Global Information Assurance Certification Paper

SANS GCFW Practical Assigment

v. 2.0

GIAC Enterprises, On-line Fortune Sayings

By Annie De Montigny-Leboeuf

April 7, 2004

Abstract

GIAC Enterprises is a small e-business delivering fortune sayings to on-line customers. These fortune sayings are similar to those found inside of Chinese fortune cookies. Customers subscribed to GIAC services on a monthly basis. The service entitles the customers to one fortune saying every day. Customers can access their fortune saying by electronic mail, and/or by connecting to GIAC Enterprise's web site (using a login and password). This paper describes the network design and infrastructure to support the business operation.

GIAC Entreprises is a starting business and must operate based on a very tight budget. The network is therefore a Personal Computer based network, with open-source software. The network security is taken seriously, and much efforts are put towards the protection of the assets.

Our description will include a basic overview of the business operation and traffic flows. The network components will be described with much details, especially the components of the perimeter network. The perimeter components of GIAC network are the border gateway, the primary firewall, and the Virtual Private Network (VPN) gateway. For these systems, we provide the complete scripts that implements GIAC's security policy.

Assigment 1 - Security Architecture

After ten months of market study, planning and publicity, GIAC Enterprises is ready to initiate its first pilot year of business operation. The few hundreds of "registered" customers are looking forward to receiving their first on-line, personalized, fortune sayings for Easter 2004.

GIAC Enterprises is a small e-business delivering fortune sayings to on-line customers. This e-business has recognized a trend among young people who wish to include spiritual insights into their busy life. The inspiration and awakening that fortune sayings can provide make people's day more joyful and fructuous. This business is also attractive to people who already consult psychics and read horoscopes.

Business operation at glance

Partners

GIAC enterprises is the result of a 6-people effort. These people are considered equal partners. All partners are located in the Ottawa valley area. Two of them live at the same address, where the business network is set-up. The other four partners require Virtual Private Network access to this network from their home (the access requirements will be detailed later on).

Partners are colorful and imaginative individuals willing to give their time and effort for the success of the company. With creativity, humor, and wisdom, they have managed within the first ten months to build a database of over 2000 different fortune sayings, distributed according to 8 groups of age and personality. Through personal contacts, psychic fairs, and tradeshows the team has recruited close to 350 subscribers.

All partners have agreed to dedicate a minimum of 15 hours per week for the next 2 years (excluding the first ten months of preparation) to GIAC enterprise.

The four people working remotely are responsible for updating the database with new fortune sayings. They must also find the time to respond to the public and customer's information requests and comments. They also take turn for advertising the company at tradeshows and other events (13 events have been identified and will be targeted over the next twelve months).

The two people on-site are qualified for maintaining and securing the network. Aside from the networking tasks, one of the two fellow is responsible for all the corporate administrative details, while the other does software development to enhance GIAC enterprises' FORTUNE® engine.

Customers

Customers subscribe on a monthly basis, and deals are available on 3-month and 12-month subscriptions. The service entitles the customer to one fortune saying every day. GIAC Enterprise guarantees delivery of fortune saying by 6hAM, customer local time.

Customers can access their fortune saying by electronic mail, and/or by connecting to GIAC Enterprise's web site (using a login and password). The subscription price is the same whether the customer chooses

- electronic mail delivery service,
- web site login,
- electronic mail delivery service and web site login.

The web site login option was added to the primary delivery method (e-mail) to accommodate customers who travel or who wish to obtain their fortune saying at a later time during the day (e.g. from work). The web login option also caters to customers who are hesitant in giving away their e-mail address.

At the time of writing, GIAC enterprise is putting together the required documentation to file for a Certificate Signing Request (CSR) to VeriSign Certificate Authority (CA). A widely recognized CA for GIAC enterprise's SSL enabled web server will provide customers with the ability to register and pay their subscription on-line. Meanwhile, GIAC Enterprise is accepting cheques and cash payment at tradeshows and events.

Suppliers

Identifying the right profile for each customer is believed to be an important aspect for GIAC's enterprises success. A customer must be able to relate to the fortune sayings with his or her own life experience.

GIAC enterprise is out-sourcing Market studies and polls from a well established polling company located in Montreal. This supplier has also prepared for GIAC Enterprises a short questionnaire for new customers. The goal of this questionnaire is to best identify which of the 8 profiles a new customer best fits with. Of course, the questionnaire must be subtle enough so that the customer is unaware he or she is being profiled.

Because existing customers are often met at tradeshows, GIAC's enterprises partners are able to appreciate whether or not the questionnaire is helpful. The supplier has agreed to revisit or enhance this questionnaire at GIAC enterprises' demand.

Network infrastructure

Before describing the network lay-out, it is appropriate to explain some choices made with regard to hardware and operating systems.

When the project started to take form a year ago, it was decided that the security of this e-business would not be the sole responsibility of one guru. A bulletproof architecture can

fall apart if the only person who understands the network's dynamic is absent (or temporarily unavailable). Therefore, it was agreed that the design would be kept simple enough so that anyone of the partners would be able to take over. Tutorials on how to make modifications to the systems would be carefully written and revised by all partners. These documents will be kept up-to-date as changes to the network are made.

This is a PC-based network. The only "specialized" networking equipments are layer-2 (data link) switches. The server, firewall and gateways are built on Personal Computers (PC). The PCs are Intel based and, aside from the network cards, they all have the same hardware components (we will explain why shortly).

There are two different operating system (OS) versions in the network:

- Redhat 9 Linux kernel 2.4.20-30.9, and
- Open BSD 3.4.

The co-existence of two different operating system families adds a little diversity, and therefore contributes to enhancing the security of the network. This adds a second line of defense in times of attack, especially if a relay (e.g. a mail relay) has a different OS and application than the actual server (e.g. mail server). Moreover, "two" is still small enough to remain easily manageable in terms of system updates and patches. This also contributes to the security of the network by allowing the system administrator to keep the systems up-to-date in a reasonable amount of time. The idea is to stay as close as possible to "day-zero" when vulnerability is announced.

Redhat 9 was chosen because all partners are well acquainted with the Redhat Linux environment. Redhat 9 is the operating system running on the teleworkers' workstation as well as on many of the network components. OpenBSD 3.4 was chosen as the second operating system for three reasons: (1) because the OpenBSD project team efforts place emphasis on security (http://www.openbsd.org); (2) while not as obvious as Linux in terms of getting things to work, OpenBSD is popular enough to allow us to find detailed instructions and explanations on the web; and (3), well, we are Canadian!

Now, the reason for choosing the same hardware for all PCs is to allow major reconfigurations to go smoothly. A major reconfiguration could involve a change of OS for a gateway or a server. We purchased a few extra hard drives in prevision of this. The idea is to proceed with the installation from one of the workstation (on a new hard drive). Once everything is set-up and tested, we would then shut down the computer that needs maintenance (most likely at night if this turns out to be the least busy hours), change its hard drive and reboot the system.

It is already foreseen that the border gateway (currently a Linux RH9 kernel 2.4.20-30.9) would be changed to OpenBSD 3.4. The reason for this change is because the primary firewall is also Linux RH9 kernel 2.4.20-30.9 with NetFilter. Because the admin had more experience with NetFilter rules than ipfilter rules, both the firewall and the gateway were chosen to run Linux for the beginning of operation. Now that the admin has gained

some experience with ipfilter (http://www.ipfilter.org). This will contribute to add defense in depth to the architecture.

The other foreseen change, yet less urgent, is with Linux RH9. Red Hat 9 can still be used right now, and into the future. But Red Hat will not be maintaining a support program for it after April 2004. Fortunately, at least one other company, "Progeny", will be carrying on with the security notifications via email, and the software updates, for Red Hat 7-9 (See www.progeny.com/products/transition). At the time of writing, not all GIAC partners are ready for a change. An alternative OS is yet to be determined.

Installation and System updates

The guideline is to install systems with a minimal number of packages (what is needed and nothing more). In particular, compilers are removed from all systems once they are configured. This mitigates the risk that an attacker can compile code and execute programs.

Network services and client applications are not installed if not required. In particular, the firewalls and the border gateway do not offer network services. Remote maintenance is not required.

We will run all network processes in a restricted-user/chroot environment, so the amount of damage would be seriously limited if an attacker were able to compromise one of the modules.

User workstations' updates are the responsibility of the users. Redhat systems are upgraded using the RedHat Update Agent (up2date), with the "do not install packages after retrieval" configuration setting. The user then decides what he wishes to update. The traffic involved is SSL.

The system administrator located at GIAC enterprises site is responsible for updating the servers, firewalls and gateways. Patches and updates are downloaded from the admin workstation. The downloaded files are then copied onto a 1GB USB memory stick, and moved to the system to be patched. The system administrator can download the patches and updates from ftp, http, or https sites. He can also access CVS pserver using SSH port forwarding. CVS access using SSH is the method the admin prefers for retrieving OpenBSD updates.

Because the network is of relatively small scale and because components are located in the same room, a centralized syslog server is considered unnecessary at this time.

Network Lay-out

The network lay-out is depicted in the figure below:

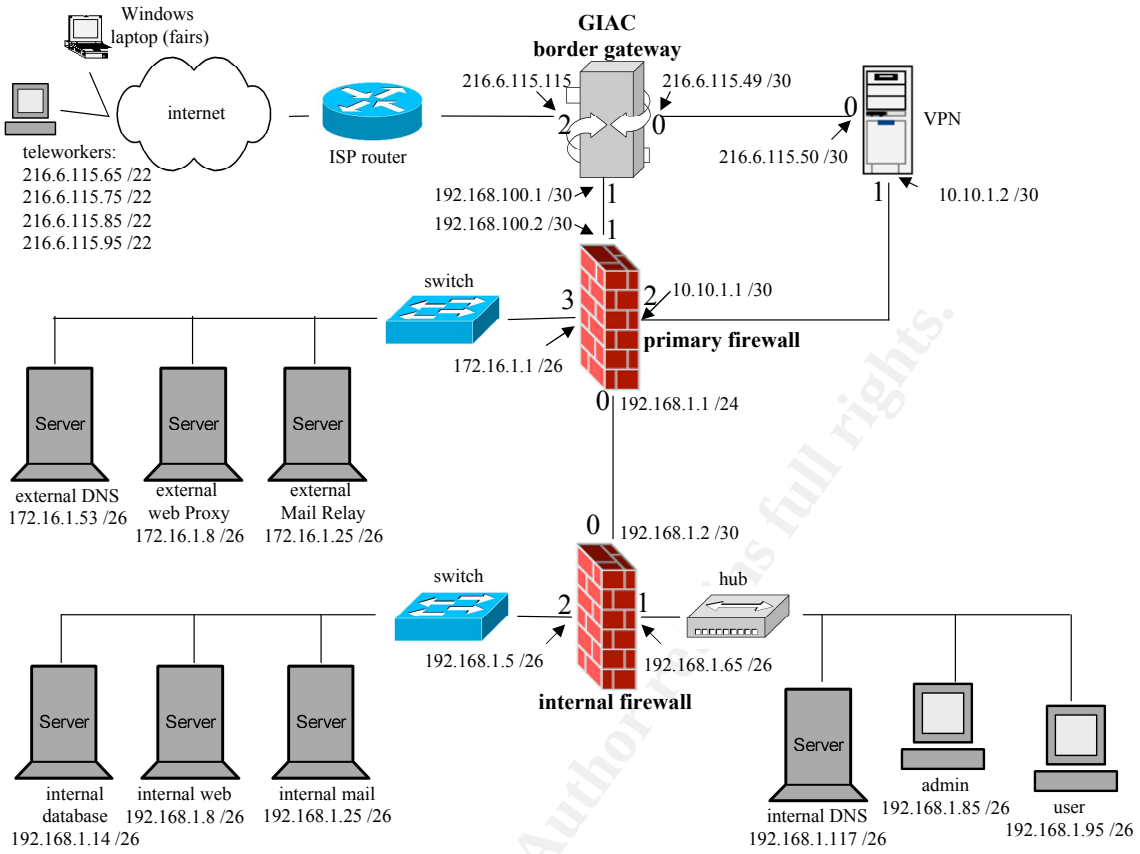


Figure 1

Disclosure: the routable IP addresses appearing in this document do not really belong to GIAC enterprises, they were borrowed from an ISP provider of the area for which it was easy to find information about prices, services, and addresse scheme.

Address scheme

GIAC enterprises and the four teleworkers are registered with the same Internet Service Provider. For each line, this ISP provides one static IP address, with an extra charge of 5$ per additional static IP address.

GIAC Enterprises has subscribed to a 1.5M Commercial ADSL line for the office with 2 static IP address: one for the border gateway, and one for the VPN gateway. The company has subscribe to four 1M Residential ADSL line for the teleworkers. Each teleworker has been assigned a static IP address.

Aside for the external interfaces of the border and VPN gateways, the address scheme of GIAC Enterprises network uses private addresses (from RFC 1918 address space).

- Primary Border Non-Trusted Network (NTN)[1] (border gateway to primary firewall): 192.168.100.0/30
- Encrypted Border NTM (border gateway to VPN gateway): 216.6.115.49/30
- Unencrypted NTN (VPN gateway to primary firewall): 10.10.1.0/26
- Service network: 172.16.1.0/26
- Internal network: 192.168.1.0/24
  - Internal Protected network: 192.168.1.0/26
  - Internal User network: 192.168.1.64/26

The IP addresses of all systems are given in Figure 1.

The address scheme was chosen with very distinct address space to recognize quickly the location of the source and destination appearing in logs or traffic traces. A color scheme to differentiate between the distinct sections of the network is provided in Figure 2. The scheme choice obviously has a major impact on routing, static routes are required on several systems for network interconnectivity.

Infrastructure components

This section describes the components of the network, with details on how they are configured.

[1] Non-Trusted Network (NTN) is not a common term. I made it up to refer to a zone where the traffic is within the corporate network, but has not yet been filtered to comply with the corporate security policies. I chose to use this term instead of DMZ since this term is sometime used to refer to what we called here the "Service Network". The "Service Network", is hosting the servers to which the internet can connect (i.e. where inbound connections are allowed).

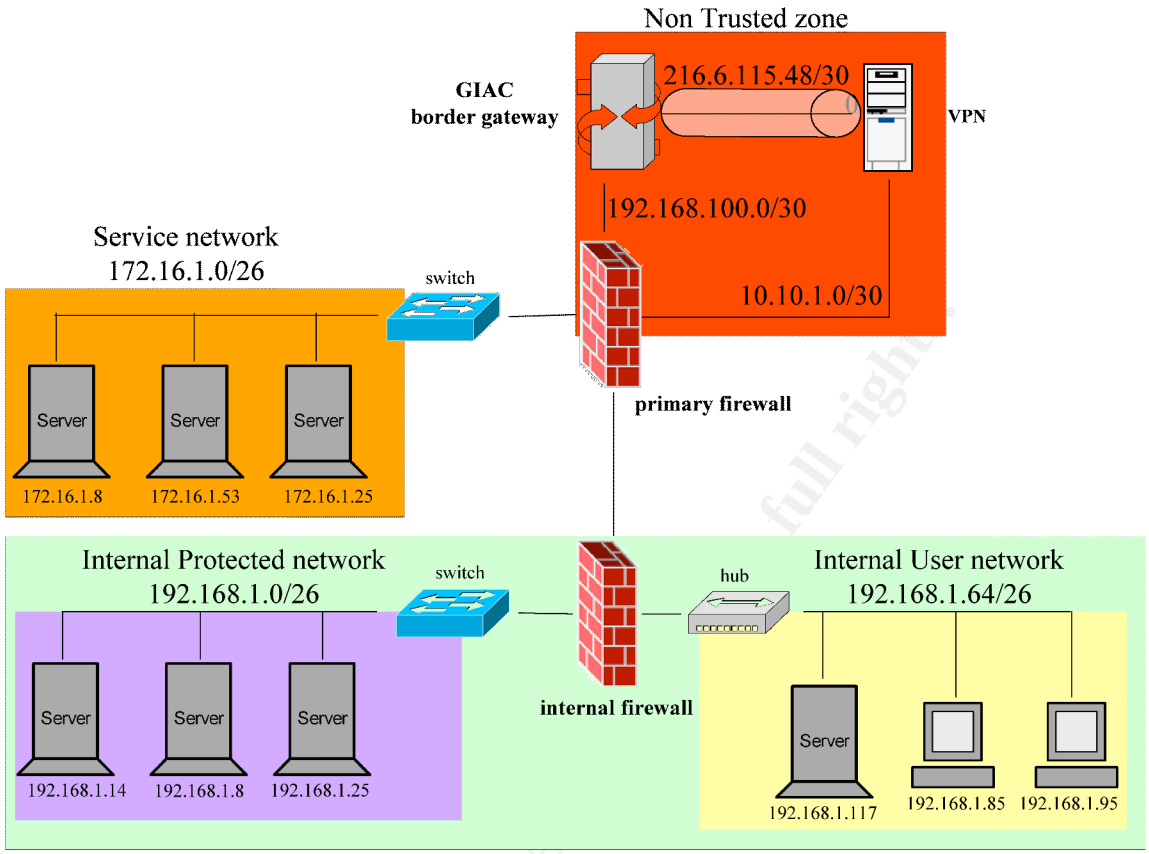


Figure 2

Border gateway

The border gateway is a hardened Linux Red hat 9.0 (kernel 2.4.20-30.9) with minimal installation. It has the roaring penguin (http://www.roaringpenguin.com) package to support the PPPoE link from our ISP. It is running NetFilter with stateless rules to filter out some of the undesired traffic. This pre-filtering reduces some of the load on the primary firewall. A strict "deny all unless specifically allowed" policy is used.
The border gateway is configured to do port forwarding to relay inbound traffic to the appropriate server. The Network Address translation (NAT) is done at the border gateway as well. The NAT rule is written so that the VPN's routable IP address remains unchanged. While there are some workarounds for FreeSwan ipsec to function despite NATed addresses[2], it is much simpler to avoid NAT situations along the tunnel.

Primary firewall

The primary firewall is a hardened Linux Linux Red hat 9.0 (kernel 2.4.20-30.9) with minimal installation. It is running NetFilter with stateful packet filtering to prevent unsolicited (or "out of sync") packets to pass through. A strict "deny all unless specifically allowed" policy is used. The rules discriminate based on fields at the network and transport layers. We choose not to inspect content at the application layer.

[2] See http://www.freeswan.org/freeswan_trees/freeswan-2.05/doc/HowTo.html

While this reduces the load on the firewall, it also comes with the risk of letting some attack go through undetected. For this reason, we will have relays further down the road to add another layer of protection between the clients and our servers.

VPN gateway
The VPN gateway is a hardened Linux Red hat 9.0 (kernel 2.4.20-30.9) with minimal installation. It is running FreeS/WAN v 2.05 (http://www.freeswan.org/). FreeS/WAN is an open source implementation of IPSec. The VPN gateway is also running NetFilter as a stateful packet filtering firewall. Access to the GIAC network is restricted to the teleworkers.

HTTP/FTP proxy (located on the service network)
The proxy web/ftp server is a hardened OpenBSD 3.4 with minimal installation. It is running Squid 2.5 (http://www.squid-cache.org/). This machine contains no useful information. It is there to segregate the client from the server and prevents them from communicating directly. This can help protect the server from certain attack. Due to budget constrain that prevents us from deploying two proxy servers, Squid is configured to serve inbound and outbound request at the same time. Internal users and teleworkers need to be configured their browser to direct their FTP, HTTP, and HTTPS requests to the proxy on port TCP/3128 (the default port). The sources of outbound requests are restricted to IP addresses of GIAC enterprises (to prevent the Squid from becoming an open proxy to anyone on the internet). The Squid proxy is listening on TCP/80 and TCP/443 for internet requests to access the web server. We use Squid's httpd-accelerator with Jeanne (http://www.ists.dartmouth.edu/IRIA) to specify exactly wich files and directories are accessible on the web server.

HTTP server (located on the Internal Protected Network)
The web server is a hardened linux Redhat 9 kernel 2.4.20-30.9 with minimal installation. It is running Apache 1.3.29 (http://www.apache.org). The httpd.conf file has been modified to obfuscate the banner (which by default reveals the Apache version). PHP (http://www.php.net) has been installed, with built-in MySQL support, to handle dynamic content. The delivery application has been carefully written in-house. It has been audited for common vulnerabilities.

Mail Relay (located on the service network)
The mail relay is a hardened OpenBSD 3.4 with minimal installation. It relays mail to and from GIAC enterprises mail server. The mail relay contains no mailbox or any user accounts information. It is running Postfix 2.0 patchlevel 19 (www.postfix.org) which has a good record for security and is simple to configure. Amavisd-new (www.ijs.si/software/amavisd) is the virus scanner which processes email from postfix. SpamAssassin (www.spamassassin.org) is the main anti-spam component which works by comparing messages to a ruleset and by using a statistical analysis. In addition to the SpamAssassin spam detection software, we will be using two online SPAM databases:

DCC (www.rhyolite.com/anti-spam/dcc) and Vipul's Razor (razor.sourceforge.net). The details of installation can be found at http://www.oddquad.org/linux/anti-spam.html.

Mail Server (Internal Protected network)
The mail server is a hardened linux Redhat 9 kernel 2.4.20-30.9 with minimal installation. It is running Qmail (http://www.qmail.org/top.html). The reason for choosing Qmail was simply to have something different from Postfix, installed on the OpenBSD relay. Although this does not ensure that they would not be vulnerable to the same exploits, we still can hope they'd react differently, especially since the operating systems are different. It strips out headers to hide information about the system.

External DNS server
The external DNS server is a hardened OpenBSD 3.4 with minimal installation. It is configured with BIND 9.2.3 (http://www.isc.org). It contains the information that is required by internet users. The external DNS's role requires only that it respond to queries for which it knows the answers. Therefore, it is configured to be non-recursive. This will prevent internet users form using our limited resources. Zone transfers are only allowed to our ISP DNS server.
Reverse queries (IP addresses to names) issued from the internet are allowed for inqueries concerning GIAC's routable IP addresses (vpn and gateway). This is because GIAC is well intended and willing to cooperate with other companies if our routable IP addresses are found in their logs. In the event, allowing reverse mapping for our routable IP addresses would help these companies to find GIAC enterprises. The DNS server has an entry that gives a contact point. This contact point is an e-mail address provided by our ISP (ISP assigned us 5 e-mail addresses). This e-mail address will not be advertised anywhere else.

Internal DNS server
The internal DNS server is a hardened OpenBSD 3.4 with minimal installation. It is configured with BIND 9.2.3 and operates separately from the external DNS. The internal DNS server contains the information regarding GIAC Enterprises' addressing, and responds to queries initiated from GIAC Enterprises' IP addresses only. This DNS server is recursive. It does not allow any zone transfer. Zone transfers reveal too much information at once and is not required since our architecture does not include a secondary DNS server. It is located on the same network than the internal users because it is considered too potentially vulnerable to be located on the protected internal network. This is because it is allowed to contact other DNS on the internet for which we have no control on. Because it is allowed to send packets on the internet, it is a potential and attractive target to install a backdoor on, very much like any of our user workstation.

Internal database server
The internal DNS server is a hardened OpenBSD 3.4 with minimal installation. OpenBSD was chosen over linux because it is considered more secure. It is running MySQL 4.0 (http://www.mysql.com). This server serves as the backend for the internal

web server. It is also accessed by the teleworkers who upload new fortune sayings. While protected by two firewalls, the database server has IPFilter acting as a personal firewall. The database is what the company is all about; GIAC is willing to have redundant checks, at the risk of introducing delays, to better protect this machine.

Snort IDS sensor
The IDS sensor runs on the network administrator system located on the internal user network. It is a hardened linux Redhat 9 kernel 2.4.20-30.9. It is running Snort v2.1.2 (http://www.snort.org). We have a Shomiti taping device (http://www.finisar.com) that listens to all incoming/outgoing traffic on the database segment. The tapping device forwards the monitored traffic to the hub on the Internal User Network. Snort can therefore monitor all traffic on the Internal user network as well as all traffic related to the database.

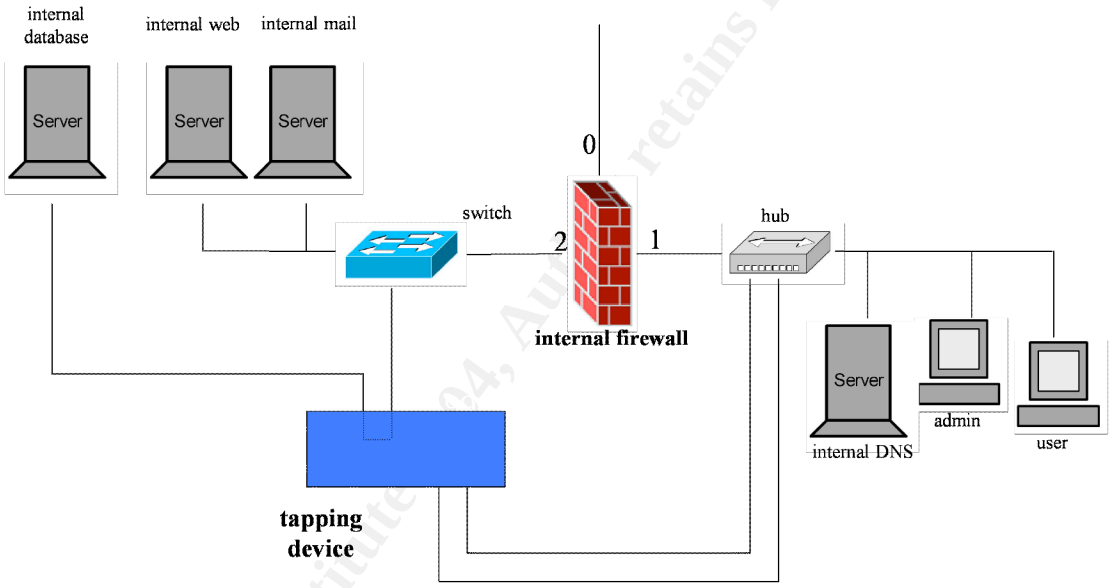


Figure 3

Access Requirements and Restrictions

Customers and General Public access

The general public and the customers will access the network through the web and mail servers. They will not contact these servers directly but through the mail relay and web proxy.

When customers access their fortune sayings from our web site, they send their logins and passwords in clear text. Because the fortune sayings are sent one at the time, GIAC Enterprises is comfortable with having them sent in the clear (whether it is through e-mail or http traffic).

Customers are reminded that their login and password are sent in clear text. They therefore should refrain from using a password common to any other account they may have. To quote:

> "*Please keep in mind that the login and password are sent in clear text (just like with hotmail or yahoo email accounts).*
> *For your own protection, GIAC Enterprises recommends that you select a password unique to this account. Using a password common with other accounts is not considered good security practice.*"

GIAC Enterprises is in the process of trying to acquire a trusted certificate from VeriSign for GIAC's SSL enabled web server. Once this is completed, customers will be able to register and pay their subscription online.

The firewall rules will plan ahead for SSL traffic, to which we refer below as HTTPS.

To allow customers and the general public to reach GIAC enterprises, DNS traffic to the external DNS server is allowed (only requests for our domain).

Traffic expected at the border gateway and the primary firewall:

- HTTP, HTTPS
  Communications initiated from the internet to the proxy web server
- SMTP
  Communications between the internet and the Mail relay, initiated from either direction.
- DNS
  Communications initiated from the internet to the External DNS server.

Systems on-site

The on-site traffic flows are broken down into two categories:

1. Personal workstations ↔ internet and servers on-site
2. Servers on-site ↔ internet and other servers on-site

1. Personal workstations (two internal users) ↔ internet and servers on-site

The two people on-site are responsible for maintaining and securing the network. One of them does software development to enhance GIAC enterprises' FORTUNE® engine.

Access by internal users (from their workstations) to the internet and to other systems on-site is limited to:

- Surf the internet with a browser (FTP/HTTP/HTTPS sites). Traffic must go through the squid proxy.
- Download programs and patches from ftp servers using an ftp client.
- Listen to real time streaming audio. Although users are asked to limit their bandwidth consumption by avoiding using real-time streaming applications, Real-time Streaming Protocol (RTSP) traffic is allowed so

that users can listen to the audio streams of some security webcast events (e.g. SANS http://www.sans.org/webcasts/).
- Use SSH to download patches. (OpenBSD system updates are done through remote CVS pserver access using SSH port forwarding).
- Send e-mails (SMTP) to the internal mail server.
- Retrieve e-mails (IMAP) from the internal mail server.
- Ping other sites, or other systems on-site to test for connectivity
- Query the internal DNS servers to resolve "names <-> IP addresses" for hosts on the local domain or on the internet. On-site users are located on the same subnet as the internal DNS server and therefore the traffic between an internal user and the DNS server will not cross the internal (nor the primary) firewall.
- Browse GIAC web server. Traffic must go through the squid proxy. Users may want to make sure it looks good.

Traffic expected at the internal firewall (with source or destination IP addresses of internal users):

- HTTP/HTTPS/FTP (TCP/3128), FTP, RTSP, SSH, SMTP, IMAP, ICMP echo request and reply
  Communications initiated from inside.

Traffic expected at the primary firewall (with source or destination IP addresses of internal users):

- HTTP/HTTPS/FTP (TCP/3128), FTP, RTSP, SSH, ICMP echo request and reply
  Communications initiated from inside.

Traffic expected at the gateway (with source or destination IP addresses of internal users):

- RTSP (TCP/554), SSH, FTP, ICMP echo request and reply
  Communications initiated from inside.

2. Servers on-site ↔ internet and other servers on site

To protect the internal systems from receiving unexpected traffic, GIAC Enterprises allows only the traffic required for business operation.

Access by systems on-site to the internet and to other systems on-site is limited to:

- Internal mail server communicates with the mail relay.
- Mail relay can communicate with the internet for SMTP traffic.
- Internal web server communicates with the database server and with the Squid proxy.
- Squid proxy can communicate with the internet for FTP, HTTP, HTTPS traffic. Note that FTP traffic is initiated from inside only. HTTP and HTTPS can be initiated from the internet (Squid then acts as a Reverse Proxy), or from GIAC's network.

- Internal DNS server can respond to enquiries from any of GIAC's internal systems. Internal DNS can also query other DNS server on the internet to recursively find responses.
- External DNS server can respond to queries issued from the internet. The external DNS also allows zone transfers to the ISP DNS server.

Traffic between servers on the Service network and servers on the Internal network will cross the primary firewall and internal firewall. Traffic between the servers on the Service network and the internet will cross the primary firewall and the border gateway. Traffic between the servers and the internal DNS server will cross one or two firewalls (depending on the location of the server issuing the query). Traffic between the internal DNS server and the internet will cross the internal firewall, the primary firewall and the border gateway.

Partners off-site (four teleworkers)

The four people working off-site are responsible for updating the database with new fortune sayings. They also respond to the public and customer's information requests and comments through e-mails.

Access to GIAC network will be allowed only after establishing a VPN connection to the VPN gateway. Once the tunnel is established, the following traffic can be expected:

- Use of a local SQL client to communicate with the remote database.
- Retrieve and send e-mails (IMAP and SMTP). Traffic is directed at the internal mail relay.
- Surf the internet using a browser configured to go through GIAC's proxy server (FTP/HTTP/HTTPS sites). Traffic must go through the squid proxy.
- Ping internal hosts to test for connectivity.

Traffic expected at the border gateway and at the vpn gateway (with source or destination IP addresses of teleworkers):

- IKE, ESP
  Communications initiated from the teleworker sites.

Traffic expected at the primary firewall (with source or destination IP addresses of teleworkers):

- MySQL, IMAP, SMTP, FTP/HTTP/HTTPS (TCP/3128), ICMP echo request and reply
  Communications initiated from the teleworker sites.

Traffic expected at the internal firewall (with source or destination IP addresses of teleworkers):

- MySQL, IMAP, SMTP, ICMP echo request and reply
  Communications initiated from the teleworker sites.

Partners on the road (at tradeshows)

There is nothing special in terms of access requirements when partners are at tradeshows. As it will become apparent shortly, partners at tradeshows have the same restrictions than the general public.

The tradeshows are generally a day or two long, and are chosen, for budget restrictions, not too far away from the main office (e.g. Montreal, Toronto, Ottawa). It is estimated that e-mail responses can wait for a day or two.

Partners on the road use a Windows laptop provided by GIAC enterprises. This laptop should not contain any sensitive data. It may contained a few recent documents or fortune sayings the partner is working on, but certainly not a complete copy of any database. The laptop is equipped with a personal firewall and anti-virus product.

To demonstrate the concepts to potential customers, the GIAC enterprises partner relies on the availability of internet access at the tradeshows. The partner access the web proxy server in the same manner as a customer would, only the partner uses special accounts:

blank login: in which he enters the first name of the person (potential client)
keying password: corresponding to one of the eight profiles (assumed to fit the best with the person).

And voilà! The customer is seduced with the personalized fortune saying! To get more, he must subscribe... cheque and cash accepted...

Traffic expected at the border gateway is already allowed for the customers and general public:

- HTTP, HTTPS
  Communications initiated from the internet (laptop with a possible dynamic address) to the proxy web server.

Suppliers

The supplier for our market studies and profiling questionaires is a fairly well established company, trusted by GIAC Enterprises. Nonetheless, the supplier is not granted access to GIAC enterprises network. This would be unnecessary. The supplier's sale representative informs GIAC enterprises of the readiness of documents. GIAC Enterprises then logs-in to supplier's SSL enabled web site and downloads the documents.

Traffic IN is already allowed for the customers and general public, and the traffic OUT, is already allowed for the partners working on site.

Asset criticality

High

GIAC Enterprises must keep the databases secure. The database that contains the fortune sayings is considered the asset with the highest value. The fortune sayings are original, adapted to today's lifestyle, and because they are specific to a customer profile, people can relate to them and get addicted ($$$). If the partners decide to close GIAC Enterprises after their two-year business pilot, they plan to sell the database, again ($$$).

The other most critical asset is the database containing customer information. Although it is at this point simply e-mail addresses and profiles (no credit card numbers). It is still of the utmost importance. GIAC enterprises will not sell nor distribute any personal information about its customers. It would be as unacceptable to leak this information accidentally or through a break-in.

Medium

The next most critical asset is the source code behind GIAC enterprises' FORTUNE® engine software. It uses a proprietary random engine along with extra verification to ensure a customer does not get the same fortune saying in a year. The code is stored on the developer workstation (on-site user) and on the Web server.

Low

All other PCs have been assigned the same asset criticality: low.

Nonetheless, all PCs must be hardened and looked after to ensure the highly important assets remain well protected.

What this criticality assessment means is that if something goes wrong (e.g. network under attack), the assets that need immediate attention are the ones with the highest importance. They should get disconnected from the network until the partner is certain it is safe to reconnect.

General Security Conduct Policy

In addition to trying to reduce bandwidth consumption, partners have agreed to comply with certain security policies.

Physical access to the databases and servers is limited to GIAC Enterprises partners. An alarm system must protect the main office site (GIAC enterprises is located in a house already protected by such a system).

All systems must be kept up-to-date, whether they are located on site or at the teleworkers sites.

The one laptop and the six partners' workstations must be equipped with anti-virus systems and personal firewalls.

No copy of the databases can be stored on computers other than the MySQL servers.

GIAC enterprises' FORTUNE® engine software is kept secure on the developer workstation (one developer) and on the internal Web server. No other copies are allowed.[3]

Budget

Fees

- 1 Monitor
- 5 new PCs
- 3 extra hard drives
- 2 KVM switches
- 1 Shomiti taping device
- 1 Commercial ADSL line (1.5Meg) Self Installation, static IP address (+ 1 additional): 55$/month
- 4 Residential ADSL lines (1Meg) Self Installation, static IP address: 45$/month
- 13 tradeshow subscription fees (mean of 600$/tradeshow)
- Travel related fees (mean of 300$/tradeshow)

Income (Subscription price: 5$/month, 10$/(3 months), 30$/(12 months))

- 200 subscribers for the 12-month package: 6000$
- 150 subscribers for the 3-month package: (1500$ or more)
- Estimated new subscriptions per tradeshow (25 for 12-month package, 8 for 3-month package, 3 for 5 month package): 13 tradeshows schedule, we'll assume that 2 may get cancelled:11x845$:9295$

Assignment 2 – Security Policy and Tutorial

Border Gateway security rules

The border gateway must be able to sustain the load of traffic that arrives at GIAC network. While in normal condition the load of traffic should be very low, the gateway must be able to handle legitimate traffic when the network is being scanned by hosts on the internet. The filtering rules are therefore stateless. This saves the border gateway from having to maintain state tables for the communications between the internet and GIAC network. The security policy is to "drop everything unless there are specific rules to allow". The filtering rules allow inbound traffic for services provided at GIAC, and

[3] A CVS server may be added at some point to keep track of changes made to the GIAC enterprises' FORTUNE® engine software. It would be located on the segment where the on-site user workstations are.

allows outbound permitted traffic. The NetFilter script also includes filtering rules to prevent "easy to detect" spoofed packet to reach the internal systems.

The border gateway is also in charge on the NAT'ing. As seen from the internet, this host and the VPN gateway are the only two visible systems on this network. All traffic arriving from the interface that connects the border gateway to the Primary firewall gets NAT'ed to the external IP address of the border gateway. The traffic from the VPN gateway remains unchanged. This simplifies the ipsec configuration.

Additionally to NAT'ing IP addresses of internal hosts, the border gateway is responsible to forward the traffic to the appropriate servers. This is done through port forwarding. For instance, a packet arriving at the border gateway with the destination address of the border gateway and destination port TCP/80 will be forwarded to the Squid proxy listening on this port.

Because the border gateway is exposed to the internet, and thus to a potentially high volume of traffic, very little logging is done here. This is to ensure that an attacker does not send a high volume of abnormal packets in the intent of overloading the border gateway (because of its logging capability), and then pass through.

Below are the actual Netfilter rules implemented on the border gateway. The rules among a given chain (e.g. INPUT, OUTPUT, FORWARD), are given priority by Netfilter based on the order in which they appear. If a rule is loose (i.e. is satisfied by many types of packet) and placed before a rule that is more restrictive, there is a danger that the traffic gets allowed before NetFilter reaches the restrictive rule. This is something to keep in mind when ordering the rules. Placing rules for most popular traffic before rules for less popular traffic is also a good practice in terms of processing performance.

Here at GIAC, we are more concern about the readability of the script than the performance. It is of our policy to ensure that any partner can understand and be able to update the rules. Therefore the rules have been grouped primarily by traffic flow. This allows for easy examination of the policy.

```
#!/bin/sh
# The script configures a Netfilter firewall for a
# border gateway with 3 interfaces.

IPTABLES="/sbin/iptables"   # Location of iptables binary
DEPMOD="/sbin/depmod"       # Location of modules dependency handler
MODPROBE="/sbin/modprobe"   # Location of modules loader

################################################
# SET KERNEL VARIABLES
################################################

# see http://ipsysctl-tutorial.frozentux.net/chunkyhtml
# for detailed information on kernel variables.  This link
```

```
# also describes cautions and issues for each variable.

# IP_FORWARDING off
# Disable IP forwarding until the default policy of DROP
# is applied to each default chain, and the GIAC rules are applied.
# This eliminates the possibility of packets getting through
# before the rules have been applied.
echo 0 > /proc/sys/net/ipv4/ip_forward

# TCP_SYNCOOKIES on
# Prevents SYN Flood attack
echo 1 > /proc/sys/net/ipv4/tcp_syncookies

# LOG_MARTIANS on
# Log impossible to resolve addresses (e.g. not in routing tables)
echo 1 > /proc/sys/net/ipv4/conf/all/log_martians

# ICMP_ECHO_IGNORE_BROADCASTS on
# Ignores ICMP messages sent to broadcast addresses. To keep from being a SMURF amplifier.
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

# RP_FILTER off
# While setting rp_filter on is generally a good idea because it prevents certain spoofing
# scenarios from happening, it can produce undesired effect if one is using special policy
# routings.  What rp_filter does is it verifies that the source address used in the packets
# correlates with the routing table. We set it off because we impose static routes in several
# GIAC multi-homed systems. The gateway should only look in its routing table for determining
# where to forward a given packet (not to validate where it came from).
# We will thighten up some of our iptables rule to validate the source of packets and the
# interface through which packet arrives.
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter

# ACCEPT_SOURCE_ROUTE off
# Do not accept packets with IP source route options enabled. These are generally malicious and
# are considered a security risk.
echo 0 > /proc/sys/net/ipv4/conf/all/accept_source_route

# ACCEPT_REDIRECTS off
# Do not accept ICMP redirect packets. Linux accepts them by default. These packets can attempt
# to redirect traffic to an attacker's system and are therefore considered a security risk.
echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects


################################################
#  LOAD MODULES
################################################

# see http://www.ecst.csuchico.edu/~dranch/LINUX/ipmasq/examples/rc/firewall-2.4
# for a description of the modules.  This link also describes how certain of these
# modules are automatically loaded given certain iptables commands are invoked.
# Loading the modules manually as we do below cleans up kernel auto-loading timing issues.

$DEPMOD -a                # Verifies that all modules have required dependencies
$MODPROBE ip_tables       # Loads the main body of the iptables module
$MODPROBE iptable_filter  # Loads the filtering modules (also loads ipt_LOG, ipt_REJECT)
$MODPROBE ip_conntrack    # Loads the stateful connection tracking
```

```
$MODPROBE ip_conntrack_ftp  # Loads the FTP tracking mechanism
$MODPROBE iptable_nat       # Loads the NAT functionality (MASQ)
$MODPROBE ip_nat_ftp        # Loads the FTP NAT (required to support PASV FTP)
$MODPROBE ipt_limit         # Allows to set a limit on packet rate (hits per sec, min, or hour)
$MODPROBE ipt_state         # Allows to catch packets with various IP and TCP flags settings
$MODPROBE ipt_unclean       # Allows to catch packets with invalid IP and TCP flags settings
$MODPROBE ipt_TTL           # Allows to change the ttl value of packets



################################################
# CLEAR OUT FIREWALL RULES
################################################
$IPTABLES -F              # flush default table (same as $IPTABLES --table filter)
$IPTABLES -X              # delete every non built-in chain
$IPTABLES -Z              # zero out counters in all chains
$IPTABLES --table nat -F    # flush nat table
$IPTABLES --table mangle -F # flush mangle table
                  # mangle is used for specialized packet alteration,
                  # GIAC uses it to change TTL values of outbound packets.



################################################
# SET DEFAULT DROP POLICY FOR BUILT-IN CHAINS
################################################

$IPTABLES --policy INPUT DROP      # traffic targeted at the firewall
$IPTABLES --policy OUTPUT DROP     # traffic leaving the firewall
$IPTABLES --policy FORWARD DROP    # traffic crossing the firewall


################################################
#GLOBAL VARIABLES
################################################

LOGLEVEL=notice

# The next 6 variables are concerned with this host (the border gateway)
EXTIF=eth2             # interface to internet (external interface)
IF_TO_FW=eth1          # interface to primary firewall
IF_TO_VPN=eth0         # interface to vpn
EXTIP=216.6.115.115    # External IP (facing internet)
IP_TO_FW=192.168.100.1  # IP facing vpn
IP_TO_VPN=216.6.115.49  # IP facing vpn

FW=192.168.100.2       # IP of firewall
VPN=216.6.115.50       # IP of vpn

SERVICE=172.16.1.0/26        # service network range
SERVICE_SQUID=172.16.1.8      # IP of squid proxy
SERVICE_MAIL=172.16.1.25      # IP of mail relay
SERVICE_DNS=172.16.1.53       # IP of external dns

INTERNAL=192.168.1.0/24       # internal network range
INTERNAL_USERS=192.168.1.64/26 # internal user range
```

```
INTERNAL_DNS=192.168.1.117     # IP of dns
INTERNAL_ADMIN=192.168.1.85    # IP of administrator's workstation

ANY=0.0.0.0/0             # internet
ISP_DNS=216.6.44.5          # IP address of our ISP DNS

TELEWORKER1=216.6.115.65   # is allowed special access (vpn)
TELEWORKER2=216.6.115.75   # is allowed special access (vpn)
TELEWORKER3=216.6.115.85   # is allowed special access (vpn)
TELEWORKER4=216.6.115.95   # is allowed special access (vpn)

PORT_HTTP_PROXY=80      # HTTP port expected at squid proxy (configurable)
PORT_HTTPS_PROXY=443    # HTTPS port expected at squid proxy (configurable)
PORT_MAIL_RELAY=25      # MAIL port expected at mail relay (configurable)
PORT_DNS_SERVER=53      # DNS port expected at DNS server (configurable)


#######################################################
#  STATIC ROUTES
#######################################################

# to reach service net
route add -net 172.16.1.0 netmask 255.255.255.192 gw $FW dev $IF_TO_FW

# to reach internal net
route add -net 192.168.1.0 netmask 255.255.255.0 gw $FW dev $IF_TO_FW


#######################################################
# DROP Filter Rules based on source/dest IP addr
# Apply this before allowing traffic
#######################################################

# Block packets with Reserved/Private Addresses (src or destination)
# Data pulled from http://www.iana.org/assignments/ipv4-address-space.
# This creates a rule for each of these defined IANA_RESERVED addresses and
# IANA_Multicast addresses by looping through each of them.

IANA_RESERVED="0 1 2 5 7 23 27 31 36 37 39 41 42 58 59 70 71 72 73 74 75 76 77 \
78 79 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 \
105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 \
124 125 126 127 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 \
189 190 197 223 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255"
for NET in $IANA_RESERVED; do
    # drop spoofing
    $IPTABLES -t nat --append PREROUTING -i $EXTIF -s $NET.0.0.0/8 -j DROP
    # drop recon
    $IPTABLES -t nat --append PREROUTING -i $EXTIF -d $NET.0.0.0/8 -j DROP
done

IANA_MULTICAST="224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239"
for NET in $IANA_MULTICAST; do
    # drop spoofing
    $IPTABLES -t nat --append PREROUTING -i $EXTIF -s $NET.0.0.0/8 -j DROP
    # drop recon
    $IPTABLES -t nat --append PREROUTING -i $EXTIF -d $NET.0.0.0/8 -j DROP
```

```
done

# IANA PRIVATE IPv4 address space via RFC 1918.
$IPTABLES -t nat --append PREROUTING -i $EXTIF -s 10.0.0.0/8 -j DROP
$IPTABLES -t nat --append PREROUTING -i $EXTIF -d 10.0.0.0/8 -j DROP
$IPTABLES -t nat --append PREROUTING -i $EXTIF -s 172.16/12 -j DROP
$IPTABLES -t nat --append PREROUTING -i $EXTIF -d 172.16/12 -j DROP
$IPTABLES -t nat --append PREROUTING -i $EXTIF -s 192.168.0.0/16 -j DROP
$IPTABLES -t nat --append PREROUTING -i $EXTIF -d 192.168.0.0/16 -j DROP
$IPTABLES -t nat --append PREROUTING -i $EXTIF -s 192.88.99.0/24 -j DROP
$IPTABLES -t nat --append PREROUTING -i $EXTIF -d 192.88.99.0/24 -j DROP

# Spoofed IP addresses that belong to GIAC on-site address space
# The following command creates a chain for spoofed TCP packets.
$IPTABLES -t nat --append PREROUTING -i $EXTIF -s $EXTIP -j DROP
$IPTABLES -t nat --append PREROUTING -i $EXTIF -s $IP_TO_VPN -j DROP
$IPTABLES -t nat --append PREROUTING -i $EXTIF -s $VPN -j DROP



#######################################################
# PORT FORWARDING AND NAT SUPPORT
#######################################################

#------------------------------------------------------
# NAT SUPPORT FOR OUTBOUND COMMUNICATIONS
#------------------------------------------------------

# VPN address will not be NAT'ed, all other IP addresses will.
$IPTABLES --table nat --append POSTROUTING -o $EXTIF -d \! $VPN -j MASQUERADE


# Do not let out packets with our private IP addresses scheme. At this
# point, such packets should have been pre-filtered or pre-NAT'ed.
# If this is about to happen, something went wrong... LOG and DROP!
$IPTABLES -t nat --append POSTROUTING -o $EXTIF -s $SERVICE -j LOG \
        --log-level $LOGLEVEL --log-prefix "private scheme leak: "
$IPTABLES -t nat --append POSTROUTING -o $EXTIF -s $INTERNAL -j LOG \
        --log-level $LOGLEVEL --log-prefix "private scheme leak: "
$IPTABLES -t nat --append POSTROUTING -o $EXTIF -s $IP_TO_FW -j LOG \
        --log-level $LOGLEVEL --log-prefix "private scheme leak: "
$IPTABLES -t nat --append POSTROUTING -o $EXTIF -s $FW -j LOG \
        --log-level $LOGLEVEL --log-prefix "private scheme leak: "
$IPTABLES -t nat --append POSTROUTING -o $EXTIF -s $SERVICE -j DROP
$IPTABLES -t nat --append POSTROUTING -o $EXTIF -s $INTERNAL -j DROP
$IPTABLES -t nat --append POSTROUTING -o $EXTIF -s $IP_TO_FW -j DROP
$IPTABLES -t nat --append POSTROUTING -o $EXTIF -s $FW -j DROP


#------------------------------------------------------
# OVERWRITE TTL OF OUTBOUND PACKETS
#------------------------------------------------------
# Change the ttl value of outbound packets so this network looks
# flat from outside.  This should slow down reconnaissance and
# mapping attempts against GIAC's site.
```

```
# Packets emitted by the VPN gateway are left alone (no mangling for them).
$IPTABLES -t mangle --append POSTROUTING -o $EXTIF -d \! $VPN -j TTL --ttl-set 64



#------------------------------------------------
# PORT FORWARDING SUPPORT FOR INBOUND COMMUNICATIONS
#------------------------------------------------

# Forward incoming HTTP packets to the squid proxy (to the port squid is listening on)
$IPTABLES -t nat --append PREROUTING -p tcp -d $EXTIP --dport 80 -j DNAT \
        --to-destination $SERVICE_SQUID:$PORT_HTTP_PROXY
# Make responses on the internal network go through the gateway
$IPTABLES -t nat --append POSTROUTING -p tcp -d $SERVICE_SQUID \
        --dport $PORT_HTTP_PROXY -j SNAT  --to-source $IP_TO_FW

# Forward incoming HTTPS packets to the squid proxy (to the port squid is listening on)
$IPTABLES -t nat --append PREROUTING -p tcp -d $EXTIP --dport 443 -j DNAT \
        --to-destination $SERVICE_SQUID:$PORT_HTTPS_PROXY
# Make responses on the internal network go through the gateway
$IPTABLES -t nat --append POSTROUTING -p tcp -d $SERVICE_SQUID \
         --dport $PORT_HTTPS_PROXY -j SNAT --to-source $IP_TO_FW

# Forward incoming SMTP packets to the mail relay (to the port the relay is listening on)
$IPTABLES -t nat --append PREROUTING  -p tcp -d $EXTIP  --dport 25  -j DNAT \
        --to-destination $SERVICE_MAIL:$PORT_MAIL_RELAY
# Make responses on the internal network go through the gateway
$IPTABLES -t nat --append POSTROUTING -p tcp -d $SERVICE_MAIL \
       --dport $PORT_MAIL_RELAY -j SNAT  --to-source $IP_TO_FW

# Forward incoming UDP DNS packets to the external DNS server (to the port it is listening on)
$IPTABLES -t nat --append PREROUTING  -p udp -d $EXTIP  --dport 53  -j DNAT \
        --to-destination $SERVICE_DNS:$PORT_DNS_SERVER
# Make responses on the internal network go through the gateway
$IPTABLES -t nat --append POSTROUTING -p udp -d $SERVICE_DNS \
        --dport $PORT_DNS_SERVER \ -j SNAT  --to-source $IP_TO_FW

# Forward incoming TCP DNS packets to the external DNS server (to the port it is listening on).
$IPTABLES -t nat --append PREROUTING  -p tcp -d $EXTIP  --dport 53  -j DNAT \
        --to-destination $SERVICE_DNS:$PORT_DNS_SERVER
# Make responses on the internal network go through the gateway
$IPTABLES -t nat --append POSTROUTING -p tcp -d $SERVICE_DNS \
         --dport $PORT_DNS_SERVER -j SNAT  --to-source $IP_TO_FW



#######################################################
# INBOUND CONNECTIONS TO GIAC SERVERS (service network)
#######################################################

# Although the traffic is expected to originate from the internet (i.e. inbound traffic)
# We must explicitly allow outbound traffic as well since our rules are stateless.

# Allow inbound/outbound HTTP traffic
$IPTABLES --append FORWARD -p tcp -i $EXTIF -d $SERVICE_SQUID \
```

```
        --dport $PORT_HTTP_PROXY -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW -s $SERVICE_SQUID \
         --sport $PORT_HTTP_PROXY -j ACCEPT

# Allow inbound/outbound HTTPS traffic
$IPTABLES --append FORWARD -p tcp -i $EXTIF -d $SERVICE_SQUID \
        --dport $PORT_HTTPS_PROXY -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW -s $SERVICE_SQUID \
        --sport $PORT_HTTPS_PROXY -j ACCEPT

# Allow inbound/outbound SMTP traffic
$IPTABLES --append FORWARD -p tcp -i $EXTIF -d $SERVICE_MAIL \
        --dport $PORT_MAIL_RELAY -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW -s $SERVICE_MAIL \
         --sport $PORT_MAIL_RELAY -j ACCEPT

# Allow inbound/outbound UDP DNS traffic
# Our external DNS contains records for GIAC's routable IP addresses. The records are of small
# length and therefore only UDP traffic is expected for answering DNS queries.
$IPTABLES --append FORWARD -p udp -i $EXTIF -d $SERVICE_DNS \
       --dport $PORT_DNS_SERVER -j ACCEPT
$IPTABLES --append FORWARD -p udp -i $IF_TO_FW -s $SERVICE_DNS \
       --sport $PORT_DNS_SERVER -j ACCEPT

# Allow inbound/outbound TCP DNS traffic
# DNS mechanism primarly uses UDP as a transport layer.  It falls back on TCP for transfering
# large DNS records (which we do not have).
# The TCP DNS rule appears after the the UDP rule since it is unlikely to hapened... So unlikely that
# we will log such traffic. The only reason for allowing it in is to allow our ISP's DNS server
# to do zone transfers.
$IPTABLES --append FORWARD -p tcp -i $EXTIF -d $SERVICE_DNS \
       --dport $PORT_DNS_SERVER -j LOG \
       --log-level $LOGLEVEL --log-prefix "TCP DNS inbound: "
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW -s $SERVICE_DNS \
        --sport $PORT_DNS_SERVER -j LOG \
        --log-level $LOGLEVEL --log-prefix "TCP DNS outbound: "
$IPTABLES --append FORWARD -p tcp -i $EXTIF -s $ISP_DNS -d $SERVICE_DNS \
       --dport $PORT_DNS_SERVER -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW -d $ISP_DNS -s $SERVICE_DNS \
       --sport $PORT_DNS_SERVER -j ACCEPT


# Reject IDENT packets with tcp-reset
# This is meant to quickly inform the external end point that IDENT is not supported.
# IDENT can be used on a server outside this site which tries to identify the owner
# of the process before it allows us to connect to the service we are requesting.
# If the client does not support IDENT (as in our case), the server won't receive
# IDENT response and will eventually figure out that IDENT is not supported by the client.
# The server generally, after a delay, allows the connection.
# By sending a RESET explicitly, we may allow the connection to take place faster.
# If we do not send explicitly a RESET, the communication will generally the external system will
retransmit a couple
$IPTABLES --append INPUT -p tcp --dport 113 -j REJECT --reject-with tcp-reset
$IPTABLES --append OUTPUT -p tcp --sport 113 -j ACCEPT  # required because of our DROP policy

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

```
# LOCALHOST (127.0.0.1) TRAFFIC
#########################################################################

# Log and allow localhost traffic entering and leaving lo interface
# We will give priority to this host before letting outbound user traffic through.
# Note: We will monitor for a while what traffic this host needs to send to himself.
#       If messages are rare, the iptables rules will be moved further down. Although
#       this is not a big deal in terms of firewall performance, it is good to keep
#       in mind that NetFilter goes through the rules from the top down until it finds
#       a match.  Starting with most popular rules can save some processing.
#       Use the following command to find loopback traffic in the logs:
#       grep "Local-" /var/log/messages

#$IPTABLES --append INPUT -i lo -j LOG --log-level $LOGLEVEL --log-prefix "Local-in: "
#$IPTABLES --append OUTPUT -o lo -j LOG --log-level $LOGLEVEL --log-prefix "Local-out: "
$IPTABLES --append INPUT -i lo -j ACCEPT
$IPTABLES --append OUTPUT -o lo -j ACCEPT



#########################################################################
# OUTBOUND USER TRAFFIC (local users and teleworkers already tuneled in)
#########################################################################

#-----------------------------------------------
# ALLOW OUTBOUND DNS TRAFFIC FROM INTERNAL DNS SERVER
#-----------------------------------------------

# udp dns rules appear first because they are likely to hit first (i.e. more frequent than tcp)

# ALLOW (after logging) outbound/inbound udp traffic from internal DNS
# The traffic of the internal DNS server is already logged on the system itself because
# the system is considered insecure.  It is redundant to log it here. However if the DNS
# gets compromised and the logs disappear from it,  there may be some trace over here.
$IPTABLES --append FORWARD -p udp -s $INTERNAL_DNS --sport 53 -d $ANY --dport 53 -j LOG \
        --log-level $LOGLEVEL --log-prefix "INTERNAL DNS OUT (udp): "
$IPTABLES --append FORWARD -p udp -d $INTERNAL_DNS --dport 53 -s $ANY --sport 53 -j LOG \
        --log-level $LOGLEVEL --log-prefix "INTERNAL DNS IN (udp): "
$IPTABLES --append FORWARD -p udp -s $INTERNAL_DNS --sport 53 -d $ANY --dport 53 \
         -j ACCEPT
$IPTABLES --append FORWARD -p udp -d $INTERNAL_DNS --dport 53 -s $ANY --sport 53 \
          -j ACCEPT

# ALLOW (after logging) outbound/inbound tcp traffic from internal DNS
$IPTABLES --append FORWARD -p tcp -s $INTERNAL_DNS --sport 53 -d $ANY --dport 53 -j LOG \
        --log-level $LOGLEVEL --log-prefix "INTERNAL DNS OUT (tcp): "
$IPTABLES --append FORWARD -p tcp -d $INTERNAL_DNS --dport 53 -s $ANY --sport 53 -j LOG \
        --log-level $LOGLEVEL --log-prefix "INTERNAL DNS IN (tcp): "
$IPTABLES --append FORWARD -p tcp -s $INTERNAL_DNS --sport 53 -d $ANY --dport 53 \
         -j ACCEPT
$IPTABLES --append FORWARD -p tcp -d $INTERNAL_DNS --dport 53 -s $ANY --sport 53 \
         -j ACCEPT



#-----------------------------------------------
```

```
# ALLOW OUTBOUND USER TRAFFIC
#-----------------------------------------------------
# There is no particular reason for prioritizing certain user traffic type over others.
# For clarity and readability, they are listed "by server". We tighten up the rules
# by specifying the interface from which the traffic is expected to arrive.


# Allow outbound/inbound HTTP and HTTPS traffic from SQUID PROXY
# (and from the interface that connects the primary firewall to the border gateway)
# Note: the HTTPS traffic is logged (in part) at the primary firewall.
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW -s $SERVICE_SQUID -m multiport \
        --dport 80,443 -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $EXTIF    -d $SERVICE_SQUID -m multiport \
        --sport 80,443 -j ACCEPT


# !!!!!!!!!!!!!!!!!!!!!!!!        FTP      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
# FTP traffic is a bit tricky bcause it has concurrent related session (ftp-command, ftp-data).
# We will use stateful rules to keep track of FTP.  This should not harm the gateway
# since only outbound FTP is allowed, and since GIAC has few users.
# We tighten up the rule by describing exactly what to expect in terms of connections.
# The rules will be written in a different form at the primary firewall. While we expect
# that the rules below and those at the primary firewall will allow the same type of connections,
# we prefer to write them in a different manner.  If one method should fail and allow unexpected
# traffic, the other may block it.

# Port 21 (ftp-command) from the SQUID PROXY
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW -s $SERVICE_SQUID --dport 21 -m state \
        --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $EXTIF --sport 21 -m state --state ESTABLISHED \
         -j ACCEPT

# Port 21 (ftp-command) from internal users
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW -s $INTERNAL_USERS --dport 21 -m state \
        --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $EXTIF --sport 21 -m state --state ESTABLISHED \
         -j ACCEPT

#Allow ftp-data session in ACTIVE mode (ftp servers connects to us from its tcp/20 port)
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW --sport 20 -m state \
         --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $EXTIF --dport 20 -m state --state ESTABLISHED \
          -j ACCEPT

#Allow ftp-data session in PASSIVE FTP (we connect to the ftp server to the specified high port)
$IPTABLES --append FORWARD -p tcp --sport 1024:65535 --dport 1024:65535 -m state \
        --state ESTABLISHED,RELATED -j ACCEPT


#!!!!!!!!!!!!!!!!!!!!!!!!!!       SMTP ALREADY ALLOWED          !!!!!!!!!!!!!!!!!!!!!
# allow outbound SMTP traffic from MAIL RELAY
# (and from the interface that connects the primary firewall to the border gateway)
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW -s $SERVICE_MAIL --dport 25 -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $EXTIF    -d $SERVICE_MAIL --sport 25 -j ACCEPT

# Allow outbound SSH traffic from INTERNAL USERS (used to download patch for OpenBSD)
```

```
# (and from the interface that connects the primary firewall to the border gateway)
# Note: Teleworkers do not use this kind of traffic (implicitly drop for them)
# Note: this traffic is logged (in part) at the primary firewall.
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW -s $INTERNAL_USERS --dport 22 -j LOG \
        --log-level $LOGLEVEL --log-prefix "OUTBOUND SSH: "
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW -s $INTERNAL_USERS --dport 22 -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $EXTIF    -d $INTERNAL_USERS --sport 22 -j ACCEPT

# allow outbound RTSP traffic from IPs in range of INTERNAL USERS
# (and from the interface that connects the primary firewall to the border gateway)
# Note: Teleworkers do not use this kind of traffic (implicitly drop for them)
$IPTABLES --append FORWARD -p tcp -i $IF_TO_FW -s $INTERNAL_USERS --dport 554 -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $EXTIF    -d $INTERNAL_USERS --sport 554 -j ACCEPT

#------------------------------------------------------
# INBOUND COMMUNICATIONS USING VPN ACCESS
#------------------------------------------------------

# log InternetKey Exchange (IKE)
$IPTABLES --append FORWARD -p udp -i $EXTIF --sport 500 --dport 500 -j LOG \
        --log-level $LOGLEVEL --log-prefix "IKE-in: "
$IPTABLES --append FORWARD -p udp -i $IF_TO_VPN --sport 500 --dport 500 -j LOG \
        --log-level $LOGLEVEL --log-prefix "IKE-out: "

# log Encapsulated Security Payload (ESP)
$IPTABLES --append FORWARD -p 50 -i $EXTIF -j LOG --log-level $LOGLEVEL \
        --log-prefix "ESP-in: "
$IPTABLES --append FORWARD -p 50 -i $IF_TO_VPN -j LOG --log-level $LOGLEVEL \
        --log-prefix "ESP-out: "

# Allow IKE for teleworkers to VPN gateway
$IPTABLES --append FORWARD -p udp -i $EXTIF     -s $TELEWORKER1 --sport 500 -d $VPN \
         --dport 500 -j ACCEPT
$IPTABLES --append FORWARD -p udp -i $IF_TO_VPN -d $TELEWORKER1 --dport 500 \
         -s $VPN --sport 500 -j ACCEPT
$IPTABLES --append FORWARD -p udp -i $EXTIF     -s $TELEWORKER2 --sport 500 -d $VPN \
         --dport 500 -j ACCEPT
$IPTABLES --append FORWARD -p udp -i $IF_TO_VPN -d $TELEWORKER2 --dport 500 -s $VPN \
         --sport 500 -j ACCEPT
$IPTABLES --append FORWARD -p udp -i $EXTIF     -s $TELEWORKER3 --sport 500 -d $VPN \
         --dport 500 -j ACCEPT
$IPTABLES --append FORWARD -p udp -i $IF_TO_VPN -d $TELEWORKER3 --dport 500 -s $VPN \
         --sport 500 -j ACCEPT
$IPTABLES --append FORWARD -p udp -i $EXTIF     -s $TELEWORKER4 --sport 500 -d $VPN \
         --dport 500 -j ACCEPT
$IPTABLES --append FORWARD -p udp -i $IF_TO_VPN -d $TELEWORKER4 --dport 500 -s $VPN \
         --sport 500 -j ACCEPT

# Allow ESP for teleworkers to VPN gateway
$IPTABLES --append FORWARD -p 50 -i $EXTIF     -s $TELEWORKER1 -d $VPN -j ACCEPT
$IPTABLES --append FORWARD -p 50 -i $IF_TO_VPN -d $TELEWORKER1 -s $VPN -j ACCEPT
$IPTABLES --append FORWARD -p 50 -i $EXTIF     -s $TELEWORKER2 -d $VPN -j ACCEPT
$IPTABLES --append FORWARD -p 50 -i $IF_TO_VPN -d $TELEWORKER2 -s $VPN -j ACCEPT
$IPTABLES --append FORWARD -p 50 -i $EXTIF     -s $TELEWORKER3 -d $VPN -j ACCEPT
$IPTABLES --append FORWARD -p 50 -i $IF_TO_VPN -d $TELEWORKER3 -s $VPN -j ACCEPT
$IPTABLES --append FORWARD -p 50 -i $EXTIF     -s $TELEWORKER4 -d $VPN -j ACCEPT
```

```
$IPTABLES --append FORWARD -p 50 -i $IF_TO_VPN -d $TELEWORKER4 -s $VPN -j ACCEPT



$IPTABLES -N ICMP_TRAFFIC
# We accept in (from interface $EXTIF) ICMP Unreachable messages and
# time exceeded messages. These packets are often legitimate, and when they are,
# they must reach the other end point for the sake of the communication.
# This gateway cannot tell whether they are legitimate or not since it is
# mainly stateless. So we allow those explicitly.
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 3 -i $EXTIF -j ACCEPT
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 11 -j ACCEPT
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 8 -j ACCEPT
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 0 -j ACCEPT

# Following ICMP types will be logged by type.  They are rare, or can be malicious.
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 4 -j LOG --log-level $LOGLEVEL \
        --log-prefix "ICMP-Source quench: "
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 5 -j LOG --log-level $LOGLEVEL \
        --log-prefix "ICMP-Redirect: "
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 9 -j LOG --log-level $LOGLEVEL \
        --log-prefix "ICMP-Router adv: "
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 10 -j LOG --log-level $LOGLEVEL \
        --log-prefix "ICMP-Router select: "
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 30 -j LOG --log-level $LOGLEVEL \
         --log-prefix "ICMP-Traceroute: "

#DROP all ICMP traffic unless explicitly ACCEPTED above
$IPTABLES --append ICMP_TRAFFIC -j DROP

# Append ICMP rules to chains
$IPTABLES --append INPUT -p icmp -j ICMP_TRAFFIC
$IPTABLES --append FORWARD -p icmp -j ICMP_TRAFFIC
$IPTABLES --append OUTPUT -p icmp -j ICMP_TRAFFIC



#####################################################
# ENABLE IP FORWARDING
#####################################################
# now that the rules have been applied, allow IP forwarding
echo "1" > /proc/sys/net/ipv4/ip_forward

# End of script
```

Primary Firewall security rules

The Primary Firewall is also running NetFilter, but with stateful rules. When a new connection is allowed and established, Netfilter creates an entry in a state table. The packets arriving at the primary firewall and matching a connection in the table of accepted and established connection will also be allowed through. NetFilter also has a mechanism to match related traffic with established connections. Related traffic can be ICMP error messages for instance. This allows for example to place a rule at the end of the script to DROP ICMP error messages. The only ICMP error messages allowed in would therefore be those that are related to established connections.

As with the border gateway, the default policy is to DROP traffic unless it is specifically permitted.

Below is the actual script running on the primary firewall.

```
#!/bin/sh
# The script configures a Netfilter firewall for a
# primary firewall with 4 interfaces.

IPTABLES="/sbin/iptables"   # Location of iptables binary
DEPMOD="/sbin/depmod"       # Location of kernel modules dependency handler
MODPROBE="/sbin/modprobe"   # Location of kernel modules loader

################################################
# SET KERNEL VARIABLES
################################################

# kernel variables have the same values as those of the border gateway
# They have been removed from this printed version to save paper...
.
.
.

################################################
#  LOAD MODULES
################################################

# see http://www.ecst.csuchico.edu/~dranch/LINUX/ipmasq/examples/rc/firewall-2.4
# for a description of the modules.  This link also describes how certain of these
# modules are automatically loaded given certain iptables commands are invoked.
# Loading the modules manually as we do below cleans up kernel auto-loading timing issues.

$DEPMOD -a                # Verifies that all modules have required dependencies
$MODPROBE ip_tables       # Loads the main body of the iptables module
$MODPROBE iptable_filter  # Loads the filtering modules (also loads ipt_LOG, ipt_REJECT)
$MODPROBE ip_conntrack    # Loads the stateful connection tracking
$MODPROBE ip_conntrack_ftp  # Loads the FTP tracking mechanism
$MODPROBE iptable_nat     # Loads the NAT functionality (MASQ)
$MODPROBE ip_nat_ftp      # Loads the FTP NAT (required to support PASV FTP)
$MODPROBE ipt_limit       # Allows to set a limit on packet rate (hits per sec/min/hr)
```

```
$MODPROBE ipt_state        # Allows to catch packets with various IP and TCP flags settings
$MODPROBE ipt_unclean      # Allows to catch packets with invalid IP and TCP flags settings



################################################
# CLEAR OUT FIREWALL RULES
################################################
$IPTABLES -F              # flush default table (same as $IPTABLES --table filter)
$IPTABLES -X              # delete every non built-in chain
$IPTABLES -Z              # zero out counters in all chains
$IPTABLES --table nat -F    # flush nat table
$IPTABLES --table mangle -F # flush mangle table
                  # mangle is used for specialized packet alteration,
                  # it is not used in the rules at this time.



################################################
# SET DEFAULT DROP POLICY FOR BUILT-IN CHAINS
################################################

$IPTABLES --policy INPUT DROP      # traffic targeted at the firewall
$IPTABLES --policy OUTPUT DROP     # traffic leaving the firewall
$IPTABLES --policy FORWARD DROP    # traffic crossing the firewall


################################################
#GLOBAL VARIABLES
################################################

LOGLEVEL=notice

# The next 8 variables are concerned with this host (the primary firewall)
EXTIF=eth1            # interface to border gateway (external interface)
IF_TO_VPN=eth2        # interface to vpn gateway
IF_TO_SERVICE=eth3    # interface to service network
IF_TO_INTERNAL=eth0   # interface to internal network
EXTIP=192.168.100.2       # External IP (facing border gateway)
IP_TO_VPN=10.10.1.1       # IP facing vpn gateway
IP_TO_SERVICE=172.16.1.1  # IP facing service network
IP_TO_SERVICE=192.168.1.1  # IP facing service network

GW=192.168.100.1       # IP of gateway (directly connected to this firewall)
VPN=10.10.1.2          # IP of vpn (directly connected to this firewall)
INTERNAL_FW=192.168.1.2 # IP of internal firewall (directly connected to the firewall)

SERVICE=172.16.1.0/26        # service network range
SERVICE_SQUID=172.16.1.8     # IP of squid proxy
SERVICE_MAIL=172.16.1.25     # IP of mail relay
SERVICE_DNS=172.16.1.53      # IP of external dns

INTERNAL=192.168.1.0/24       # internal network range
INTERNAL_USERS=192.168.1.64/26 # internal user range
INTERNAL_DNS=192.168.1.117    # IP of dns
```

```
INTERNAL_ADMIN=192.168.1.85    # IP of administrator's workstation
INTERNAL_WEB=192.168.1.8       # IP of web server
INTERNAL_MAIL=192.168.1.25     # IP of mail server

ANY=0.0.0.0/0            # internet

TELEWORKER1=216.6.115.65   # is allowed special access (vpn)
TELEWORKER2=216.6.115.75   # is allowed special access (vpn)
TELEWORKER3=216.6.115.85   # is allowed special access (vpn)
TELEWORKER4=216.6.115.95   # is allowed special access (vpn)

PORT_OUTBOUND_PROXY=3128  # port expected at squid for outbound user traffic (configurable)
PORT_HTTP_PROXY=80        # HTTP port expected at squid proxy (configurable)
PORT_HTTPS_PROXY=443      # HTTPS port expected at squid proxy (configurable)
PORT_MAIL_RELAY=25        # MAIL port expected at mail relay (configurable)
PORT_DNS_SERVERS=53       # DNS port expected at DNS servers (configurable)

PORT_HTTP_SERVER=80     # HTTP port expected at internal web server (configurable)
PORT_HTTPS_SERVER=443   # HTTPS port expected at internal web server (configurable)
PORT_SMTP_SERVER=25     # SMTP port expected at internal mail server (configurable)



#########################################################
#  STATIC ROUTES
#########################################################

# To reach VPN clients (teleworkers)
# This is required to ensure that the packets with source IP address
# of our teleworkers are forwarded to the VPN gateway.
# Add a new route for each new teleworkers.
route add  $TELEWORKER1 gw $VPN dev $IF_TO_VPN
route add  $TELEWORKER2 gw $VPN dev $IF_TO_VPN
route add  $TELEWORKER3 gw $VPN dev $IF_TO_VPN
route add  $TELEWORKER4 gw $VPN dev $IF_TO_VPN


# To reach VPN encrypted net (not directly attached to the firewall)
# Note: can be commented out, since in normal operation (i.e. other than testing),
# the firewall does not have to know how to reach this zone.
route add -net  216.6.115.48 netmask 255.255.255.252 gw 10.10.1.2 dev eth2


#########################################################
# START FILTERING PACKETS!
#########################################################
# STATEFUL PACKET FILTERING
#
# This firewall does stateful packet filtering
#
# For each type of traffic, we break the rule into two part to tighten up the security:
# (1) We tell NetFilter to ACCEPT certain NEW connections.
#     We tighten up the rule by specifying interfaces (in and out), destination IP
#     and destination port.
#
# (2) Because we specify the direction in rule (1), we need to create another rule to allow
```

```
#     ESTABLISHED connections in the other direction.  Because Netfilter maintains src/dst
#     information in the state table, this rule can be loose in terms of direction and ports.
#     It does not even have to specify the protocol.  This rule can thus be used to allow
#     RELATED traffic (e.g. ICMP error messages).
# It would be redundant to place rule(2) for each type of traffic we allow.  It will therefore
# be written once, and before any other rule.  This means that ESTABLISHED connections are
# given priority over NEW ones. We trust our two internal users to be reasonable and refrain
# from downloading large files during busy hours.  GIAC's bandwidth is limited.


# Allow ESTABLISHED connections and related traffic (see comment above concerning "rule (2)").
$IPTABLES --append FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT


###################################
# LOG AND DROP A FEW KNOWN SCANS
###################################
# Note: --tcp-flags takes two arguments, the first argument is the flag(s) to be examined
#       and the second argument is the flag(s) that must be set.
# Note: we also use the limit module to limit the logging rate to 3hits/hour

# ACK SCAN (typically from a packet crafter, e.g. hping)
$IPTABLES --append FORWARD -p tcp --tcp-flags ALL ACK -m limit --limit 3/hour \
        -j LOG --log-level $LOGLEVEL --log-prefix "ack scan: "
$IPTABLES --append FORWARD -p tcp --tcp-flags ALL ACK -j DROP

# NULL SCAN (typically from a packet crafter, e.g. hping)
$IPTABLES --append FORWARD -p tcp --tcp-flags ALL NONE -m limit --limit 3/hour \
        -j LOG --log-level $LOGLEVEL --log-prefix "null scan: "
$IPTABLES --append FORWARD -p tcp --tcp-flags ALL NONE -j DROP

# FIN SCAN
$IPTABLES --append FORWARD -p tcp --tcp-flags ALL FIN -m limit --limit 3/hour \
        -j LOG --log-level $LOGLEVEL --log-prefix "fin scan: "
$IPTABLES --append FORWARD -p tcp --tcp-flags ALL FIN -j DROP

# XMAS TREE SCAN
$IPTABLES --append FORWARD -p tcp --tcp-flags ALL FIN,URG,PSH -m limit --limit 3/hour \
        -j LOG --log-level $LOGLEVEL --log-prefix "fin scan: "
$IPTABLES --append FORWARD -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP

# FIN/SYN SCAN
$IPTABLES --append FORWARD -p tcp --tcp-flags SYN,FIN SYN,FIN -m limit --limit 3/hour \
        -j LOG --log-level $LOGLEVEL --log-prefix "fin scan: "
$IPTABLES --append FORWARD -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP



#####################################################
# INBOUND CONNECTIONS TO GIAC SERVERS (service network)
#####################################################
# We will allow connections initiated from the internet to systems on the service network.
# Because established connections are allowed at this point, we will specify what NEW
# inbound connections are ACCEPTED.
```

```
# Allow inbound HTTP connections to the SQUID proxy, and on the port the proxy expects.
$IPTABLES --append FORWARD -p tcp -i $EXTIF -o $IF_TO_SERVICE -d $SERVICE_SQUID \
        --dport $PORT_HTTP_PROXY -m state --state NEW -j ACCEPT

# Allow HTTP connections from the SQUID proxy to the internal WEB server, and on the port
# the WEB servers expects.
$IPTABLES --append FORWARD -p tcp -i $IF_TO_SERVICE -o $IF_TO_INTERNAL \
         -s $SERVICE_SQUID -d $INTERNAL_WEB \
        --dport $PORT_HTTP_SERVER -m state --state NEW -j ACCEPT


# Allow inbound HTTPS connections to the SQUID proxy, and on the port the proxy expects.
$IPTABLES --append FORWARD -p tcp -i $EXTIF -o $IF_TO_SERVICE -d $SERVICE_SQUID \
        --dport $PORT_HTTPS_PROXY -m state --state NEW -j ACCEPT

# Allow HTTPS connections from the SQUID proxy to the internal WEB server, and on the port
# the WEB servers expects.
$IPTABLES --append FORWARD -p tcp -i $IF_TO_SERVICE -o $IF_TO_INTERNAL \
          -s $SERVICE_SQUID  -d $INTERNAL_WEB \
          --dport $PORT_HTTPS_SERVER -m state --state NEW -j ACCEPT


# Allow inbound SMTP connections to the MAIL relay, and on the port the relay expects.
$IPTABLES --append FORWARD -p tcp -i $EXTIF -o $IF_TO_SERVICE -d $SERVICE_MAIL \
        --dport $PORT_MAIL_RELAY -m state --state NEW -j ACCEPT

# Allow SMTP connections from the MAIL relay to the internal mail server, and on the port
# the mail server expects. This will allow the relay to forward e-mails the clients have sent.
$IPTABLES --append FORWARD -p tcp -i $IF_TO_SERVICE -o $IF_TO_INTERNAL  \
         -s $SERVICE_MAIL -d $INTERNAL_MAIL \
         --dport $PORT_SMTP_SERVER -m state --state NEW -j ACCEPT

# Allow DNS queries from the Mail relay and Squid proxy to the INTERNAL DNS server
# These two systems may need to resolve IP addresses to forward traffic to the internet
$IPTABLES --append FORWARD -p udp -i $IF_TO_SERVICE -o $IF_TO_INTERNAL  \
          -s $SERVICE_MAIL -d $INTERNAL_MAIL \
         --dport $PORT_DNS_SERVERS -m state --state NEW -j ACCEPT
$IPTABLES --append FORWARD -p udp -i $IF_TO_SERVICE -o $IF_TO_INTERNAL  \
          -s $SERVICE_SQUID -d $INTERNAL_MAIL \
         --dport $PORT_DNS_SERVERS -m state --state NEW -j ACCEPT


# Allow inbound UDP DNS traffic to the EXTERNAL DNS server
# Even though UDP is not a stateful protocol, NetFilter can keep track of UDP conversations
# by recording source and destination of packets and using timeouts.
$IPTABLES --append FORWARD -p udp -i $EXTIF -o $IF_TO_SERVICE -d $SERVICE_DNS \
        --dport $PORT_DNS_SERVERS -m state --state NEW -j ACCEPT

# Allow inbound TCP DNS traffic from our ISP DNS server to the external DNS server
# (to allow zone transfer)
# Note: When the packet reaches this firewall, the source is no longer the IP address of
#       the ISP_DNS, it is the IP of the internal interface of the gateway.
# The gateway is in charge for allowing in TCP DNS request provided that they
# come from the ISP DNS.  Here, to let the packet through, do not filter based on
# source IP address.
$IPTABLES --append FORWARD -p tcp -i $EXTIF -o $IF_TO_SERVICE -d $SERVICE_DNS \
```

```
        --dport $PORT_DNS_SERVERS -m state --state NEW -j ACCEPT


#########################################################################
# LOCALHOST (127.0.0.1) TRAFFIC
#########################################################################

# Log and allow localhost traffic entering and leaving lo interface
# We will give priority to this host before letting outbound user traffic through.
# Note: We will monitor for a while what traffic this host needs to send to himself.
#       If messages are rare, the iptables rules will be moved further down. Although
#       this is not a big deal in terms of firewall performance, it is good to keep
#       in mind that NetFilter goes through the rules from top to down until it finds
#       a match.  Starting with most popular rules can save some processing.
#       Use the following command to find loopback traffic in the logs:
#       grep "Local-" /var/log/messages

#$IPTABLES --append INPUT -i lo -j LOG --log-level $LOGLEVEL --log-prefix "Local-in: "
#$IPTABLES --append OUTPUT -o lo -j LOG --log-level $LOGLEVEL --log-prefix "Local-out: "
$IPTABLES --append INPUT -i lo -j ACCEPT
$IPTABLES --append OUTPUT -o lo -j ACCEPT



#########################################################################
# OUTBOUND USER TRAFFIC
#########################################################################


#-----------------------------------------------------
# ALLOW OUTBOUND DNS TRAFFIC FROM INTERNAL DNS SERVER
#-----------------------------------------------------

# The internal DNS Server is allowed to be recursive to respond to queries for GIAC internal systems.
# This means that the internal DNS will contact other DNS servers on the internet. Most of this
# traffic will be carried in UDP packets.  Some records may exceed 492 bytes and will be transferred
# using TCP instead.  When this happens, our DNS server will initiate a TCP connection with the
# DNS Server with large records.
# In short, We will allow outbound UDP/TCP connections initiated from the internal DNS server.

# udp dns rules appear first because they are likely to hit first (i.e. more frequent than tcp)

# ALLOW outbound udp connections from the internal DNS server to other DNS servers on the internet
$IPTABLES --append FORWARD -p udp -i $IF_TO_INTERNAL -o $EXTIF -s $INTERNAL_DNS \
        --sport 53 -d $ANY --dport 53 -m state --state NEW -j ACCEPT


# ALLOW outbound tcp connections from the internal DNS server to other DNS servers
# DNS over TCP is less common than over UDP.  It usually means large data transfer.
# We log the first packet of the TCP handshake to monitor the destination address.
# We may want to check later that the destination is a legitimate DNS server.
# Note: --tcp-flags takes two argument, the first argument is the flag(s) to be examined
#       and the second argument is the flag(s) that must be set. It is sufficient for us
#       to examine the SYN flag (whatever the other flags are).  Disregarding the values
#       of the other flags is even safer. If of our machine gets compromised, it may try
#       to evade security rules by setting other flags as well.
```

```
#       If the other endpoint is a Windows 2000 for instance, it is possible to initiate a
#       connection with it by setting the FIN flag along with the SYN flag.
#
# In short, if at least the SYN flag is set, then LOG.

$IPTABLES --append FORWARD -p tcp -s $INTERNAL_DNS --sport 53 -d $ANY --dport 53 \
        --tcp-flags SYN SYN -j LOG \
        --log-level $LOGLEVEL --log-prefix "INTERNAL DNS OUT (tcp): "
$IPTABLES --append FORWARD -p tcp -i $IF_TO_INTERNAL -o $EXTIF -s $INTERNAL_DNS \
         --sport 53 -d $ANY --dport 53 -m state --state NEW -j ACCEPT

#-----------------------------------------------------
# ALLOW OUTBOUND USER TRAFFIC
#-----------------------------------------------------
# There is no particular reason for prioritizing certain user traffic type over others.
# The rules for the most popular types (traffic involving the SQUID PROXY) are placed first,
# followed by the rules for the other kinds of traffic.

# Allow outbound HTTP connections from the SQUID PROXY
# (only if from the interface that connects the primary firewall to the service network)
$IPTABLES --append FORWARD -p tcp -i $IF_TO_SERVICE -o $EXTIF -s $SERVICE_SQUID \
         --dport 80 -m state --state NEW -j ACCEPT

# Allow outbound HTTPS connections from the SQUID PROXY
# (only if from the interface that connects the primary firewall to the service network)
# We log the first packet of the TCP handshake of HTTPS connections to monitor the
# destination address.  Since this is encrypted traffic, we like to know with whom
# this is being established. We may want to check later that the communication was legitimate.
# Therefore, if the packet is destined to port 443 (default HTTPS) and that at
# least the SYN flag is set, then LOG.
$IPTABLES --append FORWARD -p tcp -i $IF_TO_SERVICE -s $SERVICE_SQUID --dport 443 \
         --tcp-flags SYN SYN -j LOG --log-level $LOGLEVEL --log-prefix "OUTBOUND HTTPS: "
$IPTABLES --append FORWARD -p tcp -i $IF_TO_SERVICE -o $EXTIF -s $SERVICE_SQUID \
          --dport 443 -m state --state NEW -j ACCEPT

# Allow FTP, HTTP and HTTPS connections from the INTERNAL users and TETEWORKERS
# to the SQUID PROXY on port TCP/3128.
$IPTABLES --append FORWARD -p tcp -i $IF_TO_INTERNAL -o $IF_TO_SERVICE \
        -s $INTERNAL_USERS -d $SERVICE_SQUID \
        --dport $PORT_OUTBOUND_PROXY  -m state --state NEW -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $IF_TO_VPN -o $IF_TO_SERVICE -s $TELEWORKER1 \
        -d $SERVICE_SQUID --dport $PORT_OUTBOUND_PROXY  -m state --state NEW -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $IF_TO_VPN -o $IF_TO_SERVICE -s $TELEWORKER2 \
        -d $SERVICE_SQUID --dport $PORT_OUTBOUND_PROXY  -m state --state NEW -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $IF_TO_VPN -o $IF_TO_SERVICE -s $TELEWORKER3 \
        -d $SERVICE_SQUID --dport $PORT_OUTBOUND_PROXY  -m state --state NEW -j ACCEPT
$IPTABLES --append FORWARD -p tcp -i $IF_TO_VPN -o $IF_TO_SERVICE -s $TELEWORKER4 \
        -d $SERVICE_SQUID --dport $PORT_OUTBOUND_PROXY  -m state --state NEW -j ACCEPT


# Allow SMTP connections from the internal MAIL server to the MAIL relay.
# This allows e-mails written by GIAC employees to be forwarded to mail relay.
$IPTABLES --append FORWARD -p tcp -i $IF_TO_INTERNAL -o $IF_TO_SERVICE \
        -s $INTERNAL_MAIL -d $SERVICE_MAIL  \
        --dport $PORT_MAIL_RELAY  -m state --state NEW -j ACCEPT
```

```
# Allow outbound SMTP connections from the MAIL RELAY to the internet
# This allows the e-mails written by GIAC employees to be forwarded to their destination.
$IPTABLES --append FORWARD -p tcp -i $IF_TO_SERVICE -o $EXTIF -s $SERVICE_MAIL \
        --dport 25 -m state --state NEW -j ACCEPT


# Allow outbound FTP connections from the squid proxy to the internet.
# FTP needs special attention in terms of firewall rules. It starts with a ftp-command connection
# to a destination port 21, and then ftp-data sessions are established for the actual data transfer.
# For Netfilter to track down the legitimate ftp-data sessions,  the modules ip_conntrack_ftp and
# ip_nat_ftp must be explicitly loaded.
$IPTABLES --append FORWARD -p tcp -i $IF_TO_SERVICE -o $EXTIF -s $SERVICE_SQUID \
        --dport 21 -m state --state NEW -j ACCEPT

# Allow outbound FTP connections from the internal users to the internet.
# The modules ip_conntrack_ftp and ip_nat_ftp must be explicitly loaded (see above).
# Note: Teleworkers can only ftp-out using the squid proxy.  When the teleworkers are tunneled in,
#       they use a Browser configured to direct ftp, http, and https queries to GIAC's proxy.

$IPTABLES --append FORWARD -p tcp -i $IF_TO_INTERNAL -o $EXTIF -s $INTERNAL_USERS \
        --dport 21 -m state  --state NEW -j ACCEPT


# Log and allow outbound SSH connections from INTERNAL USERS (e.g. to download patch for
# OpenBSD).
# We log the first packet of the TCP handshake of SSH connections to monitor the
# destination address.  Since this is encrypted traffic, we like to know with whom
# this is being established. We may want to check later that the communication was legitimate.
# Note: Teleworkers do not use this king of traffic (implicitly DROP if no rules to ALLOW)
$IPTABLES --append FORWARD -p tcp -i $IF_TO_INTERNAL -s $INTERNAL_USERS --dport 22 \
        --tcp-flags SYN SYN -j LOG --log-level $LOGLEVEL --log-prefix "OUTBOUND SSH: "
$IPTABLES --append FORWARD -p tcp -i $IF_TO_INTERNAL -o $EXTIF -s $INTERNAL_USERS \
        --dport 22  -m state --state NEW -j ACCEPT

# allow outbound RTSP traffic from IPs in range of INTERNAL USERS
# (and from the interface that connects the primary firewall to the border gateway)
# Note: Teleworkers do not use this kind of traffic (implicitly DROP if no rules to ALLOW)
$IPTABLES --append FORWARD -p tcp -i $IF_TO_INTERNAL -o $EXTIF -s $INTERNAL_USERS \
        --dport 554 -m state  --state NEW -j ACCEPT


#########################################################################
# policy rules for FORWARDING, RECEIVING, and TRANSMITING ICMP traffic
#########################################################################

# Most of the rules at the firewall are for forwarding traffic (the input
# and output traffic being implicitly DROP).  This is because this
# host is not suppose to communicate with other hosts.  We do however
# allow ping traffic to be emitted or directed at this box.
# The border gateway takes care of blocking external ping requests
# directed at internal hosts. So we do not have to worry about those.

# The following command creates a NEW (-N) user-defined chain
# for ICMP traffic that will be added later on to the FORWARD,
# INPUT, OUTPUT chains.
$IPTABLES -N ICMP_TRAFFIC
```

```
# We accept in ICMP Unreachable messages and time exceeded messages.
# These packets are often legitimate, and when they are, they must reach the
# other end point for the sake of the communication. However, we are using
# stateful filtering rules at this firewall, and we are now almost at the
# end of the file.
# If these messages were legitimate, the stateful rules placed above
# should have let these in as "RELATED" traffic.
# Therefore, if we get to this point, we will LOG and DROP
# (DROP is implicit if no rules to ALLOW)
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 3 -i $EXTIF -j LOG \
        --log-level $LOGLEVEL --log-prefix "unrelated ICMP-Unreach: "
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 11 -j LOG --log-level $LOGLEVEL \
        --log-prefix "unrelated ICMP-Time-exceed: "

# ALLOW ping requests and reply (The border gateway takes care of blocking
# external ping requests directed at internal hosts. So we do not have to worry about those.
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 8 -j ACCEPT
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 0 -j ACCEPT

# Explicitly DROP all ICMP traffic unless explicitly ACCEPTED above (just to make sure)
$IPTABLES --append ICMP_TRAFFIC -j DROP

# Append ICMP rules to chains INPUT, FORWARD, OUTPUT
$IPTABLES --append INPUT -p icmp -j ICMP_TRAFFIC
$IPTABLES --append FORWARD -p icmp -j ICMP_TRAFFIC
$IPTABLES --append OUTPUT -p icmp -j ICMP_TRAFFIC


#####################################################
# END OF RULES
#####################################################
# If NetFilter reaches this point, it will use the DEFAULT
# DROP policy for INPUT, OUTPUT, and FORWARD chains.


#####################################################
# ENABLE IP FORWARDING
#####################################################
# now that the rules have been applied, allow IP forwarding
echo "1" > /proc/sys/net/ipv4/ip_forward

# End of script
```

VPN Security rules

The VPN gateway is a Linux Redhat 9, kernel 2.4.20-30.9, running the latest version of FreeS/WAN at this time, namely 2.05. The workstations of the teleworkers are also running Redhat 9 with FreeS/WAN 2.05.

We provide in this section the ipsec configuration for the VPN gateway and for one teleworker (teleworker 1). For each of the other teleworkers, new entries will be added to the configuration file of the VPN gateway, and, obviously, the configuration for the workstations of teleworker 2, 3, and 4 will be similar to the one of teleworker1.

The ipsec Tunnel to be established is depicted on the Figure 4. Teleworker1 and the VPN gateway are both ipsec enabled. The teleworker needs to access the database server and the Mail server on the Internal Protected network. For this we create a tunnel named teleworker1_internal that provides the teleworker access to internal hosts on 192.168.1.0/26. The teleworker also has permission to use the proxy server to surf the net (this provides him with a few additional layers of protection). To grant the teleworker this access, we create another tunnel teleworker1_service, which allows him to access hosts on 172.16.1.0/26. The NetFilter firewall is then responsible for restricting access to the services and systems GIAC policy allows.

We start with the ipsec configuration of both the VPN gateway and of the teleworker., and then we will include the NetFilter configuration of the VPN gateway

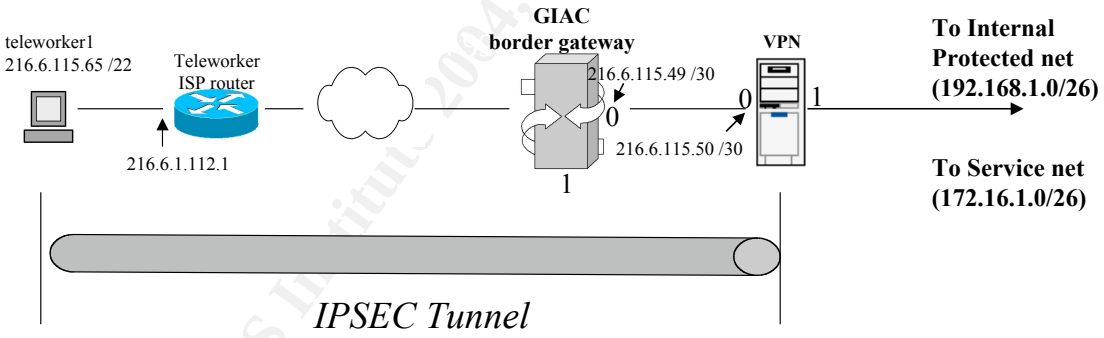


Figure 4

IPSEC configuration

The policy uses automatic keying via ISAKMP. The keys used for encryption and authentification of packets are established by pluto (FreeS/WAN ipsec keying deamon) running on the security gateway and on the workstations of the teleworkers. To allow teleworkers to authenticate themselves to the security gateway, we choose to use a public-key mechanism. The authentication is done as part of ESP encryption. This means that the first IP header of ESP packet is not authenticated. While it would be more secure to have the whole packet authenticated with the Authentication Header (AH) protocol, the tunnel would break if there were a NAT'ing device along the path between the VPN gateway and the teleworker's workstation. This is because NAT changes fields

that AH authenticates. By sticking with ESP for authentication, NAT becomes less of a problem[4].

We use FreeS/WAN default Security Association (SA) definition. The encryption algorithm is 3DES-CBC (RFC 2451), and the hash algorithm is HMAC-MD5 (RFC 2403). This is considered secured enough for GIAC's need, and light-weight enough for fast processing.

We configured FreeS/WAN on the VPN gateway and on the teleworker systems to allow two different tunnels. One tunnel allows connection to the internal network, and the other allows connections to the service network. The most important tunnel is the first one (to the internal network). It is through this one that the teleworkers can update the database and access the mail server. The second tunnel (to the service network) allows the teleworkers to access the Squid proxy for HTTP, HTTPS, and browser based FTP. Although the teleworkers do not have to tunnel in to access these internet services, they can if they wishes to add extra screening and protection to their own personal firewall.

For each system running FreeS/WAN, two files need to be modified. The first one, ipsec.secrets contains the public key and the private key. The second file, ipsec.conf contains the details of the security policy for each tunnel. They both are located in the /etc directory.

We give below the basic instructions on how to configure these files for the VPN gateway and one teleworker workstation. More details can be found at (www.freeswan.org).

1. To generate an RSA key pair on the teleworker workstation (teleworker1) and the VPN gateway (vilaincanard), we used:

On teleworker1:
```
[root@teleworker1]# ipsec rsasigkey  2048 > /etc/ipsec.secrets
```

On vilaincanard:
```
[root@vilaincanard etc]# ipsec rsasigkey  2048 > /etc/ipsec.secrets
```

This generates a 2048-bit public/private key pair for each system. The public/private key pair associated with teleworker1 is stored locally in the /etc/ipsec.secrets file. Similarly, the public/private key pair associated with vilaincanard is stored locally in the /etc/ipsec.secrets file.

2. Then /etc/ipsec.secrets must be edited on both systems to insert a line at the beginning and at the end of the file as shown below. The first line must start at the first column, but all others (including the last one) must be preceded by white-space. The lines added appear below in bold.

[4] Using NAT with ESP still needs some workaround as described at http://fixmyfirewall.com/ip_security_and_nat.html

/etc/ipsec.secrets on vilaincanard:

```
@vpngw: RSA {
       # RSA 2048 bits   vilaincanard   Fri Mar 12 10:06:08 2004
       #pubkey=0sAQNEQ4I/gOuCwA227flBHe5DGpSMOxJcmo+…
       …
      }
```

/etc/ipsec.secrets on teleworker1:

```
@teleworker1: RSA {
       # RSA 2048 bits   LinuxRH9   Sun Mar  7 05:50:17 2004
       #pubkey=0sAQOmjPiZABmx3bmMXDbCtvqWpIltY+…
       …
      }
```

The /etc/ipsec.secrets files are much longer than what is shown above. Each file contains, among other things, the private key, and the public key. Keys span over several lines. The public keys are partly shown above, they are preceded by "pubkey=".

The /etc/ipsec.secrets files must be kept secret (mode 0600). In particular, they must not be transmitted over insecure channels.

3. The files /etc/ipsec.conf on vilaincanard and teleworker1 were edited to include the following parameters (see the man page of ipsec.conf(5) for the meaning of each parameter):

/etc/ipsec.conf on villain canard (the VPN gateway):

```
version      2.0    # conforms to second version of ipsec.conf specification

# basic configuration
config setup
       # Debug-logging controls:  "none" for (almost) none, "all" for lots.
       interfaces="ipsec0=eth0"
       klipsdebug=none
       plutodebug=none


# Add connections here.
conn %default
       type=tunnel #default
       auth=esp # default
       authby=rsasig

# VPN connection for teleworker1 to database and mail server
conn teleworker1_internal
       # Left (security gateway)
       left=216.6.115.50
       leftsubnet=192.168.1.0/26
       leftnexthop=216.6.115.49
       leftid=@vpngw
       # RSA 2048 bits, pubkey:
      leftrsasigkey=0sAQNEQ4I/gOuCwA227flBHe5DGpSMOxJcmo+…
       #
       # Right (Teleworker)
       right=216.6.115.65
       rightnexthop=216.6.112.1
```

```
        rightid=@teleworker1
       # RSA 2048 bits, pubkey:
       rightrsasigkey=0sAQOmjPiZABmx3bmMXDbCtvqWpIltY+…
        #
        # To authorize this connection, but not actually
        # start it, at startup:
        auto=add

# VPN connection for teleworker1 to web proxy
conn teleworker1_service
        # Left (security gateway)
        left=216.6.115.50
        leftsubnet=172.16.1.0/26
        leftnexthop=216.6.115.49
        leftid=@vpngw
        # RSA 2048 bits, pubkey:
               leftrsasigkey=0sAQNEQ4I/gOuCwA227flBHe5DGpSMOxJcmo+…
        #
        # Right (Teleworker)
        right=216.6.115.65
        rightnexthop=216.6.112.1
        rightid=@teleworker1
               # RSA 2048 bits, pubkey:
               rightrsasigkey=0sAQOmjPiZABmx3bmMXDbCtvqWpIltY+…
        #
        # To authorize this connection, but not actually
        #start it, at startup:
        auto=add
```

The `leftrsasigkey` parameter must be set to the `pubkey` value appearing in `/etc/ipsec.secrets` on vilaincanard (the line is several hundred characters long, but must not be split into multiple lines). Similarly, `rightrsasigkey` must be set to the public-key value of teleworker1. Note that these values need not be kept secret.

The /etc/ipsec.conf is a little different on the client side (teleworker1) as shown below.

/etc/ipsec.conf on teleworker1

```
version      2.0    # conforms to second version of ipsec.conf specification

# basic configuration
config setup
        # Debug-logging controls:  "none" for (almost) none, "all" for lots.
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none


# Add connections here.
conn %default
        type=tunnel #default
        auth=esp #default
        authby=rsasig

# VPN connection to database and mail server
conn teleworker1_internal
        left=216.6.115.65
        leftnexthop=216.6.112.1
        leftid=@teleworker1
```

```
        # RSA 2048 bits, pubkey:
       leftrsasigkey=0sAQOmjPiZABmx3bmMXDbCtvqWpIltY+…
        #
        # Right (Remote VPN gateway)
        right=216.6.115.50
        rightnexthop=216.6.115.49
        rightsubnet=192.168.1.0/26
        rightid=@vpngw
        # RSA 2048 bits, pubkey:
       rightrsasigkey=0sAQNEQ4I/gOuCwA227flBHe5DGpSMOxJcmo+…
        #
        # To authorize this connection, but not actually
        #start it, at startup
        auto=add

# VPN connection to web proxy
conn teleworker1_service
        left=216.6.115.65
        leftnexthop=216.6.112.1
        leftid=@teleworker1
        # RSA 2048 bits, pubkey:
       leftrsasigkey=0sAQOmjPiZABmx3bmMXDbCtvqWpIltY+…
        #
        # Right (Remote VPN gateway)
        right=216.6.115.50
        rightnexthop=216.6.115.49
        rightsubnet=172.16.1.0/26
        rightid=@vpngw
        # RSA 2048 bits, pubkey:
       rightrsasigkey=0sAQNEQ4I/gOuCwA227flBHe5DGpSMOxJcmo+…
        #
        # To authorize this connection, but not actually
        #start it, at startup
        auto=add
```

To ensure the configuration files are loaded properly when ipsec is started, start ipsec on each end from a terminal and issue the command "ipsec verify" as shown below. No traffic between the end point is exchanged during this. This command only allows ipsec to verify the local configuration.

```
[root@vilaincanard etc]# service ipsec start
ipsec_setup: Starting FreeS/WAN IPsec 2.05...
ipsec_setup: Using /lib/modules/2.4.20-30.9/kernel/net/ipsec/ipsec.o
[root@vilaincanard etc]# ipsec verify
Checking your system to see if IPsec got installed and started correctly:
Version check and ipsec on-path                                 [OK]
Linux FreeS/WAN 2.05
Checking for IPsec kernel support: found KLIPS                  [OK]
Checking that pluto is running                                  [OK]
Two or more interfaces found, checking IP forwarding            [OK]
Checking NAT and MASQUERADEing                                  [OK]
[root@vilaincanard etc]#

[annie@teleworker1]# service ipsec start
ipsec_setup: Starting FreeS/WAN IPsec 2.05...
ipsec_setup: Using /lib/modules/2.4.20-30.9/kernel/net/ipsec/ipsec.o
[annie@teleworker1]# ipsec verify
Checking your system to see if IPsec got installed and started correctly:
Version check and ipsec on-path                                 [OK]
Linux FreeS/WAN 2.05
Checking for IPsec kernel support: found KLIPS                  [OK]
```

```
Checking that pluto is running                                    [OK]
[annie@teleworker1]#
```

If there errors appear in the output, refer to the FreeS/WAN's documentation, and verify the messages logged in the file "\var\log\secure". The problem may turn out to be a simple typo.

The tunnels are initiated from the teleworker side. Ipsec must be running at all times at the VPN gateway to allow incoming connections from the teleworkers.

Assuming ipsec has been started on each side, the teleworker can start any of the two tunnels by issuing the command "`ipsec auto --up` <name of tunnel>" as shown below. Once a tunnel has been initiated, the teleworker's system will encrypt all communications directed at systems behind the vpn gateway. If the teleworker initiates the tunnel named "teleworker1_internal", he will be able to contact the systems in the internal network range (192.168.1.0/26), provided that the firewall rules on the VPN gateway, primary and internal firewalls allow such traffic. The traffic is encrypted between the teleworker and VPN gateway, and goes unencrypted through the two firewalls. The teleworker can initiate the tunnel named "teleworker1_service" to contact the systems located on the service network (172.16.1.0/26). This tunnel can be initiated without having to turn off the other one. That is, both tunnels can be initiated at the same time.

```
[annie@teleworker1]# ipsec auto --up teleworker1_internal
104 "teleworker1_internal" #1: STATE_MAIN_I1: initiate
106 "teleworker1_internal" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "teleworker1_internal" #1: STATE_MAIN_I3: sent MI3, expecting MR3
004 "teleworker1_internal" #1: STATE_MAIN_I4: ISAKMP SA established
112 "teleworker1_internal" #2: STATE_QUICK_I1: initiate
004 "teleworker1_internal" #2: STATE_QUICK_I2: sent QI2, IPsec SA established
{ESP=>0x9fc57997 <0x8ecf52d8}
[annie@teleworker1]# ping -c 1 192.168.1.14
PING 192.168.1.14 (192.168.1.14) 56(84) bytes of data
64 bytes from 192.168.1.14: icmp_seq=1 ttl =62 time=6.38 ms
--- 192.168.1.14 ping statistics ---
1 packet transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 6.381/6.381/6.381/0.000 ms
[annie@teleworker1]# ipsec auto --up teleworker1_service
112 "teleworker1_service" #3: STATE_QUICK_I1: initiate
004 "teleworker1_service" #3: STATE_QUICK_I2: sent QI2, IPsec SA established
{ESP=>0x9fc57998 <0x8ecf52d9}
[annie@teleworker1]# mozilla&
```

In the example above, the teleworker first initiates a secure ipsec chanel to the internal network, issues a ping request to test the connection with system "192.168.1.14". He then starts the second tunnel, and open a browser to surf the internet. The browser is configured with GIAC proxy. All traffic, except the IKE exchange (ISAKMP), is encrypted using ESP as shown below. The traffic was captured using tcpdump on the teleworker system, and the output was produced using ethereal (print summary). The traffic traces contains a few ARP requests that do not belong to the ipsec communications.

Source	Destination	Protocol	Info

Source	Destination	Protocol	Info
00:0c:29:11:43:7a	ff:ff:ff:ff:ff:ff	ARP	Who has 216.6.115.49? Tell 216.6.115.65
00:01:03:bf:c8:ea	00:0c:29:11:43:7a	ARP	216.6.115.49 is at 00:01:03:bf:c8:ea
216.6.115.65	216.6.115.50	ISAKMP	Identity Protection (Main Mode)
216.6.115.50	216.6.115.65	ISAKMP	Identity Protection (Main Mode)
216.6.115.65	216.6.115.50	ISAKMP	Identity Protection (Main Mode)
216.6.115.50	216.6.115.65	ISAKMP	Identity Protection (Main Mode)
216.6.115.65	216.6.115.50	ISAKMP	Identity Protection (Main Mode)
216.6.115.50	216.6.115.65	ISAKMP	Identity Protection (Main Mode)
216.6.115.65	216.6.115.50	ISAKMP	Quick Mode
216.6.115.50	216.6.115.65	ISAKMP	Quick Mode
216.6.115.65	216.6.115.50	ISAKMP	Quick Mode
00:01:03:bf:c8:ea	00:0c:29:11:43:7a	ARP	Who has 216.6.115.65? Tell 216.6.115.115
00:0c:29:11:43:7a	00:01:03:bf:c8:ea	ARP	216.6.115.65 is at 00:0c:29:11:43:7a
216.6.115.65	216.6.115.50	ESP	ESP (SPI=0x45884ff9)
216.6.115.50	216.6.115.65	ESP	ESP (SPI=0xef7f5ca5)
216.6.115.65	216.6.115.50	ISAKMP	Quick Mode
216.6.115.50	216.6.115.65	ISAKMP	Quick Mode
216.6.115.65	216.6.115.50	ISAKMP	Quick Mode
216.6.115.65	216.6.115.50	ESP	ESP (SPI=0x45884ffa)
216.6.115.50	216.6.115.65	ESP	ESP (SPI=0xef7f5ca6)
216.6.115.65	216.6.115.50	ESP	ESP (SPI=0x45884ffa)
216.6.115.65	216.6.115.50	ESP	ESP (SPI=0x45884ffa)
216.6.115.50	216.6.115.65	ESP	ESP (SPI=0xef7f5ca6)
216.6.115.50	216.6.115.65	ESP	ESP (SPI=0xef7f5ca6)
216.6.115.50	216.6.115.65	IP	Fragmented IP protocol (proto=ESP 0x32, off=1480)
216.6.115.50	216.6.115.65	ESP	ESP (SPI=0xef7f5ca6)
216.6.115.65	216.6.115.50	ESP	ESP (SPI=0x45884ffa)
216.6.115.50	216.6.115.65	ESP	ESP (SPI=0xef7f5ca6)
216.6.115.65	216.6.115.50	ESP	ESP (SPI=0x45884ffa)
216.6.115.65	216.6.115.50	ESP	ESP (SPI=0x45884ffa)
216.6.115.50	216.6.115.65	ESP	ESP (SPI=0xef7f5ca6)
00:0c:29:11:43:7a	00:01:03:bf:c8:ea	ARP	Who has 216.6.115.49? Tell 216.6.115.65
00:01:03:bf:c8:ea	00:0c:29:11:43:7a	ARP	216.6.115.49 is at 00:01:03:bf:c8:ea

To tear down the tunnels, the teleworker issues the following commands:

```
[annie@teleworker1]# ipsec auto --down teleworker1_service
[annie@teleworker1]# ipsec auto --down teleworker1_internal
```

NetFilter Configuration of the VPN gateway

The NetFilter rules on the VPN gateway are important to restrict access to only the services and systems the security policy allows. Assuming the teleworker's system gets compromised and that the attacker initiates a tunnel to GIAC's network, the firewall on the VPN gateway becomes our first line of defense. The border offers no protection since it lets all encrypted traffic from the teleworkers to the VPN gateway pass through. The

primary firewall is then our second line of defence. All traffic from the teleworkers arrives unencrypted at the primary firewall.

The NetFilter configuration of the VPN is the following:

```
#!/bin/sh
# The script configures a Netfilter firewall for a
# VPN gateway with 2 interfaces.

IPTABLES="/sbin/iptables"   # Location of iptables binary
DEPMOD="/sbin/depmod"       # Location of kernel modules dependency handler
MODPROBE="/sbin/modprobe"   # Location of kernel modules loader

################################################
# SET KERNEL VARIABLES
################################################

# kernel variables have the same values as those of the border gateway
# They have been removed from this printed version to save paper...
.
.
.

################################################
#  LOAD MODULES
################################################

# see http://www.ecst.csuchico.edu/~dranch/LINUX/ipmasq/examples/rc/firewall-2.4
# for a description of the modules.  This link also describes how certain of these
# modules are automatically loaded given certain iptables commands are invoked.
# Loading the modules manually as we do below cleans up kernel auto-loading timing issues.

$DEPMOD -a                # Verifies that all modules have required dependencies
$MODPROBE ip_tables       # Loads the main body of the iptables module
$MODPROBE iptable_filter  # Loads the filtering modules (also loads ipt_LOG, ipt_REJECT)
$MODPROBE ip_conntrack    # Loads the stateful connection tracking
$MODPROBE ip_conntrack_ftp  # Loads the FTP tracking mechanism
$MODPROBE iptable_nat     # Loads the NAT functionality (MASQ)
$MODPROBE ip_nat_ftp      # Loads the FTP NAT (required to support PASV FTP)
$MODPROBE ipt_limit       # Allows to set a limit on packet rate (hits per sec, min, or hour)
$MODPROBE ipt_state       # Allows to catch packets with various IP and TCP flags settings
$MODPROBE ipt_unclean     # Allows to catch packets with invalid IP and TCP flags settings



################################################
# CLEAR OUT FIREWALL RULES
################################################
$IPTABLES -F              # flush default table (same as $IPTABLES --table filter)
$IPTABLES -X              # delete every non built-in chain
$IPTABLES -Z              # zero out counters in all chains
$IPTABLES --table nat -F    # flush nat table
$IPTABLES --table mangle -F # flush mangle table
```

```
# mangle is used for specialized packet alteration,
# it is not used in the rules at this time.


################################################
# SET DEFAULT DROP POLICY FOR BUILT-IN CHAINS
################################################

$IPTABLES --policy INPUT DROP      # traffic targeted at the firewall
$IPTABLES --policy OUTPUT DROP     # traffic leaving the firewall
$IPTABLES --policy FORWARD DROP    # traffic crossing the firewall


################################################
#GLOBAL VARIABLES
################################################

LOGLEVEL=notice

# The next 4 variables are concerned with this host (the primary firewall)
EXTIF=eth0              # interface to border gateway (external interface)
IPSECIF=ipsec+              # any interface beginning with ipsec
IF_TO_FW=eth1          # interface to primary firewall
EXTIP=216.6.115.50      # External IP (facing border gateway)
IP_TO_FW=10.10.1.2      # IP facing vpn gateway


GW=216.6.115.49         # IP of gateway (directly connected to this gateway)
FW=10.10.1.1           # IP of primary firewall  (directly connected to this gateway)


SERVICE=172.16.1.0/26         # service network range
SERVICE_SQUID=172.16.1.8      # IP of squid proxy

INTERNAL=192.168.1.0/24       # internal network range
INTERNAL_USERS=192.168.1.64/26 # internal user range
INTERNAL_MAIL=172.16.1.25      # IP of mail relay
INTERNAL_DB=192.168.1.14       # IP of database
INTERNAL_DNS=192.168.1.117     # IP of internal DNS (unused, cannot be reached)


TELEWORKER1=216.6.115.65   # is allowed special access (vpn)
TELEWORKER2=216.6.115.75   # is allowed special access (vpn)
TELEWORKER3=216.6.115.85   # is allowed special access (vpn)
TELEWORKER4=216.6.115.95   # is allowed special access (vpn)

PORT_WEB_PROXY=3128     # port expected at PROXY for outbound HTTP/HTTPS connections
(configurable)
PORT_SMTP_SERVER=25     # SMTP port expected at mail server (configurable)
PORT_IMAP_SERVER=143    # IMAP port expected at mail server (configurable)
PORT_DB_SERVER=1433     # MySQL port expected at the database server (configurable)


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

```
#  STATIC ROUTES
#####################################################

# To reach the service network
route add -net 172.16.1.0 netmask 255.255.255.192 gw $FW dev $IF_TO_FW

# To reach the internal network
route add -net 192.168.1.0 netmask 255.255.255.0 gw $FW dev $IF_TO_FW


#########################################################################
# LOCALHOST (127.0.0.1) TRAFFIC
#########################################################################

# Log and allow localhost traffic entering and leaving lo interface
# We will give priority to this host before letting outbound user traffic through.
# Note: We will monitor for a while what traffic this host needs to send to himself.
#       If messages are rare, the iptables rules will be moved further down. Although
#       this is not a big deal in terms of firewall performance, it is good to keep
#       in mind that NetFilter goes through the rules from the top down until it finds
#       a match.  Starting with most popular rules can save some processing.
#       Use the following command to find loopback traffic in the logs:
#       grep "Local-" /var/log/messages

#$IPTABLES --append INPUT -i lo -j LOG --log-level $LOGLEVEL --log-prefix "Local-in: "
#$IPTABLES --append OUTPUT -o lo -j LOG --log-level $LOGLEVEL --log-prefix "Local-out: "
$IPTABLES --append INPUT -i lo -j ACCEPT
$IPTABLES --append OUTPUT -o lo -j ACCEPT


#########################################################################
# INBOUND VPN ACCESS
#########################################################################

# log Internet Key Exchange (IKE)
$IPTABLES --append INPUT -p udp -i $EXTIF --sport 500 --dport 500 -j LOG \
        --log-level $LOGLEVEL --log-prefix "IKE-in: "
$IPTABLES --append OUTPUT -p udp -o $EXTIF --sport 500 --dport 500 -j LOG \
        --log-level $LOGLEVEL --log-prefix "IKE-out: "

# log Encapsulated Security Payload (ESP)
$IPTABLES --append INPUT -p 50 -i $EXTIF -j LOG --log-level $LOGLEVEL \
        --log-prefix "ESP-in: "
$IPTABLES --append OUTPUT -p 50 -o $EXTIF -j LOG --log-level $LOGLEVEL \
        --log-prefix "ESP-out: "


# Allow ESTABLISHED connections and related traffic between teleworkers and vpn gateway
$IPTABLES --append INPUT -s $TELEWORKER1 -m state --state ESTABLISHED,RELATED \
         -j ACCEPT
$IPTABLES --append OUTPUT -d $TELEWORKER1 -m state --state ESTABLISHED,RELATED \
         -j ACCEPT
$IPTABLES --append INPUT -s $TELEWORKER2 -m state --state ESTABLISHED,RELATED \
         -j ACCEPT
$IPTABLES --append OUTPUT -d $TELEWORKER2 -m state --state ESTABLISHED,RELATED \
         -j ACCEPT
```

```
$IPTABLES --append INPUT -s $TELEWORKER3 -m state --state ESTABLISHED,RELATED \
        -j ACCEPT
$IPTABLES --append OUTPUT -d $TELEWORKER3 -m state --state ESTABLISHED,RELATED \
        -j ACCEPT
$IPTABLES --append INPUT -s $TELEWORKER4 -m state --state ESTABLISHED,RELATED \
        -j ACCEPT
$IPTABLES --append OUTPUT -d $TELEWORKER4 -m state --state ESTABLISHED,RELATED \
        -j ACCEPT

# Allow NEW IKE connections from teleworkers to the VPN gateway
$IPTABLES --append INPUT -p udp -i $EXTIF -s $TELEWORKER1 --sport 500 -d $EXTIP --dport 500 \
        -m state --state NEW -j ACCEPT
# Allow NEW IKE connections from teleworkers to VPN gateway
$IPTABLES --append INPUT -p udp -i $EXTIF -s $TELEWORKER2 --sport 500 -d $EXTIP --dport 500 \
        -m state --state NEW -j ACCEPT
# Allow NEW IKE connections from teleworkers to VPN gateway
$IPTABLES --append INPUT -p udp -i $EXTIF -s $TELEWORKER3 --sport 500 -d $EXTIP --dport 500 \
        -m state --state NEW -j ACCEPT
# Allow NEW IKE connections from teleworkers to VPN gateway
$IPTABLES --append INPUT -p udp -i $EXTIF -s $TELEWORKER4 --sport 500 -d $EXTIP --dport 500 \
        -m state --state NEW -j ACCEPT

# Allow NEW ESP traffic initiated from the teleworkers to the VPN gateway
$IPTABLES --append INPUT -p 50 -i $EXTIF  -s $TELEWORKER1 -d $EXTIP -m state --state NEW \
         -j ACCEPT
$IPTABLES --append INPUT -p 50 -i $EXTIF  -s $TELEWORKER2 -d $EXTIP -m state --state NEW \
         -j ACCEPT
$IPTABLES --append INPUT -p 50 -i $EXTIF  -s $TELEWORKER3 -d $EXTIP -m state --state NEW \
         -j ACCEPT
$IPTABLES --append INPUT -p 50 -i $EXTIF  -s $TELEWORKER4 -d $EXTIP -m state --state NEW
         -j ACCEPT


#####################################################
# TRAFFIC BETWEEN INTERFACES eth+ AND ipsec+
#####################################################
# Allow any forwarded traffic to or from ipsec interfaces
$IPTABLES --append FORWARD -i ipsec+ -j ACCEPT
$IPTABLES --append FORWARD -o ipsec+ -j ACCEPT


#####################################################
# CONNECTIONS TO INTERNAL NETWORK
#####################################################

# STATEFUL PACKET FILTERING
# We will allow connections initiated from the internet to systems on the service network.
#
# For each type of traffic, we break the rule into two part to tighten up the security:
# (1) We tell NetFilter to ACCEPT NEW or ESTABLISHED inbound connections.
#     We tighten up the rule by specifying interface, destination IP
#     and destination port.
#
# (2) Because we specify the direction in rule (1), we need to create another rule to allow
#     ESTABLISHED connections in the other direction.  Because Netfilter maintains src/dst
#     information in the state table, this rule can be loose in terms of direction and ports.
```

```
#     It does not even have to specify the protocol.  This rule can thus be use to allow
#     RELATED traffic (e.g. ICMP error messages).
# It would be redundant to place rule(2) for each type of traffic we allow.  It will therefore
# be written once, and before any other rule.  This means that ESTABLISHED connections are
# given priority over NEW ones.

# Allow ESTABLISHED connections and related traffic (see comment above concerning "rule (2)").
$IPTABLES --append FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# Because established connections are allowed at this point, we will specify what NEW
# inbound connections are ACCEPTED.


# Allow SMTP and IMAP connections to the MAIL server, and on the ports the server expects.
# This allows the teleworkers to send (SMTP) e-mails,and retrieve (IMAP) e-mails.
# Traffic must first go through the primary firewall.
$IPTABLES --append FORWARD -p tcp -o $IF_TO_FW -s $TELEWORKER1 -d $INTERNAL_MAIL \
        -m multiport --dport $PORT_SMTP_SERVER,$PORT_IMAP_SERVER -m state --state NEW -j
ACCEPT
$IPTABLES --append FORWARD -p tcp -o $IF_TO_FW -s $TELEWORKER2 -d $INTERNAL_MAIL \
        -m multiport --dport $PORT_SMTP_SERVER,$PORT_IMAP_SERVER -m state --state NEW -j
ACCEPT
$IPTABLES --append FORWARD -p tcp -o $IF_TO_FW -s $TELEWORKER3 -d $INTERNAL_MAIL \
        -m multiport --dport $PORT_SMTP_SERVER,$PORT_IMAP_SERVER -m state --state NEW \
        -j ACCEPT
$IPTABLES --append FORWARD -p tcp -o $IF_TO_FW -s $TELEWORKER4 -d $INTERNAL_MAIL \
        -m multiport --dport $PORT_SMTP_SERVER,$PORT_IMAP_SERVER -m state --state NEW \
        -j ACCEPT



# Allow MySQL connections to the database server, and on the ports the server expects.
# Traffic must first go through the primary firewall.
$IPTABLES --append FORWARD -p tcp -o $IF_TO_FW -s $TELEWORKER1 -d $INTERNAL_DB \
        --dport $PORT_DB_SERVER -m state --state NEW -j ACCEPT
$IPTABLES --append FORWARD -p tcp -o $IF_TO_FW -s $TELEWORKER2 -d $INTERNAL_DB \
        --dport $PORT_DB_SERVER -m state --state NEW -j ACCEPT
$IPTABLES --append FORWARD -p tcp -o $IF_TO_FW -s $TELEWORKER3 -d $INTERNAL_DB \
        --dport $PORT_DB_SERVER -m state --state NEW -j ACCEPT
$IPTABLES --append FORWARD -p tcp -o $IF_TO_FW -s $TELEWORKER4 -d $INTERNAL_DB \
        --dport $PORT_DB_SERVER -m state --state NEW -j ACCEPT


#####################################################
# CONNECTIONS TO RELAYS (service network)
#####################################################

# Allow FTP, HTTP and HTTPS connections from TELEWORKERS to the SQUID proxy, and on
# the port the proxy expects. SQUID will go and get the files for the teleworkers.
$IPTABLES --append FORWARD -p tcp -o $IF_TO_FW -s $TELEWORKER1 -d $SERVICE_SQUID \
        --dport $PORT_WEB_PROXY -m state --state NEW -j ACCEPT
$IPTABLES --append FORWARD -p tcp -o $IF_TO_FW -s $TELEWORKER2 -d $SERVICE_SQUID \
        --dport $PORT_WEB_PROXY -m state --state NEW -j ACCEPT
$IPTABLES --append FORWARD -p tcp -o $IF_TO_FW -s $TELEWORKER3 -d $SERVICE_SQUID \
        --dport $PORT_WEB_PROXY -m state --state NEW -j ACCEPT
$IPTABLES --append FORWARD -p tcp -o $IF_TO_FW -s $TELEWORKER4 -d $SERVICE_SQUID \
```

```
        --dport $PORT_WEB_PROXY -m state --state NEW -j ACCEPT


#########################################################################
# CONNECTIONS TO THE INTERNET
#########################################################################

# Allow outbound direct connections from the teleworkers to the internet.
# None! they are on their own...



#########################################################################
# policy rules for FORWARDING, RECEIVING, and TRANSMITING ICMP traffic
#########################################################################
# We allow ping traffic to be emitted from or directed at this box.
# In particular, if the teleworkers cannot get through using the VPN,
# they may want to test connectivity with this box.  We allow it
# from anywhere so that the admins can test from where ever they want.

# The following command creates a NEW (-N) user-defined chain
# for ICMP traffic that will be added later on to the FORWARD,
# INPUT, OUTPUT chains.
$IPTABLES -N ICMP_TRAFFIC

# We accept in ICMP Unreachable messages and time exceeded messages.
# These packets are often legitimate, and when they are, they must reach the
# other end point for the sake of the communication. However, we are using
# stateful filtering rules at this firewall, and we are now almost at the
# end of the file.
# If these messages were legitimate, the stateful rules placed above
# should have let these in as "RELATED" traffic.
# Therefore, if we get to this point, we will LOG and DROP
# (DROP is implicit if no rules to ALLOW)
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 3 -i $EXTIF -j LOG \
        --log-level $LOGLEVEL --log-prefix "unrelated ICMP-Unreach: "
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 11 -j LOG --log-level $LOGLEVEL \
        --log-prefix "unrelated ICMP-Time-exceed: "

# ALLOW ping requests and reply (The border gateway takes care of blocking
# external ping requests directed at internal hosts. So we do not have to
# worry about those.
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 8 -j ACCEPT
$IPTABLES --append ICMP_TRAFFIC -p icmp --icmp-type 0 -j ACCEPT

# Explicitly DROP all ICMP traffic unless explicitely ACCEPTED above (just to make sure)
$IPTABLES --append ICMP_TRAFFIC -j DROP

# Append ICMP rules to chains INPUT, FORWARD, OUTPUT
$IPTABLES --append INPUT -p icmp -j ICMP_TRAFFIC
$IPTABLES --append FORWARD -p icmp -j ICMP_TRAFFIC
$IPTABLES --append OUTPUT -p icmp -j ICMP_TRAFFIC


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

```
# END OF RULES
#######################################################
# If NetFilter reaches this point, it will use the DEFAULT
# DROP policy for INPUT, OUTPUT, and FORWARD chains.


#######################################################
# ENABLE IP FORWARDING
#######################################################
# now that the rules have been applied, allow IP forwarding
echo "1" > /proc/sys/net/ipv4/ip_forward

# End of script
```

Tutorial on how to set-up a NetFilter border gateway

This section provides guideline on how to set-up a border gateway using Linux and NetFilter.
The installation is threefold:

1. Linux installation and configuration
2. NetFilter installation and configuration
3. Connection set-up with the ISP link.

Linux installation and configuration

Install a Linux system with an up-to-date Linux kernel. Here we used the RedHat 9 Linux distribution with kernel 2.4.20-30.9. The installation, requires that a compiler (e.g. gcc) be included as well has the kernel source. This will allow to recompile the kernel to include neat NetFilter modules.

Secure the Operating System. Several resources are available for this purpose including the following:

- Bastille-Linux (http://www.bastille-linux.org)
  a set of hardening scripts available in RPM format.
- Hardening Linux Systems (http://www.linux-mag.com/2002-09/guru_01.html)
  a guide in hardening linux systems.
- Red Hat Linux Security Guide (http://www.redhat.com/docs/manuals/linux/RHL-9 Manual/security-guide/)
  an overview of security best practices from Red Hat.
- Linux Security HOWTO (http://tldp.org/HOWTO/Security-HOWTO/index.html)
  an overview of security issues that face the administrator of Linux systems.

NetFilter installation and configuration

Install the latest version of Netfilter found at http://www.netfilter.org/. Netfilter alone is fairly straightforward to install. Assuming the downloaded compressed file is iptables-1.2.9.tar.bz2 (the latest version of iptables currently available), here is how to proceed:

1. Decompress the file:
   # bzip2 iptables-1.2.9.tar.bz2
   # tar –xvf iptables-1.2.9.tar
2. Go to the new directory:
   # cd iptables-1.2.9
3. Compile the package:
   # make KERNEL_DIR=<<where-you-built-your-kernel>>
4. Install the shared libraries, and the binary:
   # make install KERNEL_DIR=<<where-you-built-your-kernel>>

It gets a little more complicate if you'd like to install certain "add on" functionalities to NetFilter that are not included in the base package. The following link describes a number of interesting extensions to enhance the capability of your firewall: http://www.netfilter.org/patch-o-matic/pom-extra.html. Among other thing, there is a

module for basic passive OS fingerprinting (osf), and another for port scan detection (psd). This is how I got the module for modifying TTL values (ipt_TTL).

To install these patches, download the latest version of " patch-o-matic" (also known as "p-o-m"). p-o-m is a script that guides you through the process of choosing/selecting the patches you want to apply, and automatically patch the kernel for you. p-o-m is available for download at the NetFilter web site (http://www.netfilter.org/).

Assuming the downloaded compressed file is patch-o-matic-20031219.tar.bz2 (the latest version as of Today), follow these steps:

1. Decompress the file:
    # bzip2 iptables-1.2.9.tar.bz2
    # tar –xvf iptables-1.2.9.tar

2. Make sure the dependencies are made already. If unsure:
    # cd /usr/src/<<name of yourkernel>>/
    # make dep

3. Go back to the directory where you decompressed the latest p-o-m, e.g.:
    # cd /root/patch-o-matic

4. run p-o-m:
    #./runme extra

p-o-m will go through the patches one by one and will prompt you to decide whether or not to install it.

Once you have installed all the patches you wished to apply, the next step is to recompile your kernel and install it. While configuring your kernel, you will see new options in "Networking Options -> Netfilter Configuration". Choose the options you need, recompile and install your new kernel.

This tutorial will not explain how to do this. Instead, you can follow the instruction at http://www.linuxvoodoo.com/resources/howtos/kernel/kernel1.php. The instructions go through each step to take for compiling the kernel and making modifications to your boot loader.

Once your new kernel is installed, you can install and compile NetFilter according to the instructions above.

There are good tutorials available for NetFilter. The one I prefer is from FrozenTux available at http://iptables-tutorial.frozentux.net/iptables-tutorial.html.

To start NetFilter, simply issue the following command:
    # service iptables start

When customizing the rules for your needs, start with the rules that implement your security policy. Place them in a script as we did for the border gateway of this practical. Once the script has been run and your rules applied, it is a good idea to prompt NetFilter to display the rules. Use the command:

```
# iptables –L
```

This will tell you if some of your rules are redundant. More importantly, you will see how Netfilter interpreted them. You may want to revise the order in which you placed the rules based on what you see.

When you are satisfied with your primary security policy, you can do a second round, and play with the neat stuff you have installed with p-o-m. Always verify that you haven't broken anything. Have fun!

Connection set-up with the internet

The last thing you need to do is to connect with the internet. In GIAC Enterprise case, the ISP link is on PPPoE. We installed a package named "roaring penguin" to help set-up the connection. The package can be downloaded at http://www.roaringpenguin.com

This package greatly facilitates the installation set-up. The HOWTO that comes with it describes every step and the purpose of the configuration. All you need is to have the information about your ISP account handy.

Assignment 3 – Verify the Firewall Policy

Validation Plan

In this section we validate the security policies implemented by the firewalls. Please note that this does not constitute a thorough assessment of the GIAC enterprises security policy. A complete assessment would also challenge the security measures implemented on the servers and relays. The systems should have been hardened, and secured using best security practice. For instance, it would be of the utmost importance to validate that the restricted access to files and directories on the web server is effective, that the relays cannot be used as open proxys from where spam could be relayed, or attack against other site could be launched. Obviously, a vulnerability assessment on all the systems is also required. Lastly, because of the experience in passive profiling one of GIAC partner has, the traffic outside the border gateway will be monitored passively for a week to see what transpire of GIAC's enterprises network. This could very well call for modifications of some firewall rules. We concentrate here on testing the firewall rules as described in the previous section.

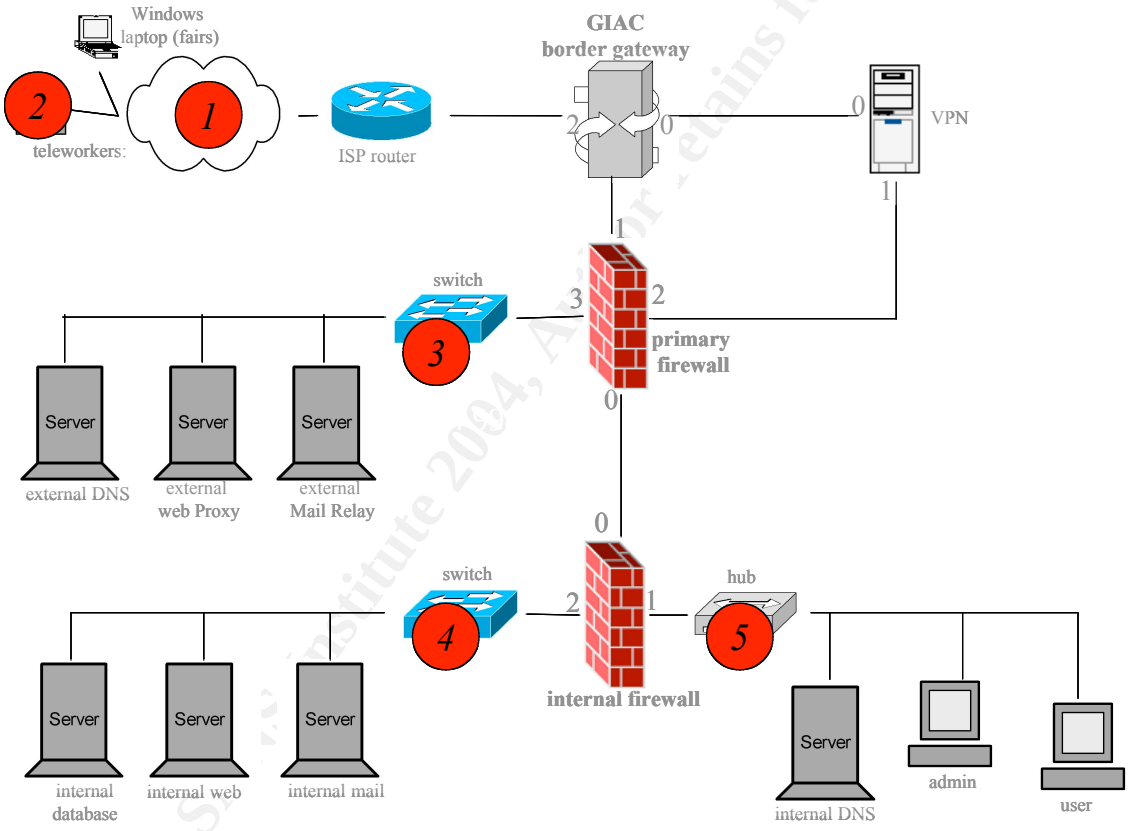
The firewall rules will be tested using primarily hping2 (http://www.hping.org). The reason for choosing hping2 is that it allows conducting the assessment with a minimal number of packets (thus smaller traffic trace to analyze), and it is flexible enough to craft all IP, TCP, UDP, and ICMP packets we wish to test. Remember that the intent here is

not to assess that all ports are closed on a target behind the firewall, but rather to test whether the firewall will let the scan reach the target.

The tool will be called from shell scripts. There will be as many scripts as there are locations from where to launch the scans. To test the firewall rules of the border gateway, the primary firewall, and the VPN firewall, we will conduct the assessment from five access points:

1. Assessment from the internet;
2. Assessment from an ipsec tunnel initiated from a teleworker's system;
3. Assessment from the service network;
4. Assessment from the Internal Protected Network;
5. Assessment from the Internal User Network;

Figure 5



A laptop running Linux and hping2-rc2 is used as the scanner. Note that hping2-rc1 does not allow to specify the protocol field of an IP packet. Therefore, to test ipsec packets (e.g. ESP), hping2-rc2 is required.

The assessment will be conducted late at night, and all partners will participate to facilitate and speed up the process. There is no cost associate with this assessment. The

connection to the internet will be down for a very short period of time. 15 minutes[5] should be sufficient to conduct the firewall rules assessment from outside of the border gateway. The rest of the assessment can be conducted while the network is up[6].

The systems under tests here are:

- The border gateway
- The VPN gateway
- The Primary firewall
- The Internal firewall

When scanning the network, all interfaces of all systems under test will be turned into promiscuous mode to monitor the traffic. This will be done using tcpdump. For instance, to monitor all three interfaces of the border gateway (named luke), we issue the commands:

```
[root@luke]# tcpdump -i eth0 -n -w name_of_Assessment_gw_eth0.pcap
[root@luke]# tcpdump -i eth1 -n -w name_of_Assessment_gw_eth1.pcap
[root@luke]# tcpdump -i eth2 -n -w name_of_Assessment_gw_eth2.pcap
```

Each command tells tcpdump on wich interface to listen (-i eth+), that it should not issue DNS queries (-n), and to save the traffic into a file in the libpcap format (-w). The name of the file is chosen so that it clearly indicates the context of the capture.

The following paragraphs describe the outline of the plan for each assessment. GIAC partners have already conducted the Assessment from the internet. The validation results for this assessment are provided in a subsequent section.

1. Assessment from the internet

The laptop acting as the scanner is connected to the external interface of the border gateway. This implies that the connection with the internet is interrupted. The packets will be directed at either the border gateway's IP address or the VPN gateway IP address (both routable IP address). Some of the packets will also be destined to private addresses to test if they are blocked at the border gateway.

The key points to validate are:

Traffic to validate	**Intended behaviour**
Do packets with reserved source addresses get dropped?	DROP at the border gateway
Do packets with reserved destination addresses get dropped?	DROP at the border gateway
Is the legitimate traffic destined to the relays on the service network is properly forwarded, and can the response be received?	ACCEPT at the border gateway, the Primary firewall and Internal

[5] A little more time should be plan to test, from outside the network, the security measures implemented on the servers. I would recommend to do this test a day or two later, and benefit from the experience acquired during the firewall rules validation.

[6] Even the vulnerability assessment can be conducted while the link to the internet is up. The rate of the scan should be reduced to limit the impact on bandwidth and cpu. GIAC has few systems, this can be done overnight.

	Firewall
Can the primary firewall drop what the stateless gateway cannot?	DROP at the primary firewall
Can the legitimate traffic destined to the VPN gateway get through?	ACCEPT at the border gateway
Is the non-legitimate traffic destined to the VPN gateway dropped?	DROP at the border gateway
Do ICMP error messages get through?	ACCEPT at the border gateway, and if legitimate, at the other firewalls
Does the IP TTL value of each response packet set to 64?	Must be 64
Do the rules to log traffic really work?	LOG
Are ident TCP SYN packet (port 113) rejected with a RST?	REJECT with RST

2. Assessment from a tunnel initiated by a teleworker

In this particular case, the scanner will not be the laptop. We choose teleworker1 as the scanner. After installing hping2-rc2 and the appropriate script on his system, he will start ipsec, initiate the two configured tunnels (one to the service network, and one to the internal protected network). It is only then that he will launch the script. The targets to which to direct traffic have IP addresses in the range on the internal private network(172.168.1.0/26), or in the range of the service network (172.16.1.0/26). Any other destination address is a "NO GO"!! The traffic would not be tunneled in both rather sent on the internet. We do not want that…

The key points to validate are:

Traffic to validate	**Intended behaviour**
Can the teleworker reach the database and the mail server on the Internal Protected Network (ports TCP/25, TCP/443)?	ACCEPT at the VPN, the primary firewall and the Internal firewall
When the target is the database server, are requests other than MySQL dropped?	DROP at the VPN
When the target is the mail server, are requests other than mail dropped?	DROP at the VPN
Can the teleworkers reach the Squid proxy on the service network on port TCP/3128?	ACCEPT at the VPN and the Primary firewall
When the target is the squid proxy, are requests targeted at other ports dropped?	DROP at the VPN
Is the traffic logged according to the rules?	LOG

3. Assessment from the Service Network

The Linux laptop is used and connected to a port on the switch. This configuration allows to remain connected with the internet during the assessment. The source address will be spoof as if coming from the systems on the service network (172.16.1.0/26). We will spoof the IP address of the Squid proxy, the IP address of the Mail Relay, and the IP address of the External DNS server. The external DNS server is not suppose to contact anyone. Other unused addresses, within the range of the service network and not within this range, will be spoofed as well to verify that the packets get dropped.

Because the source addresses are spoofed, we should not expect the laptop to receive the responses. As always, we rely on the traffic trace to examine the behaviours.
The traffic will be directed at:

- The internet (the teleworkers IP address will do)
- Systems on the Internal Protected Network
- Systems on the Internal User Network
- The primary firewall
- The border gateway
- The VPN gateway

Key points to validate:

Traffic to validate	**Intended behaviour**
Can the Squid proxy initiate and maintain FTP connections with hosts on the internet?	ACCEPT at the Primary firewall and the border gateway
Can the Squiq proxy initiate HTTP/HTTPS connections with hosts on the internet?	ACCEPT at the Primary firewall and the border gateway
Can the Squid proxy forward packets to the web server on the internal network?	ACCEPT at the Primary firewall and Internal firewall
Can the Mail relay forward SMTP traffic to the internet?	ACCEPT at the Primary firewall and the border gateway
Can the Mail relay forward SMTP traffic to the Mail server on the internal network?	ACCEPT at the Primary firewall and Internal firewall
Can hosts on this network contact the internal DNS server? (To reach destinations on the internet, the Mail relay and Squid proxy may need to resolve the the IP address associated with a given fully qualified domain name.)	ACCEPT at the Primary firewall and Internal firewall
Does any other type of traffic get dropped?	DROP at the Primary firewall
If a scan is initiated from a host on the service network, does it get logged on the primary firewall?	LOG at the Primary firewall
Other than ARP, and ping, do all the other types of traffic get	DROP, these boxes

dropped if directed at the firewalls (i.e. border gateway, primary firewall, VPN gateway)?	only forwards traffic.

4. Assessment from the Internal Protected Network

The Linux laptop is used and connected to a port on the switch. This configuration allows GIAC network to remain connected with the internet during the assessment. The source address will be spoofed as if coming from the systems on the internal protected network (192.168.1.0/26). We will spoof the database server, the Mail server and the web server. Other "faked" (unused) addresses will be spoofed as well to verify that the packets get dropped. Some of these other spoofed addresses will be within the range of the Internal Protected Network (i.e. between 192.168.1.1 to 192.168.1.1.63) and some will be out of this range.

Because the source addresses are spoofed, we should not expect the laptop to receive the responses. As always, we rely on the traffic trace to determine whether the rules were defeated or not.

For this assessment, we will also test the rules implemented on the internal firewalls. Because these rules have not been described in this document, we will describe the validation based on the flow of traffic described in the first part on this practical.

The traffic will be directed at:

- The internet (the teleworkers' IP addresses will do)
- Systems on the Service Network
- Systems on the Internal User Network
- The primary firewall
- The internal firewall
- The border gateway
- The VPN gateway

Key points to validate:

Traffic to validate	**Intended behaviour**
Can the web server initiates connections with the Squid proxy located on the service network?	ACCEPT at the Internal and Primary firewall
Can the Mail server initiate SMTP connections with the Mail relay located on the service network?	ACCEPT at the Internal and Primary firewall
Is any traffic initiated from the database dropped? (other than ARP)	DROP at the Internal firewall
Does any other kind of traffic get dropped?	DROP at the Internal firewall
If a scan is initiated from a host on the service network, does it get logged on the primary firewall?	LOG at the Internal firewall

5. Assessment from the Internal User Network

The Linux laptop is used and connected to a port on the hub. This configuration allows GIAC network to remain connected with the internet during the assessment. The source address will be spoofed as if coming from the systems on the internal protected network (192.168.1.64/26). We will spoof the admin workstation, the other user workstation, and the internal DNS server. Other "faked" (unused) addresses will be spoofed as well to verify that the packets get dropped.

Because the source addresses are spoofed, we should not expect the laptop to receive the responses. As always, we rely on the traffic trace to determine whether the rules were defeated or not.

For this assessment, we will also test the rules implemented on the internal firewalls. Because these rules have not been described in this document, we will describe the validation based on the flow of traffic described in the first part on this practical.

The traffic will be directed at:

- The internet (the teleworkers' IP addresses will do)
- Systems on the Service Network
- Systems on the Internal Protected Network
- The primary firewall
- The internal firewall
- The border gateway
- The VPN gateway

Key points to validate:

Traffic to validate	**Intended behaviour**
Can the user and the admin initiate TCP/3128 connections with the Squid proxy located on the service network?	ACCEPT at the Internal and Primary firewall
Can the user and the admin initiate TCP/25 and TCP/443 connections with the Mail server located on the Internal Protected Network?	ACCEPT at the Internal firewall
Can the user and the admin initiate FTP sessions in Passive and Active mode with FTP servers on the internet?	ACCEPT at the Internal firewall, Primary firewall, and border gateway
Can the user and the admin initiate SSH session with SSH servers on the internet?	ACCEPT at the Internal firewall, Primary firewall, and border gateway
Can the admin and the user initiate TCP/554 sessions with RTSP servers.	ACCEPT at the Internal firewall, Primary firewall, and border gateway

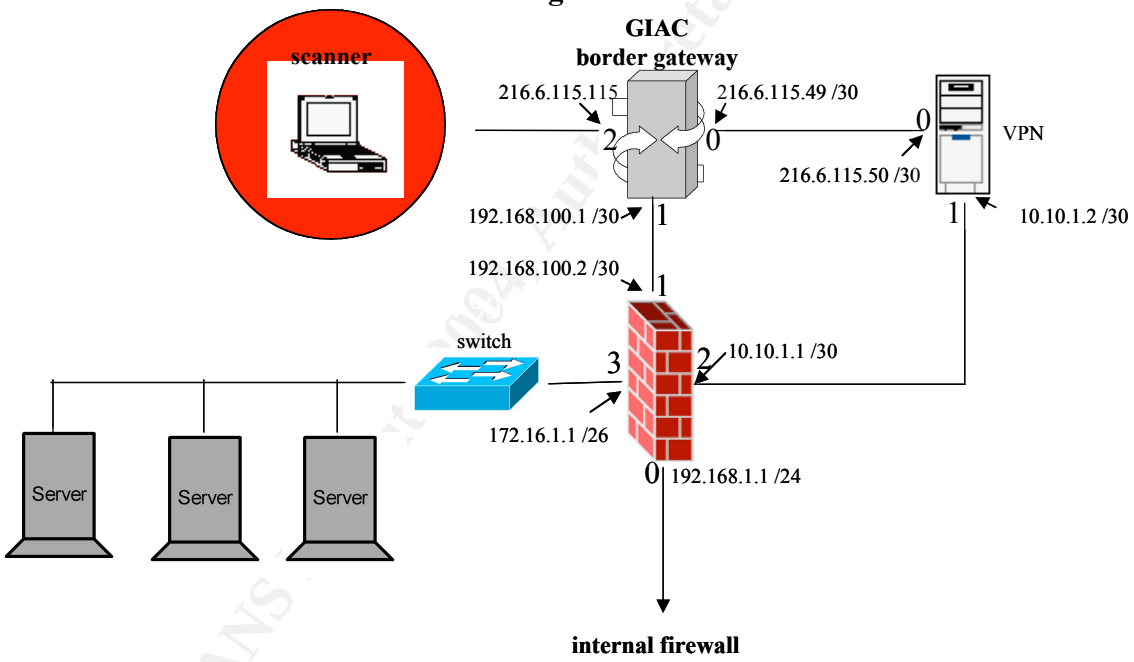
Can the internal DNS server contact other DNS server on the internet?	ACCEPT at the Internal firewall, Primary firewall, and border gateway
Does any other kind of traffic get dropped?	DROP at the Internal firewall
If a scan is initiated from a host on the service network, does it get logged on the primary firewall?	LOG at the Internal firewall

Assessment and Results

In this section we provide the results of the Assessment conducted from the internet. The complete script is provided in Appendix A.

Assessment from the internet.

Figure 6



Key points concerning the set-up:

- The script to run from the scanner side is script_ValidateRule_fromInternet.sh (see Appendix A)
- We run the script from a laptop directly connected to the external interface of the border gateway. The IP address assigned to the laptop is 216.6.115.113, with default gateway being 216.6.115.115 (our border gateway).
- The PPPoE interface is turned down at the border gateway, and the external interface becomes eth2 (instead of ppp0). This implies a modification in the NetFilter script on the border gateway.

- Tcpdump is listening on all three interfaces of the border gateway, on all four interfaces of the primary firewall, and on the two interfaces of the VPN gateway.

The script is broken down into 8 sections:

- TEST 1 DROP packets with private/reserved src/dst IP at the border gateway
  - "TEST 1a" spoof reserve/multicast: DROP
  - "TEST 1b" spoof private: DROP
  - "TEST 1c" spoof gw IP: DROP
  - "TEST 1d" spoof vpn IP: DROP
  - "TEST 1e" reserved destination IP: DROP
- TEST 2 ALLOW access to GIAC services
  - "TEST 2a" SYN to TCP/80: ALLOW & FORWARD
  - "TEST 2b" SYN to TCP/443: ALLOW & FORWARD
  - "TEST 2c" SYN to TCP/25: ALLOW & FORWARD
  - "TEST 2d" UDP to UDP/53: ALLOW & FORWARD
  - "TEST 2e" SYN to TCP/53: LOG, ALLOW & FORWARD
  - "TEST 2f" a complete TCP handshake on an accepted port
- TEST 3 ALLOW VPN access to teleworkers
  - "TEST 3a" IKE: LOG and ALLOW
  - "TEST 3b" ESP: LOG and ALLOW
- TEST 4 DROP at border gateway traffic for non-offered and unused services
  - First, TCP: any destination port other than 20, 21, 22, 25, 53, 80, 443, 554
    - "TEST 4a" SYN to TCP port < 1024: DROP
    - "TEST 4b" SYN to TCP port > 1024: DROP
    - "TEST 4c" SYN to TCP/113 (ident) DROP and RESET
  - Then UDP: any destination port other than 53, 500
    - "TEST 4d" UDP port < 1024: DROP
    - "TEST 4e" UDP port > 1024: DROP
  - Finally, UDP with destination port 500, but with source port other than 500
    - "TEST 4f" Not IKE: DROP
- TEST 5 DROP at the primary firewall traffic scan packet
  - "TEST 5a" ACK to TCP/80 : LOG, and DROP
  - "TEST 5b" SYN/ACK to TCP/80: LOG, and DROP
- TEST 6 ALLOW IN icmp error messages
  - "TEST 6a" Destination Unreachable: ALLOW
  - "TEST 6b" Time exceeded: ALLOW
- TEST 7 ALLOW IN icmp echo request/reply
  - "TEST 7a" Echo Reply: ALLOW
  - "TEST 7b" Echo Request: ALLOW
- TEST 8 DROP unaccepted icmp
  - "TEST 8a" Redirect: LOG, and DROP
  - "TEST 8b" Mask address: DROP

The traffic captured during the assessment appears in the tables below. The format is of ethereal (print summary). We have added a column entitled "TAG". These tags refer to the subtest for which the packet was transmitted.

Note that the "info" column of ethereal is of little use here since ethereal got confused by the crafted packets. For instance packets 3 to 8 of Table 1 are SYN packets directed at port 80. They were not part of any established connection, but had data in them (the reference tag for easy trace back).

- Table 1 shows the traffic seen on the link between the scanner and the border gateway,
- Table 2 shows the traffic between the border gateway and the primary firewall,
- Table 3 shows the traffic between the primary firewall and the service network
- Table 4 shows the traffic between the border gateway and the VPN gateway.

No traffic was seen between the VPN gateway and the primary firewall, nor between the primary firewall and the internal firewall. This is simply because the script was not written to send well-formed requests to the targeted relays, but rather to see if the traffic would reach these relays. The IP and TCP headers were crafted to appear as regular traffic, but the application layer was not.

Here is how to validate the results:

TEST1

We see that the packets with the tag TEST1 appear only in table 1, this means that none of the traffic with a reserved IP address as the source or the destination got through. They were dropped at the border gateway. The packets for TEST1a to TEST1d were SYN packets directed at the border gateway on port 80. TEST1e was a SYN packet directed at on port 80 of private address (the one of our web server). If the sources and destinations had been legitimate, these packets would have gone through since they targeted a port on which we offer a service. Therefore, TEST1 validates our DROP rule on the border gateway for such Traffic.

TEST2

Packets transmitted during TEST2 are all allowed traffic. TEST2a, TEST2b, TEST2c are SYN packets directed at the IP address of the border gateway and on ports supported at GIAC (TCP/80, TCP/443, TCP/25 respectively). TEST2d is a UDP packet to port 53, also allowed according to the security policy. TEST2e is a TCP SYN packet to port 53 with the source IP of our ISP DNS Server. Finally, TEST2e is a complete three-way handshake to port TCP/80. The intent was to verify that all packets associated with a given connection could pass through the stateful rules of the Primary firewall and the stateless rule of the border gateway. All packets associated with TEST2 can reach their destination, in either direction (see Table 1, Table 2 and Table 3). TEST2 is thereby validated.

TEST3

Packets transmitted during TEST3 are allowed traffic directed at the VPN gateway. TEST3a is an IKE packet and TEST3b is an ESP packet. They both show up on Table1

and Table 4. Moreover, the traffic showed up in the /var/log/messages file of the border gateway, named luke, as shown below:

```
Apr  5 15:49:05 luke kernel: IKE-in: IN=eth2 OUT=eth0 SRC=216.6.115.65
DST=216.6.115.50 LEN=38 TOS=0x00 PREC=0x00 TTL=63 ID=62724 PROTO=UDP
SPT=500 DPT=500 LEN=18
Apr  5 15:49:07 luke kernel: ESP-in: IN=eth2 OUT=eth0 SRC=216.6.115.65
DST=216.6.115.50 LEN=80 TOS=0x00 PREC=0x00 TTL=63 ID=64512 PROTO=ESP
SPI=0x54455354
```

This validate TEST3.

Note that although the traffic is not shown here, we did ensure that the VPN rules would allow the traffic, by initiating a real tunnel from Teleworker1 workstation. A proper ipsec connection cannot be simulated with hping2.

TEST4

TEST4 ensures that the default DROP policy is applied if traffic is directed at ports for which GIAC does not provide any service. Arbitrary TCP and UDP ports were targeted in TEST4a, TEST4b, TEST4d and TEST4e. TEST4c was a TCP SYN targeted to port 113, this was to verify that the REJECT iptables rule was effective. The TCP RST packet did show up in Table1. TEST4f was to test that UDP packets destined to port 500 must also come with a source port of 500. None of the packets of TEST4 get passed the border gateway. TEST4 is validated.

TEST5

Packets of TEST5 were meant to test the stateful filtering rules of the primary firewall, and to test the logging of packets associated with certain scan or reconnaissance techniques. TEST5a is a ACK to port 80, and TEST5b a SYN/ACK to port 80 is a simple FIN to port 80. These packets get through the border gateway, but are DROP at the primary firewall. This can be seen by observing that none of these packets appear in Table 3, while they do appear in Table 1 and 2.

The “ack scan” message appears in the logs (/var/log/messages) of the primary firewall, named lea:

```
Apr  5 15:49:22 lea kernel: ack scan: IN=eth1 OUT=eth3
SRC=192.168.100.1 DST=172.16.1.8 LEN=40 TOS=0x00 PREC=0x00 TTL=62
ID=42539 PROTO=TCP SPT=2390 DPT=80 WINDOW=512 RES=0x00 ACK URGP=0
```

TEST5 is validated.

TEST6

TEST6 verifies the ICMP error messages get passed the border gateway, to test this, an ICMP Unreachable message, and a Time Exceeded message were sent to the address of the VPN gateway (routable). Both packets show up in Table 1 and 4. TEST6 is validated.

TEST7

TEST7 verifies that ICMP Echo Requests and Reply passes can reach the VPN gateway. We sent a Request and a Reply, of course, the Reply received no answer, but the request did. These packets show up in Table 1 and Table 4. TEST7 is validated.

TEST8

TEST8 verifies that undesired ICMP packets get dropped at the border gateway. These packets only appear in Table1. Therefore they got dropped. Note that no message appeared in the LOG, this is probably due to the fact that the iptables script turns off the kernel variable "ACCEPT_REDIRECTS". We could rerun the test to see, we won't… the important thing is the DROP. TEST8 is validated.

Table 1

Assessment: Validate rules from the internet
Link: between the scanner and the border gateway

No.	Time	Source	Destination	Protocol	Info	TAG
1	0	00:0c:29:62:dc:be	ff:ff:ff:ff:ff:ff	ARP	Who has 216.6.115.115? Tell 216.6.115.113	
2	0.000049	00:01:03:bf:c8:ea	00:0c:29:62:dc:be	ARP	216.6.115.115 is at 00:01:03:bf:c8:ea	
3	0.001878	224.0.1.22	216.6.115.115	HTTP	Continuation	TEST 1a
4	2.010935	10.10.10.10	216.6.115.115	HTTP	Continuation	TEST 1b
5	4.020654	216.6.115.115	216.6.115.115	HTTP	Continuation	TEST 1c
6	6.037825	216.6.115.50	216.6.115.115	HTTP	Continuation	TEST 1d
7	8.050457	216.6.115.113	192.168.1.1	HTTP	Continuation	TEST 1e
8	10.06144	216.6.115.113	216.6.115.115	HTTP	Continuation	TEST 2a
9	10.06316	216.6.115.115	216.6.115.113	TCP	80 > 1120 [SYN, ACK] Seq=521459709 Ack=1459711276 Win=5840 Len=0	TEST 2a
10	10.06755	216.6.115.113	216.6.115.115	TCP	1120 > 80 [RST] Seq=1459711276 Ack=0 Win=0 Len=0	TEST 2a
11	10.07973	216.6.115.113	216.6.115.115	SSL	Continuation Data	TEST2b
12	10.08557	216.6.115.115	216.6.115.113	TCP	443 > 1120 [SYN, ACK] Seq=516997555 Ack=1459711276 Win=5840 Len=0	TEST2b
13	10.08619	216.6.115.113	216.6.115.115	TCP	1120 > 443 [RST] Seq=1459711276 Ack=0 Win=0 Len=0	TEST2b
14	10.09754	216.6.115.113	216.6.115.115	SMTP	Command: TEST 2XXXX	TEST2c
15	10.09866	216.6.115.115	216.6.115.113	TCP	25 > 1120 [SYN, ACK] Seq=515346885 Ack=1459711276 Win=5840 Len=0	TEST2c
16	10.09904	216.6.115.113	216.6.115.115	TCP	1120 > 25 [RST] Seq=1459711276 Ack=0 Win=0 Len=0	TEST2c
17	10.13902	216.6.115.113	216.6.115.115	DNS	Unknown operation (10)[Malformed Packet]	TEST2d
18	12.15549	216.6.44.5	216.6.115.115	DNS	Zone change notification[Short Frame]	TEST2e
19	14.23404	216.6.115.113	216.6.115.115	TCP	1053 > 80 [SYN] Seq=160017248 Ack=0 Win=32120 Len=0	TEST2f
20	14.23485	216.6.115.115	216.6.115.113	TCP	80 > 1053 [SYN, ACK] Seq=519852692 Ack=160017249 Win=5792 Len=0	TEST2f
21	14.23704	216.6.115.113	216.6.115.115	TCP	1053 > 80 [ACK] Seq=160017249 Ack=519852693 Win=32120 Len=0	TEST2f
22	14.23983	216.6.115.113	216.6.115.115	TCP	1053 > 80 [FIN, ACK] Seq=160017249 Ack=519852693 Win=32120 Len=0	TEST2f
23	14.24561	216.6.115.115	216.6.115.113	TCP	80 > 1053 [FIN, ACK] Seq=519852693 Ack=160017250 Win=5792 Len=0	TEST2f
24	14.247	216.6.115.113	216.6.115.115	TCP	1053 > 80 [ACK] Seq=160017250 Ack=519852694 Win=32120 Len=0	TEST2f
25	14.25514	216.6.115.65	216.6.115.50	ISAKMP	[Malformed Packet]	TEST3a
26	15.06128	00:01:03:bf:c8:ea	00:0c:29:62:dc:be	ARP	Who has 216.6.115.113? Tell 216.6.115.115	---------
27	15.06212	00:0c:29:62:dc:be	00:01:03:bf:c8:ea	ARP	216.6.115.113 is at 00:0c:29:62:dc:be	---------

28	16.32156	216.6.115.65	216.6.115.50	ESP	ESP (SPI=0x54455354)	TEST3b
29	18.34026	216.6.115.113	216.6.115.115	TCP	2901 > 7 [SYN] Seq=1053100400 Ack=860455640 Win=512 Len=10	TEST4a
30	20.36868	216.6.115.113	216.6.115.115	X11	Requests: AllocColor[Unreassembled Packet]	TEST4b
31	22.33194	216.6.115.113	216.6.115.115	TCP	2893 > 113 [SYN] Seq=590233014 Ack=291387090 Win=512 Len=0	TEST4c
32	22.77194	216.6.115.115	216.6.115.113	TCP	113 > 2893 [RST, ACK] Seq=0 Ack=590233015 Win=0 Len=0	TEST4c
33	24.35541	216.6.115.113	216.6.115.115	NBNS	Unknown operation (10)[Malformed Packet]	TEST4d
34	26.37821	216.6.115.113	216.6.115.115	UDP	Source port: 1311 Destination port: 6000	TEST4e
35	28.40169	216.6.115.113	216.6.115.115	ISAKMP	[Malformed Packet]	TEST4f
36	30.37168	216.6.115.113	216.6.115.115	HTTP	Continuation	TEST5a
37	30.38471	216.6.115.113	216.6.115.115	HTTP	Continuation	TEST5b
38	40.50037	216.6.115.113	216.6.115.50	ICMP	Destination unreachable	TEST6a
39	42.52239	216.6.115.113	216.6.115.50	ICMP	Time-to-live exceeded	TEST6b
40	44.499	216.6.115.113	216.6.115.50	ICMP	Echo (ping) reply	TEST7a
41	46.52219	216.6.115.113	216.6.115.50	ICMP	Echo (ping) request	TEST7b
42	46.54461	216.6.115.50	216.6.115.113	ICMP	Echo (ping) reply	TEST7b
43	46.56683	216.6.115.113	216.6.115.50	ICMP	Redirect	TEST8a
44	48.61553	216.6.115.113	216.6.115.50	ICMP	Address mask request	TEST8b

Table 2

Assessment: Validate rules from the internet
Link: between the border gateway and the primary firewall)

No.	Time	Source	Destination	Protocol	Info	TAG
1	0	00:01:03:be:ed:bf	ff:ff:ff:ff:ff:ff	ARP	Who has 192.168.100.2? Tell 192.168.100.1	---------
2	0.000117	00:10:5a:9e:1b:6c	00:01:03:be:ed:bf	ARP	192.168.100.2 is at 00:10:5a:9e:1b:6c	--------
3	0.000132	192.168.100.1	172.16.1.8	HTTP	Continuation	TEST2a
4	0.001544	172.16.1.8	192.168.100.1	TCP	80 > 1120 [SYN, ACK] Seq=521459709 Ack=1459711276 Win=5840 Len=0	TEST2a
5	0.005969	192.168.100.1	172.16.1.8	TCP	1120 > 80 [RST] Seq=1459711276 Ack=0 Win=0 Len=0	TEST2a
6	0.018258	192.168.100.1	172.16.1.8	SSL	Continuation Data	TEST2b
7	0.023955	172.16.1.8	192.168.100.1	TCP	443 > 1120 [SYN, ACK] Seq=516997555 Ack=1459711276 Win=5840 Len=0	TEST2b
8	0.024611	192.168.100.1	172.16.1.8	TCP	1120 > 443 [RST] Seq=1459711276 Ack=0 Win=0 Len=0	TEST2b
9	0.035995	192.168.100.1	172.16.1.25	SMTP	Command: TEST 2XXXX	TEST2c
10	0.03705	172.16.1.25	192.168.100.1	TCP	25 > 1120 [SYN, ACK] Seq=515346885 Ack=1459711276 Win=5840 Len=0	TEST2c
11	0.037453	192.168.100.1	172.16.1.25	TCP	1120 > 25 [RST] Seq=1459711276 Ack=0 Win=0 Len=0	TEST2c
12	0.07757	192.168.100.1	172.16.1.53	DNS	Unknown operation (10)[Malformed Packet]	TEST2d
13	2.094078	192.168.100.1	172.16.1.53	DNS	Zone change notification[Short Frame]	TEST2e
14	4.172579	192.168.100.1	172.16.1.8	TCP	1053 > 80 [SYN] Seq=160017248 Ack=0 Win=32120 Len=0	TEST2f
15	4.173237	172.16.1.8	192.168.100.1	TCP	80 > 1053 [SYN, ACK] Seq=519852692 Ack=160017249 Win=5792 Len=0	TEST2f

16	4.175463	192.168.100.1	172.16.1.8	TCP	1053 > 80 [ACK] Seq=160017249 Ack=519852693 Win=32120 Len=0	TEST2f
17	4.178249	192.168.100.1	172.16.1.8	TCP	1053 > 80 [FIN, ACK] Seq=160017249 Ack=519852693 Win=32120 Len=0	TEST2f
18	4.184003	172.16.1.8	192.168.100.1	TCP	80 > 1053 [FIN, ACK] Seq=519852693 Ack=160017250 Win=5792 Len=0	TEST2f
19	4.185419	192.168.100.1	172.16.1.8	TCP	1053 > 80 [ACK] Seq=160017250 Ack=519852694 Win=32120 Len=0	TEST2f
20	4.997206	00:10:5a:9e:1b:6c	00:01:03:be:ed:bf	ARP	Who has 192.168.100.1? Tell 192.168.100.2	---------
21	4.99725	00:01:03:be:ed:bf	00:10:5a:9e:1b:6c	ARP	192.168.100.1 is at 00:01:03:be:ed:bf	---------
22	20.310217	192.168.100.1	172.16.1.8	HTTP	Continuation	TEST5a
23	20.32315	192.168.100.1	172.16.1.8	HTTP	Continuation	TEST5b

Table 3

Assessment: Validate from internet
Link: between primary firewall and the service network

No.	Time	Source	Destination	Protocol	Info	TAG
1	0	00:50:fc:9d:ec:cd	ff:ff:ff:ff:ff:ff	ARP	Who has 172.16.1.8? Tell 172.16.1.1	---------
2	0.000501	00:0c:29:12:68:c2	00:50:fc:9d:ec:cd	ARP	172.16.1.8 is at 00:0c:29:12:68:c2	--------
3	0.00051	192.168.100.1	172.16.1.8	HTTP	Continuation	TEST2a
4	0.001211	172.16.1.8	192.168.100.1	TCP	80 > 1120 [SYN, ACK] Seq=521459709 Ack=1459711276 Win=5840 Len=0	TEST2a
5	0.005798	192.168.100.1	172.16.1.8	TCP	1120 > 80 [RST] Seq=1459711276 Ack=0 Win=0 Len=0	TEST2a
6	0.018043	192.168.100.1	172.16.1.8	SSL	Continuation Data	TEST2b
7	0.023649	172.16.1.8	192.168.100.1	TCP	443 > 1120 [SYN, ACK] Seq=516997555 Ack=1459711276 Win=5840 Len=0	TEST2b
8	0.024462	192.168.100.1	172.16.1.8	TCP	1120 > 443 [RST] Seq=1459711276 Ack=0 Win=0 Len=0	TEST2b
9	0.035809	00:50:fc:9d:ec:cd	ff:ff:ff:ff:ff:ff	ARP	Who has 172.16.1.25? Tell 172.16.1.1	---------
10	0.036262	00:0c:29:12:68:c2	00:50:fc:9d:ec:cd	ARP	172.16.1.25 is at 00:0c:29:12:68:c2	---------
11	0.03627	192.168.100.1	172.16.1.25	SMTP	Command: TEST 2XXXX	TEST2c
12	0.036718	172.16.1.25	192.168.100.1	TCP	25 > 1120 [SYN, ACK] Seq=515346885 Ack=1459711276 Win=5840 Len=0	TEST2c
13	0.037287	192.168.100.1	172.16.1.25	TCP	1120 > 25 [RST] Seq=1459711276 Ack=0 Win=0 Len=0	TEST2c
14	0.077414	00:50:fc:9d:ec:cd	ff:ff:ff:ff:ff:ff	ARP	Who has 172.16.1.53? Tell 172.16.1.1	---------
15	0.077898	00:0c:29:12:68:c2	00:50:fc:9d:ec:cd	ARP	172.16.1.53 is at 00:0c:29:12:68:c2	---------
16	0.077912	192.168.100.1	172.16.1.53	DNS	Unknown operation (10)[Malformed Packet]	TEST2d
17	0.094076	192.168.100.1	172.16.1.53	DNS	Zone change notification[Short Frame]	TEST2e
18	4.17142	192.168.100.1	172.16.1.8	TCP	1053 > 80 [SYN] Seq=160017248 Ack=0 Win=32120 Len=0	TEST2f
19	4.171913	172.16.1.8	192.168.100.1	TCP	80 > 1053 [SYN, ACK] Seq=519852692 Ack=160017249 Win=5792 Len=0	TEST2f
20	4.174235	192.168.100.1	172.16.1.8	TCP	1053 > 80 [ACK] Seq=160017249 Ack=519852693 Win=32120 Len=0	TEST2f
21	4.177035	192.168.100.1	172.16.1.8	TCP	1053 > 80 [FIN, ACK] Seq=160017249 Ack=519852693 Win=32120 Len=0	TEST2f
22	4.182611	172.16.1.8	192.168.100.1	TCP	80 > 1053 [FIN, ACK] Seq=519852693 Ack=160017250 Win=5792 Len=0	TEST2f

23	4.184193	192.168.100.1	172.16.1.8	TCP	1053 > 80 [ACK] Seq=160017250 Ack=519852694 Win=32120 Len=0	TEST2f

Table 4

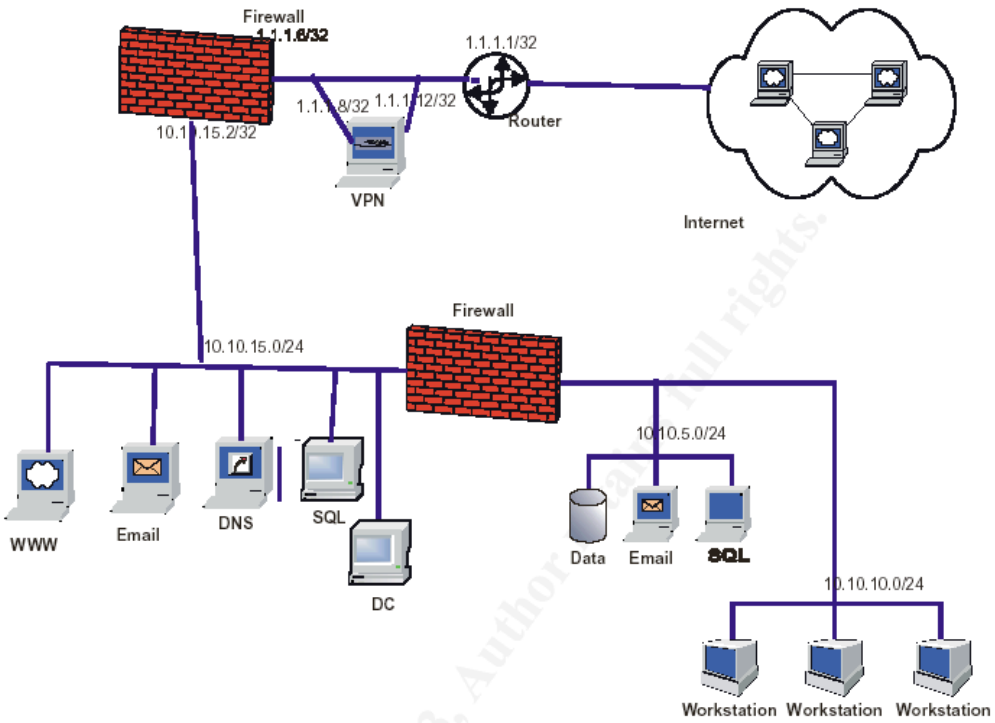
Assessment: Validate rules from the internet
Link : between the border gateway and the VPN gateway

No.	Time	Source	Destination	Protocol	Info	TAG
1	0	00:04:76:b6:df:e9	ff:ff:ff:ff:ff:ff	ARP	Who has 216.6.115.50? Tell 216.6.115.49	-------
2	0.005501	00:0c:29:f9:b6:2a	00:04:76:b6:df:e9	ARP	216.6.115.50 is at 00:0c:29:f9:b6:2a	------
3	0.005519	216.6.115.65	216.6.115.50	ISAKMP	[Malformed Packet]	TEST3a
4	2.066481	216.6.115.65	216.6.115.50	ESP	ESP (SPI=0x54455354)	TEST3b
5	26.245215	216.6.115.113	216.6.115.50	ICMP	Destination unreachable	TEST6a
6	28.267206	216.6.115.113	216.6.115.50	ICMP	Time-to-live exceeded	TEST6b
7	30.243823	216.6.115.113	216.6.115.50	ICMP	Echo (ping) reply	TEST7a
8	32.267101	216.6.115.113	216.6.115.50	ICMP	Echo (ping) request	TEST7b
9	32.289293	216.6.115.50	216.6.115.113	ICMP	Echo (ping) reply	TEST7b
10	36.998161	00:0c:29:f9:b6:2a	00:04:76:b6:df:e9	ARP	Who has 216.6.115.49? Tell 216.6.115.50	----------
11	36.998206	00:04:76:b6:df:e9	00:0c:29:f9:b6:2a	ARP	216.6.115.49 is at 00:04:76:b6:df:e9	----------

Assignment 4 – Design Under Fire

The network design under fire here is the one of Lesa Ludwig. The practical can be found at http://www.giac.org/practical/GCFW/Lesa_Ludwig_GCFW.pdf

The network is depicted in the figure below, drawn by Lesa.



Attack against a firewall

The attack we choose here does not target a particular firewall product. We will simply try to flood the perimeter with non-legitimate packets, hoping that the border router is stateless and thus let these packets go through, and that the primary firewall has logging capability.

What we try to accomplish here is to overwhelm the firewall with packets that get logged. If we succeed and that the firewall gets disabled but the IP forwarding capability remains, we could then possibly defeat Lesa's security policy and explore the network.

One of the tools we'll use here is hping2. The packets will have abnormal settings. Hping2 allows us to craft most fields of IP, ICMP,UDP, and TCP. By modifying the offset field of the TCP header for instance, and adding data to the TCP packets, we can craft any TCP options we wish. We will also play with TCP and IP flags.

Packets that typically get logged are those of nmap and other scanning tools. We can craft similar packets, or better, use these tools all together.

We do not have a very powerful system from where to launch the flood, however, it may be sufficient to keep flooding for a long period of time for the attack to work.

Obviously, we do not need to get responses back for all packets we send. We can spoof the majority of the packets, and only send very few probes from time to time to see if packets that were blocked before gets through now.

Mitigation
Lesa's border router drops many known spoof packets, so if we had spoofed reserved or private addresses, the attack would not have worked. Moreover, Lesa's primary firewall has rate-limit capability for logging, therefore the attack chosen had very little chance to succeed.

Distributed Denial of Service attack

This attack will used compromised hosts to attempt a denial of service attack on Lesa's border network. This is attack is twofold:

- Compromise 50 DSL/Cable computers;
- Launch from these compromised systems an attack that may overwhelm the systems on the perimeter network.

Compromise 50 DSL/Cable computer
An obvious choice for a target OS is Microsoft Windows systems. It is the preferred OS for many home users. More over, home users tend to delay installing patches. There are still many Windows boxes vulnerable to the DCOM exploit (http://packetstormsecurity.nl/0307-exploits/dcom.c). The tool will allow us to get a shell on the compromised hosts.

Windows boxes are easy to identify remotely. We will use hping2 to send ICMP echo requests to about 500 systems at a slow rate over 24 hours (~1 ping every two minutes). All traffic will be saved to a traffic trace.

Because of the slow rate and the fact that ICMP packets are common, our scanning technique should not trigger many IDS systems.

The important characteristic of our crafted ICMP echo requests is the non-zero value of the ICMP code field. The idea comes from a tool named Xprobe (www.sys-security.com). Windows systems are apparently the only hosts that overwrite the ICMP code in their response to set it to 0 (the normal value). All other OSes echo the value they received.

It is then a simple matter of defining tcpdump filters to identify IP addresses of hosts who responded with a value of zero in their ICMP echo Reply. My guess is that from 500 IP addresses scanned on a range that belong to a popular ISP, there are chances that we received close to 200 responses, of which 150 would be Windows boxes. Note that while we do not know which version of Windows they are running, chances are they are mostly Windows 2000 and Windows XP systems. Assuming half of those have not been patched, our DCOM exploit may work against more than 50 hosts.

Launch the Attack
Once we get a shell on these hosts, we can install an exploit that can launch a denial of service attack against our target. One choice is to use a Flooder tool, of any kind, to direct traffic at the border gateway of Lesa's network.

Here we choose to assume that like most networks, the border gateway is protected by a Cisco router. A Cisco advisory posted in September 2003 reveals Cisco IOS contains a denial of service vulnerability that can be triggered by a series of non-standard IP protocol packets (not TCP/UDP/ICMP) having TTL=0 or TTL=1, directed to one of the router's interface IP addresses. A successful attack causes the router to stop processing traffic on the targeted interface.

Affected Products are:
Cisco IOS 11.x
Cisco IOS 12.0 through 12.2
The Cisco Advisory can be found at http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml

The interesting thing about exploiting this vulnerability is that only a few packets are required to fool the system into believing that the queue is full. An exploit code, written in C, can be downloaded from
http://archives.neohapsis.com/archives/fulldisclosure/2003-q3/0580.html

The name of the file is shadowchode.tar.gz

The exploit is straightforward to use. We can generate a binary executable for Windows and launch it from our compromised hosts.

Very few traffic will be generated from each of the compromised host (4 packets I believe). This makes it quite attractive since it is a bit stealthy.

Lesa's access lists do not deny the traffic produced by the exploit, the IOS version is running is 12.2, which could be vulnerable. We cannot know for sure since patches were available in October 2003, date at which Lesa's report was produced.

Mitigation
Patches to protect systems from this attack are available from Cisco. The Advisory provides some workarounds to mitigate the attack against unpatched systems. The acl to add are the following:
access-list 101 deny 53 any any
access-list 101 deny 55 any any
access-list 101 deny 77 any any
access-list 101 deny 103 any any

Attack to compromise an internal machine

The path we chose in order to reach an internal system is through the mail relay and mail server. The goal is to send e-mails containing a malicious custom written windows root kit to valid e-mail addresses of internal hosts. Finding valid e-mails is not too difficult; getting people to open attachment is not so hard either. The primary challenge is to pass through the virus scanner.

A search through the news group did not provide much help for finding valid e-mail addresses belonging to Lesa's domain. Apparently, the users are not big fans of chat rooms neither. What we happen to know however is where the Office is located. We will post an ad near the entrance of the building to advertise extremely good deals on a couple of accessories: kitchen tools, gardening tools, books, and pets. The only point of contact will be an e-mail address from Yahoo. Assuming we get at least one response, not only have we found a valid e-mail address, but we also have an idea of certain things of interest to the victim. Moreover, Lesa does not mention anything about Banner stripping; therefore, it is possible that the header of the received e-mail contains information about the user's system, the mail client, and possibly the mail server.

We would respond politely to the victime saying that the item in question is already sold. A week later, we would send the target an e-mail, using the same yahoo address, and attaching the rootkit in a tarball file, entiltled "For sale". The root kit would install itself upon opening of the archive.

The challenge is to pass through the virus scanner undetected. Since our rootkit is custom written, the attack has good chance to get through.

According to Lesa's document, the external email server runs Mail Sweeper 4.3 for SMTP to filter unwanted email attachments.

A search through SecurityFocus's website informs us that a vulnerability in Mail Sweeper version 4.3.10 and prior, may cause the software to fail detecting malicious zip archives. The bugtraq id is 8982. More information can be found on this vulnerability at (http://www.securityfocus.com/bid/8982)

The attack may have worked.

Mitigation

Companies must remind the users not to open attachment in e-mails unless they know in advance what they contain. This is very difficult to enforce since humans tend to be curious by nature. Stripping off information from e-mail headers is however more easy to control. It reduces the risk of divulgating information about the configuration of systems attached to the network.

List of References

OpenBSD
http://www.openbsd.org

Progeny
http://www.progeny.com/products/transition

IPFilter
http://www.ipfilter.org

Roaring pPenguin
http://www.roaringpenguin.com

FreeS/WAN v 2.05
http://www.freeswan.org/

Squid 2.5
http://www.squid-cache.org/

DCC
http://www.rhyolite.com/anti-spam/dcc

Vipul's Razor
http://razor.sourceforge.net

hping2
(http://www.hping.org)

Qmail
http://www.qmail.org/top.html

BIND 9.2.3
http://www.isc.org

MySQL 4.0
http://www.mysql.com

Snort v2.1.2
http://www.snort.org

FreeS/WAN
www.freeswan.org

Finisar
http://www.finisar.com

Bastille-Linux
http://www.bastille-linux.org

Hardening Linux Systems
http://www.linux-mag.com/2002-09/guru_01.html

Red Hat Linux Security Guide
http://www.redhat.com/docs/manuals/linux/RHL-9 Manual/security-guide/

Linux Security HOWTO
http://tldp.org/HOWTO/Security-HOWTO/index.html

Netfilter
http://www.netfilter.org/

The Linux Kernel: Compiling Options
http://www.linuxvoodoo.com/resources/howtos/kernel/kernel1.php

FrozenTux
http://iptables-tutorial.frozentux.net/iptables-tutorial.html

Lesa Ludwig's practical
http://www.giac.org/practical/GCFW/Lesa_Ludwig_GCFW.pdf

Cisco Advisory,
http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml

Security Focus
http://www.securityfocus.com/bid/8982

Appendix A: Script to assess the firewall rules from the internet

```
#!/bin/sh
# Script using hping2 to test policy rules
# name: script_ValidateRules_fromInternet.sh
# requires two argument:
#      $1: the IP addresse of border gateway
#      $2: the IP addresse of the vpn gateway

# This penetration tests is design to test the NetFilter rules from an outsider's point of view
# The script should be run from a system located outside of GIAC enterprises network.

# To validate the policy rules,
# 1) monitor the traffic on all interfaces of the border gateway;
# 2) monitor the traffic on all interfaces of the primary firewall;
# 3) monitor the traffic on all interfaces of the internal firewall;
# 5) Verify logs of all three systems

# !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
# NOTE: This script is not a complete assessment; the objective is to test rules of gateways
# and firewalls.
#
# There are security policies that are implemented on the servers themselves (such as policies
# regarding DNS zone transfers, access restriction to http files and directories, etc.).
#
# For each systems located at GIAC enterprises, the system admin wrote a tutorial on how the
# services running were configured and secured.  The person conducting the audit should read
# these tutorials carefully and try to defeat (circumvent) the security policies implemented.
#
# The validation of system security, and a vulnerability assessment are out of the scope
# of this script.
# !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


#################################################################
# 1) DROP packets with private or reserved src/dst IP at the border gateway
#################################################################
echo "       TEST 1 DROP spoof source at the border gateway "
echo "Gotchas: Since we spoof, hping may not see the responses"
echo "         do not rely on hping output, verify with the traffic traces..."
#
hping2 -c 1 -d 10 --sign "TEST 1a"  --syn --destport 80 --spoof 224.0.1.22  $1   # spoof reserve/multicast: DROP
hping2 -c 1 -d 10 --sign "TEST 1b"  --syn --destport 80 --spoof 10.10.10.10 $1   # spoof private: DROP
hping2 -c 1 -d 10 --sign "TEST 1c"  --syn --destport 80 --spoof $1          $1   # spoof own IP:  DROP
hping2 -c 1 -d 10 --sign "TEST 1d"  --syn --destport 80 --spoof $2          $1   # spoof vpn IP:  DROP
hping2 -c 1 -d 10 --sign "TEST 1e"  --syn --destport 80 192.168.1.1   # reserved destination: DROP

echo "       TEST 1 done!"
echo


#############################################
# 2) ALLOW access to GIAC services from the internet.
#############################################
echo "       TEST 2 ALLOW access to GIAC services"

hping2 -c 1  -d 10 --sign "TEST 2a" --syn --destport 80    $1   # TCP/80:  ALLOW & FORWARD
hping2 -c 1  -d 10 --sign "TEST 2b" --syn --destport 443   $1   # TCP/443: ALLOW & FORWARD
hping2 -c 1  -d 10 --sign "TEST 2c" --syn --destport 25    $1   # TCP/25:  ALLOW & FORWARD
```

```
hping2 -c 1  -d 10 --sign "TEST 2d" --udp --destport 53    $1   # UDP/53:  ALLOW & FORWARD
hping2 -c 1  -d 10 --sign "TEST 2e" --syn --destport 53 --spoof 216.6.44.5 $1 # TCP/53:LOG,ALLOW&FORWARD

# TEST 2f is a complete tcp  handshake on an accepted port
# we use telnet client since hping2 resets the connection before the handshake is completed
telnet $1 80 <<END
^]
quit
END
echo "       TEST 2 done!"
echo

##################################
# 3) ALLOW VPN access to teleworkers
##################################
echo "       TEST 3 ALLOW VPN access to teleworkers"
echo "gotchas: do not expect response, check traffic trace at GIAC site"
# We spoof the IP of one of our teleworker (216.6.115.65)

# IKE: LOG and ALLOW
hping2 -c 1 -d 10 --sign "TEST 3a" --udp --baseport 500 --destport 500  --spoof 216.6.115.65  $2

# ESP: LOG and ALLOW
hping2 -c 1 -d 10 --sign "TEST 3b" --rawip --ipproto 50 --data 60       --spoof 216.6.115.65  $2
echo "       TEST 3 done!"
echo

#####################################################################
# 4) DROP at border gateway traffic for non-offered and unused services
#####################################################################
echo "       TEST 4 DROP at border gateway traffic for non-offered and unused services"

# TCP: pick any destination port other than 20, 21, 22, 25, 53, 80, 443, 554
hping2 -c 1  -d 10 --sign "TEST 4a" --syn --destport 7       $1   # TCP port < 1024: DROP
hping2 -c 1  -d 10 --sign "TEST 4b" --syn --destport 6000  $1   # TCP port > 1024: DROP

# TCP ident
hping2 -c 1  -d 10 --sign "TEST 4c" --syn --destport 113   $1   # DROP and RESET

# UDP: pick any destination port other than 53, 500
hping2 -c 1 -d 10 --sign "TEST 4d" --udp --destport 137       $1   # UDP port < 1024: DROP
hping2 -c 1 -d 10 --sign "TEST 4e" --udp --destport 6000      $1   # UDP port > 1024: DROP

# UDP: destination port 500, but with source port other than 500 (DROP)
hping2 -c 1 -d 10 --sign "TEST 4f" --udp --baseport 1025 --destport 500   $1 # Not IKE: DROP
echo "       TEST 4 done!"
echo

#####################################################################
# 5) DROP at the primary firewall traffic out of sync
#####################################################################
echo "       TEST 5 DROP at the primary firewall traffic out of sync"

hping2 -c 1  -d 10 --sign "TEST 5a" --ack       --destport 80    $1   # ACK:  DROP
hping2 -c 1  -d 10 --sign "TEST 5b" --syn --ack --destport 80    $1   # SYN/ACK:  DROP
echo "       TEST 5 done!"
echo


##################################
# 6) ALLOW IN icmp error messages
##################################
```

```
echo "       TEST 6 ALLOW IN icmp error messages"

# Destination is the destination of the VPN so we see the traffic OFF the gateway
hping2 -c 1 -d 10 --sign "TEST 6a" --icmp --icmptype 3    $2   # Destination Unreachable: ALLOW
hping2 -c 1 -d 10 --sign "TEST 6b" --icmp --icmptype 11   $2   # Time exceeded: ALLOW
echo "       TEST 6 done!"
echo

##################################
# 7) ALLOW IN icmp echo request/reply
##################################
echo "       TEST 7 ALLOW IN icmp echo request/reply"

# Destination is the destination of the VPN so we see the traffic OFF the gateway
hping2 -c 1 -d 10 --sign "TEST 7a" --icmp --icmptype 0   $2   # Echo Reply: ALLOW
hping2 -c 1 -d 10 --sign "TEST 7b" --icmp --icmptype 8   $2   # Echo Request: ALLOW
echo "       TEST 7 done!"
echo

##################################
# 8) DROP undesired icmp
##################################
echo "       TEST 8 DROP undesired icmp"

# Destination is the destination of the VPN so we see the traffic OFF the gateway
hping2 -c 1 -d 10 --sign "TEST 8a" --icmp --icmptype 5   $2   # Redirect:     LOG, DROP
hping2 -c 1 -d 10 --sign "TEST 8b" --icmp-addr           $2   # Mask address: DROP
echo "       TEST 8 done!"
echo
```