF_EXT -p tcp -s $SRV_GW \
     -m multiport --dports 80,443 -m state --state NEW -j ACCEPT

# SSH: outgoing ssh connections to the internet
$IPTABLES -A FORWARD -i $IF_SRV -o $IF_EXT -p tcp -s $SRV_GWA \
     --dport 22 -m state --state NEW -j ACCEPT

# SSH: incoming ssh connections from the internet
$IPTABLES -A FORWARD -i $IF_EXT -o $IF_SRV -p tcp -d $SRV_GWA \
     --dport 22 -m state --state NEW -j ACCEPT

# SSH: incoming ssh connections from corporate segment to SSH gateway
$IPTABLES -A FORWARD -i $IF_INT -o $IF_SRV -p tcp -d $SRV_GWA \
     --dport 22 -m state --state NEW -j ACCEPT

# NTP: allow incoming ntp connections from anywhere
$IPTABLES -A FORWARD -i $IF_EXT -o $IF_DMZ -p udp -d $DMZ_NTP \
     --dport 123 -m state --state NEW -j ACCEPT

# NTP: allow outgoing ntp connections only from our ntp server
$IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -p udp -s $DMZ_NTP \
     --dport 123 -m state --state NEW -j ACCEPT

# DNS: allow outgoing udp dns requests only from our public DNS server
$IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -p udp -s $DMZ_NS \
     --dport 53 -m state --state NEW -j ACCEPT

# DNS: allow outgoing tcp dns requests only from our public DNS server
$IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -p tcp -s $DMZ_NS \
     --dport 53 -m state --state NEW -j ACCEPT

# DNS: allow incoming udp dns requests from anyone
$IPTABLES -A FORWARD -i $IF_EXT -o $IF_DMZ -p udp -d $DMZ_NS \
     --dport 53 -m state --state NEW -j ACCEPT

# MAIL: No IMAP/POP traffic is allowed since it must travel through
# SSH/VPN tunnels only to the internal server (handled by routing on vs5)

# MAIL: anyone can connect to our external mail server
$IPTABLES -A FORWARD -i $IF_EXT -o $IF_DMZ -p tcp -d $DMZ_MAIL \
     --dport 25 -m state --state NEW -j ACCEPT
```

```
114.   # MAIL: only our public mail server can connect to the internet
115.   $IPTABLES -A FORWARD -i $IF_DMZ -o $IF_EXT -p tcp -s $DMZ_MAIL \
116.        --dport 25 -m state --state NEW -j ACCEPT

117.   # MAIL: our internal mail server may only use our external mail server
118.   $IPTABLES -A FORWARD -i $IF_SRV -o $IF_DMZ -p tcp -s $SRV_MAIL \
119.        -d $DMZ_MAIL --dport 25 -m state --state NEW -j ACCEPT

120.   # MAIL: only the external mail server may connect to the internal one
121.   $IPTABLES -A FORWARD -i $IF_DMZ -o $IF_SRV -p tcp -s $DMZ_MAIL \
122.        -d $SRV_MAIL --dport 25 -m state --state NEW -j ACCEPT

123.   #
124.   # NAT: convert our external services to/from their internal servers.
125.   #

126.   $IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_GW -j DNAT \
127.        --to-destination $SRV_GW
128.   $IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $SRV_GW -j SNAT \
129.        --to-source $EXT_GW
130.   $IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_GWA -j DNAT \
131.        --to-destination $SRV_GWA
132.   $IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $SRV_GWA -j SNAT \
133.        --to-source $EXT_GWA
134.   $IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_GWB -j DNAT \
135.        --to-destination $SRV_GWB
136.   $IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $SRV_GWB -j SNAT \
137.        --to-source $EXT_GWB
138.   $IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_NTP -j DNAT \
139.        --to-destination $DMZ_NTP
140.   $IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $DMZ_NTP -j SNAT \
141.        --to-source $EXT_NTP
142.   $IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_NS -j DNAT \
143.        --to-destination $DMZ_NS
144.   $IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $DMZ_NS -j SNAT \
145.        --to-source $EXT_NS
146.   $IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_WWW -j DNAT \
147.        --to-destination $DMZ_WWW
148.   $IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $DMZ_WWW -j SNAT \
149.        --to-source $EXT_WWW
150.   $IPTABLES -t nat -A PREROUTING -i $IF_EXT -d $EXT_MAIL -j DNAT \
151.        --to-destination $DMZ_MAIL
152.   $IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $SRV_MAIL -j SNAT \
153.        --to-source $EXT_MAIL
154.   #
155.   # we are done...
156.   #
```

Lines 5-32 define the configuration parameters for this router. These parameters include internal and external host addresses for servers and interface names for network segments. Lines 33-36 include the common firewall configuration settings found in rc.common-fw (described above).

Lines 40-45 allow SSH connections to the router that come from the directly connected console or network management workstation. Lines 46-50 allows any internal hosts (including this router) to log messages to the network logging host.

Lines 51-56 allow the east site to open new VPN tunnel connections on the west site. Lines 57-59 allows outgoing VPN connections between the east and west sites. Lines 60-62 allow incoming VPN connections from the west site VPN gateway. Lines 63-65 allow connections from the east site corporate network segment to the east site VPN gateway. Lines 66-68 allow the east site network management server to connect to the east site VPN gateway. Lines 69-71 allows traffic to be sent between the east site VPN gateway and the east site corporate network segment.

Lines 75-78 allows the corporate network hosts to access the east site file server.

Lines 58-63 allow incoming http and https connections to the public web server. Lines 64-69 allows the web proxy to access http and https content for internal users.

Lines 85-90 allow the SSH gateway to accept incoming connections and establish outgoing connections to the Internet.

Lines 91-93 allow the corporate network hosts to connect to the SSH gateway.

Lines 94-96 allow incoming NTP connections to our time server. Lines 97-99 only allow outbound ntp connections from our time server.

Lines 100-105 allow outbound tcp and udp DNS requests from our external DNS server. Lines 106-108 allow incoming udp DNS requests directed towards our external DNS server.

Lines 111-116 allow incoming and outgoing e-mail traffic from our external mail server. Note: no IMAP/POP3 traffic is allowed, since it must be tunneled inside of SSH or VPN connections. Lines 117-122 allows traffic to pass between the east site internal and external mail servers.

Lines 126-153 define the network address translation rules for our public services. Each service has two rules. The first rule provides translation of the destination addresses on incoming packets (DNAT). The second rule provides translation of the source addresses on outgoing packets (SNAT). Each of these rules are needed, without the DNAT rule, incoming packets would not be forwarded to the correct server. Without SNAT, internal host IP numbers would “leak” outside of our network and could allow unauthorized persons to map parts of our internal network.

VPN Configuration for west-east site tunnel

Here are the configuration files for the west and east site VPN gateways. This first file will create the west endpoint of an OpenVPN tunnel to the east site. It is loaded on the west site VPN gateway gw1b.example.com.

gw1b-vpn.conf file listing

```
 1.    dev tun
 2.    port 5000
 3.    ; user nobody
 4.    ; group nobody
 5.    comp-lzo
 6.    verb 3
 7.    #shaper 1000
 8.    remote 10.0.3.85
 9.    ifconfig 10.1.40.6 10.2.40.6
10.    up /etc/vpn/ip-up.sh
11.    tls-server
12.    #auth alg=none
13.    #cipher alg=none
14.    #keysize 192
15.    ca /etc/vpn/cacert.crt
16.    cert /etc/vpn/gw1b.crt
17.    key /etc/vpn/gw1b.key
18.    dh /etc/vpn/dh2048.pem
19.    tls-cipher RC4-MD5
```

Lines 1 and 2, configures OpenVPN to use the network tunnel device for VPN traffic on UDP port 5000. Lines 3 and 4 are commented out, but if active will have the OpenVPN daemon change its user and group to an account with limited capabilities. Line 5 compresses VPN traffic using the LZO library. Line 6 sets the verbosity level of the OpenVPN daemon to medium (status and some debugging messages are printed). Uncommenting line 7 will turn on traffic shaping, the value of 1000 will limit VPN traffic to approximately two-thirds of a T-1 line (1000Kb/s out of 1540Kb/s bandwidth).

Line 8 identifies the IP address of the system located at the other end of the UDP tunnel. Note: this is not the VPN addresses, just the tunnel. Line 9 configures the VPN address for the two ends of the VPN link (for our example, 10.1.40.6 is gw1b's end, and 10.2.40.6 is gw2b's end. Line 10 provides the name of the helper script to add route information when starting the VPN link. Note: These path names are relative to the root of the VPN file system, not to the development system.

The rest of the file configures the SSL library. Lines 16-18 load gw1b's certificate information into OpenVPN. Lines 19 and 20 configure the encryption engine to use the MD5 algorithm with 2048 length keys.

This next file creates the east endpoint of the OpenVPN tunnel to the west site. It is loaded on the east site VPN gateway gw2b.example.com.

gw2b-vpn.conf file listing

```
 1.    dev tun
 2.    port 5000
 3.    ; user nobody
 4.    ; group nobody
 5.    comp-lzo
 6.    verb 3
 7.    #shaper 1000
 8.    remote 10.0.3.37
 9.    ifconfig 10.2.40.6 10.1.40.6
10.    up /etc/vpn/ip-up.sh
11.    tls-client
12.    #auth alg=none
13.    #cipher alg=none
14.    #keysize 192
15.    ca /etc/vpn/cacert.crt
16.    cert /etc/vpn/gw2b.crt
17.    key /etc/vpn/gw2b.key
18.    dh /etc/vpn/dh2048.pem
19.    tls-cipher RC4-MD5
```

Lines 1 and 2, configures OpenVPN to use the network tunnel device for VPN traffic on UDP port 5000. Lines 3 and 4 are commented out, but if active will have the OpenVPN daemon change its user and group to an account with limited capabilities. Line 5 compresses VPN traffic using the LZO library. Line 6 sets the verbosity level of the OpenVPN daemon to medium (status and some debugging messages are printed). Uncommenting line 7 will turn on traffic shaping, the value of 1000 will limit VPN traffic to approximately two-thirds of a T-1 line (1000Kb/s out of 1540Kb/s bandwidth).

Line 8 identifies the IP address of the system located at the other end of the UDP tunnel. Note: this is not the VPN addresses, just the tunnel. Line 9 configures the VPN address for the two ends of the VPN link (for our example, 10.1.40.6 is gw1b’s end, and 10.2.40.6 is gw2b’s end. Line 10 provides the name of the helper script to add route information when starting the VPN link. Note: These path names are relative to the root of the VPN file system, not to the development system. The rest of the file configures the SSL library. Lines 16-18 load gw2b’s certificate information into OpenVPN. Lines 19 and 20 configure the encryption engine to use the MD5 algorithm with 2048 length keys.

GIAC Enterprises VPN Installation and Configuration Tutorial

Introduction

This tutorial describes process of installing, configuring and verifying the virtual private network (VPN) between the GIAC Enterprises west and east sites. In order to successfully use this tutorial, there are several prerequisites:

- You must have root access to the GIAC Enterprises internal development system or equivalent. The development system is used to produce this tutorial is an IBM-PC compatible system with dual processor (Celeron 500Mhz), 288MB memory, and 32GB disk. It runs the Red Hat Fedora Core-1 Linux operating system with all current patches installed.
- You must have the complete development environment installed. Specifically, the C compiler, library headers and standard libraries.
- You must have a copy of the GIAC Enterprises standard UML root file system. This root file system is the starting point for the custom root file systems used by the User Mode Linux (UML) virtual machines. It is normally available on the development machine under the path name: /var/vnuml/root_fs_dist.
- A basic understanding of Linux/Unix commands and how to compile programs would be helpful.

Several packages were evaluated as possible candidates for VPN solutions. GIAC Enterprises adopted two: OpenSSH (secure shell) and OpenVPN. Both OpenSSH and OpenVPN are open source projects. The home pages for OpenSSH and OpenVPN are located at http://www.openssh.org and http://openvpn.sourceforge.net.

OpenSSH was chosen for use by “road warriors” because it is compatible with the PuTTY and WinSCP utilities adopted for laptop and desktop users. SSH performance is further improved when the blowfish encryption algorithm is used.

Another commonly used VPN for UNIX and Linux systems implements IPSEC. An example IPSEC implementation for Linux is the Frees/wan package. OpenVPN was chosen instead of Frees/wan for two main reasons:

- IPSEC does not allow network address translation of the VPN endpoints.
- Typical IPSEC implementations (like Frees/wan) require kernel mode modifications to the host system in order to function.

OpenVPN addresses both of these issues, providing SSL encrypted tunnels by using user mode programs that support NAT for the endpoints. An additional benefit is that the VPN tunnels can be encapsulated using UDP packets.

Organization of this Tutorial

The GIAC Enterprises VPN tutorial provides a step-by-step guide on how to configure, install, operate, and verify the proper functioning of GIAC Enterprises’

Virtual Private Network. We will show how to setup the VPN between the west (using the gw1b gateway) and the east (using the gw2b gateway) sites. This tutorial contains information that must be entered from the keyboard, *”They will be written in italic typeface, like this”.* Information printed by the system will be written in monotype typeface, like this.

In addition to the installation and configuration of OpenVPN itself, this tutorial will also describe the steps necessary to create a certificate authority (CA) for GIAC Enterprises. This CA allows GIAC Enterprises to self-sign SSL certificates for each VPN link instead of having them signed by an external authority like Verisign.

References

In addition to the OpenVPN and OpenSSL manual pages, the following reference sources were used in creating this tutorial:

> “OpenVPN HOWTO”, James Yonan, contained in the OpenVPN source code distribution located at http://openvpn.org/howto.html.
> “Setting up VPN using OpenVPN”, Project VOLANS, located at http://mia.ece.uic.edu/~papers/volans/openvpn.html.
> “How to set up your own Certification Authority”, Project VOLANS, located at http://mia.ece.uic.edu/~papers/volans/settingupCA.html.
> “User Mode Linux”, The User Mode Linux Project, located at http://user-mode-linux.sourceforge.net.
> “Virtual Network User Mode Linux”, The Virtual Network User Mode Linux (VNUML) project, located at http://jungla.dit.upm.es/~vnuml.

Note: In this tutorial, we will use the term “host system” in a restricted sense with regards to the network simulation script used to verify the VPN connection. This term does not refer to an actual machine instead it refers to the process that started the vnumlparser.pl network simulation script. Whenever a reference is made “to entering a command on the host system”, this means that the command is to be entered into the shell of the terminal session that started the simulation.

1. Download the OpenVPN source code to the development machine.

There are two required packages to provide VPN service: OpenVPN and the LZO compression library. If they have not already been downloaded, do so now by entering the following:

```
# cd /usr/local/src
# wget \
    http://unc.dl.sourceforge.net/sourceforge/openvpn/openvpn-1.6_beta6.tar.gz
# wget http://www.oberhumer.com/opensource/lzo/download/lzo-1.0 8.tar.gz
# tar zxf lzo-1.08.tar.gz
# tar zxf openvpn-1.6_beta6.tar.gz
```

2. **Create a custom UML distribution for VPN gateways**

Create a custom root file system for use by the VPN gateways by copying the standard GIAC Enterprises UML file system and mount it on the /mnt/uml mount point. Right now, we are generating a template file system that contains information used by both VPN gateways. Later in the tutorial, we will use it to produce a file system customized for each gateway.

```
# cp /var/vnuml/root_fs_dist /var/vnuml/root_fs_vpn_temp
# #
# # mount the uml root file system for the gateway (VPN) hosts
# #
# mount -o loop /var/vnuml/root_fs_vpn_temp /mnt/uml
```

3. **Compile and install the OpenVPN and LZO packages**

Enter the following commands in order to configure, compile, and install the OpenVPN and LZO packages into the template VPN file system. NOTE: Each of theses commands will produce a large amount of output that has not been included for space reasons.

```
# cd lzo-1.08
# ./configure --prefix=/mnt/uml/usr/local
# make
# make install
# cd ../openvpn-1.6_beta6
# ./configure --prefix=/mnt/uml/usr/local
# make install
```

4. **Check to see if required OpenSSL libraries are available**

OpenVPN requires that the OpenSSL utilities and libraries be available for use. It is normally available with most modern Linux distributions. To check and see if they are installed on the system, enter the following:

```
# rpm -qi openssl
```

If the system responds “package openssl is not installed”, then you must install OpenSSL on the system. You can find the latest version by using the www.rpmfind.com web site. Once the Openssl libraries have been installed on the development system, copy the required libraries over to the template root file system:

```
# cd /mnt/uml/lib
# cp /lib/libssl.so.4 /lib/libssl.so.0.9.7a .
# cp /lib/libcrypto.so.4 /lib/libcrypto.so.0.9.7a .
# cd ../usr/lib
# cp /usr/lib/libgssapi_krb5.so.2* .
# cp /usr/lib/libkrb5.so* .
# cp /usr/lib/libk5crypto.so.* .
```

5. Check to see if a signing certificate has already been created

Use the “openssl ca” command to determine if a signing certificate has already been created. If one has already been created, skip to step 7.

Enter the openssl command:

```
# openssl ca
```

If the output looks something like this:

```
Using configuration from /etc/CertAuth/openssl.cnf
error loading the config file '/etc/CertAuth/openssl.cnf'
27722:error:02001002:system library:fopen:No such file or
directory:bss_file.c:104:fopen('/etc/CertAuth/openssl.cnfx','rb')
27722:error:2006D080:BIO routines:BIO_new_file:no such file:bss_file.c:107:
27722:error:0E064072:configuration file routines:CONF_load:no such
file:conf_def.c:197:
#
```

continue onto step 6 (a signing certificate does not exist). If the output looks something like:

```
Using configuration from /etc/CertAuth/openssl.cnf
#
```

then a certificate already has already been created. In this case, continue on to step 7.

6. Create a signing certificate

To create a new signing certificate, we create a directory to hold them, build a new OpenSSL configuration file, and create the certificate. In this tutorial, we will use the directory: /etc/CertAuth.

```
# #
# # certificate authority files do not exist, create them
# #
# cd /etc
# mkdir CertAuth; cd CertAuth
# mkdir certs
# echo '01' > serial
# touch index.txt
```

Create the new OpenSSL configuration file for GIAC Enterprises.

```
# cat > /etc/CertAuth/openssl.cnf
#
# OpenSSL configuration file for GIAC Enterprise's self-signed certificates
# The original configuration file is in /usr/share/ssl/openssl.cnf
#
####################################################################
[ ca ]
default_ca      = CA_default                    # The default ca section
####################################################################
[ CA_default ]
```

```
dir              = /etc/CertAuth                 # Where everything is kept
certificate      = $dir/cacert.crt               # The CA certificate
serial           = $dir/serial
database         = $dir/index.txt                # database index file.
new_certs_dir    = $dir/certs                    # default place for new certs.
private_key      = $dir/cacert.key               # The private key
certs            = $dir/certs                    # where the issued certs are
kept
default_days     = 700                           # how long to certify for
default_crl_days= 300                            # how long before next CRL
default_md       = md5                           # which md to use.
policy           = CertAuthCA_policy
x509_extensions = certificate_extensions         # The extensions to add to the
cert

# For the CA policy

[ CertAuthCA_policy ]
commonName              = supplied
stateOrProvinceName     = supplied
countryName             = supplied
emailAddress            = supplied
organizationName        = supplied
organizationalUnitName  = optional

[ certificate_extensions ]
basicConstraints=CA:FALSE
####################################################################
[ req ]
default_bits            = 2048
default_keyfile         = /etc/CertAuth/cacert.key
default_md              = md5
prompt                  = no
distinguished_name      = root_ca_distinguished_name
x509_extensions         = root_ca_extensions

[ root_ca_distinguished_name ]
commonName              = GIAC Enterprises Self-signed Certificate Authority
stateOrProvinceName     = Nevada
countryName             = US
emailAddress            = certs@example.com
organizationName        = Root Certification Authority

[ root_ca_extensions ]
basicConstraints        = CA:true
^D
```

Generate the signing certificate. The “openssl req” command arguments used are:

- req - we are doing X.509 Certificate Signing Request (CSR) Management.
- -nodes - we do not require a pass phrase to sign certificates.
- -new - generate a new key and certificate.
- -x509 - use the X509 Certificate display and signing utility.
- -keyout cacert.key -- place our private key in the cacert.key file.
- -out cacert.crt - place the generated certificate in the file “cacert.crt”.
- -days 10000 - this key is valid for 10000 days.

For more information read the openssl and x509 manual pages.

```
# #
# # point to our ssl configuration
# #
# export OPENSSL_CONF=/etc/CertAuth/openssl.cnf
# #
# # create our root certificate
# #
# openssl req -nodes -new -x509 -keyout cacert.key -out cacert.crt -days 10000
Generating a 2048 bit RSA private key
....................+++
.......................+++
writing new private key to 'cacert.key'
-----
```

It is a good idea to verify the new certificate

```
# #
# # verify the new CA certificate
# #
# openssl x509 -in cacert.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 0 (0x0)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: CN=GIAC Enterprises Self-signed Certificate Authority,
ST=Nevada, C=US/emailAddress=certs@example.com, O=Root Certification Authority
        Validity
            Not Before: Feb 13 18:24:40 2004 GMT
            Not After : Jul  1 18:24:40 2031 GMT
        Subject: CN=GIAC Enterprises Self-signed Certificate Authority,
ST=Nevada, C=US/emailAddress=certs@example.com, O=Root Certification Authority
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:d2:5f:d0:ac:35:2c:2a:b5:08:bb:6f:48:42:f7:
                    db:45:c2:9a:4c:cb:31:0a:16:54:5b:63:bb:75:60:
                    e9:94:f5:cd:fa:c8:29:8a:f2:aa:cf:48:60:fe:45:
                    8e:3c:49:d2:75:6e:47:9e:34:c5:ed:dd:d2:1a:5f:
                    77:a2:9c:67:a0:40:9e:d5:78:b9:28:c8:f7:bc:43:
                    6d:d4:c4:26:bb:f3:c0:ad:05:3e:11:65:cd:9d:36:
                    a0:55:29:e5:37:de:50:45:57:67:83:98:ce:66:19:
                    df:ce:86:d1:fb:1f:fc:42:e4:08:a9:3c:89:90:6c:
                    89:7b:ed:c2:fa:64:b6:9c:00:82:5b:71:9c:36:ba:
                    7c:5f:74:24:8c:6d:d1:45:c2:71:a1:7f:86:0e:d1:
                    4f:28:9e:20:b2:2d:c2:d1:e4:e7:74:f7:11:cd:b9:
                    c9:db:70:1e:60:9f:8a:52:40:82:34:2e:d5:dd:9a:
                    39:7a:f1:90:17:e6:bd:17:aa:88:c0:98:8e:3e:d4:
                    c7:e2:0d:b9:74:85:64:79:21:48:12:49:82:a9:07:
                    0f:e0:44:9f:ce:f3:1f:8b:30:c5:44:d4:a5:bd:d8:
                    0b:25:b4:76:b1:5c:8b:c0:3d:c8:7c:81:d1:36:cd:
                    dc:19:71:ed:a9:3a:eb:6d:03:a6:a5:d1:32:86:5b:
                    cc:f7
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
            CA:TRUE
    Signature Algorithm: md5WithRSAEncryption
        09:28:33:70:e1:7f:27:1d:2d:c0:6c:0b:18:f9:be:99:43:3f:
        82:5e:b3:49:4f:9f:0b:46:b8:75:6e:63:dd:eb:b9:96:62:62:
```

```
        68:f9:aa:cf:f3:c0:6c:09:84:10:3e:04:90:8c:81:e2:5f:bf:
        b4:f1:7d:a3:40:ca:f7:e2:19:92:bd:f3:06:53:f1:f8:34:fd:
        4f:1d:20:77:df:c7:15:e5:42:18:ec:68:af:a0:f8:62:a1:32:
        35:18:eb:8b:a4:5a:75:7a:6b:c5:38:56:e5:3e:99:2f:e9:41:
        38:54:24:7b:f8:46:31:a4:f2:11:11:3b:35:51:e6:85:8b:94:
        e4:da:01:74:86:62:4f:a4:99:29:a4:36:e7:5c:a6:3b:54:49:
        08:71:88:75:7f:43:a6:eb:f6:7f:ec:cb:bb:21:a6:20:fb:dd:
        9c:a0:ea:b3:98:ef:6f:de:d9:6c:e4:b5:b9:7f:86:e2:21:b8:
        16:87:46:a8:d0:d2:6d:8b:b9:9d:75:02:d8:48:e3:ac:30:dd:
        3a:1f:26:3a:f4:19:42:33:d4:e7:fd:fb:26:b1:03:cd:f0:5c:
        4e:a6:c4:eb:77:0d:0e:bc:86:1c:7a:b0:dc:64:9f:f6:85:3f:
        c7:31:b0:cc:d7:e6:6b:02:af:e7:3e:d3:40:3c:f6:f7:52:2d:
        1e:1a:75:b3
```

Finally, we need to create the Diffie-Hellman encryption engine parameter file to use 2048 bit keys. Note: IT REALLY CAN TAKE A LONG TIME TO GENERATE THE FILE, BE PATIENT.

```
# openssl dhparam -out dh2048.pem 2048
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time

*** OUTPUT DELETED to save space ***
#
```

7. Create certificates for both VPN endpoint clients

Now we are ready to start the process of generating SSL certificates for each of the VPN endpoints (west and east site). The previous steps were in preparation of step 7 and probably will not need to be repeated (except for mounting the VPN root file systems). NOTE: You can skip this step if a certificate has already been generated (i.e. if a partner company provides a certificate to be used for a cooperative VPN link).

For this tutorial we will be generating a certificate for both the west (gw1b) and the east site (gw2b).

Generate the user certificate. The “openssl req” command arguments used are:

- req - we are doing X.509 Certificate Signing Request (CSR) Management.
- -nodes - do not require a DES password to use the certificate.
- -new - generate a new certificate.
- -keyout gw1b.key - place the generated key in the file “gw1b.key”.
- -out gw1b.csr - place the certificate signing request in the file “gw1b.csr”.
- -config /usr/share/ssl/openssl.cnf - use the default system configuration file for openssl. Note: You should make sure that if you are generating user certificates on the same system that is the CA, you do not accidentally use the CA’s openssl.cnf. It will not include all of the necessary information in the request. You must use this argument if you had previously entered the “export OPENSSL_CONF” command listed in step 6.

For more information read the openssl and x509 manual pages.

```
# #
# # create certificate for gw1b
# #
# openssl req -nodes -new -keyout gw1b.key -out gw1b.csr -config
       /usr/share/ssl/openssl.cnf
Generating a 1024 bit RSA private key
.....................++++++
...........++++++
writing new private key to 'gw1b.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:Nevada
Locality Name (eg, city) [Newbury]:Las Vegas
Organization Name (eg, company) [My Company Ltd]:GIAC Enterprises West Site
Organizational Unit Name (eg, section) []:IT
Common Name (eg, your name or your server's hostname) []:gw1b
Email Address []:certs@example.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

The lower part of the request includes contact information for the certificate. This information is accessible to anyone that can establish a SSL session with the VPN endpoint. Because of this, you should consider not including sensitive or personally identifiable information in these fields. Such information could be used to perform a social engineering attack.

Now, we do the same thing for the other endpoint gw2b.

```
# #
# # create certificate for gw2b
# #
# openssl req -nodes -new -keyout gw2b.key -out gw2b.csr -config
       /usr/share/ssl/openssl.cnf
Generating a 1024 bit RSA private key
.....................++++++
......++++++
writing new private key to 'gw2b.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:New York
Locality Name (eg, city) [Newbury]:New York
Organization Name (eg, company) [My Company Ltd]:GIAC Enterprises East Site
```

```
Organizational Unit Name (eg, section) []:IT
Common Name (eg, your name or your server's hostname) []:gw2b
Email Address []:certs@example.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
#
```

If we were not doing our own certificate signing, we would then send the gw1b.csr and gw2b.csr files to the signing authority.

8. Sign the VPN endpoint client certificate requests

Once a certificate signing request has been received, or generated as above. It needs to be signed by the Certificate Authority. In order to sign a certificate, you must know the pass phrase that was used, if any, to generate the CA (see step 6 above). We are not using pass phrases in this tutorial.

This process takes the certificate signing request file for the client and produces a signed certificate as output. For the west site client, these files are gw1b.csr and gw1b.crt respectively.

For more information read the “openssl ca” man page.

```
# export OPENSSL_CONF=/etc/CertAuth/openssl.cnf
# openssl ca -out gw1b.crt -in gw1b.csr
Using configuration from /etc/CertAuth/openssl.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :PRINTABLE:'Nevada'
localityName          :PRINTABLE:'Las Vegas'
organizationName      :PRINTABLE:'GIAC Enterprises West Site'
organizationalUnitName:PRINTABLE:'IT'
commonName            :PRINTABLE:'gw1b'
emailAddress          :IA5STRING:'certs@example.com'
Certificate is to be certified until Jan 13 18:46:09 2006 GMT (700 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
#
```

Now generate the other endpoint’s certificate:

```
# openssl ca -out gw2b.crt -in gw2b.csr
Using configuration from /etc/CertAuth/openssl.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :PRINTABLE:'New York'
```

```
localityName           :PRINTABLE:'New York'
organizationName       :PRINTABLE:'GIAC Enterprises East Site'
organizationalUnitName:PRINTABLE:'IT'
commonName             :PRINTABLE:'gw2b'
emailAddress           :IA5STRING:'certs@example.com'
Certificate is to be certified until Jan 13 18:47:50 2006 GMT (700 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
#
```

At this point, you would normally send the gw1b.crt and gw2b.crt files back to their respective site's network staff. However, since we are building custom root file systems, we will not have to do this.

9. Create the customized root file systems

In this step, we will begin customizing the VPN end point file systems. One of the end point systems will have to be designated as the server and the other one as the client. For convenience we will treat gw1b (the west site system) as the server, and gw2b (east site) as the client.

First we need to unmount the template file system:

```
# cd /
# umount /mnt/uml
```

Copy the template file system to the two custom file systems (one for each end point)

```
# cp /var/vnuml/root_fs_vpn_temp /var/vnuml/root_fs_gw1b
# cp /var/vnuml/root_fs_vpn_temp /var/vnuml/root_fs_gw2b
```

Create two additional mount points for the custom file systems (the mount points may already exist)

```
# mkdir /mnt/uml1 /mnt/uml2
```

Mount the custom file systems:

```
# mount -o loop /var/vnuml/root_fs_vpn_gw1b /mnt/uml1
# mount -o loop /var/vnuml/root_fs_vpn_gw2b /mnt/uml2
```

Create empty directory trees to hold each of the VPN configurations:

```
# cd /mnt/uml1/etc
# mkdir vpn vpn/bin
# chmod -R 700 /mnt/uml1/etc/vpn

# cd /mnt/uml2/etc
# mkdir vpn vpn/bin
# chmod -R 700 /mnt/uml2/etc/vpn
```

10. Create the VPN server-side (gw1b) configuration file

Create a text file called /mnt/uml1/etc/vpn/gw1b-vpn.conf containing the following lines. Do not enter the numbers at the beginning of each line; they are used below to describe what purpose each line has in the configuration. Any line in this file that begins with a pound sign (“#”) or a semi-colon are considered comments and will not affect the configuration.

```
# cat > /mnt/uml1/etc/vpn/gw1b-vpn.conf

  1.    dev tun
  2.    port 5000
  3.    ; user nobody
  4.    ; group nobody
  5.    comp-lzo
  6.    verb 3
  7.    #shaper 1000
  8.    remote 10.2.20.6
  9.    ifconfig 10.1.40.6 10.2.40.6
 10.    up /etc/vpn/ip-up.sh
 11.    tls-server
 12.    #auth alg=none
 13.    #cipher alg=none
 14.    #keysize 192
 15.    ca /etc/vpn/cacert.crt
 16.    cert /etc/vpn/gw1b.crt
 17.    key /etc/vpn/gw1b.key
 18.    dh /etc/vpn/dh2048.pem
 19.    tls-cipher RC4-MD5
```

Lines 1 and 2, configures OpenVPN to use the network tunnel device for VPN traffic on UDP port 5000. Lines 3 and 4 are commented out, but if active will have the OpenVPN daemon change its user and group to an account with limited capabilities. Line 5 compresses VPN traffic using the LZO library. Line 6 sets the verbosity level of the OpenVPN daemon to medium (status and some debugging messages are printed). Uncommenting line 7 will turn on traffic shaping, the value of 1000 will limit VPN traffic to approximately two-thirds of a T-1 line (1000Kb/s out of 1540Kb/s bandwidth).

Line 8 identifies the IP address of the system located at the other end of the UDP tunnel. Note: this is not the VPN addresses, just the tunnel. Line 9 configures the VPN address for the two ends of the VPN link (for our example, 10.1.40.6 is gw1b’s end, and 10.2.40.6 is gw2b’s end. Line 10 provides the name of the helper script to add route information when starting the VPN link. Note: These path names are relative to the root of the VPN file system, not to the development system. The rest of the file configures the SSL library. Lines 16-18 load gw1b’s certificate information into OpenVPN. Lines 19 and 20 configure the encryption engine to use the MD5 algorithm with 2048 length keys.

Note: The IP numbers used in lines 8 and 9 must be the same as those used below in the gw2b-vpn.conf configuration file.

11. Complete the configuration for the VPN server side (gw1b)

All of the commands in this step take place in the root_fs_vpn_gw1b file system (/mnt/uml1/etc/vpn). Make sure that you are in the correct place.

Change the gw1b's /etc/vpn directory.

```
# cd /mnt/uml1/etc/vpn
```

Copy the certificates and keys needed by gw1b from the certificate authority:

```
# cp /etc/CertAuth/cacert.crt cacert.crt
# cp /etc/CertAuth/gw1b.crt gw1b.crt
# cp /etc/CertAuth/gw1b.key gw1b.key
# cp /etc/CertAuth/dh2048.pem dh2048.pem
```

Finally, we need to create the ip-up.sh shell file.

```
# cat > ip-up.sh
```

it contains the command:

```
#!/bin/bash
route add -net 10.2.0.0 netmask 255.255.255.0 gw $5
```

Make sure that this file is executable:

```
# chmod 700 /etc/vpn/ip-up.sh
```

This will add an entry to the routing table to direct traffic for the east site network through the VPN tunnel. The name of the tunnel device will be passed from the OpenVPN daemon as the fifth argument to the helper script.

12. Create the VPN client-side (gw2b) configuration file

This is done the same way as for the client side, except for changing some of the parameters in the configuration files. We are going to create a text file called /mnt/uml2/etc/vpn/gw2b-vpn.conf. Again, don't type in the line numbers.

```
# cat > /mnt/uml2/etc/vpn/gw2b-vpn.conf
```

```
 1.    dev tun
 2.    port 5000
 3.    ; user nobody
 4.    ; group nobody
 5.    comp-lzo
 6.    verb 3
 7.    #shaper 1000
 8.    remote 10.1.20.6
 9.    ifconfig 10.2.40.6 10.1.40.6
10.    up /etc/vpn/ip-up.sh
11.    tls-client
12.    #auth alg=none
```

```
13.    #cipher alg=none
14.    #keysize 192
15.    ca /etc/vpn/cacert.crt
16.    cert /etc/vpn/gw2b.crt
17.    key /etc/vpn/gw2b.key
18.    dh /etc/vpn/dh2048.pem
19.    tls-cipher RC4-MD5
```

Lines 1 and 2, configures OpenVPN to use the network tunnel device for VPN traffic on UDP port 5000. Lines 3 and 4 are commented out, but if active will have the OpenVPN daemon change its user and group to an account with limited capabilities. Line 5 compresses VPN traffic using the LZO library. Line 6 sets the verbosity level of the OpenVPN daemon to medium (status and some debugging messages are printed). Uncommenting line 7 will turn on traffic shaping, the value of 1000 will limit VPN traffic to approximately two-thirds of a T-1 line (1000Kb/s out of 1540Kb/s bandwidth).

Line 8 identifies the IP address of the system located at the other end of the UDP tunnel. Note: this is not the VPN addresses, just the tunnel. Line 9 configures the VPN address for the two ends of the VPN link (for our example, 10.1.40.6 is gw1b's end, and 10.2.40.6 is gw2b's end. Line 10 provides the name of the helper script to add route information when starting the VPN link. Note: These path names are relative to the root of the VPN file system, not to the development system. The rest of the file configures the SSL library. Lines 16-18 load gw2b's certificate information into OpenVPN. Lines 19 and 20 configure the encryption engine to use the MD5 algorithm with 2048 length keys.

Note: The IP numbers used in lines 8 and 9 must be the same as those used above in the gw1b-vpn.conf configuration file.

13. Complete the configuration for the VPN client side (gw2b)

All of the commands in this step take place in the root_fs_vpn_gw2b file system (/mnt/uml2/etc/vpn). Make sure that you are in the correct place.

Change the gw2b's /etc/vpn directory.

```
# cd /mnt/uml2/etc/vpn
```

Copy the certificates and keys needed by gw2b from the certificate authority:

```
# cp /etc/CertAuth/cacert.crt cacert.crt
# cp /etc/CertAuth/gw1b.crt gw2b.crt
# cp /etc/CertAuth/gw1b.key gw2b.key
# cp /etc/CertAuth/dh2048.pem dh2048.pem
```

Finally, we need to create the ip-up.sh shell file.

```
# cat > ip-up.sh
```

it contains the command:

```
#!/bin/bash
route add -net 10.1.0.0 netmask 255.255.255.0 gw $5
```

Make sure that this file is executable:

```
# chmod 700 /etc/vpn/ip-up.sh
```

This will add an entry to the routing table to direct traffic for the west site network through the VPN tunnel. The name of the tunnel device will be passed from the OpenVPN daemon as the fifth argument to the helper script.

14. Unmount the custom file systems

We now need to unmount the file systems that we have customized. It is also a good idea to perform a file system check on them to make sure that they have not been corrupted.

```
# umount /mnt/uml1
# umount /mnt/uml2
# e2fsck root_fs_vpn_gw1b
e2fsck 1.34 (25-Jul-2003)
root_fs_vpn_gw1b: clean, 25251/114912 files, 327771/460800 blocks
# e2fsck root_fs_vpn_gw2b
e2fsck 1.34 (25-Jul-2003)
root_fs_vpn_gw2b: clean, 25251/114912 files, 327771/460800 blocks
#
```

15. Cleaning things up

We have now successfully customized the root file systems for the VPN gateways. If desired, now we can remove the private keys and certificates for the VPN gateways that are stored on the Certificate Authority. This is not necessary, but if the Certificate Authority system is compromised, then all traffic sent on the VPN link could be comprised as well.
Note: Do not just use the “rm” command, since it will only remove the directory entry while leaving the file contents still on disk. Use “shred” instead.

```
# shred -u /etc/CertAuth/gw1b.key /etc/CertAuth/gw1b.crt
# shred -u /etc/CertAuth/gw2b.key /etc/CertAuth/gw2b.crt
```

16. Preparing to Test the VPN configuration by simulating the network

One of the big advantages to using User Mode Linux (http://user-mode-linux.sourceforge.net) at GIAC Enterprises is that it is extremely easy to test network configurations without affecting any of the production systems. Using the vnumlparser.pl tool developed by the Virtual Network User Mode Linux (VNUML) project (http://jungla.dit.upm.es/~vnuml) makes it even easier.

As a preliminary test to verify that the configuration files are correct, we will create a small network focused around the two VPN gateways. If we can successfully establish a VPN tunnel, the configuration files are correct, and we can then verify that the VPN security policies are correct. We will do that in a later step. Note: This test file assumes that no firewall filtering or NAT rules loaded on any of the routers. For testing purposes the external IP addresses for the router1 is set to 10.0.3.1 and the east site is set to 10.0.3.100.

The vnumlparser.pl script has already been installed on the development system. There is also a test simulation contained in the file vpn-test.xml. A copy of this file is listed below. This file creates a simple test platform of the GIAC corporate network. This simulation contains the following systems (but not the virtual servers): router1, router2, router3, netmgr, www1, corp1, gw1b, netmgr2, www2, corp2, and gw2b. Please refer to the vnuml language manual for more information (http://jungla.dit.upm.es/~vnuml/doc/1.3/reference).

To start the test network simulation, execute the following command:

```
# vnumlparser.pl -t vpn-test.xml -o /tmp/vpn-test -v
```

The “vnumlprser.pl” command arguments used are:

- -t vpn-test.xml -- create the network simulation using the description file “vpn-test.xml” (see below).
- -o /tmp/vpn-test -- create the individual host system’s log files prefixed by “/tmp/vpn-test”. For example, this will create a log file for the gw1b host with the name: /tmp/vpn-test.gw1b.
- -v -- verbose output, display a log of the commands used to generate the network simulation.

What will happen after this command completes, is the network simulation will be executing as a series of UML instances. Each instance will be accessible by using the “slogin” or “ssh” client program to login into the instance. Each instance will have its name added to the host system’s /etc/host file. Note: only version 1 SSH connections are supported in this manner. This is not a large security risk, since the version 1 SSH connections are internal to the host machine running the network simulation.

Contents of vpn-test.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/var/vnuml/vnuml.dtd">
<vnuml>
<global>
   <version>1.3</version>
   <simulation_name>vpn-test</simulation_name>
   <ssh_key>/root/.ssh/identity.pub</ssh_key>
   <automac/>
   <ip_offset>100</ip_offset>
   <host_mapping/>
</global>
<net name="west0"/>
```

```
<net name="west1"/>
<net name="west2"/>
<net name="west3"/>
<net name="west4"/>    <!-- west site vpn network segment -->
<net name="east0"/>
<net name="east1"/>
<net name="east2"/>
<net name="east3"/>
<net name="east4"/>
<net name="east5"/>    <!-- east site vpn network segment -->
<net name="INET"/>
<!-- west site network -->
<vm name="netmgr">
   <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
   <if id="1" net="west0">
      <ipv4>10.1.10.251</ipv4>
   </if>
</vm>
<vm name="router1">
   <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
   <if id="1" net="INET">
      <ipv4>10.0.3.1</ipv4>
   </if>
   <if id="2" net="west0">
      <ipv4>10.1.10.249</ipv4>
   </if>
   <if id="3" net="west1">
      <ipv4>10.1.1.249</ipv4>
   </if>
   <route type="inet" gw="10.1.1.248">10.1.1.0/24</route>
   <route type="inet" gw="10.1.1.248">10.1.20.0/24</route>
   <route type="inet" gw="10.1.1.248">10.1.30.0/24</route
   <route type="inet" gw="10.0.3.2">default</route>
   <forwarding type="ip" />
</vm>
<vm name="www1">
   <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
   <if id="1" net="west0">
      <ipv4>10.1.10.3</ipv4>
   </if>
   <if id="2" net="west1">
      <ipv4>10.1.1.3</ipv4>
   </if>
   <route type="inet" gw="10.1.1.249">default</route>
   <forwarding type="ip" />
</vm>
<vm name="router2">
   <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
   <if id="1" net="west1">
      <ipv4>10.1.1.248</ipv4>
   </if>
   <if id="2" net="west0">
      <ipv4>10.1.10.4</ipv4>
   </if>
   <if id="3" net="west2">
      <ipv4>10.1.20.248</ipv4>
   </if>
   <if id="4" net="west3">
      <ipv4>10.1.30.248</ipv4>
   </if>
   <route type="inet" gw="10.1.1.249">default</route>
   <forwarding type="ip" />
</vm>
```

```
<vm name="corp1">
   <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
   <if id="1" net="west3">
      <ipv4>10.1.30.1</ipv4>
   </if>
   <route type="inet" gw="10.1.30.248">default</route>
</vm>
<vm name="gw1b">
   <filesystem type="cow">/var/vnuml/root_fs_vpn_gw1b</filesystem>
   <if id="1" net="west0">
      <ipv4>10.1.10.6</ipv4>
   </if>
   <if id="2" net="west2">
      <ipv4>10.1.20.6</ipv4>
   </if>
   <if id="3" net="west4">
      <ipv4>10.1.40.6</ipv4>
   </if>
   <route type="inet" gw="10.1.20.248">default</route>
</vm>
<!-- east site network -->
<vm name="netmgr2">
   <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
   <if id="1" net="east0">
      <ipv4>10.2.10.251</ipv4>
   </if>
</vm>
<vm name="router3">
   <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
   <if id="1" net="INET">
      <ipv4>10.0.3.100</ipv4>
   </if>
   <if id="2" net="east0">
      <ipv4>10.2.10.249</ipv4>
   </if>
   <if id="3" net="east1">
      <ipv4>10.2.1.249</ipv4>
   </if>
   <if id="4" net="east3">
      <ipv4>10.2.30.249</ipv4>
   </if>
   <if id="5" net="east4">
      <ipv4>10.2.20.249</ipv4>
   </if>
   <route type="inet" gw="10.0.3.2">default</route>
   <forwarding type="ip" />
</vm>
<vm name="www2">
   <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
   <if id="1" net="east0">
      <ipv4>10.2.10.3</ipv4>
   </if>
   <if id="2" net="east1">
      <ipv4>10.2.1.3</ipv4>
   </if>
   <route type="inet" gw="10.2.1.249">default</route>
   <forwarding type="ip" />
</vm>
<vm name="corp2">
   <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
   <if id="1" net="east3">
      <ipv4>10.2.30.1</ipv4>
   </if>
```

```
      <route type="inet" gw="10.2.30.249">default</route>
   </vm>
   <vm name="gw2b">
      <filesystem type="cow">/var/vnuml/root_fs_vpn_gw2b</filesystem>
      <if id="1" net="east0">
         <ipv4>10.2.10.6</ipv4>
      </if>
      <if id="2" net="east4">
         <ipv4>10.2.20.6</ipv4>
      </if>
      <if id="3" net="east5">
         <ipv4>10.2.40.6</ipv4>
      </if>
      <route type="inet" gw="10.2.20.249">default</route>
      <forwarding type="ip" />
   </vm>
   <!-- host connection -->
   <host>
      <hostif net="INET">
         <ipv4>10.0.3.2</ipv4>
      </hostif>
      <route type="inet" gw="10.0.3.1">10.1.0.0/16</route>
      <route type="inet" gw="10.0.3.100">10.2.0.0/16</route>
   </host>
</vnuml>
```

17. Starting the VPN Tunnel

From the host system, open two new terminal sessions, one to gw1b (west site gateway) and the other to gw2b (east site gateway). (Note: these are the simulated gateways, not the production ones). For this tutorial, we assume that you are working from the console with the

```
# xterm -e “slogin -1 gw1b” &
# xterm -e “slogin -1 gw2b” &
```

In the gw1b window, enter the commands:

```
gw1b:~# cd /etc/vpn
gw1b:/etc/vpn# openvpn --config gw1b-vpn.conf &
```

In the gw2b window, enter the commands:

```
gw2b:~# cd /etc/vpn
gw2b:/etc/vpn# openvpn --config gw2b-vpn.conf &
```

This will result in output on gw1b that is similar to the following (items of particular interest are shown in **bold**):

```
Thu Feb 19 21:19:12 2004 0: OpenVPN 1.5.0 i686-pc-linux-gnu [SSL] [LZO] built
on Feb 14 2004
Thu Feb 19 21:19:12 2004 1: Diffie-Hellman initialized with 2048 bit key
Thu Feb 19 21:19:12 2004 3: LZO compression initialized
```

The above two messages indicates that the server side of the connection (gw1b) was able to initialize the 2048 bit save Diffie-Hellman key that was created in step 11, and LZO compression is active.

```
Thu Feb 19 21:19:12 2004 4: Control Channel MTU parms [ L:1300 D:138 EF:38 EB:0
ET:0 ]
Thu Feb 19 21:19:12 2004 5: TUN/TAP device tun0 opened
Thu Feb 19 21:19:12 2004 6: /sbin/ifconfig tun0 10.1.40.6 pointopoint 10.2.40.6
mtu 1258
Thu Feb 19 21:19:13 2004 7: /etc/vpn/ip-up.sh tun0 1258 1300 10.1.40.6
10.2.40.6 init
Thu Feb 19 21:19:13 2004 8: Data Channel MTU parms [ L:1300 D:1300 EF:42 EB:19
ET:0 ]
Thu Feb 19 21:19:13 2004 9: Local Options hash (VER=V3): 'b316f64b'
Thu Feb 19 21:19:13 2004 10: Expected Remote Options hash (VER=V3): '8657019a'
Thu Feb 19 21:19:13 2004 11: UDPv4 link local (bound): [undef]:5000
Thu Feb 19 21:19:13 2004 12: UDPv4 link remote: 10.2.20.6:5000
Thu Feb 19 21:19:13 2004 13: read UDPv4 [ECONNREFUSED]: Connection refused
(code=111)
Thu Feb 19 21:19:15 2004 14: TLS: tls_pre_decrypt: first response to initial
packet from 10.2.20.6:5000, sid=b708d8c0 a6473816
Thu Feb 19 21:19:15 2004 15: VERIFY OK: depth=1, /CN=GIAC.Enterprises.Self-
signed.Certificate.Authority/ST=Nevada/C=US/emailAddress=certs@example.com/O=Ro
ot.Certification.Authority
Thu Feb 19 21:19:15 2004 16: VERIFY OK: depth=0,
/CN=gw1b/ST=Nevada/C=US/emailAddress=certs@example.com/O=GIAC.Enterprises.West.
Site/OU=IT
Thu Feb 19 21:19:15 2004 17: Data Channel Decrypt: Cipher 'BF-CBC' initialized
with 128 bit key
Thu Feb 19 21:19:15 2004 18: Data Channel Decrypt: Using 160 bit message hash
'SHA1' for HMAC authentication
Thu Feb 19 21:19:15 2004 19: Data Channel Encrypt: Cipher 'BF-CBC' initialized
with 128 bit key
Thu Feb 19 21:19:15 2004 20: Data Channel Encrypt: Using 160 bit message hash
'SHA1' for HMAC authentication
Thu Feb 19 21:19:15 2004 21: Control Channel: TLSv1, cipher TLSv1/SSLv3 RC4-
MD5, 1024 bit RSA
Thu Feb 19 21:19:15 2004 22: Peer Connection Initiated with 10.2.20.6:5000
```

The bolded message “Connection refused,” indicates that gw1b is waiting for the other VPN endpoint on gw2b to come up. This message will repeat until the other endpoint is available.

The “VERIFY OK” messages shows the information describing the certificates that was used to establish the encrypted SSL connection. The final two messages, shows that the VPN link is using a 1024 bit SSL control channel and that the other endpoint is active on port 5000 of gw2b (10.2.20.6)

The output displayed on gw2b is very similar, with the same interpretation as above.

```
Thu Feb 19 21:19:14 2004 0: OpenVPN 1.5.0 i686-pc-linux-gnu [SSL] [LZO] built
on Feb 14 2004
Thu Feb 19 21:19:14 2004 2: LZO compression initialized
Thu Feb 19 21:19:14 2004 3: Control Channel MTU parms [ L:1300 D:138 EF:38 EB:0
ET:0 ]
Thu Feb 19 21:19:14 2004 4: TUN/TAP device tun0 opened
Thu Feb 19 21:19:14 2004 5: /sbin/ifconfig tun0 10.2.40.6 pointopoint 10.1.40.6
mtu 1258
Thu Feb 19 21:19:15 2004 6: /etc/vpn/ip-up.sh tun0 1258 1300 10.2.40.6
```

```
10.1.40.6 init
Thu Feb 19 21:19:15 2004 7: Data Channel MTU parms [ L:1300 D:1300 EF:42 EB:19
ET:0 ]
Thu Feb 19 21:19:15 2004 8: Local Options hash (VER=V3): '8657019a'
Thu Feb 19 21:19:15 2004 9: Expected Remote Options hash (VER=V3): 'b316f64b'
Thu Feb 19 21:19:15 2004 10: UDPv4 link local (bound): [undef]:5000
Thu Feb 19 21:19:15 2004 11: UDPv4 link remote: 10.1.20.6:5000
Thu Feb 19 21:19:15 2004 12: TLS: tls_pre_decrypt: first response to initial
packet from 10.1.20.6:5000, sid=d0f3d4a7 0dbe96f5
Thu Feb 19 21:19:15 2004 13: Data Channel Encrypt: Cipher 'BF-CBC' initialized
with 128 bit key
Thu Feb 19 21:19:15 2004 14: Data Channel Encrypt: Using 160 bit message hash
'SHA1' for HMAC authentication
Thu Feb 19 21:19:15 2004 15: VERIFY OK: depth=1, /CN=GIAC.Enterprises.Self-
signed.Certificate.Authority/ST=Nevada/C=US/emailAddress=certs@example.com/O=Ro
ot.Certification.Authority
Thu Feb 19 21:19:15 2004 16: VERIFY OK: depth=0,
/CN=gw1b/ST=Nevada/C=US/emailAddress=certs@example.com/O=GIAC.Enterprises.West.
Site/OU=IT
Thu Feb 19 21:19:15 2004 17: Data Channel Decrypt: Cipher 'BF-CBC' initialized
with 128 bit key
Thu Feb 19 21:19:15 2004 18: Data Channel Decrypt: Using 160 bit message hash
'SHA1' for HMAC authentication
Thu Feb 19 21:19:15 2004 19: Control Channel: TLSv1, cipher TLSv1/SSLv3 RC4-
MD5, 1024 bit RSA
Thu Feb 19 21:19:15 2004 20: Peer Connection Initiated with 10.1.20.6:5000
```

18. Basic Functionality Test of VPN Tunnel

To determine that the VPN tunnel is functioning at a basic level, look at the network routing table entries on each VPN gateway. The items of interest are bolded.

```
gw1b:/etc/vpn# netstat -ran
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.2.40.6       0.0.0.0         255.255.255.255 UH        0 0          0 tun0
192.168.1.36    0.0.0.0         255.255.255.252 U         0 0          0 eth0
10.1.40.0       0.0.0.0         255.255.255.0   U         0 0          0 eth3
10.1.10.0       0.0.0.0         255.255.255.0   U         0 0          0 eth1
10.1.20.0       0.0.0.0         255.255.255.0   U         0 0          0 eth2
10.2.40.0       10.2.40.6       255.255.255.0   UG        0 0          0 tun0
0.0.0.0         10.1.20.248     0.0.0.0         UG        0 0          0 eth2
```

These two lines show that the client (gw2b) end of the VPN tunnel is located on the tun0 interface and that a routing entry for the opposite VPN (10.2.40.0/24) is available.

Gw2b shows similar entries for the server (gw1b) end of the VPN tunnel:

```
gw2b:/etc/vpn# netstat -ran
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.1.40.6       0.0.0.0         255.255.255.255 UH        0 0          0 tun0
192.168.1.56    0.0.0.0         255.255.255.252 U         0 0          0 eth0
10.2.20.0       0.0.0.0         255.255.255.0   U         0 0          0 eth2
10.1.40.0       10.1.40.6       255.255.255.0   UG        0 0          0 tun0
10.2.40.0       0.0.0.0         255.255.255.0   U         0 0          0 eth3
10.2.10.0       0.0.0.0         255.255.255.0   U         0 0          0 eth1
```

```
0.0.0.0         10.2.20.249     0.0.0.0         UG        0 0          0 eth2
gw2b:/etc/vpn#
```

Now, we want to see if packets are actually sent across the VPN link (ping test):

```
gw1b:/etc/vpn# ping -c 3 10.2.40.6
PING 10.2.40.6 (10.2.40.6) 56(84) bytes of data.
64 bytes from 10.2.40.6: icmp_seq=1 ttl=64 time=8.86 ms
64 bytes from 10.2.40.6: icmp_seq=2 ttl=64 time=8.03 ms
64 bytes from 10.2.40.6: icmp_seq=3 ttl=64 time=8.25 ms

--- 10.2.40.6 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2123ms
rtt min/avg/max/mdev = 8.037/8.384/8.862/0.357 ms

On gw2b, we get the following:

gw2b:/etc/vpn# ping -c 3 10.1.40.6
PING 10.1.40.6 (10.1.40.6) 56(84) bytes of data.
64 bytes from 10.1.40.6: icmp_seq=1 ttl=64 time=7.59 ms
64 bytes from 10.1.40.6: icmp_seq=2 ttl=64 time=7.99 ms
64 bytes from 10.1.40.6: icmp_seq=3 ttl=64 time=8.33 ms

--- 10.1.40.6 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2126ms
rtt min/avg/max/mdev = 7.596/7.975/8.331/0.309 ms
```

Since there is no packet loss in either direction, the VPN tunnel is functioning at a basic level.

19. Basic Functionality Test of VPN Tunnel Encryption

The previous step shows that packets are transmitted over the VPN link, but did not verify that their contents are being correctly encrypted. To verify that the packets are being sent encrypted, we use the tcpdump utility to capture the sent packets and look at their contents.

Start the tcpdump utility on gw1b so it listens on the network interface:

```
gw1b:/etc/vpn# tcpdump -nn -X -i eth2

tcpdump: listening on eth2
```

At this point, gw1b will wait until network traffic is received on eth2.

On gw2b, enter the following command to request the time of day from gw1b:

```
gw2b:/etc/vpn# telnet 10.1.20.6 daytime
Trying 10.1.20.6...
Connected to 10.1.20.6.
Escape character is '^]'.
Thu Feb 19 21:50:45 2004
Connection closed by foreign host.
```

Since we are using telnet on the regular network interface, we should expect to see unencrypted network traffic with tcpdump:

```
21:50:45.505473 arp who-has 10.1.20.6 tell 10.1.20.248
0x0000   0001 0800 0604 0001 fefd 0000 0403 0a01        ................
0x0010   14f8 0000 0000 0000 0a01 1406                   ............
21:50:45.505682 arp reply 10.1.20.6 is-at fe:fd:0:0:6:2
0x0000   0001 0800 0604 0002 fefd 0000 0602 0a01        ................
0x0010   1406 fefd 0000 0403 0a01 14f8                   ............
21:50:45.513776 10.2.20.6.1027 > 10.1.20.6.13: S 2470422499:2470422499(0) win
5840 <mss 1460,sackOK,timestamp 146108 0,nop,wscale 0> (DF) [tos 0x10]
0x0000   4510 003c 9d5f 4000 3c06 653e 0a02 1406        E..<._@.<.e>....
0x0010   0a01 1406 0403 000d 933f a7e3 0000 0000        .........?......
0x0020   a002 16d0 7b37 0000 0204 05b4 0402 080a        ....{7..........
0x0030   0002 3abc 0000 0000 0103 0300                   ..:.........
21:50:45.514165 10.1.20.6.13 > 10.2.20.6.1027: S 2490106758:2490106758(0) ack
2470422500 win 5792 <mss 1460,sackOK,timestamp 147956 146108,nop,wscale 0> (DF)
0x0000   4500 003c 0000 4000 4006 fead 0a01 1406        E..<..@.@.......
0x0010   0a02 1406 000d 0403 946c 0386 933f a7e4        .........l...?..
0x0020   a012 16a0 a16d 0000 0204 05b4 0402 080a        .....m..........
0x0030   0002 41f4 0002 3abc 0103 0300                   ..A...:.....
21:50:45.527593 10.2.20.6.1027 > 10.1.20.6.13: . ack 1 win 5840
<nop,nop,timestamp 146108 147956> (DF) [tos 0x10]
0x0000   4510 0034 9d60 4000 3c06 6545 0a02 1406        E..4.`@.<.eE....
0x0010   0a01 1406 0403 000d 933f a7e4 946c 0387        .........?...l..
0x0020   8010 16d0 d002 0000 0101 080a 0002 3abc        ..............:.
0x0030   0002 41f4                                       ..A.
21:50:45.528687 10.1.20.6.13 > 10.2.20.6.1027: P 1:27(26) ack 1 win 5792
<nop,nop,timestamp 147956 146108> (DF)
0x0000   4500 004e 4273 4000 4006 bc28 0a01 1406        E..NBs@.@..(....
0x0010   0a02 1406 000d 0403 946c 0387 933f a7e4        .........l...?..
0x0020   8018 16a0 c769 0000 0101 080a 0002 41f4        .....i........A.
0x0030   0002 3abc 5468 7520 4665 6220 3139 2032        ..:.Thu.Feb.19.2
0x0040   313a 3530 3a34 3520 3230 3034 0d0a             1:50:45.2004..
21:50:45.529026 10.1.20.6.13 > 10.2.20.6.1027: F 27:27(0) ack 1 win 5792
<nop,nop,timestamp 147956 146108> (DF)
0x0000   4500 0034 4274 4000 4006 bc41 0a01 1406        E..4Bt@.@..A....
0x0010   0a02 1406 000d 0403 946c 03a1 933f a7e4        .........l...?..
0x0020   8011 16a0 d017 0000 0101 080a 0002 41f4        ..............A.
0x0030   0002 3abc                                       ..:.
21:50:45.535576 10.2.20.6.1027 > 10.1.20.6.13: . ack 27 win 5840
<nop,nop,timestamp 146108 147956> (DF) [tos 0x10]
0x0000   4510 0034 9d61 4000 3c06 6544 0a02 1406        E..4.a@.<.eD....
0x0010   0a01 1406 0403 000d 933f a7e4 946c 03a1        .........?...l..
0x0020   8010 16d0 cfe8 0000 0101 080a 0002 3abc        ..............:.
0x0030   0002 41f4                                       ..A.
21:50:45.540431 10.2.20.6.1027 > 10.1.20.6.13: F 1:1(0) ack 28 win 5840
<nop,nop,timestamp 146108 147956> (DF) [tos 0x10]
0x0000   4510 0034 9d62 4000 3c06 6543 0a02 1406        E..4.b@.<.eC....
0x0010   0a01 1406 0403 000d 933f a7e4 946c 03a2        .........?...l..
0x0020   8011 16d0 cfe6 0000 0101 080a 0002 3abc        ..............:.
0x0030   0002 41f4                                       ..A.
21:50:45.540965 10.1.20.6.13 > 10.2.20.6.1027: . ack 2 win 5792
<nop,nop,timestamp 147956 146108> (DF)
0x0000   4500 0034 4275 4000 4006 bc40 0a01 1406        E..4Bu@.@..@....
0x0010   0a02 1406 000d 0403 946c 03a2 933f a7e5        .........l...?..
0x0020   8010 16a0 d016 0000 0101 080a 0002 41f4        ..............A.
0x0030   0002 3abc
```

Now, resend the request over the VPN tunnel:

```
gw2b:/etc/vpn# telnet 10.1.40.6 daytime
Trying 10.1.40.6...
Connected to 10.1.40.6.
Escape character is '^]'.
```

```
Thu Feb 19 21:52:56 2004
Connection closed by foreign host.
```

Looking at the packet data, we see only “garbage” (this is good, the data is encrypted).

```
21:53:44.691524 10.2.20.6.5000 > 10.1.20.6.5000: udp 101 (DF)
0x0000   4500 0081 0000 4000 3c11 025e 0a02 1406        E.....@.<..^....
0x0010   0a01 1406 1388 1388 006d 7deb 3076 b635        .........m}.0v.5
0x0020   6b3c 8a90 b76f fdd5 19da 5bf5 061e 8585        k<...o....[.....
0x0030   51e4 46d1 c40c 0dbb 5696 f03f 637d 28e7        Q.F.....V..?c}(.
0x0040   25bd 055a 4d53 8d60 cd94 586a 0574 9bc9        %..ZMS.`..Xj.t..
0x0050   30d9                                           0.
21:53:44.693688 10.1.20.6.5000 > 10.2.20.6.5000: udp 101 (DF)
0x0000   4500 0081 0000 4000 4011 fe5d 0a01 1406        E.....@.@..]....
0x0010   0a02 1406 1388 1388 006d a8d5 30c7 e300        .........m..0...
0x0020   c9d2 dfd4 387e e934 b215 ff3f a376 4712        ....8~.4...?.vG.
0x0030   b78a 6bd2 b439 d5fd 665e 6108 388d 1589        ..k..9..f^a.8...
0x0040   0392 599d 4cd1 ab58 cc2d 4019 af92 94df        ..Y.L..X.-@.....
0x0050   f0de                                           ..
21:53:44.704005 10.2.20.6.5000 > 10.1.20.6.5000: udp 93 (DF)
0x0000   4500 0079 0000 4000 3c11 0266 0a02 1406        E..y..@.<..f....
0x0010   0a01 1406 1388 1388 0065 7d1f 309c a59c        .........e}.0...
0x0020   69c5 7ec0 0742 163e 3628 7534 b525 4831        i.~..B.>6(u4.%H1
0x0030   86a4 1696 507b 2911 2771 1839 61d6 92f7        ....P{).'q.9a...
0x0040   f78a 265b 9c99 d321 0209 e723 3d4d 0686        ..&[...!...#=M..
0x0050   872f                                           ./
21:53:44.711073 10.1.20.6.5000 > 10.2.20.6.5000: udp 117 (DF)
0x0000   4500 0091 0000 4000 4011 fe4d 0a01 1406        E.....@.@..M....
0x0010   0a02 1406 1388 1388 007d c7df 307e 6ca0        .........}..0~l.
0x0020   7e6d 5cad 8dee 7868 c558 cb49 babb 9683        ~m\...xh.X.I....
0x0030   1678 1b1c ed3d 9792 d10a a53e 3a9b 8e33        .x...=.....>:..3
0x0040   ab26 5b50 dd2e 2d87 a66c 76b5 e874 afb5        .&[P..-..lv..t..
0x0050   090c                                           ..
21:53:44.711995 10.1.20.6.5000 > 10.2.20.6.5000: udp 93 (DF)
0x0000   4500 0079 0000 4000 4011 fe65 0a01 1406        E..y..@.@..e....
0x0010   0a02 1406 1388 1388 0065 462d 300b a363        .........eF-0..c
0x0020   8f3e 4209 dd6f 5e47 c49d 279a 8bee 4669        .>B..o^G..'...Fi
0x0030   7014 95c9 1b09 fcd3 7360 80bf 85d6 7471        p.......s`....tq
0x0040   cbc9 b0eb 174d 0b4e 8b8a babe 7fa4 b928        .....M.N.......(
0x0050   0eba                                          ..
21:53:44.722843 10.2.20.6.5000 > 10.1.20.6.5000: udp 93 (DF)
0x0000   4500 0079 0000 4000 3c11 0266 0a02 1406        E..y..@.<..f....
0x0010   0a01 1406 1388 1388 0065 7583 30e1 8480        .........eu.0...
0x0020   13b6 5a50 928f 80fb ab5a cea3 24c2 3814        ..ZP.....Z..$.8.
0x0030   1523 ba3d af2c 19ef bb10 59ab 2527 68b6        .#.=.,....Y.%'h.
0x0040   f0ef 11eb c4ee e09e c373 462b ed76 b8f4        .........sF+.v..
0x0050   a89c                                           ..
21:53:44.730948 10.2.20.6.5000 > 10.1.20.6.5000: udp 93 (DF)
0x0000   4500 0079 0000 4000 3c11 0266 0a02 1406        E..y..@.<..f....
0x0010   0a01 1406 1388 1388 0065 df44 306b 2f30        .........e.D0k/0
0x0020   5c49 1692 40d6 723b 2675 853e 6f23 d188        \I..@.r;&u.>o#..
0x0030   c299 9ed6 0ed8 90c5 1276 04c4 9b5b 4a11        .........v...[J.
0x0040   1da5 c300 9169 157a 341b 6c15 5e0e b0b6        .....i.z4.l.^...
0x0050   adc7                                           ..
21:53:44.736686 10.1.20.6.5000 > 10.2.20.6.5000: udp 93 (DF)
0x0000   4500 0079 0000 4000 4011 fe65 0a01 1406        E..y..@.@..e....
0x0010   0a02 1406 1388 1388 0065 210c 303f 0798        .........e!.0?..
0x0020   9bc5 5cc0 310b b537 a3d3 4432 a75e 2f81        ..\.1..7..D2.^/.
0x0030   c9dd bd5b 14f2 e857 d8fb 9b6a 1a20 0910        ...[...W...j....
0x0040   be9a ff1a 69b2 6fbb fed9 47ba 0559 eeb7        ....i.o...G..Y..
0x0050   c815                                           ..
```

```
21:53:49.935957 arp who-has 10.1.20.248 tell 10.1.20.6
0x0000   0001 0800 0604 0001 fefd 0000 0602 0a01        ................
0x0010   1406 0000 0000 0000 0a01 14f8                   ............
21:53:49.937271 arp reply 10.1.20.248 is-at fe:fd:0:0:4:3
0x0000   0001 0800 0604 0002 fefd 0000 0403 0a01        ................
0x0010   14f8 fefd 0000 0602 0a01 1406                   ............
```

As a final test, we are going to remove the tcpdump on eth2, and start it up on the VPN tunnel. This should show the unencrypted data inside of the VPN tunnel.

```
gw1b:/etc/vpn# tcpdump -nn -X -i tun0
tcpdump: listening on tun0
```

Send another daytime request through the VPN tunnel from gw2b:

```
gw2b:/etc/vpn# telnet 10.1.40.6 daytime
Trying 10.1.40.6...
Connected to 10.1.40.6.
Escape character is '^]'.
Thu Feb 19 21:52:56 2004
Connection closed by foreign host.
```

Looking at the packet capture from the VPN tunnel interface, we see the data as expected.

```
21:55:05.552860 10.2.40.6.1030 > 10.1.40.6.13: S 2757928700:2757928700(0) win
4872 <mss 1218,sackOK,timestamp 159079 0,nop,wscale 0> (DF) [tos 0x10]
0x0000   4510 003c cb9b 4000 4006 0b02 0a02 2806        E..<..@.@.....(.
0x0010   0a01 2806 0406 000d a462 a6fc 0000 0000        ..(......b......
0x0020   a002 1308 1507 0000 0204 04c2 0402 080a        ................
0x0030   0002 6d67 0000 0000 0103 0300                   ..mg........
21:55:05.553138 10.1.40.6.13 > 10.2.40.6.1030: S 2778422102:2778422102(0) ack
2757928701 win 4824 <mss 1218,sackOK,timestamp 160911 159079,nop,wscale 0> (DF)
0x0000   4500 003c 0000 4000 4006 d6ad 0a01 2806        E..<..@.@.....(.
0x0010   0a02 2806 000d 0406 a59b 5b56 a462 a6fd        ..(.......[V.b..
0x0020   a012 12d8 9fa2 0000 0204 04c2 0402 080a        ................
0x0030   0002 748f 0002 6d67 0103 0300                   ..t...mg....
21:55:05.574571 10.2.40.6.1030 > 10.1.40.6.13: . ack 1 win 4872
<nop,nop,timestamp 159079 160911> (DF) [tos 0x10]
0x0000   4510 0034 cb9c 4000 4006 0b09 0a02 2806        E..4..@.@.....(.
0x0010   0a01 2806 0406 000d a462 a6fd a59b 5b57        ..(......b....[W
0x0020   8010 1308 cd45 0000 0101 080a 0002 6d67        .....E........mg
0x0030   0002 748f                                       ..t.
21:55:05.575426 10.1.40.6.13 > 10.2.40.6.1030: P 1:27(26) ack 1 win 4824
<nop,nop,timestamp 160911 159079> (DF)
0x0000   4500 004e 2fd8 4000 4006 a6c3 0a01 2806        E..N/.@.@.....(.
0x0010   0a02 2806 000d 0406 a59b 5b57 a462 a6fd        ..(.......[W.b..
0x0020   8018 12d8 c4ab 0000 0101 080a 0002 748f        ..............t.
0x0030   0002 6d67 5468 7520 4665 6220 3139 2032        ..mgThu.Feb.19.2
0x0040   313a 3535 3a30 3520 3230 3034 0d0a             1:55:05.2004..
21:55:05.575590 10.1.40.6.13 > 10.2.40.6.1030: F 27:27(0) ack 1 win 4824
<nop,nop,timestamp 160911 159079> (DF)
0x0000   4500 0034 2fd9 4000 4006 a6dc 0a01 2806        E..4/.@.@.....(.
0x0010   0a02 2806 000d 0406 a59b 5b71 a462 a6fd        ..(.......[q.b..
0x0020   8011 12d8 cd5a 0000 0101 080a 0002 748f        .....Z........t.
0x0030   0002 6d67                                       ..mg
21:55:05.606454 10.2.40.6.1030 > 10.1.40.6.13: . ack 27 win 4872
<nop,nop,timestamp 159079 160911> (DF) [tos 0x10]
0x0000   4510 0034 cb9d 4000 4006 0b08 0a02 2806        E..4..@.@.....(.
0x0010   0a01 2806 0406 000d a462 a6fd a59b 5b71        ..(......b....[q
0x0020   8010 1308 cd2b 0000 0101 080a 0002 6d67        .....+........mg
0x0030   0002 748f                                      ..t.
21:55:05.607218 10.2.40.6.1030 > 10.1.40.6.13: F 1:1(0) ack 28 win 4872
<nop,nop,timestamp 159079 160911> (DF) [tos 0x10]
0x0000   4510 0034 cb9e 4000 4006 0b07 0a02 2806        E..4..@.@.....(.
0x0010   0a01 2806 0406 000d a462 a6fd a59b 5b72        ..(......b....[r
0x0020   8011 1308 cd29 0000 0101 080a 0002 6d67        .....)........mg
0x0030   0002 748f                                      ..t.
21:55:05.607401 10.1.40.6.13 > 10.2.40.6.1030: . ack 2 win 4824
<nop,nop,timestamp 160911 159079> (DF)
0x0000   4500 0034 2fda 4000 4006 a6db 0a01 2806        E..4/.@.@.....(.
0x0010   0a02 2806 000d 0406 a59b 5b72 a462 a6fe        ..(.......[r.b..
0x0020   8010 12d8 cd59 0000 0101 080a 0002 748f        .....Y........t.
0x0030   0002 6d67                                      ..mg
```

So, data sent over the VPN tunnel is transmitted in encrypted form.

20. Terminating the Network Simulation

To terminate the network simulation, enter the following command on the host system:

```
# vnumlparser.pl -P vpn-test.xml
```

21. Installation of Root File Systems on Production VPN Gateways

Once the VPN configuration has been verified, the custom root file systems must be copied over to the production systems. In order to do this, the network simulation must be terminated (step 19 above). Then the custom root file systems can be copied to the vs3 (west site) or vs5 (east site) servers by using secure copy (scp) over the management network segment. After copying the file system images, the VPN gateways can be rebooted during a maintenance window or sooner if necessary. Alternately, the configuration files can also be manually loaded into the currently active VPN gateways and OpenVPN manually restarted. However, the procedure to accomplish this is beyond the scope of this tutorial.

Assignment 3
Verify the Firewall Policy

The part of the study describes the planning, execution, and results of a technical validation of the primary firewall security policies for GIAC Enterprises. While reviewing the network design, it is apparent that GIAC has two "primary" firewalls, one each for the west and east sites. Since the east site firewall (router3) provides the same protection as both of the west site firewalls, router3 was chosen for the validation study. Since the ruleset for router3 is a superset of both router1 and router2 for the west site, much of the validation will apply directly to them as well.

Validation Plan

Any technical evaluation of security policies must start with a in-depth review of the documents that describe the network design and security policies. Without an understanding of the intent of the policies, a correct validation would be difficult.

Also, a review of the actual firewall configuration files was performed. In this review, it was apparent that GIAC developed the scripts so the maintenance and validation processes were simplified. The scripts are structured so that all of the site specific information is located at the beginning of the main script file. This allows changes to be made in a single location without having to manually review every line in the script to verify that all of the changes were applied correctly.

Since this validation testing is done on the production firewall, it should be done with the network disconnected from the Internet at the ISP demarcation point. This is to prevent unauthorized traffic from being passed into the GIAC network in case there is a failure of the firewall or other systems during the test. The testing will be conducted during one of the regularly scheduled maintenance windows. For the east site, the maintenance window is Thursday mornings between 1:00 and 6:00 am local time). Since the GIAC network is divided between two sites, one site can provide network services while the other one is disconnected for testing.

Based on the network design, the validation portion for testing the ingress portion of the primary firewall will be divided into three parts:

- No firewall rules loaded: this will show what the potential exposure to the internal network will be if the firewall fails completely in an "open" state. Because the firewall rules also include the NAT rules to map external IP addresses to internal servers, this will only expose the firewall itself and not any internal hosts. Therefore, we will not scan for any internal hosts.
- NAT and routing rules loaded but no "rc.common-fw" rules: this will show the network exposure without the self-protection measures for the firewall. Also, the default policies are set to DROP inside of rc.common-fw. So we should expect to see any open ports on the NATed hosts regardless of filtering rules. The expected exposure during this portion of the test is that

information regarding the internal network could "leak out" from the firewall during external probes.

- The full firewall configuration: shows the network exposure that should be expected when the normal production firewall is operating properly. The security policy states that only a single port per host should be visible, except for web and DNS servers.

Because GIAC will use its network simulation framework to develop the test suite and expected results, the amount of effort and cost associated with validating the firewall configuration is minimized. The initial development and generation of the test suite and expected results can be done in approximately 3 hours. This can be done outside of the maintenance window, since they do not affect the primary firewall.

Analysis and reporting of the actual test results, will take another two hours.

The largest amount of time will be spent in performing the test scans against the physical network. Since the scans will cover all 8 subnets and all 65535 available ports, this will take most of the available maintenance window. The cost of the validation for a single firewall, assuming a personnel cost of $150/hour and 10 hours is $1500.00. To validate the other site's firewall would cost an additional $1050, since the test suite has already been created.

Total cost for both firewalls: $2550.00.

The only hardware needed is a small laptop and a 4-port network hub. The laptop will be connected to the external interface of the primary router to function as a surrogate Internet. It will generate test packets directed at the primary firewall. However, if a second laptop is available, it would simplify egress (outbound) testing, since by changing its IP address, we can mimic a server without risk to the production server.

Conducting the validation

The validation is performed in two parts: ingress filtering to test inbound firewall rules, and egress filtering for outbound firewall rules for selected hosts. Egress filter validation is only performed on the public web, NTP, and DNS servers since they are the only hosts protected directly by the primary router. The other hosts on the network are behind an additional internal firewall The egress validation will consist of running a tcp connect scan directed at an external Internet host (10.0.3.2).

Important note about the "nmap" port scanner

In our validation testing, we will be using a modified version of the "nmap" port scanning utility. The default version of nmap will discard the intermediate results of a scan whenever the scan takes longer than a specified time (default is 15 minutes). For our testing, we have patched the latest version of nmap (3.50) with an updated version of a patch developed by Colin Phipps described in the nmap development mailing list, August 12, 2002. Colin's patch was to nmap 2.54BETA29 and needed to be modified to work with the current version of nmap. His original mail message can be found at: http://seclists.org/lists/nmap-dev/2002/Jul-Sep/0037.html. The modified version of the patch can be found in Appendix 2.

Ingress Filtering Tests

To validate the ingress firewall rules, we will use the nmap packet generator to perform a port scan against each of the public IP addresses protected by the west site border router. The predefined scans that we will be using are: syn, tcp connection, udp, and xmas. The syn scan performs a "stealth" scan by only performing a part of the normal three-way handshake for tcp network connections. The tcp connection scan performs a normal handshake for each of the scanned ports. The udp scan will send a udp packet to each of the scanned ports. The xmas scan generates packets with the FIN, URG, and PUSH flags turned on.

To get an indication of which firewall rules are activated during each scan, we will run each test with three different configurations for the border firewall. The first set of tests will be run with no firewall settings other than the default policies. The second test run will configure the firewall settings with the NAT and routing rules, but without the rules contained in the rc.common-fw rules. The final test run will configure the firewall with the complete rule set used by the production network.

Test Sequence 1: Use default iptables policies only

The first test is the syn scan with the default policies. The nmap command used was:

```
# nmap -v -sS -n -r -P0 -p 1-65535 -T5 -oN test-nonesyn 10.0.3.1 10.0.3.33-39
```

The options used are:

- -v -- verbose mode,
- -sS -- use the syn scan mode
- -n -- do not resolve any addresses with DNS,
- -r -- do not randomize the port probes. This is specified so we can monitor the test's progress with tcpdump,
- -P0 -- do not ping the host to determine if it is up,
- -p 1-65535 -- scan all ports between 1 and 65535 inclusive,
- -T5 -- generate the packets as fast as possible,
- -oN test-nonesyn -- store the scan results in the file: test-nonesyn
- 10.0.3.1 10.0.3.33-39 -- scans all of the public IP numbers and the Internet facing interface of the border router.

Results:

```
# nmap 3.50 scan initiated Sat Feb 28 21:09:45 2004 as: nmap -v -sS -n
-r -P0 -p 1-65535 -T5 -oN test-nonesyn 10.0.3.1 10.0.3.33-39
Interesting ports on 10.0.3.1:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
774/tcp open  unknown

Results for host 10.0.3.33 incomplete due to host timeout
All 786 scanned ports on 10.0.3.33 are: filtered

Results for host 10.0.3.34 incomplete due to host timeout
All 798 scanned ports on 10.0.3.34 are: filtered

Results for host 10.0.3.35 incomplete due to host timeout
All 803 scanned ports on 10.0.3.35 are: filtered

Results for host 10.0.3.36 incomplete due to host timeout
All 797 scanned ports on 10.0.3.36 are: filtered

Results for host 10.0.3.37 incomplete due to host timeout
All 724 scanned ports on 10.0.3.37 are: filtered

Results for host 10.0.3.38 incomplete due to host timeout
All 318 scanned ports on 10.0.3.38 are: filtered

Results for host 10.0.3.39 incomplete due to host timeout
All 1083 scanned ports on 10.0.3.39 are: filtered

# Nmap run completed at Sat Feb 28 22:55:14 2004 -- 8 IP addresses (8
hosts up) scanned in 6329.412 seconds
```

We see that the border router has the common services available by default, and none of the public IP addresses show any listening services. This is due to the fact there is no address translation being performed in this test.

The second test run uses the same nmap options except that the scan type is the tcp (-sT) connect mode. Again, we see the same services are available as with the syn scan.

```
# nmap 3.50 scan initiated Sat Feb 28 23:08:48 2004 as: nmap -v -sT -n
-r -P0 -p 1-65535 -T5 -oN test-nonetcp 10.0.3.1 10.0.3.33-39
Interesting ports on 10.0.3.1:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
```

```
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
774/tcp open  unknown
```

The third test performs a UDP scan (-sU). The results showed that there were many ports responding (showing a closed status supposedly indicates that a server is bound to the port). The fact was that there were not supposed to be any udp ports active. This was verified by running a “netstat -an” command on the router and servers. Further research indicated that the cause of this misleading information was the Linux kernel. Linux has code built into its network stack that limits the number of “port unreachable” replies that will be sent within a certain amount of time.

> ‟… the Linux kernel (in net/ipv4/icmp.h) limits destination unreachable message generation to 80 per 4 seconds, with a 1/4 second penalty if that is exceeded.”
>
> (extracted from: http://www.insecure.org/nmap/data/nmap_manpage.html).

```
# nmap 3.50 scan initiated Sat Feb 28 23:14:21 2004 as: nmap -v -sU -n
-r -P0 -p 1-65535 -T5 -oN test-noneudp 10.0.3.1
Results for host 10.0.3.1 incomplete due to host timeout
Interesting ports on 10.0.3.1:
(The 64661 ports scanned but not shown below are in state: open)
PORT     STATE  SERVICE
2/udp    closed unknown
3/udp    closed unknown
4/udp    closed unknown
5/udp    closed rje
6/udp    closed unknown
7/udp    closed echo
42/udp   closed nameserver
69/udp   closed tftp
109/udp  closed pop2
143/udp  closed imap
169/udp  closed unknown
209/udp  closed qmtp
243/udp  closed unknown
269/udp  closed unknown
309/udp  closed unknown
340/udp  closed unknown
383/udp  closed unknown
409/udp  closed unknown
443/udp  closed https
483/udp  closed unknown
509/udp  closed unknown
543/udp  closed unknown
        . . .
        . . .
        . . .
2411/udp closed unknown
2412/udp closed unknown
2413/udp closed unknown
2414/udp closed unknown
2415/udp closed unknown

# Nmap run completed at Sat Feb 28 23:30:04 2004 -- 1 IP address (1
host up) scanned in 942.820 seconds
```

The last test with the default policy configuration is the xmas scan (-sX) with the expected results.

```
# nmap 3.50 scan initiated Sat Feb 28 23:11:36 2004 as: nmap -v -sX -n
-r -P0 -p 1-65535 -T5 -oN test-nonexmas 10.0.3.1
Interesting ports on 10.0.3.1:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
774/tcp open  unknown

# Nmap run completed at Sat Feb 28 23:12:33 2004 -- 1 IP address (1
host up) scanned in 57.179 seconds
```

Test Sequence 2: No rc.common-fw rules, NAT and Filtering are active

Now, we begin the second sequence of testing. In this set, we have configured the firewall with the NAT and Filtering rules. We do not include any of the settings found in rc.common-fw file, so the default ACCEPT policies are still in force.

The syn scan:

```
# nmap 3.50 scan initiated Sat Feb 28 23:44:02 2004 as: nmap -v -sS -n
-r -P0 -p 1-65535 -T5 -oN test-noncommonsyn 10.0.3.1 10.0.3.33-39
Interesting ports on 10.0.3.1:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
774/tcp open  unknown

Interesting ports on 10.0.3.33:
```

```
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.34:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.35:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.36:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.37:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
```

```
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
772/tcp open  unknown

Interesting ports on 10.0.3.38:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.39:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

# Nmap run completed at Sat Feb 28 23:54:12 2004 -- 8 IP addresses (8
hosts up) scanned in 610.382 seconds
```

The results are what we expected, a limited number of services active on each of the servers. Since results are returned for each of the servers, we know that NAT is functioning correctly. Also, keep in mind, since the default policies are ACCEPT, the filter rules are evaluated but have no effect.

We should get the same results with the tcp connect scan (-sT).

```
# nmap 3.50 scan initiated Sat Feb 28 23:35:26 2004 as: nmap -v -sT -n
-r -P0 -p 1-65535 -T5 -oN test-noncommonudp 10.0.3.1 10.0.3.33-39
Interesting ports on 10.0.3.1:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
```

```
515/tcp open  printer
774/tcp open  unknown

Interesting ports on 10.0.3.33:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.34:
(The 65525 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
53/tcp  open  domain
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.35:
(The 65524 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
80/tcp  open  http
111/tcp open  sunrpc
113/tcp open  auth
443/tcp open  https
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.36:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown
```

```
Interesting ports on 10.0.3.37:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
772/tcp open  unknown

Interesting ports on 10.0.3.38:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.39:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

# Nmap run completed at Sat Feb 28 23:43:04 2004 -- 8 IP addresses (8
hosts up) scanned in 458.275 seconds
```

We then ran the udp scan (-sU), and as discussed in the first test sequence, we have omitted the results since they are completely bogus.

```
# nmap 3.50 scan initiated Sat Feb 28 23:55:22 2004 as: nmap -v -sU -n
-r -P0 -p 1-65535 -T5 -oN test-noncommonudp 10.0.3.1 10.0.3.33-39

# Nmap run completed at Sun Feb 29 02:00:43 2004 -- 8 IP addresses (8
hosts up) scanned in 7521.220 seconds
```

We now run the xmas scan to finish the second test sequence. The results are the same as the other scan types, as expected.

```
# nmap 3.50 scan initiated Sun Feb 29 09:34:29 2004 as: nmap -v -sX -n
-r -P0 -p 1-65535 -T5 -oN test-nocommonxmas 10.0.3.1 10.0.3.33-39
Interesting ports on 10.0.3.1:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
774/tcp open  unknown

Interesting ports on 10.0.3.33:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.34:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.35:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.36:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
```

```
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.37:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
772/tcp open  unknown

Interesting ports on 10.0.3.38:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

Interesting ports on 10.0.3.39:
(The 65526 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
111/tcp open  sunrpc
113/tcp open  auth
515/tcp open  printer
770/tcp open  unknown

# Nmap run completed at Sun Feb 29 09:45:09 2004 -- 8 IP addresses (8
hosts up) scanned in 639.161 seconds
```

Test Sequence 3: Full Configuration

In this sequence we run the same four scan types but the router has been loaded with full configuration files. Anti-spoofing, bad packet elimination, NAT, and

Filtering are all active. The default policies have been changed to DROP.

The first scan is the syn scan. We would expect to see only those services that have been explicitly allowed to pass through the firewall.

```
# nmap 3.50 scan initiated Sun Feb 29 11:42:58 2004 as: nmap -v -sS -n
-r -P0 -p 1-65535 -T5 -oN test-fullsyn 10.0.3.1 10.0.3.33-39
Results for host 10.0.3.1 incomplete due to host timeout
All 0 scanned ports on 10.0.3.1 are: unknown

Results for host 10.0.3.33 incomplete due to host timeout
All 0 scanned ports on 10.0.3.33 are: unknown

Results for host 10.0.3.34 incomplete due to host timeout
All 0 scanned ports on 10.0.3.34 are: unknown

Results for host 10.0.3.35 incomplete due to host timeout
Interesting ports on 10.0.3.35:
PORT    STATE  SERVICE
80/tcp  closed http
443/tcp closed https

Results for host 10.0.3.36 incomplete due to host timeout
Interesting ports on 10.0.3.36:
PORT   STATE SERVICE
22/tcp open  ssh

Results for host 10.0.3.37 incomplete due to host timeout
All 0 scanned ports on 10.0.3.37 are: unknown

Results for host 10.0.3.38 incomplete due to host timeout
Interesting ports on 10.0.3.38:
PORT   STATE SERVICE
25/tcp open  smtp

Results for host 10.0.3.39 incomplete due to host timeout
All 0 scanned ports on 10.0.3.39 are: unknown

# Nmap run completed at Sun Feb 29 13:43:00 2004 -- 8 IP addresses (8
hosts up) scanned in 7201.732 seconds
```

The syn scan shows only the following services:

- 80 (http) and 443 (https) are available on the public web server (www1).
- 22 (ssh) is available on the SSH gateway (gw1a).
- 25 (smtp) is available on the public e-mail server (mail1)

Note: that even though port 53/TCP is permitted outbound according to the security policy. It is not visible to the Internet and so should not appear. No other services should appear according to the security policy.

Then run the tcp connect scan (-sT), the results should be the same:

```
# nmap 3.50 scan initiated Sun Feb 29 14:05:31 2004 as: nmap -v -sT -n
-r -P0 -p 1-65535 -T5 -oN test-fulltcp 10.0.3.1 10.0.3.33-39
All 65535 scanned ports on 10.0.3.1 are: filtered

All 65535 scanned ports on 10.0.3.33 are: filtered

All 65535 scanned ports on 10.0.3.34 are: filtered

Interesting ports on 10.0.3.35:
(The 65533 ports scanned but not shown below are in state: filtered)
PORT    STATE  SERVICE
80/tcp  closed http
443/tcp closed https

Interesting ports on 10.0.3.36:
(The 65534 ports scanned but not shown below are in state: filtered)
PORT   STATE SERVICE
22/tcp open  ssh

All 65535 scanned ports on 10.0.3.37 are: filtered

Interesting ports on 10.0.3.38:
(The 65534 ports scanned but not shown below are in state: filtered)
PORT   STATE SERVICE
25/tcp open  smtp

All 65535 scanned ports on 10.0.3.39 are: filtered

# Nmap run completed at Sun Feb 29 15:57:19 2004 -- 8 IP addresses (8
hosts up) scanned in 6708.367 seconds
```

Since we are now running the full firewall configuration, we should expect to see reasonable results from the udp packet scan.

```
# nmap 3.50 scan initiated Sat Feb 28 14:09:13 2004 as: nmap -v -sU -n
-r -P0 -p 1-65535 -T5 -oN test-fulludp 10.0.3.1 10.0.3.33-39
Results for host 10.0.3.1 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.1 are: filtered

Results for host 10.0.3.33 incomplete due to host timeout
Interesting ports on 10.0.3.33:
(The 65534 ports scanned but not shown below are in state: open)
PORT    STATE  SERVICE
123/udp closed ntp

Results for host 10.0.3.34 incomplete due to host timeout
Interesting ports on 10.0.3.34:
(The 65534 ports scanned but not shown below are in state: open)
PORT   STATE  SERVICE
53/udp closed domain

Results for host 10.0.3.35 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.35 are: filtered

Results for host 10.0.3.36 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.36 are: filtered
```

```
Results for host 10.0.3.37 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.37 are: filtered

Results for host 10.0.3.38 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.38 are: filtered

Results for host 10.0.3.39 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.39 are: filtered

# Nmap run completed at Sat Feb 28 16:14:42 2004 -- 8 IP addresses (8
hosts up) scanned in 7529.056 seconds
```

The results do show that the permitted udp services are visible to the internet. These services are:

- 123 (NTP) is available on the public time server (ntp1).
- 53 (DNS) is available on the external DNS server (ns1).

We now run the final xmas scan test (-sX). Since the firewall is fully configured, nothing should appear in the results. If any port appears, then the anti-spoofing and invalid packet filtering rules have failed.

```
# nmap 3.50 scan initiated Sun Feb 29 15:57:53 2004 as: nmap -v -sX -n
-r -P0 -p 1-65535 -T5 -oN test-fullxmas 10.0.3.1 10.0.3.33-39
Results for host 10.0.3.1 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.1 are: filtered

Results for host 10.0.3.33 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.33 are: filtered

Results for host 10.0.3.34 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.34 are: filtered

Results for host 10.0.3.35 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.35 are: filtered

Results for host 10.0.3.36 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.36 are: filtered

Results for host 10.0.3.37 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.38 are: filtered

Results for host 10.0.3.38 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.38 are: filtered

Results for host 10.0.3.39 incomplete due to host timeout
All 65535 scanned ports on 10.0.3.39 are: filtered

# Nmap run completed at Sun Feb 29 17:59:50 2004 -- 8 IP addresses (8
hosts up) scanned in 7316.748 seconds
```

No services appear, so the invalid packet filtering rules are functioning.

Egress Filtering Tests

For the egress testing we will use nmap's tcp connect and udp port scans to determine what ports on the Internet we can see from our public systems.

We start with the public web server (www1):

```
www1:~# nmap -r -n -v -sT -T5 -P0 -p 1-65535 10.0.3.2

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-03-03 19:08 UTC
Host 10.0.3.2 appears to be up ... good.
Initiating Connect() Scan against 10.0.3.2 at 19:08
Skipping host 10.0.3.2 due to host timeout

Nmap run completed -- 1 IP address (1 host up) scanned in 900.068 seconds
```

To make sure that things are as we expect, we use tcpdump to look at www1's traffic. All we see are "host unreachable" messages. This is what we expect, since there are no outbound firewall rules.

```
[root@fedora root]# tcpdump -i INET -nn
tcpdump: listening on INET
11:32:01.528006 arp who-has 10.0.3.1 tell 10.0.3.2
11:32:01.529553 arp reply 10.0.3.1 is-at fe:fd:0:0:2:1
11:32:01.529616 10.0.3.2 > 10.1.1.12: icmp: host 10.0.3.2 unreachable - admin
prohibited [tos 0xc0]
11:32:01.529668 10.0.3.2 > 10.1.1.12: icmp: host 10.0.3.2 unreachable - admin
prohibited [tos 0xc0]
11:32:01.529952 10.0.3.2 > 10.1.1.12: icmp: host 10.0.3.2 unreachable - admin
prohibited [tos 0xc0]
11:32:01.531511 10.0.3.2 > 10.1.1.12: icmp: host 10.0.3.2 unreachable - admin
prohibited [tos 0xc0]
11:32:01.532763 10.0.3.2 > 10.1.1.12: icmp: host 10.0.3.2 unreachable - admin
prohibited [tos 0xc0]
11:32:01.533976 10.0.3.2 > 10.1.1.12: icmp: host 10.0.3.2 unreachable - admin
prohibited [tos 0xc0]
11:32:02.540140 10.0.3.2 > 10.1.1.12: icmp: host 10.0.3.2 unreachable - admin
prohibited [tos 0xc0]
11:32:03.570939 10.0.3.2 > 10.1.1.12: icmp: host 10.0.3.2 unreachable - admin
prohibited [tos 0xc0]
    .  .  .
    .  .  .
    .  .  .
```

Run the udp port scan. We should not see any ports.

```
www1:~# nmap -r -n -v -T5 -P0 -p 1-65535 -sU 10.0.3.2

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-03-03 21:18 UTC
Host 10.0.3.2 appears to be up ... good.
Initiating UDP Scan against 10.0.3.2 at 21:18
Skipping host 10.0.3.2 due to host timeout

Nmap run completed -- 1 IP address (1 host up) scanned in 900.363 seconds
www1:~#
```

Now we are going to run the tcp connect scan from the public time server. Again, we should not see any ports open.

```
ntp1:~# nmap -r -n -v -sT -T5 -P0 -p 1-65535 10.0.3.2

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-03-03 19:34 UTC
Host 10.0.3.2 appears to be up ... good.
Initiating Connect() Scan against 10.0.3.2 at 19:34
Skipping host 10.0.3.2 due to host timeout

Nmap run completed -- 1 IP address (1 host up) scanned in 901.314 seconds
```

The udp scan from the time server. We don't see any ports open since the target system is not our upstream time server and the ntp daemon is not active on ntp1. Otherwise we would see one port: 123/UDP.

```
ntp1:~# nmap -r -n -v -sU -T5 -P0 -p 1-65535 10.0.3.2

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-03-03 19:53 UTC
Host 10.0.3.2 appears to be up ... good.
Initiating UDP Scan against 10.0.3.2 at 19:53
Skipping host 10.0.3.2 due to host timeout

Nmap run completed -- 1 IP address (1 host up) scanned in 901.911 seconds
```

Now, we run the tcp connect scan from our external name server. We should not see any ports.

```
ns1:~# nmap -r -n -v -sT -T5 -P0 -p 1-65535 10.0.3.2

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-03-03 20:10 UTC
Host 10.0.3.2 appears to be up ... good.
Initiating Connect() Scan against 10.0.3.2 at 20:10
Skipping host 10.0.3.2 due to host timeout

Nmap run completed -- 1 IP address (1 host up) scanned in 900.570 seconds
```

The results of the udp scan from our external name server are as expected, no ports. We would only see an open port during an active DNS request.

```
ns1:~# nmap -r -n -v -sU -T5 -P0 -p 1-65535 10.0.3.2

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-03-03 20:26 UTC
Host 10.0.3.2 appears to be up ... good.
Initiating UDP Scan against 10.0.3.2 at 20:26
Skipping host 10.0.3.2 due to host timeout

Nmap run completed -- 1 IP address (1 host up) scanned in 900.141 seconds
```

For the tcp connect scan from the external mail server, we should see the SMTP port on the target system.

```
mail1:~# nmap -sT -P0 -n -v -r -p 1-65535 10.0.3.2

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-03-03 23:29 UTC
Host 10.0.3.2 appears to be up ... good.
Initiating Connect() Scan against 10.0.3.2 at 23:29
Interesting ports on 10.0.3.2:
(The 65534 ports scanned but not shown below are in state: filtered)
Unable to find nmap-services!  Resorting to /etc/services
PORT   STATE SERVICE
25/tcp open  smtp
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 900.345 seconds
```

For the udp scan from mail1, there should not be any ports visible.

```
mail1:~# nmap -r -n -v -T5 -P0 -p 1-65535 -sU 10.0.3.2

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-03-03 21:39 UTC
Host 10.0.3.2 appears to be up ... good.
Initiating UDP Scan against 10.0.3.2 at 21:39
Skipping host 10.0.3.2 due to host timeout

Nmap run completed -- 1 IP address (1 host up) scanned in 900.737 seconds
```

Evaluation of Results and Recommendations

The results show that the Primary firewalls are fully implementing the security policy as described. This was verified by comparing the actual test results with the expected results obtained from the simulated network.

We recommend that a similar validation be performed for ingress/egress filtering on the VPN and egress filtering on the SSH gateway. Testing of ingress filtering for the SSH gateway was included in the current validation since it is exposed to general internet hosts. Performing these additional validations will improve the overall network security policies in case a VPN or SSH system should be compromised.

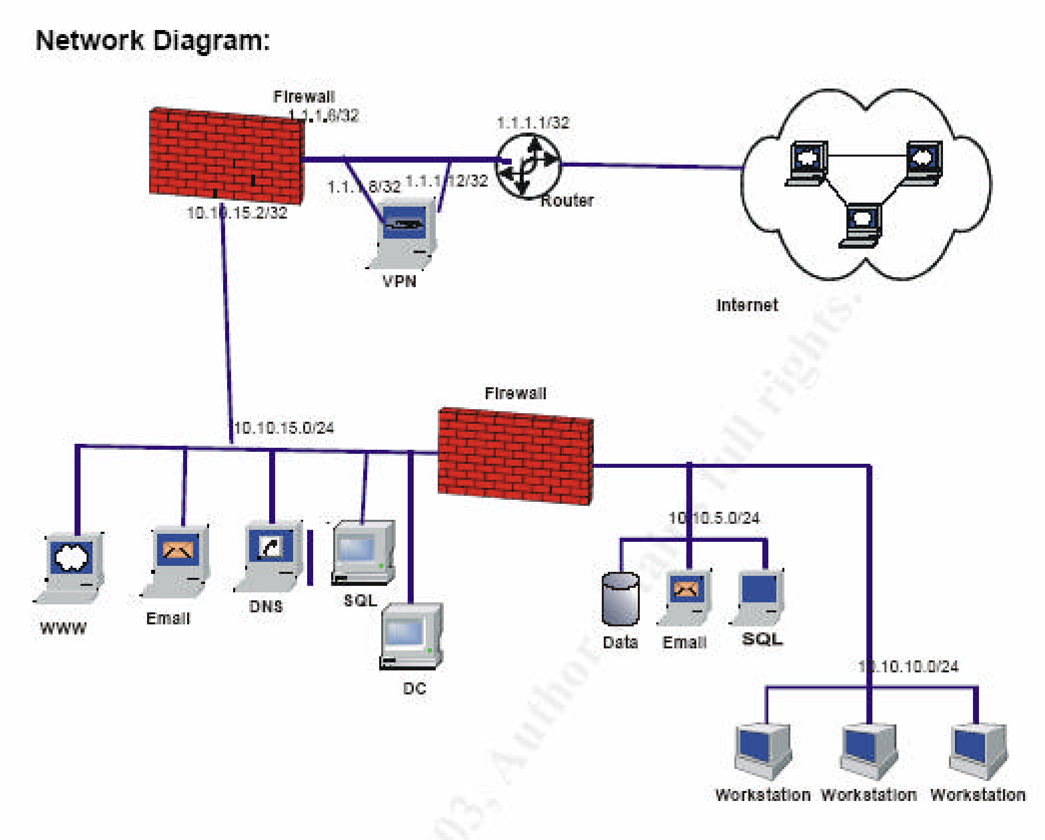
Consideration should also be given in developing a set of standard network sessions. This test set could be used for regression testing as well as network security policy validation. For example, capturing a session where an e-mail message is received, a user surfing the web, etc. An easy method to accomplish this is to capture an actual session using the tcpdump utility and then replay it later using the “tcpreplay” utility (http://tcpreplay.sourceforge.net/). The tcprelay project is managed by Aaron Turner and is available under a BSD style license. By capturing the replayed session and comparing it to the test set, validation is simplified. It may be possible that this comparison could be automated.

ASSIGNMENT 4:
Design Under Fire

The network design chosen for this section was developed and presented by Lesa Ludwig in her GCFW Practical of October 2003, available at the GIAC website: http://www.giac.org/practical/GCFW/Lesa_Ludwig_GCFW.pdf. The following short description and network diagram have been extracted from the document referenced above. This description is used as source information that would normally be obtained by other reconnaissance methods like public web pages and the whois & nmap utilities. The description has been edited to include only the information that could be reasonably determined using these methods.

Border router: cisco 1711 ios 12.2

Primary Firewalls: GIAC Enterprises has chosen to use an IPCop version 1.3.0 firewall. This firewall is fork of Smoothwall, an open source firewall that runs on a hardened Linux kernel and is designed to take advantage of older hardware. … With version 1.3 IPCop has moved to IPTables. IPtables has several advantages over its predecessor IPchains. IPtables is capable of stateful inspection of TCP, UDP and ICMP packets. IPtables simplifies how rules are processed and allows NAT to be separate from packet filtering which makes it easier to manage and more logical to apply. Finally IPTables improves on filtering capabilities and allows the administrator to rate-limit on both connections and logging to protect the network from flooding attacks.



Distributed Denial of Service Attack and Countermeasures

While, there are many approaches that can be used to compromise systems, for most of them there is risk of detection. The approach that will be used in this study provides a high probability of successfully compromising a system while minimizing the risk of detection.

There are several items that we will be using:

- A system running a UNIX like (Solaris, Linux, *BSD) system connected to the internet. In this study, we are using Fedora Core-1 Linux.
- The Snort network sniffer package (http://www.snort.org).
- The whois utility, usually found as part of the operating system distribution.
- p0f, this is used to "fingerprint" operating systems.
- nmap, used to scan systems for open ports.
- Nessus, scans hosts for known vulnerabilities.
- TFN2K, Tribal Flood Network is a distributed utility that is used to launch a denial of service attack against an Internet host.

The easiest way to identify possible target systems to compromise is to passively look for hosts that may have already been compromised. At the time of this writing, there were several worms and viruses with high activity on the Internet. Two of the most active are called "mydoom" and "DameWare". While we will be using mydoom as part of our attack, the techniques presented will probably work with any virus or worm that installs a backdoor.

How do we determine what are good worms/viruses to target? We use publicly available information from sources like the Bugtraq mailing list (bugtraq@securityfocus.com), web sites like www.neohapsis.com, isc.sans.org, IRC chat rooms, or Usenet newsgroups.

One reason that mydoom was chosen as a target vector is the availability of ready-to-use exploit code. On Feb. 10, 2004, Fabien from the K-OTik security group posted code to the bugtraq mailing list. This code exploited the mydoom upload/exec backdoor. With this backdoor, you can upload and execute arbitrary code on a computer system infected by the mydoom virus.

Mydoom is spread by opening an infected e-mail or downloading an infected file. One of the side effects of mydoom is to open a backdoor command shell that listens on one or more ports. The ports used depend on the specific variant of the mydoom infection. The most common port used is 3127.

We then install the Snort packet filtering software on a host connected to the Internet. If we log all probes, we can easily build a list of hosts that may have already been compromised. For example, over a three day period of monitoring, snort captured portscans from 63 unique IP addresses. Fourteen of these were probing port 3127 (mydoom).

We can continue to monitor in this manner until we gather enough candidate hosts. This method is completely passive with no risk of discovery. The problem with this approach is that the amount of time needed to gather the list of candidate systems is unknown.

If time is short and we need to build a list of candidate hosts quickly, we would use a port scanning tool like nmap to scan a section of the Internet for hosts and find out which ports are open on them. Scanning in this manner is “noisy”, and can usually be detected by tools like snort and other intrusion detection systems.

These tools have command line settings that can be used to alter its scanning behavior to try and hide from a target’s defenses. This can be done by changing the number of probes, the amount of time between probes, and the type of options in the packets.

Once we have a list of candidates, we may want to do some further research on the systems to make sure we pick undefended systems. This research is to help make sure that our activities are undetected. The ideal systems would be a home-based system directly connected to a cable modem. Typically, these systems have little or no protection features. Also, these systems are more likely to not have up-to-date patches installed.

We use the whois service to lookup the domain registration information for a candidate host. The information returned includes the domain name, IP number block, register domain owner, and ISP information.

Once we have finalized the list of our 50 compromised hosts, we upload the Tribal Flood Network 2000 (TFN2K) client and agents to the systems using the mydoom Upload/exploit code described above. It might be a good idea to have several additional compromised systems available for use as decoys. These decoys will help hide the host originating the attack from detection. These decoys will use the cryptographic version of netcat (cryptcat) to establish a communication channel to control the TFN2K clients.

At this point, we can attack the network described above by specifying the IP number and possibly the port number of the target systems. Based on the network diagram, there are two targets that if rendered inoperable would take the entire network out of service and one host that would greatly impact usability. The two critical systems are the border router (1.1.1.1/32) and the primary firewall (1.1.1.6/32). The other system that would greatly impact service is the name server (the external IP number of the DNS server can be determined from the whois record).

There are several ways of achieving our goal of denying service, two of the most common is to overwhelm the device or to use an exploit to render the device incapable of servicing requests. An example of using an exploit to crash the device would be to send an ill-formed packet as described in http://www.k-otik.com/exploits/07.22.ciscodos.sh.php. Note: This exploit is used only as an example. It is for an earlier version of the Cisco IOS (operating system) and the version 12.2 used by the described network should not be vulnerable. If the router was vulnerable to this attack, then a single host could be used instead of the 50 compromised systems.

The other approach is to overwhelm the target device with outstanding requests. This may cause the target to exhaust its available resources. An example of this type of attack is to use the TFN2K agent network of compromised machines to send large numbers of packets with random source addresses and the SYN bit (request to open a new connection) set. The target system will send a reply packet to the spoofed source address and wait for the final part of the connection open handshake sequence. Since the source address is random and the reply packet will be ignored, the necessary response is never sent back to the target. The target system will wait for the reply thereby consuming resources for the pending connection. If there are enough pending requests, the target system can run out of resources to allow new ones to be created. This results in a denial of service until the pending requests can be cleared.

These approaches can also be used against the primary firewall and DNS servers with slight variations. The firewall is based on IPCOP 1.3 which uses the Linux kernel, so it may be vulnerable to exploits for IPCOP and/or Linux. Similarly, the DNS server is based on Windows 2000 and may be vulnerable to

Windows exploits.

It may be possible to counter these attacks by making sure that the devices have the latest patches installed. By having the system patched, they should be protected against older exploits. Also, the devices should be checked to make sure they are configured to take advantage of any countermeasures that may be available. For example, Linux provides the "syn_cookies" capability which when active, protects against a syn flood by not allocating resources for a new connection until a full handshake has been received.

To counter DNS and other host attacks, you could have multiple DNS servers located on separate networks and set the expiration and time-to-live values on the DNS records to short intervals. Having short time limits would allow you to change IP numbers of targeted servers. This would allow systems to issue requests that would be directed to the new IP number.

If a host on the service or internal networks is repeatedly attacked, the firewall or border router can be configured to drop all incoming packets from the attacking systems. In some cases, the amount of incoming data may be so great as to completely consume all of the available bandwidth of the Internet connection. In this case, the solution would be to contact your upstream ISP and have them drop the packets. This escalation may have to go through several levels of service providers depending on the number and location of the attacking systems.

Attack Plan to Compromise an Internal System

In this scenario, the goal is to compromise one of the internal systems located on the 10.10.10.0/24 network of Lesa Ludwig's network design shown above. An internal desktop system on this network segment was chosen for the following reasons:

- These systems tend to not be as up-to-date on patch levels as the perimeter defenses and service machines.
- They systems may not be monitored as closely as servers.
- Important system configuration settings and applications on these systems may be altered in insecure ways by unsuspecting users. A high percentage of desktop users use an administrator level account for routine usage.
- Because of the large number of exploits available, we will be targeting machines running Microsoft Windows (NT/W2K/XP) with the NTFS file system.

The most difficult part of compromising the target is to get a malicious program (also known as "malware") past the perimeter defenses. Most defenses are directed outward toward the Internet and control inbound access to internal systems. A direct inbound attack towards a 10.10.10.0/24 system would be very difficult due to the network address translation and packet filtering performed by the firewalls. Any attempt that was able to overcome the NAT issue would probably still be logged. A better approach would be to some how get the target system to initiate an outbound connection and request that the malware be downloaded and executed by the system itself.

So how do we get our malware payload installed onto the computer? We will be using a social engineering technique to identify a potential target and hopefully have them unsuspectingly trigger the installation. This requires a plausible scenario and a user without much computer security experience.

We will target a salesman in the marketing department. We will call the sales number and posing as a potential customer, obtain more information on the target, particularly the salesman's name and mailing address. This can be done several times to get a number of possible targets.

Next, we develop or modify an application that would make the target's job "easier, quicker, and more profitable". For example: "a free version of a client contact manager". This application would provide some contact management functionality, but would also contain the malware payload. As part of the application startup, an attempt would be to install this payload on the system.

An information package is developed and sent to the salesman. In the package, we include either a cd-rom containing the trojaned application, or a link to a web page to download it. The package would also contain a helpful guide telling the user that in order to run the program they may have to "click ok" if their system's firewall or anti-virus software issues any warning messages during startup.

Once the payload is executed on the target system, there are many possible exploits available to use to compromise the system. For our purposes, we will simply create a scheduled backdoor command shell using the "tini" application.

Tini was written by Arne Vidstron (available at http://ntsecurity.nu/toolbox/tini). Since tini is quite small (< 3kb) it is contained inside of the application executable in a data area. It is encrypted using a simple XOR operation and has been modified to use a network port other than its default of 7777, in this example we will use the port 54321.

The procedure to create the backdoor is as follows:

- The first thing that we want to do is hide our executable. We will do this by using a feature of the NTFS file system called "alternate data streams". Alternate data streams allow you to store files in an area of the file system that is normally not displayed to the user. We will store our version of tini as "C:\Documents and Settings\%USER%\Favorites:notepad.exe", where "%USER%" is the current user. Note: it is not necessary for this user to have administrator privileges.
- Modify the task scheduler using the "at" command to execute the tini tool after business hours (i.e. 3:00am local time).

- On another compromised remote system, open a netcat session at the appropriate time and connect to the compromised system. If the account running the tini, doesn't have administrator privileges, you can use other techniques (like buffer overflows) to escalate privileges or use null session exploits to grab the password file for later off-line cracking. In any case once you can connect to the target system with a command shell, any number of exploits will work.

Countermeasures for Compromised Internal System

It is important to note that this compromise did not rely on penetrating the perimeter defenses from outside. It took advantage of human nature to get an initial foothold and then used the common practice of allowing outgoing connections from internal machines to the internet. There are several steps that can be used to prevent this attack from succeeding:

- Train users to be very wary of using unknown or unsolicited software and web sites. It may be useful to have an dedicated system that is not connected to the Internet for evaluating software products. Even better is to have any new software packages sent to the IT support group for a security evaluation prior to running it.
- Have the firewalls use time-based egress filtering; don't allow outgoing connections outside of normal business hours. Alternately, you could filter all outgoing connections through proxy systems.
- Have the firewalls, log all outgoing connections. This would detect (but not prevent) the network connection for the backdoor.
- Install a personal firewall (like zone alarm) and other monitoring utilities (like winpatrol) on the internal desktop systems. See the web pages: http://zonelabs.com and http://www.winpatrol.com.

Appendix 1: Full vnumlparser.pl configuration file for test network simulation

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/var/vnuml/vnuml.dtd">
<vnuml>
  <global>
    <version>1.3</version>
    <simulation_name>vpn-test</simulation_name>
    <ssh_key>/root/.ssh/identity.pub</ssh_key>
    <automac/>
    <ip_offset>64</ip_offset>
    <host_mapping/>
  </global>
  <net name="west0"/>
  <net name="west1"/>
  <net name="west2"/>
  <net name="west3"/>
  <net name="west4"/>    <!-- west site vpn network segment -->
  <net name="west5"/>    <!-- west site web - database point-to-point -
->
  <net name="east0"/>
  <net name="east1"/>
  <net name="east2"/>
  <net name="east3"/>
  <net name="east4"/>
  <net name="east5"/>    <!-- east site vpn network segment -->
  <net name="INET"/>

<!-- west site network -->

  <vm name="netmgr">
    <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
    <if id="1" net="west0">
      <ipv4>10.1.10.251</ipv4>
    </if>
  </vm>
  <vm name="router1">
    <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
    <if id="1" net="INET">
      <ipv4>10.0.3.1</ipv4>
    </if>
    <if id="2" net="west0">
      <ipv4>10.1.10.1</ipv4>
    </if>
    <if id="3" net="west1">
      <ipv4>10.1.1.1</ipv4>
    </if>
    <route type="inet" gw="10.1.1.2">10.1.20.0/24</route>
    <route type="inet" gw="10.1.1.2">10.1.30.0/24</route>
    <route type="inet" gw="10.0.3.2">default</route>
    <forwarding type="ip" />
  </vm>
  <vm name="ntp1">
    <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
    <if id="1" net="west1">
```

```
        <ipv4>10.1.1.10</ipv4>
      </if>
      <if id="2" net="west0">
        <ipv4>10.1.10.10</ipv4>
      </if>
      <route type="inet" gw="10.1.1.1">default</route>
      <forwarding type="ip" />
    </vm>
    <vm name="ns1">
      <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
      <if id="1" net="west1">
        <ipv4>10.1.1.11</ipv4>
      </if>
      <if id="2" net="west0">
        <ipv4>10.1.10.11</ipv4>
      </if>
      <route type="inet" gw="10.1.1.1">default</route>
      <forwarding type="ip" />
    </vm>
    <vm name="www1">
      <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
      <if id="1" net="west1">
        <ipv4>10.1.1.12</ipv4>
      </if>
      <if id="2" net="west0">
        <ipv4>10.1.10.12</ipv4>
      </if>
      <route type="inet" gw="10.1.1.1">default</route>
      <forwarding type="ip" />
    </vm>
    <vm name="mail1">
      <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
      <if id="1" net="west1">
        <ipv4>10.1.1.21</ipv4>
      </if>
      <if id="2" net="west0">
        <ipv4>10.1.10.21</ipv4>
      </if>
      <route type="inet" gw="10.1.1.1">default</route>
      <forwarding type="ip" />
    </vm>
    <vm name="router2">
      <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
      <if id="1" net="west1">
        <ipv4>10.1.1.2</ipv4>
      </if>
      <if id="2" net="west0">
        <ipv4>10.1.10.2</ipv4>
      </if>
      <if id="3" net="west2">
        <ipv4>10.1.20.1</ipv4>
      </if>
      <if id="4" net="west3">
        <ipv4>10.1.30.1</ipv4>
      </if>
      <route type="inet" gw="10.1.1.1">default</route>
      <forwarding type="ip" />
```

```
</vm>
<vm name="corp1">
  <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
  <if id="1" net="west3">
    <ipv4>10.1.30.10</ipv4>
  </if>
  <route type="inet" gw="10.1.30.1">default</route>
</vm>

<vm name="intweb">
  <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
  <if id="1" net="west2">
    <ipv4>10.1.20.10</ipv4>
  </if>
  <if id="2" net="west0">
    <ipv4>10.1.10.10</ipv4>
  </if>
  <if id="3" net="west5">
    <ipv4>10.1.50.10</ipv4>
  </if>
  <route type="inet" gw="10.1.20.1">default</route>
</vm>

<vm name="srvweb">
  <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
  <if id="1" net="west2">
    <ipv4>10.1.20.11</ipv4>
  </if>
  <if id="2" net="west0">
    <ipv4>10.1.10.11</ipv4>
  </if>
  <if id="3" net="west5">
    <ipv4>10.1.50.11</ipv4>
  </if>
  <route type="inet" gw="10.1.20.1">default</route>
</vm>

<vm name="dbsrv">
  <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
  <if id="1" net="west5">
    <ipv4>10.1.50.50</ipv4>
  </if>
  <if id="2" net="west0">
    <ipv4>10.1.10.50</ipv4>
  </if>
</vm>

<vm name="gw1">
  <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
  <if id="1" net="west2">
    <ipv4>10.1.20.31</ipv4>
  </if>
  <if id="2" net="west0">
    <ipv4>10.1.10.31</ipv4>
  </if>
  <route type="inet" gw="10.1.20.1">default</route>
</vm>
```

```
  <vm name="gw1a">
    <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
    <if id="1" net="west2">
      <ipv4>10.1.20.32</ipv4>
    </if>
    <if id="2" net="west0">
      <ipv4>10.1.10.32</ipv4>
    </if>
    <route type="inet" gw="10.1.20.1">default</route>
  </vm>

  <vm name="gw1b">
    <filesystem type="cow">/var/vnuml/root_fs_vpn_gw1b</filesystem>
    <if id="1" net="west2">
      <ipv4>10.1.20.33</ipv4>
    </if>
    <if id="2" net="west0">
      <ipv4>10.1.10.33</ipv4>
    </if>
    <if id="3" net="west4">
      <ipv4>10.1.40.2</ipv4>
    </if>
    <route type="inet" gw="10.1.20.1">default</route>

  </vm>

  <vm name="intns1">
    <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
    <if id="1" net="west2">
      <ipv4>10.1.20.34</ipv4>
    </if>
    <if id="2" net="west0">
      <ipv4>10.1.10.34</ipv4>
    </if>
    <route type="inet" gw="10.1.20.1">default</route>
  </vm>

  <vm name="imail1">
    <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
    <if id="1" net="west2">
      <ipv4>10.1.20.41</ipv4>
    </if>
    <if id="2" net="west0">
      <ipv4>10.1.10.41</ipv4>
    </if>
    <route type="inet" gw="10.1.20.1">default</route>
  </vm>

<!-- east site network -->

  <vm name="router3">
    <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
    <if id="1" net="INET">
      <ipv4>10.0.3.100</ipv4>
    </if>
    <if id="2" net="east0">
      <ipv4>10.2.10.1</ipv4>
    </if>
    <if id="3" net="east1">
      <ipv4>10.2.1.1</ipv4>
    </if>
    <if id="4" net="east3">
      <ipv4>10.2.30.1</ipv4>
    </if>
    <if id="5" net="east4">
      <ipv4>10.2.20.1</ipv4>
    </if>
    <route type="inet" gw="10.0.3.2">default</route>
    <forwarding type="ip" />
  </vm>
  <vm name="www2">
    <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
    <if id="1" net="east0">
      <ipv4>10.2.10.3</ipv4>
    </if>
    <if id="2" net="east1">
      <ipv4>10.2.1.3</ipv4>
    </if>
    <route type="inet" gw="10.2.1.1">default</route>
    <forwarding type="ip" />
  </vm>
  <vm name="corp2">
    <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
    <if id="1" net="east3">
      <ipv4>10.2.30.100</ipv4>
    </if>
    <route type="inet" gw="10.2.30.1">default</route>
  </vm>
  <vm name="netmgr2">
    <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
    <if id="1" net="east4">
      <ipv4>10.2.20.21</ipv4>
    </if>
    <if id="2" net="east0">
      <ipv4>10.2.10.21</ipv4>
    </if>
    <forwarding type="ip" />
  </vm>
  <vm name="gw2a">
    <filesystem type="cow">/var/vnuml/root_fs_tutorial</filesystem>
    <if id="1" net="east4">
      <ipv4>10.2.20.22</ipv4>
    </if>
    <if id="2" net="east0">
      <ipv4>10.2.10.22</ipv4>
    </if>
    <route type="inet" gw="10.2.20.1">default</route>
    <forwarding type="ip" />
  </vm>
  <vm name="gw2b">
    <filesystem type="cow">/var/vnuml/root_fs_vpn_gw2b</filesystem>
    <if id="1" net="east4">
      <ipv4>10.2.20.23</ipv4>
    </if>
    <if id="2" net="east0">
      <ipv4>10.2.10.23</ipv4>
    </if>
    <if id="3" net="east5">
      <ipv4>10.2.40.6</ipv4>
    </if>
    <route type="inet" gw="10.2.20.1">default</route>
    <forwarding type="ip" />
  </vm>

<!-- host connection -->

  <host>
    <hostif net="INET">
      <ipv4>10.0.3.2</ipv4>
    </hostif>
    <route type="inet" gw="10.0.3.1">10.1.0.0/16</route>
    <route type="inet" gw="10.0.3.100">10.2.0.0/16</route>
  </host>
</vnuml>
```

Appendix 2: Revised nmap patch

```
*** nmap.cc-dist    2003-12-01 17:09:39.000000000 -0800
--- nmap.cc  2004-02-27 18:33:03.000000000 -0800
***************
*** 1001,1013 ****
      }

      if (currenths->timedout) {
!       log_write(LOG_NORMAL|LOG_SKID|LOG_STDOUT,"Skipping host %s due to host
timeout\n", currenths->NameIP(hostname, sizeof(hostname)));
        log_write(LOG_MACHINE,"Host: %s (%s)\tStatus: Timeout",
                 currenths->targetipstr(), currenths->HostName());
!     } else {
        printportoutput(currenths, &currenths->ports);
        printosscanoutput(currenths);
!     }

      if (o.debugging) log_write(LOG_STDOUT, "Final times for host: srtt: %d
rttvar: %d  to: %d\n", currenths->to.srtt, currenths->to.rttvar, currenths-
>to.timeout);
      log_write(LOG_NORMAL|LOG_SKID|LOG_STDOUT|LOG_MACHINE,"\n");
--- 1001,1015 ----
      }

      if (currenths->timedout) {
!       log_write(LOG_NORMAL|LOG_SKID|LOG_STDOUT,"Results for host %s
incomplete due to host timeout\n",
!                currenths->NameIP(hostname, sizeof(hostname)));
        log_write(LOG_MACHINE,"Host: %s (%s)\tStatus: Timeout",
                 currenths->targetipstr(), currenths->HostName());
!     }
!     {
        printportoutput(currenths, &currenths->ports);
        printosscanoutput(currenths);
!     }

      if (o.debugging) log_write(LOG_STDOUT, "Final times for host: srtt: %d
rttvar: %d  to: %d\n", currenths->to.srtt, currenths->to.rttvar, currenths-
>to.timeout);
      log_write(LOG_NORMAL|LOG_SKID|LOG_STDOUT|LOG_MACHINE,"\n");
*** portlist.cc-dist      2003-09-20 02:03:00.000000000 -0700
--- portlist.cc     2004-02-28 13:51:06.000000000 -0800
***************
*** 507,512 ****
--- 507,516 ----
    } else if (state_counts[PORT_UNFIREWALLED] >
           state_counts[PORT_CLOSED]) {
      ignored = PORT_UNFIREWALLED;
+   } else if (state_counts[PORT_OPEN] > state_counts[PORT_CLOSED] +
+                                      state_counts[PORT_FIREWALLED] +
+                                      state_counts[PORT_UNFIREWALLED]) {
+     ignored = PORT_OPEN;
    } else ignored = PORT_CLOSED;

    if (state_counts[ignored] < 10)
*** scan_engine.cc-dist   2004-02-27 18:19:32.000000000 -0800
--- scan_engine.cc  2004-02-27 18:27:27.000000000 -0800
***************
*** 1803,1823 ****
      }
    } while(changed && ++tries < 100);
```

```
!   openlist = testinglist;

    if (o.debugging || o.verbose)
      log_write(LOG_STDOUT, "The %s took %ld %s to scan %d ports.\n",
scantype2str(scantype), (long) time(NULL) - starttime, (((long) time(NULL) -
starttime) == 1)? "second" : "seconds",  numports);

!   for (current = openlist; current;  current = (current->next >= 0)?
&scan[current->next] : NULL) {
!     if (scantype == IPPROT_SCAN)
!       target->ports.addPort(current->portno, IPPROTO_IP, NULL, PORT_OPEN);
!     else if (scantype != UDP_SCAN)
!       target->ports.addPort(current->portno, IPPROTO_TCP, NULL, PORT_OPEN);
!     else
!       target->ports.addPort(current->portno, IPPROTO_UDP, NULL, PORT_OPEN);
!   }

!  superscan_timedout:

    free(scan);
    close(rawsd);
--- 1803,1831 ----
      }
    } while(changed && ++tries < 100);

!  superscan_timedout:    /* cph - give a hint about open ports anyway */

    if (o.debugging || o.verbose)
      log_write(LOG_STDOUT, "The %s took %ld %s to scan %d ports.\n",
scantype2str(scantype), (long) time(NULL) - starttime, (((long) time(NULL) -
starttime) == 1)? "second" : "seconds",  numports);

!   /* cph 2001/08/07 - all ports on both openlist and testinglist are
!    * potentially open, must mark them all as open for the results.
!    * Watch out for the messy logic here. */
!   current = openlist;
!   do {
!     for (; current; current = (current->next >= 0) ? & scan[current->next] :
NULL) {
!       if (scantype == IPPROT_SCAN)
!      target->ports.addPort(current->portno, IPPROTO_IP, NULL, PORT_OPEN);
!       else if (scantype != UDP_SCAN)
!      target->ports.addPort(current->portno, IPPROTO_TCP, NULL, PORT_OPEN);
!       else
!      target->ports.addPort(current->portno, IPPROTO_UDP, NULL, PORT_OPEN);
!     }

!     /* If we still have testinglist to do, get it now */
!     current = testinglist;
!     testinglist = NULL;
!   } while (current);

    free(scan);
    close(rawsd);
```

References

General Reference used throughout this study: GCFW (GIAC Certified Firewall Analyst) Training Track, presented Oct., 2003:

Brenton, Chris. et al. Track 2.1 – TCP/IP. SANS Institute. 2003.
Brenton, Chris. et al. Track 2.2 – Packet Filters. SANS Institute. 2003.
Brenton, Chris. et al. Track 2.3 – Firewalls. SANS Institute. 2003.
Brenton, Chris. et al. Track 2.4 – Defense in Depth. SANS Institute. 2003.
Brenton, Chris. et al. Track 2.5 – VPNS. SANS Institute. 2003.
Brenton, Chris. et al. Track 2.6 – Network Design and Assessment. SANS Institute. 2003.

Reference for Design Under Fire (Part 4): GCIH (GIAC Certified Incident Handler) Training Track, presented Jan. 2004:

Skoudis, Ed. et al. Track 4.2 – Computer and Network Hacker Exploits, Part 1. SANS Institute. 2004.
Skoudis, Ed. et al. Track 4.3 – Computer and Network Hacker Exploits, Part 2. SANS Institute. 2004.
Skoudis, Ed. et al. Track 4.4 – Computer and Network Hacker Exploits, Part 3. SANS Institute. 2003.

Linksys BFSR41 Cable/DSL Router with 4-Port switch (reference web page: http://www.linksys.com/products/product.asp?grid=34&scid=29&prid=561)

Linksys WRT54G Wireless-G Broadband Router (reference web page: http://www.linksys.com/products/product.asp?grid=33&scid=35&pid=577)

Microsoft Windows XP: http://www.microsoft.com/windowsXP

Mozilla Firebird 0.7 web browser (http://mozilla.org/products/firebird)

Mozilla Thunderbird 0.4 E-mail client (http://mozilla.org/products/thunderbird)

PuTTY 0.53b (http://www.chiark.greenend.org.uk/~sgtatham/putty).

WinZip file compression/archiver 8.0 (http://www.winzip.com)

ZoneAlarm Pro 4: http://www.zonelabs.com

Red Hat Fedora Core-1 (http://fedora.redhat.com/)

User Mode Linux (http://user-mode-linux.sourceforge.net)

AboutTime4.8 network time client (http://www.arachnoid.com)

Spoofed input packets rejection criteria: Linux Firewalls, Robert L. Ziegler, 2nd edition, New Riders Publishing, 2001, page 36-37
Network architecture best practices: Linux Firewalls, 2nd edition, by Robert

Ziegler, New Riders Publishing, 2002, pg 214-232.

Construction of Firewall and NAT rules: Linux Firewalls, 2nd edition, by Robert Ziegler, New Riders Publishing, 2002, chapters 6 and 7.

Network Time Protocol (NTP) project: http://www.ntp.org

Berkeley Internet Name Domain (BIND) provided by the Internet Software Consortium (http://www.isc.org/products/BIND).

Load balancing name server lbnamed: written by Roland Schemers and currently maintained by Rob Riepel both of Stanford. http://www.stanford.edu/~riepel/lbnamed.

Apache Software Foundation's HTTP server version 1.3.29: http://httpd.apache.org

PHP Project's PHP preprocessor version 4.3.4: http://www.php.net.

OpenSSL Project's OpenSSL (secure sockets layer) library: http://www.openssl.org

Verisign Inc. provides signed SSL certificates: http://www.verisign.com.

Tutorial: “Configuring an Open Source Mail Gateway” by David Handermann. This tutorial was posted on May 31, 2003 on the Freshmeat web site, a direct link to this tutorial is: http://freshmeat.net/articles/view/857.

AmaViS (A Mail Virus Scanner), home page: http://www.amavis.org.

SpamAssassin, by the SpamAssassin development team, home page: http://www.spamassassin.org.

MIMEDefang by Roaring Penguin Software Inc., home page: http://www.roaringpenguin.com/products/mimedefang.

Clam AntiVirus, by the Clam AntiVirus development team, home page: http://www.clamav.net.

Postfix email system, by Wietse Venema, home page: http://www.postfix.org.
Perl Language, written by Larry Wall, home page: http://www.perl.org.

Snort IDS system, written by Marty Roesch, home page: http://www.snort.org.

“Snort 2.0: Intrusion Detection”, Brian Caswell, Jay Beale, James Foster, Jeffrey Posluns, et al., 2003, Syngress Publishing Inc.

OpenVPN Virtual Private Network by James Yonan jim@yonan.net, home page: http://openvpn.sourceforge.net

“Rsync” file transfer utility written by Andrew Tridgell, home page: http://rsync.samba.org.

Samba SMB/CIFS file and print server software developed and maintained by the Samba Team, home page: http://samba.org.

Chapter 15 (Securing Samba) of Samba documentation (written by Andrew Tridgell and John H. Terpstra), home page: http://de.samba.org/samba/docs/man/securing-samba.html

“OpenVPN HOWTO”, James Yonan, contained in the OpenVPN source code distribution located at http://openvpn.org/howto.html.

“Setting up VPN using OpenVPN”, Project VOLANS, located at http://mia.ece.uic.edu/~papers/volans/openvpn.html

“How to set up your own Certification Authority”, Project VOLANS, located at http://mia.ece.uic.edu/~papers/volans/settingupCA.html.

“Virtual Network User Mode Linux”, The Virtual Network User Mode Linux (VNUML) project, located at http://jungla.dit.upm.es/~vnuml.

Design under Fire. The network design chosen for analysis was presented by Lesa Ludwig’s GCFW Practical of October 2003. Her presentation is available at the GIAC website: http://www.giac.org/practical/GCFW/Lesa_Ludwig_GCFW.pdf

MyDoom Upload/Exec backdoor exploit code”: contained in an E-mail message from K-OTik Security <Special-Alerts@k-otik.com> to the bugtraq mailing list (Bugtraq@securityfocus.com) list with the Subject: “MyDoom.A Machines: The new P2P Sharing Network”.

Cisco 1711 (really IOS) DOS exploit: An example of using an exploit to crash the device would be to send an ill-formed packet as described in http://www.k-otik.com/exploits/07.22.ciscodos.sh.php

Windows backdoor listener tool: “Tini”, tini was written by Arne Vidstron and is available at http://ntsecurity.nu/toolbox/tini.

Linux network stack rate limiting for udp scans: extracted from the nmap manual page at http://www.insecure.org/nmap/data/nmap_manpage.html).

Nmap port scanner patch: “Re: OS Scan & Print Port Output after host timeout”, by Colin Phipps. Nmap development mailing list: http://seclists.org/lists/nmap-dev/2002/Jul-Sep/0037.html.

“Tcpreplay” utility, by Aaron Turner. http://tcpreplay.sourceforge.net.