Global Information Assurance Certification Paper

GCFW Practical Assignment

Sergei Ledovskij
11th of April 2001

Introduction

This document is my GCFW certification practical assignment. I have taken the corresponding SANS Firewall, Perimeter Protection and VPNs course in Sydney, Australia on February 12-15, 2001.

This practical consists of 4 assignments:

1. Security Architecture

   Define a security architecture, which contains filtering routers, firewalls, VPNs, secure remote access and internal firewalls for GIAC Enterprises, a company that sells online fortune cookies.

2. Security Policy

   Based on the security architecture, provide a security policy for the border router, primary firewall and the VPN device.

3. Security Architecture Audit

   Plan and implement a security assessment of the Primary Firewall described in Assignments 1 and 2.

4. Design Under Fire

   Select a network design from any previously posted GCFW practicals and design various attacks against it, trying to breach the security.

Table of Contents

Introduction....2
1.) Assignment I: The Design....4
1.1) Things to consider....4
1.1.1) Merger/Acquisition....4
1.1.2) Customers....4
1.1.3) Partners....4
1.1.4) Suppliers....4
1.1.5) Fortune Cookies....5
1.2) Network design....5
1.2.1) WWW server (customers)....6
1.2.2) WWW server (suppliers)....6
1.2.3) Border Router....6
1.2.4) Network Monitor Station....7
1.2.5) Outer Firewall....7
1.2.6) VPN Box....7
1.2.7) Main Switch....8
1.2.8) IDS....8
1.2.9) Inner Firewall....8
1.2.10) Database server....8
1.2.11) Maintenance LAN....8
1.2.12) Architecture notes....9
2.) Assignment II: The Policy....10
2.1) Border Router....10
2.2) Outer Firewall....11
2.3) VPN Box....14
3.) Assignment III: The Audit....18
3.1) Outer firewall audit....18
3.1.1) Scan from the Internet....18
3.1.2) Scan from the screened subnet....20
3.1.3) Scan from the internal network....22
3.1.4) Scan from the maintenance LAN....23
3.2) Denial Of Service attack....24
3.3) Recommendations for improvement....24
4.) Practical assignment (I-III) notes....25
5.) Assignment IV: Design Under Fire....26
5.1) An attack against the firewall....27
5.2) A Denial of Service attack....27
5.3) An attack plan to compromise an internal system....28
Attachment I: Firewall configuration....30
Attachment II: VPN Box configuration....32
Attachment III: Resources....33

1.) Assignment I: The Design

This part of the practical describes the design of the GIAC Enterprises' E-Business network environment.

This particular document does not take into account such means of securing the environment of GIAC Enterprises as individual host hardening, strong company security policy, etc. This document concentrates on designing and implementing secure network infrastructure for the company.

GIAC Enterprises is a growing E-Business company, which primary area of market is an online sale of fortune cookies. The company has multiple suppliers and partners and has just completed a merger/acquisition.

Many things have to be taken into account when designing a secure and well performing network environment. Certain balance has to be achieved between the ease of administration and the level of security.

In the GIAC Enterprises environment there are many influencing factors, such as customers, partners, suppliers and the just acquired company.

Each of these factors has to be looked into to see how it affects the network environment.

1.1) Things to consider

In this section of the document each of the factors influencing the design of the network environment is discussed.

1.1.1) Merger/Acquisition

Newly acquired company has to have a secure access to the production environment of GIAC Enterprises. VPN will be implemented between the networks of two companies. We will use an IPSEC solution to implement the VPN. This will require appropriate devices on both ends.

1.1.2) Customers

The customers of GIAC Enterprises will only need to have a secure WWW access to the company's web site in order to buy fortune cookies. They do not need to know about the structure underneath the web interface.

1.1.3) Partners

The partners of GIAC Enterprises will need direct access to the database of cookies, so that they can translate and resell them. This access will need to be secure, thus we can probably enforce the use of VPN technology in our security policy.

Other option would be to allow them to access the database using the web interface. For security reasons we are leaning towards the VPN solution.

1.1.4) Suppliers

The suppliers of cookies however cannot be forced and do not need to use VPN. Their job has much to do with creativity, and they cannot be bothered with complicated technical solutions. We will set up a separate WWW server for them.

The reason we want to use 2 different WWW servers (one for clients, one for suppliers) has much to do with the fact that complex application level session tracking would have to be implemented to ensure the security if just one web server was used. We also want our system to be somewhat redundant.

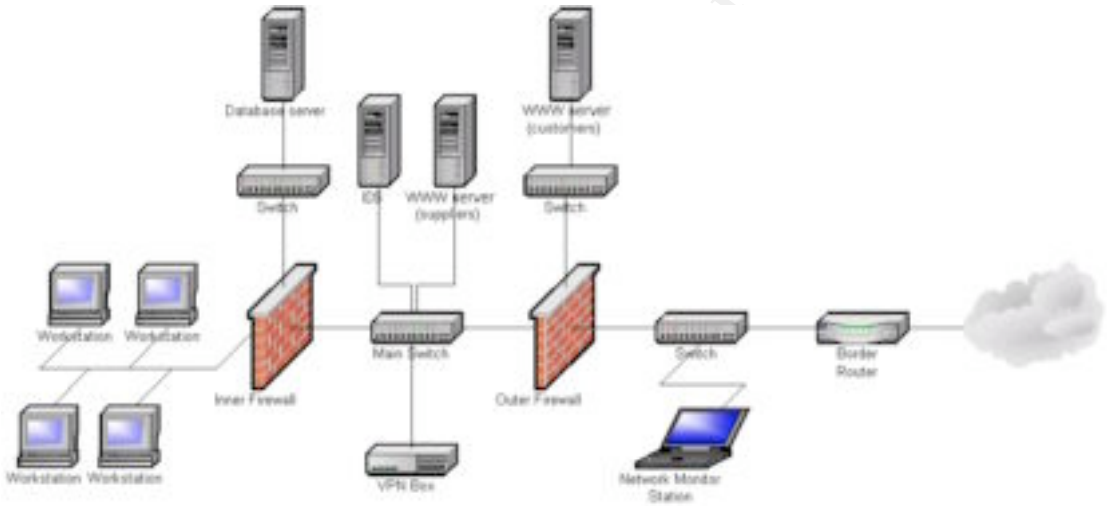
This will become clearer once we go through the diagram of the suggested network architecture.

1.1.5) Fortune Cookies

The product that GIAC Enterprises primarily sells can be considered of fairly low interest to hackers and industrial spies. The most sensitive part of the infrastructure in this case is the client information database. But of course both should be equally protected.

Also it is not very essential to have fortune cookie database updated real time on the web server. The WWW server can poll new and approved fortune cookies once in 12 hours for example.

1.2) Network design



This is the suggested design for the environment. A few designs were considered, but this one was chosen as the one that provides the best balance between security, functionality, scalability and the ease of administration.

We will assume that GIAC Enterprises has registered a class-C network for this environment; we will refer to this class-C network as **a.b.c**.0. We will divide it into multiple subnets as follows:

Segment	***Subnet***	***Mask***
Router <-> Outer firewall	**a.b.c**.0	/27 (.224)
Outer firewall <-> Customers' web server	**a.b.c**.32	/27 (.224)
Inner firewall <-> Outer firewall	**a.b.c**.64	/27 (.224)
Inner firewall <-> Database server	**a.b.c**.96	/27 (.224)
Maintenance LAN	**a.b.c**.128	/25 (.128)

We will assign IP addresses to participating network devices as follows:

Device	*Interface*	*Address /Mask*
Router		**a.b.c.**1 /27
Outer firewall	-> Router	**a.b.c.**2 /27
Outer firewall	-> Customers' web server	**a.b.c.**33 /27
Outer firewall	-> Inner firewall	**a.b.c.**65 /27
Web server (customers)		**a.b.c.**40 /27
Web server (suppliers)		**a.b.c.**70 /27
Database server		**a.b.c.**100 /27
VPN box		**a.b.c.**67 /27
Inner firewall	-> Outer firewall	**a.b.c.**66 /27
Inner firewall	-> Database server	**a.b.c.**97 /27
Inner firewall	-> Maintenance LAN	**a.b.c.**129 /25

The hardware and software versions of the devices in the picture can be seen in the following table:

Device	*Hardware*	*Software*
Border router	Cisco	IOS 11.x
Outer firewall	Intel x86	OpenBSD 2.8
Inner firewall	Intel x86	Linux 2.4.x
VPN box	Intel x86	OpenBSD 2.8
Main switch	Cisco	

Below we will describe each component's task in this environment.

1.2.1) WWW server (customers)

For the purposes of this document we refer to the WWW server responsible for handling customer requests as "WWW server" or "customers web server".

This web server will act as a main interface for the clients. It will be polling the latest approved fortune cookies from the database server.

The server will only listen for encrypted (HTTPS) connections (TCP port 443).

1.2.2) WWW server (suppliers)

For the purposes of this document we refer to the WWW server responsible for handling suppliers' requests as "suppliers web server".

This web server will act as a front-end to the database of fortune cookies. This is where suppliers will be submitting their work.

The server will only listen for encrypted (HTTPS) connections (TCP port 443).

1.2.3) Border Router

The task of the border router is to filter out all of the "garbage" traffic. This includes spoofing

attempts (incoming traffic that claims to originate from our network), traffic from private and multicast networks and some ICMP traffic.

The router should have support for IP packet forwarding and should not be confused by "unusual" IP protocol numbers (such as 50, since we will be using IPSEC ESP protocol). Almost all IP routers should do fine.

For our needs it will be sufficient for the router to have 1 serial and 1 ethernet interface.

1.2.4) Network Monitor Station

This is in no way a requirement for this environment. The network monitor could be a **non-stationary** laptop for example.

As this machine is placed outside the Outer firewall it can be used for penetration testing during the audit phase. It can also be used to monitor the bandwidth usage and perform some basic network troubleshooting tasks at the early stages of the implementation phase.

1.2.5) Outer Firewall

This device should have at least 3 ethernet interfaces. One interface (world) will be to the router. The customers' web server will be behind the second interface in the screened subnet. The third interface will be facing the segment between the Inner and the Outer firewalls.

This firewall will be allowing VPN traffic between the VPN box and GIAC Enterprises' partners' VPN devices. It will also allow HTTPS traffic to both of the web servers. It will also allow database poll requests from the customers' web server to the database server.

It should also allow traffic coming from the maintenance LAN.

We chose to have OpenBSD 2.8 running IP Filter 3.3 act as an outer firewall.

1.2.6) VPN Box

The VPN box will act as an end-point of the encrypted traffic coming from GIAC Enterprises' partners.

We will use IPSEC to implement the VPN. In particular we will be using tunnel mode and ESP (for authentication and encryption). MD5 will be used for authentication and 3DES for encryption.

Authentication phase 1 will be implemented using Main Mode (we will not be using Aggressive Mode for security and compatibility reasons). Phase 2 of the authentication will be implemented using the variant of Quick Mode that provides Perfect Forward Secrecy (PFS).

It should be noted that routing tables should be implemented very carefully on each of the servers that GIAC's partners will access. The fact that the return traffic must go through the VPN must be accounted for.

We chose to have an OpenBSD 2.8 box act as a VPN device. We will use ISAKMPD as a security association and key exchange daemon. It should be sufficient to have one network interface in the box.

1.2.7) Main Switch

For the purposes of this document we refer to the switch between the Inner and Outer firewalls as the main switch. It should be a more sophisticated device that could handle the amount of traffic that might be generated in this environment.

It should also be possible to have one of its ports act as a hub port, sending all the traffic it sees to the IDS. This is needed to ensure the proper functionality of the IDS.

1.2.8) IDS

The location of the IDS allows it to see the traffic that bypassed the primary firewall. The usability of its location is rather questionable, since it will not be able to pick up any attacks destined at the customers' web server. Only web traffic will be allowed to the customers' web server, though. It's much more critical to have the traffic that bypassed the primary firewall and is headed to the internal network inspected closer.

1.2.9) Inner Firewall

This device should have at least 3 ethernet interfaces. One interface will be facing the segment between the Inner and the Outer firewalls. The database server will be behind the second interface in the screened subnet. The third interface will be facing the maintenance LAN.

This firewall will be allowing GIAC's partners' decapsulated traffic coming from the VPN box to the database server. It will also allow traffic between both of the web servers and the database server.

It should also allow traffic coming from the maintenance LAN.

We chose to have a Linux distribution running 2.4.x kernel act as an inner firewall. We will be using IPTables to implement the security policy.

1.2.10) Database server

For the purposes of this practical we will assume that there is a database engine listening on TCP port 3306 of the database server, through which all of the database requests are handled.

Since the focus of this practical is network security, we will not be focusing on the importance of implementing a proper authentication hierarchy on the database server.

It must be stated, however, that proper authentication methods and level of authorization given to authenticated users are as important as tight network security, especially in the stated environment. For example, it would be wise to allow read-only access to the certain parts of the database for requests that originate from the customers' web server.

It would also make sense to use PUSH technique instead of PULL. See recommendations (3.3) section for details.

1.2.11) Maintenance LAN

To keep things simple we will assume that the GIAC Enterprises' office LAN is logically and physically completely separated from the production environment described in this practical. Thus the production environment stays protected from internal attacks.

The maintenance LAN however will be a network of a few workstations located in a physically

secured place. The idea here is to make the administration of the servers' possible not only from the console, but also from the workstations located in the maintenance LAN. Both firewalls will be allowing traffic coming from the maintenance network.

1.2.12) Architecture notes

This architecture presumes that a reasonable amount of traffic is generated in the stated environment. The main switch here plays an important role, since all the database traffic goes through it. Of course having the customers' web server poll the database server only a few times a day for new and approved cookies should minimize this traffic.

We also assume that the DNS service is hosted elsewhere. The choice for having this service hosted elsewhere was influenced by the large number of recently discovered security flaws in various versions of DNS server software.

2.) Assignment II: The Policy

Good security architecture is very important. It's equally important to have a properly implemented security policy based on the security architecture.

This part of the practical describes the actual implementation of the security policy, with devices' configuration files being walked through.

2.1) Border Router

The security architecture states that the border router should be filtering "garbage" traffic. Below are excerpts from a Cisco router configuration file, which show how this could be implemented.

The first 3 octets from the class C network address of GIAC Enterprises are represented as **a.b.c** in the configuration file excerpts below.

border router config excerpts

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

Above should be applied in the global configuration mode and specifies that we should deny source-routed packets and disables all the unnecessary services.

no ip directed-broadcast
no ip unreachables
no ip proxy-arp
no cdp enable
ntp disable

Above should be applied in the external (serial) interface configuration mode and specifies that we want to deny ICMP broadcasts and ICMP unreachable messages. It also specifies that we do not want to use Cisco Discovery Protocol (CDP), which gives out information about our network to a potential attacker. It also specifies that we do not want to run an ARP proxy, nor do we want to use ntp (Network Time Protocol).

interface serial0
ip access-group 15 in

The above specifies that access-list 15 should be applied on the packets coming into the world (serial) interface. The rules of the access list 15 follow:

access-list 15 deny 10.0.0.0 0.255.255.255 log
access-list 15 deny 172.16.0.0 0.15.255.255 log
access-list 15 deny 192.168.0.0 0.0.255.255 log
access-list 15 deny 224.0.0.0 15.255.255.255 log

The above denies and logs packets that have reserved or multicast address as a source address.

access-list 15 deny 127.0.0.0 0.255.255.255 log
access-list 15 deny 0.0.0.0 0.255.255.255 log
access-list 15 deny 255.255.255.255 0.0.0.0 log

The above denies and logs localnet and broadcast packets.

*access-list 15 deny **a.b.c**.0 0.0.0.255 log*
access-list 15 permit any

The above denies and logs spoofing attempts (i.e.: packets that claim to be coming from our network, but arrive to the world interface). The second line from the above excerpt permits all the packets that did not match any of the previous rules.

2.2) Outer Firewall

We start implementing the firewall by performing a minimal installation of OpenBSD 2.8.

For the purposes of this practical we will not be concentrating on stripping unneeded services or performing any other kind of host hardening.

After the initial install has been completed and the network interface set up appropriately we need to enable IP packet forwarding. This can be achieved by having the following line in the file **/etc/sysctl.conf**:

net.inet.ip.forwarding=1

After this line has been entered the machine needs to be rebooted; alternatively the setting should be set manually to the kernel state. This can be done with the **sysctl** command (**sysctl -w net.inet.ip.forwarding=1**).

In the firewall configuration file all the IP addresses are represented the following way:

a.b.c - represents the first 3 octets from the class C network address of GIAC Enterprises
d.e.f.30 - represents the IP address of GIAC Enterprises' partner's VPN gateway

And here's the actual firewall configuration:

ipf.rules

```
#
# Sergei Ledovskij / 7.Apr.2001
# an example firewall ruleset for GCFW practical
#
# IP Filter 3.3 Syntax
#

#--------------------------------------------------------------
#
# ep0 (.2)  - (world) -> router
# xl0 (.33) - (screened subnet) -> customers web server
# xl1 (.65) - (inside segment) -> inner firewall
#
```

Above comments are purely informational. It's very important to have well commented configuration files, which eases administration dramatically.

```
#--------------------------------------------------------------
# block & log short fragments on all interfaces

block in log quick all with short
```

Above we block and log packets that are too short to contain a complete header.

```
#---------------------------------------------------------
# setup per-interface groups for traffic going in and out
#
# our default policy is to *block* everything,
# but the required traffic
#
# #---------------------------------------------------------
# Group 100 - traffic coming into the world interface
# Group 150 - traffic going out of the world interface
# --
# Group 200 - traffic coming into the screened subnet interface
# Group 250 - traffic going out of the screened subnet interface
# --
# Group 300 - traffic coming into the inside interface
# Group 350 - traffic going out of the inside interface
# #---------------------------------------------------------

block in on ep0 all head 100
block out log on ep0 all head 150

block in log on xl0 from a.b.c.32/27 to any head 200
block out log on xl0 all head 250

block in log on xl1 all head 300
block out log on xl1 all head 350
```

Above we set up per-interface input / output groups. This gives us greater flexibility in creating our rules, while keeping them simple at the same time. Please note, that our default policy is to block packets. This is just fine, since we will take care of allowing traffic both ways (in and out) with "keep-state" clause. IPFilter checks its state-table to see if the packet is a part of an established connection before going through the entire ruleset.

Please note that we do not log all of the blocked traffic coming into the world interface. This is done to minimize the amount of logs that would be otherwise generated, due to the amount of mass-scans on the Internet today.

We do log all of the blocked traffic that originates from the internal network though. This is done to ensure that we would notice break-ins and attempts to steal information.

```
#---------------------------------------------------------
# Deny reserved addresses coming into the world interface

block in log quick from 10.0.0.0/8 to any group 100
block in log quick from 192.168.0.0/16 to any group 100
block in log quick from 172.16.0.0/12 to any group 100
```

Above we block and log packets that have reserved address as the source address coming into the world interface (this is specified by RFC 1918). We have similar rules in the border router, but the performance drawback here is minimal, and it's better to be safe then sorry.

```
# Prevent IP spoofing

block in log quick from a.b.c.0/24 to any group 100
```

Above we block and log packets coming to the world interface that claim to be coming from our network. The above rule also denies packets **originating** from the border router (i.e.: **not** packets

being forwarded by it), which is just fine, since if someone got access to the router they would not gain access to the internal network from there. Of course router configuration should be done from the console (we will be allowing maintenance LAN to access it later in the file).

```
#---------------------------------------------------------
# localnet traffic should only exist on loopback interface

block in log quick from 127.0.0.0/8 to any group 100
block in log quick from any to 127.0.0.0/8 group 100
block in log quick from 127.0.0.0/8 to any group 200
block in log quick from any to 127.0.0.0/8 group 200
block in log quick from 127.0.0.0/8 to any group 300
block in log quick from any to 127.0.0.0/8 group 300

# we allow packets to traverse the loopback interface

pass in quick on lo0 all
pass out quick on lo0 all
```

Above we block and log traffic that claims to be coming from the localnet, still allowing it to traverse the loopback interface.

```
#---------------------------------------------------------
# Allow traffic necessary to the environments functionality

# Allow the world to connect to the web servers

pass in quick proto tcp from any to a.b.c.40 port = 443 flags S keep state group 100
pass in quick proto tcp from any to a.b.c.70 port = 443 flags S keep state group 100
```

Above we allow connections to the HTTPS port on both of the web servers. Note that we only allow initial connection requests (packets with the SYN bit set), creating a state table for them. Thus the return traffic coming from the web servers will be allowed to traverse back to the host originating the request. This is done to ensure that if a potential attacker breaks into the web server she will not be able to make it her launch base to attack more servers on the Internet.

```
# Allow database requests from the customers' web server to the db server

pass in quick proto tcp from a.b.c.40 to a.b.c.100 port = 3306 flags S keep state group 200
```

Above we allow connections from the customers' web server to the database server. We only allow initial connection requests and create a state table for them, thus allowing database server's reply packets to traverse back to the customers' web server.

```
# Allow VPN traffic from/to partners
# we only allow one partner to establish VPN right now
# more should be added here as needed

pass in quick proto udp from d.e.f.30 port = 500 to a.b.c.67 port = 500 keep state group 100
pass in quick proto udp from a.b.c.67 port = 500 to d.e.f.30 port = 500 keep state group 300
pass in quick proto 50 from d.e.f.30 to a.b.c.67 group 100
pass in quick proto 50 from a.b.c.67 to d.e.f.30 group 300
```

Above we allow traffic from GIAC's partners' VPN gateways. We allow ISAKMPD traffic from UDP port 500 to UDP port 500. We also allow ESP (protocol 50) to traverse the firewall.

Rules above configure access for one partner only. Using the above 4 rules as an example it should

be fairly straightforward to configure more. Of course each partner that is allowed to establish VPN should also be accounted for in the VPN box's configuration.

 # Allow traffic coming from the maintenance lan

*pass in quick proto tcp/udp from **a.b.c**.128/25 to any keep state group 300*
*pass in quick proto icmp from **a.b.c**.128/25 to any keep state group 300*

Above we allow TCP, UDP and ICMP traffic originating from the maintenance LAN. We also create a state entry for this traffic, thus allowing return traffic to traverse the firewall. ICMP keep-state is also supported in IPFilter 3.3, as seen from the second line above.

Great tool to test ipf rules is a command called **ipftest**. If the ruleset above was saved as a file called ipf.rules, then the syntax for testing it with ipftest would be **ipftest -r ./ipf.rules**. To apply the rules and enable the firewall the following command can be used: **ipf -Fa -f ./ipf.rules -E**

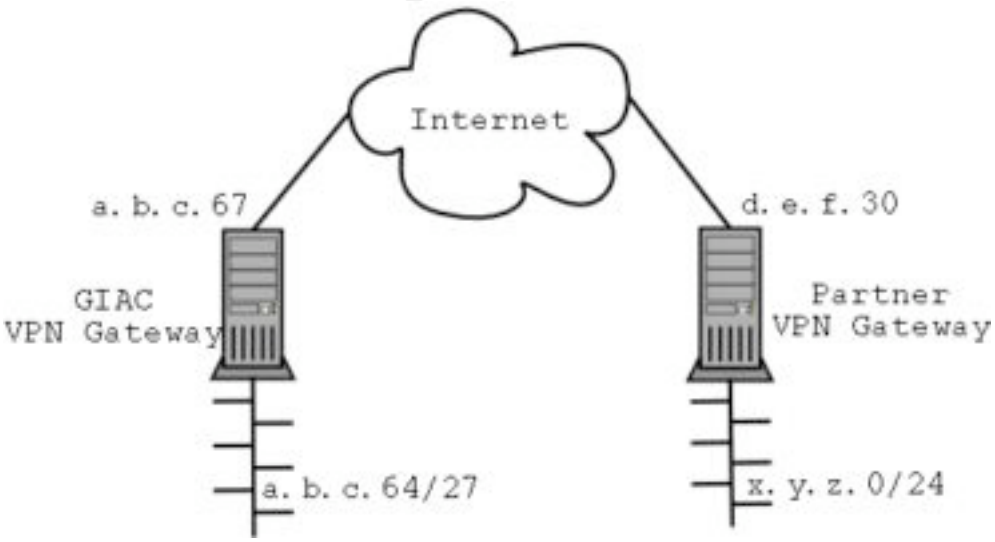
2.3) VPN Box

The implementation of the VPN as stated in our security policy is fairly straightforward.

In the configuration files all IP addresses are represented the following way:

a.b.c - represents the first 3 octets from the class C network address of GIAC Enterprises
d.e.f.30 - represents the IP address of GIAC Enterprises' partner's VPN gateway
x.y.z - represents the first 3 octets from the class C network address of one of the GIAC's partners

This is better seen from the following picture:



Please note that the picture above is purely logical, since the GIAC VPN gateway only has one network interface.

We start with an x86 hardware and perform a minimal installation of OpenBSD 2.8. The kernel handles IPSEC traffic in OpenBSD and IKE is handled by a user-land daemon ISAKMPD (or at least that's what we will use, since OpenBSD also provides support for photuris key management protocol, appropriate daemon being called photurisd).

For the purposes of this practical we will not be concentrating on stripping unneeded services or performing any other kind of VPN box hardening.

After the initial install has been completed and the network interface set up appropriately we have to tweak a few system settings to enable the VPN support.

In file **/etc/sysctl.conf** the following lines should be present:

```
net.inet.ip.forwarding=1
net.inet.esp.enable=1
```

The 1st line enables ip forwarding, so that the packets coming from the VPN device will be actually forwarded to the local network. The 2nd line enables the ESP protocol.

For maximum compatibility with different implementations of IKE daemons and the purposes of this practical we chose PresharedSecretKey as our authentication method.

Our security policy (Assignment I) states that we should be using ESP with 3DES encryption. This is how we implement this:

isakmpd.policy

```
Authorizer: "POLICY"
Licensees: "passphrase:gcfwpractical"
Conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
              esp_enc_alg != "null" -> "true";
```

This simple ISAKMPD policy states that any proposal from a remote host that authenticates using passphrase "gcfwpractical" will be accepted, as long as it contains ESP with a non-null algorithm (i.e.: packet will be encrypted). This is sufficient for us, we will be looking at configuration of accepted proposals in more detail next:

isakmpd.conf

```
# /etc/isakmpd/isakmpd.conf

[Phase 1]
d.e.f.30=               partner1-vpn-gw
```

The section [Phase1] specifies that SA negotiation phase 1 peer at address **d.e.f.30** should be known as **partner1-vpn-gw.**

```
[Phase 2]
Connections=            partner1-conn
```

The section [Phase2] specifies that we have one connection configured, that should be known as **parner1-conn**.

```
[partner1-vpn-gw]
Phase=                  1
Transport=              udp
Address=                d.e.f.30
Configuration=          Default-main-mode
Authentication=         gcfwpractical
```

The section [partner1-vpn-gw] describes the parameters that should be used when negotiating with that peer. Things like transport protocol to use, IP address and authentication data for this specific

peer are in this section. The name of the ISAKMP-configuration section to use when negotiating with this peer is also mentioned here (**Default-main-mode** in this case).

```
[partner1-conn]
Phase=              2
ISAKMP-peer=        partner1-vpn-gw
Configuration=      Default-quick-mode
Local-ID=           giac-net
Remote-ID=          partner1-net
```

The section [partner1-conn] describes the IPSEC connection properties. Things like IPSEC configuration section to use and peer to use with this connection are mentioned here. This section also specifies the names of the sections that describe local and remote client IDs.

```
[giac-net]
ID-type=            IPV4_ADDR_SUBNET
Network=            a.b.c.64
Netmask=            255.255.255.224
```

The section [giac-net] describes local client's ID. In this case it specifies that the ID type is IPv4 network address with a netmask.

```
[partner1-net]
ID-type=            IPV4_ADDR_SUBNET
Network=            x.y.z.0
Netmask=            255.255.255.0
```

The section [partner1-net] describes remote client's ID. In this case it specifies that the ID type is IPv4 network address with a netmask.

```
[Default-main-mode]
DOI=                IPSEC
EXCHANGE_TYPE=      ID_PROT
Transforms=         3DES-MD5
```

The section [Default-main-mode] describes the ISAKMP parameters to use. This section is referred to by peer configuration section above (partner1-vpn-gw). It's here that we specify that we would like to use 3DES and MD5 to protect and authenticate the ISAKMP traffic. In this section we also specify that we would like to use main mode for phase 1, we do that by specifying ID_PROT as an EXCHANGE_TYPE (if we wanted to use aggressive mode we would have specified AGGRESSIVE as an EXCHANGE_TYPE).

```
[Default-quick-mode]
DOI=                IPSEC
EXCHANGE_TYPE=      QUICK_MODE
Suites=             QM-ESP-3DES-MD5-PFS-SUITE
```

The section [Default-quick-mode] describes the IPSEC parameters to use. This section is referred to by connection configuration section above (partner1-conn). In this section we specify that we would like to use ESP and use 3DES and MD5 to protect and authenticate the IP traffic. We also specify that we would like to use the variant of QuickMode that provides for PFS (PerfectForwardSecrecy).

Since we did not specify and key exchange parameters in the ISAKMPD configuration file, the defaults will be used. The defaults are set as follows:

Main mode lifetime: 1 hour
Quick mode lifetime: 20 minutes
Diffie-Hellman group description for Main and Quick Mode (PFS): 1 (MODP_768)

To apply this configuration on OpenBSD 2.8 you should copy both files to **/etc/isakmpd** directory and run **isakmpd**. Make sure that you have set appropriate settings in **/etc/sysctl.conf** and either rebooted the machine after that or set them manually by using **sysctl** command.

The configuration described above has been tested with Linux 2.2.17 and FreeSWan 1.8 acting as the other end-point of the VPN. For the clarity of things the Linux-side configuration used in testing follows:

/etc/ipsec.conf

```
# FreeS/WAN IPSEC configuration file

# basic configuration
config setup
        interfaces=%defaultroute
        klipsdebug=none
        plutodebug=none
        plutoload=%search
        plutostart=%search

# defaults for subsequent connection descriptions
conn %default
        keyingtries=0
        spi=0x200
        esp=3des-md5-96
        espenckey=0x01234567_89abcdef_02468ace_13579bdf_12345678_9abcdef0
        espauthkey=0x12345678_9abcdef0_2468ace0_13579bdf

conn giac
        left=a.b.c.67
        leftsubnet=a.b.c.64/27
        leftnexthop=
        leftid=a.b.c.67
        right=d.e.f.30
        rightsubnet=x.y.z.0/24
        rightnexthop=
        rightid=partner1-net
        auto=add
```

As we can see the configuration of the Linux-side is also pretty straightforward. We specify the addresses of the peers and networks on both sides in [conn giac] section. And we specify that we would like to use 3DES and MD5 in [conn %default] section.

/etc/ipsec.secrets

```
# This file holds shared secrets or RSA private keys for inter-Pluto
# authentication.  See ipsec_pluto(8) manpage, and HTML documentation.

a.b.c.67 d.e.f.30  "gcfwpractical"
```

The contents of the file above specifies that the authentication between peer **a.b.c**.67 and **d.e.f.30** should be based on PSK (PresharedSecretKey) and specifies the secret to be "gcfwpractical".

The entire configuration of the ISAKMPD daemon without comments is also attached at the end of this document.

3.) Assignment III: The Audit

Each security audit starts with a carefully designed assessment plan. After that follows the gathering of information. There are many ways to gather information about the target network. The easiest way is to query WHOIS databases of appropriate NICs (Network Information Centers), such as ARIN, RIPE, etc. Sometimes that method allows us to find out physical location of the system and the responsible technical person, which allows us to perform social engineering attacks.

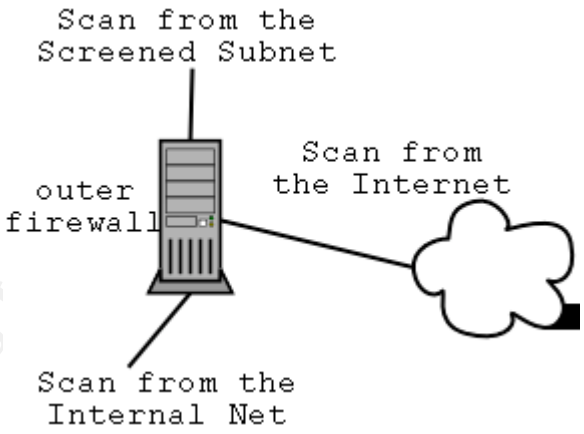
Most of the time we hear that hackers work at night. Unlike wise, if we were malicious we would be executing our scans during the network peak hours, since the attack might easily remain unnoticed, considering the amount of legit traffic.

For the purposes of this audit, however it would be best to have these scans ran on a weekend night, during the time when the network is being used the least. We must consider the fact that some of our "exotic" scans might crash some network devices on the way. Since we will also be testing network's resistance to DoS attacks it's a simple requirement for us to have this attacks executed in the off-hours. It's also essential to have appropriate technical personnel present, in case the system would need a hand getting up after a possible crash.

In this assignment we are required to perform an audit of the Outer Firewall to validate that it is actually implementing the security policy.

3.1) Outer firewall audit

We decide to perform the audit from three different logical points to make sure that the input / output rulesets of each interface are working well. This is demonstrated in the following picture:



During each scan we will be scanning two other points to validate the proper implementation of the policy. We will be also performing the validation from the maintenance lan. We will be using **nmap** port scanning tool to perform the audit.

3.1.1) Scan from the Internet

We start the audit by scanning the target network from the Internet. We setup our attacking machine on the world side of the outer firewall.

3.1.1.1) Scan of the firewall

In this section we go through the results of port scanning the firewall.

On the attacking machine we issue the following command:

nmap -sS -F -P0 a.b.c.2

The above specifies that we do not want to either ICMP or TCP ping the firewall (-P0). We also specify that we want a fast scan, which means that we only want to scan for ports listed in the services file that comes with nmap (-F). And we specify that TCP SYN scan (also known as half-open scanning) should be used (-sS).

Here's the nmap output:

Starting nmap V. 2.54BETA7 (www.insecure.org/nmap/)
*All 1075 scanned ports on (**a.b.c**.2) are: filtered*
Nmap run completed -- 1 IP address (1 host up) scanned in 1897 seconds

As seen from above the firewall blocked every packet. We had actually started a few services (sshd, identd) on the firewall, but as seen from above none of them have replied.

3.1.1.2) Scan of the screened subnet

In this section we go through the results of port scanning **a.b.c**.32/27 subnet.

On the attacking machine we issue the command:

nmap -sS -P0 -F a.b.c.32/27

The above specifies that hosts neither should nor be ICMP nor TCP pinged before the scan. We also specify that we want a fast scan, which means that we only want to scan for ports listed in the services file that comes with nmap. And we specify that TCP SYN scan (also known as half-open scanning) should be used.

Excerpts from the results of the scan follow:

*Interesting ports on (**a.b.c**.40):*
(The 1074 ports scanned but not shown below are in state: filtered)
Port State Service
443/tcp open https

The above excerpt validates that the outer firewall policy is functioning as defined, since the only service that was reported as listening was the HTTPS service on the suppliers' web server.

3.1.1.3) Scan of the internal network

In this section we go through the results of port scanning **a.b.c**.64/27 subnet.

On the attacking machine we issue the command:

nmap -sS -P0 -F a.b.c.64/27

The above specifies that hosts neither should nor be ICMP nor TCP pinged before the scan. We also specify that we want a fast scan, which means that we only want to scan for ports listed in the services file that comes with nmap. And we specify that TCP SYN scan (also known as half-open scanning) should be used.

Excerpts from the results of the scan follow:

*Interesting ports on (**a.b.c**.70):*
(The 1074 ports scanned but not shown below are in state: filtered)
Port State Service
443/tcp open https

The above excerpt validates that the outer firewall policy is functioning as defined, since the only service that was reported as listening was the HTTPS service on the suppliers' web server. Next we perform a UDP scan of the subnet. The UDP scan should find ISAKMPD listening on UDP port 500 of the VPN box. We do this by issuing the command:

nmap -sU -P0 -F a.b.c.64/27

The above specifies that hosts neither should nor be ICMP nor TCP pinged before the scan (-P0). We also specify that we want a fast scan, which means that we only want to scan for ports listed in the services file that comes with nmap (-F). And we specify that UDP scan should be performed (-sU).

Here's the output:

*All 977 scanned ports on (**a.b.c**.67) are: filtered*
Nmap run completed -- 1 IP address (1 host up) scanned in 1188 seconds

So, what's going on here? Well, since UDP is a connectionless protocol there's no certain way to make sure whether there are any UDP ports behind the firewall listening. This is especially true with nmap, since if it does not receive an ICMP port unreachable message for each port it sends probe to it assumes that the port is open. This is extremely unreliable.

We can verify that packets are actually arriving to the VPN box by running tcpdump on it and running nmap like this:

nmap -sU -P0 -g 500 -p 500 -D d.e.f.30 a.b.c.67

The above specifies that we want a UDP scan of port 500 on host **a.b.c**.67, and we want our source address to be **d.e.f.30** and the source port for this probe to be 500. By running tcpdump on the VPN box we were able to verify that the packet crafted by the above command has actually arrived to the VPN box. Here's an excerpt from the tcpdump output:

*21:36:09.463530 **d.e.f.30**.500 > **a.b.c**.67.500: [|isakmp]*

3.1.2) Scan from the screened subnet

In this section we place our attacking machine in the screened subnet. We take down the web server and give our scanning machine the IP address **a.b.c**.40. This is done to simulate the situation where a potential attack broke into the web server.

3.1.2.1) Scan of the firewall

In this section we go through the results of port scanning the firewall.

On the attacking machine we issue the following command:

nmap -sS -F -P0 a.b.c.33

The above specifies that we do not want to either ICMP or TCP ping the firewall (-P0). We also specify that we want a fast scan, which means that we only want to scan for ports listed in the services file that comes with nmap (-F). And we specify that TCP SYN scan (also known as half-open scanning) should be used (-sS).

Here's the nmap output:

Starting nmap V. 2.54BETA7 (www.insecure.org/nmap/)
*All 1075 scanned ports on (**a.b.c**.33) are: filtered*
Nmap run completed -- 1 IP address (1 host up) scanned in 1903 seconds

As seen from above the firewall blocked every packet. We had actually started a few services (sshd, identd) on the firewall, but as seen from above none of them have replied. We have also verified that firewall has logged every packet it blocked by running **ipmon** command on the firewall. Here's an excerpt from its output:

*11/04/2001 10:18:08.714642 xl0 @0:3 b **a.b.c**.40,37121 -> **a.b.c**.33,199 PR tcp len 20 40 -S IN*
*11/04/2001 10:18:08.714772 xl0 @0:3 b **a.b.c**.40,37121 -> **a.b.c**.33,559 PR tcp len 20 40 -S IN*
*11/04/2001 10:18:08.714900 xl0 @0:3 b **a.b.c**.40,37121 -> **a.b.c**.33,780 PR tcp len 20 40 -S IN*

3.1.2.2) Scan of the internal network

In this section we go through the results of port scanning **a.b.c**.64/27 subnet.

On the attacking machine we issue the command:

nmap -sS -P0 -F a.b.c.64/27

The above command did not report any ports being open on any server. This validates the fact that the firewall is working properly. We verify this by running **ipmon** command on the firewall. Here's an excerpt of its output:

*11/04/2001 10:27:29.561942 xl0 @0:3 b **a.b.c**.40,46543 -> **a.b.c**.70,2028 PR tcp len 20 40 -S IN*
*11/04/2001 10:27:29.562073 xl0 @0:3 b **a.b.c**.40,46543 -> **a.b.c**.70,560 PR tcp len 20 40 -S IN*
*11/04/2001 10:27:29.562207 xl0 @0:3 b **a.b.c**.40,46543 -> **a.b.c**.70,1450 PR tcp len 20 40 -S IN*

Next we verify that the customers' web server can connect to the database server. We verify this by issuing the following command on the attacking machine:

nmap -sS -P0 -F a.b.c.96/27

Excerpts from the nmap output follow:

*Interesting ports on (**a.b.c**.100):*
(The 1074 ports scanned but not shown below are in state: filtered)
Port State Service
3306/tcp open https

The above excerpt validates that the outer firewall policy is functioning as defined, since the only service that was reported as listening was the HTTPS service on the suppliers' web server.

3.1.2.3) Attempt to connect to the world

In this section we attempt to connect to a host that is located on the Internet, outside of our firewall. We do this to validate the firewall ruleset, and make sure keep state is working properly. We have verified that keep state works in the 'Scan of the screened subnet from the Internet' section. Now we also verify this by trying to connect to the outside world from the customers' web server.

On the customers' web server we issue the following command:

telnet 216.239.37.100 80

The above command specifies that we would like to connect to the TCP port 80 (www) of the server 216.239.37.100 (www.google.com as of 11.Apr.2001).

The output of the above follows:

Trying 216.239.37.100...

Connection does not go through. We verify that the packet was blocked by the firewall. We do this by running **ipmon** command. The output of **ipmon** follows:

*11/04/2001 10:36:25.229425 xl0 @0:3 b **a.b.c**.40,1338 -> 216.239.37.100,80 PR tcp len 20 60 -S IN*

3.1.3) Scan from the internal network

In this section we place our attacking machine in the internal subnet. We take down the suppliers' web server and give our scanning machine the IP address **a.b.c**.70. This is done to simulate the situation where a potential attacker broke into the suppliers' web server.

3.1.3.1) Scan of the firewall

In this section we go through the results of ports scanning the firewall.

On the attacking machine we issue the following command:

nmap -sS -F -P0 a.b.c.65

Here's the nmap output:

Starting nmap V. 2.54BETA7 (www.insecure.org/nmap/)
*All 1075 scanned ports on (**a.b.c**.65) are: filtered*
Nmap run completed -- 1 IP address (1 host up) scanned in 1901 seconds

As seen from above the firewall blocked every packet. We had actually started a few services (sshd, identd) on the firewall, but as seen from above none of them have replied. We have also verified that firewall has logged every packet it blocked by running **ipmon** command on the firewall. Here's an excerpt from its output:

*11/04/2001 11:57:42.584187 xl1 @0:4 b **a.b.c**.70,60372 -> **a.b.c**.65,435 PR tcp len 20 40 -S IN*
*11/04/2001 11:57:42.584312 xl1 @0:4 b **a.b.c**.70,60372 -> **a.b.c**.65,480 PR tcp len 20 40 -S IN*
*11/04/2001 11:57:42.584438 xl1 @0:4 b **a.b.c**.70,60372 -> **a.b.c**.65,131 PR tcp len 20 40 -S IN*

3.1.3.2) Scan of the screened subnet

In this section we go through the results of port scanning **a.b.c**.32/27 subnet.

On the attacking machine we issue the command:

nmap -sS -P0 -F a.b.c.32/27

The above command did not report any ports being open on any server. This validates the fact that the firewall is working properly. We verify this by running **ipmon** command on the firewall. Here's an excerpt of its output:

*11/04/2001 12:12:27.950906 xl1 @0:4 b **a.b.c**.70,37136 -> **a.b.c**.70,382 PR tcp len 20 40 -S IN*
*11/04/2001 12:27:27.951038 xl1 @0:4 b **a.b.c**.70,37136 -> **a.b.c**.70,498 PR tcp len 20 40 -S IN*
*11/04/2001 12:12:27.951165 xl1 @0:4 b **a.b.c**.70,37136 -> **a.b.c**.70,512 PR tcp len 20 40 -S IN*

3.1.3.3) Attempt to connect to the outside

In this section we attempt to connect to a host that is located on the Internet, outside of our firewall.

On the suppliers' web server we issue the following command:

telnet 216.239.37.100 80

The above command specifies that we would like to connect to the TCP port 80 (http) of the server 216.239.37.100 (www.google.com as of 11.Apr.2001).

The output of the above follows:

Trying 216.239.37.100...

Connection does not go through. We verify that the packet was blocked by the firewall. We do this by running **ipmon** command on the firewall. The output of **ipmon** follows:

*11/04/2001 12:18:06.042370 xl1 @0:4 b **a.b.c**.70,1341 -> 216.239.37.100,80 PR tcp len 20 60 -S IN*

3.1.4) Scan from the maintenance LAN

In this section we place our scanning machine in the maintenance LAN. By doing this we verify that the outer firewall is implementing the security policy properly, as our security architecture states that machines in the maintenance LAN should be allowed access to any host. We give our scanning machine the IP address **a.b.c**.130.

3.1.4.1) Scan of the firewall

In this section we go through the results of port scanning the firewall.

On the scanning machine we issue the following command:

nmap -sS -F -P0 a.b.c.65

The above specifies that we do not want to either ICMP or TCP ping the firewall (-P0). We also specify that we want a fast scan, which means that we only want to scan for ports listed in the services file that comes with nmap (-F). And we specify that TCP SYN scan (also known as half-open scanning) should be used (-sS).

Here's the nmap output:

Starting nmap V. 2.54BETA7 (www.insecure.org/nmap/)
Interesting ports on (192.168.0.65):
(The 1073 ports scanned but not shown below are in state: closed)
Port State Service
22/tcp open ssh
113/tcp open auth

The above validates that the firewall policy was implemented properly, since we've specified in the security architecture that the traffic originating from the maintenance LAN should be allowed; and above shows that both services that are listening on the firewall have responded.

3.1.4.2) Scan of the screened subnet

In this section we go through the results of port scanning **a.b.c**.32/27 subnet.

On the scanning machine we issue the command:

nmap -sS -P0 -F a.b.c.32/27

And here's an excerpt from the nmap output:

Interesting ports on (192.168.0.40):
(The 1073 ports scanned but not shown below are in state: closed)
Port State Service
113/tcp open auth
443/tcp open https

The above validates that the firewall policy was implemented properly, since we specified in the security architecture that the traffic originating from the maintenance LAN should be allowed; and above shows that both services that are listening on the customers' web server have responded.

3.1.4.3) Attempt to connect to the outside

In this section we attempt to connect to a host that is located on the Internet, outside of our firewall.

On the scanning machine we issue the following command:

telnet 216.239.37.100 80

The above command specifies that we would like to connect to TCP port 80 (www) on the server

216.239.37.100 (www.google.com as of 11.Apr.2001).

The output of the above telnet command follows:

Trying 216.239.37.100...
Connected to 216.239.37.100
Escape character is '^]'.

As seen from above the connection goes through. This validates the fact that our firewall policy was implemented as required by the security architecture.

3.2) Denial Of Service attack

We set up our attacking machine on the world side of the firewall and issue the following command:

targa2 a.b.c.2 a.b.c.2

The above specifies that bonk, jolt, land, nestea, newtear, syndrop, teardrop, winnuke, 1234, saihyousen and oshare DoS attacks should be run against the IP address **a.b.c**.2. Targa2 can be downloaded from **http://packetstorm.securify.com/groups/mixter/targa2.c** and is a great tool to test host's resistance to DoS attacks.

In this case the firewall has withstood all of the DoS attacks.

3.3) Recommendations for improvement

In this section we are required to make recommendations for improvement of the perimeter defense.

As can be seen from section 3.1 no actual security problems were discovered. The outer firewall is implementing the security policy properly.

One potential problem in the stated network design would be the possibility of high load on the Main switch that could be generated by a high amount of network traffic. To fix this potential problem it would be wise to have a stack of switches balancing the load.

It is also recommended to double all the critical components of the network. Thus it would be wise to have a firewall cluster and a web server cluster. Database cluster would also be a good idea. Cisco Local Director is a good product that could be used to balance the load in the stated environment.

In the stated network design the IDS does not analyze the traffic going to the customers' web server. It's advised to have another IDS set up in the screened subnet of the outer firewall with the purposes of analyzing the web traffic. This could also be achieved by installing 2 additional network devices in the IDS box. One could be wired to the screened subnet's switch and the other one to the switch on the world side of the outer firewall.

It could also be a good idea to have one workstation in the maintenance LAN, which would have a serial mux card installed. Serial connections can then be made between all of the switches in the environment and this workstation, which would make it easier to configure the switches.

Another potential problem is the fact that in the stated scenario the customers' web server initiates poll requests to pull new fortune cookies from the database server. It would be much wiser to have the database server push this information to the customers' web server, thus being the initiating side of the connection. That would help minimize the risk of attacker gaining access to the database server, in case she breaks into the customers' web server.

4.) Practical assignment (I-III) notes

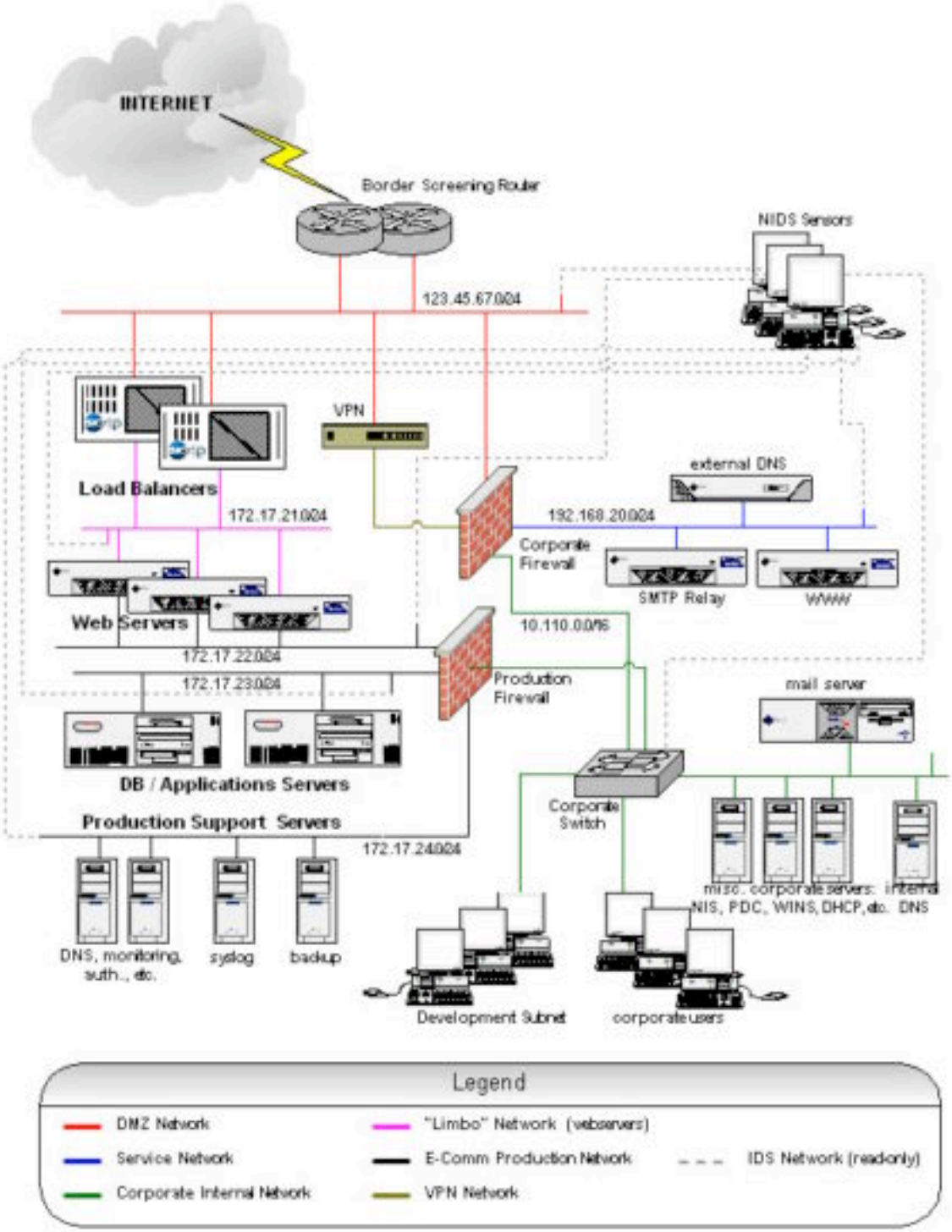
The assignments 1-3 were completed such way that they would emphasize on showing the appropriate perimeter protection techniques. Focus was on showing the knowledge and understanding of technology used to secure the network environment.

The choice of hardware and software heavily relied upon the availability of non-free products in my office. I would like to clarify that in no way was I vendor-biased and my choice of software/hardware for this practical was highly dependent on the software/hardware that I could get my hands on at the time of writing this practical.

5.) Assignment IV: Design Under Fire

For this assignment I have chosen to evaluate the network design used in Alexander Usenko's GCFW practical that can be found at the following URL:
http://www.sans.org/y2k/practical/Alexander_Usenko_GCFW.doc

And here's the design in question:



I will be using such information from Alexander's practical as type of firewall software used and

some firewall configuration rules.

The firewall software in question is Checkpoint Firewall-1 4.1.

The 1st thing that comes into mind when looking at the picture is the fact that VPN box is located outside the firewall. Even though that's quite common for many networks, this scenario leaves the VPN box itself protected only by the border router, thus allowing potential DoS attacks against the VPN box itself. There's no information in Alexander's paper regarding the hardware / software of the VPN box, thus we assume it could be anything, and that means that there could be numerous vulnerabilities if the box runs unnecessary services.

5.1) An attack against the firewall

Checkpoint Firewall-1 4.1 contains a bug when it crashes if it receives packets from the same IP address as itself, but with a different MAC address. Fortunately the border router in this scenario filters spoofed packets. So this particular exploit would not work. Example exploit code, however, can be found at:
http://www.self-evident.com/security/os/HARDWARE/firewalls/firewall-1/CPD.c

Sending extremely large fragmented packets to the firewall can exhaust Firewall-1 4.1's fragmentation logging process. More details on this attack can be found at:
http://www.demorgan.com.au/exploit/OS/Firewall-1/ip_fragment.html
This problem is fixed in SP2 for Firewall-1 4.1. However, Alexander's practical does not state whether SP2 was run on the firewall, thus this attack might work in this scenario.

There's also a fairly comprehensive list of Firewall-1 problems presented at the Black Hat Briefings 2000 that can be found at:
http://www.securiteam.com/securitynews/An_inspection_of_FireWall-1_reveals_holes.html
Most of the things mentioned in that document do not directly apply to the environment in question, and could not be easily exploited. There's a possibility that the TCP Fastmode problem could be exploited in the environment described in Alex's practical to map the internal network. I cannot be sure however, since Alex did not mention whether Fastmode was defined with any rules or not.

The URL above also mentions a rather interesting problem with the way Firewall-1 parses the FTP PORT command. This is more of an attack against the internal network, and it seems it does not apply to this scenario, since no FTP connections are defined.

Thus, the conclusion is that if the firewall software is up-to-date, there should not be any known problems that would allow us to execute a successful attack against the firewall itself.

5.2) A Denial of Service attack

For this purpose lets assume that a DoS attack is run against the network described in Alex's practical. Lets assume that there are 50 cable modem systems all UDP flooding the web server at the same time.

The question here is what will happen?

First of all the firewall will deny all UDP traffic coming to the web server, since there's no rule that would allow it.

UDP is a connectionless protocol, thus no reply has to be sent out indicating whether the packet has reached the destination.

So, if packets were spoofed (source address set to our network), then they would be blocked at the border router, since anti-spoofing ACLs are configured there.

If the packets were not spoofed, or spoofed to be coming from elsewhere, but our network, then they would be blocked at the firewall.

What really matters here is the speed at which the border router or the firewall would drop those packets. How much would it affect the load of the device in question?

Cable modem technology solutions range in speed. Let's assume that the effective upstream speed of the link between each of the attackers and the network attacked is 1Mbps. Thus, if we have 50 systems sending packets simultaneously at the maximum speed, the maximum amount of traffic they would generate is 50Mbps.

Of course the real number would be much lower, depending on how fast the systems in question can actually send the packets, uplink providers for each of those systems and the throughput of different routing devices along the way. It should be understood that all of this traffic will go through the uplink provider first, thus the effective transmission rate also depends on the rate at which the uplink provider routes those packets to the network under attack.

Let's assume that the efficiency here would be at around 50%, so the actual amount of traffic on the line would be around 25Mbps, or around 3125Kb/s. There's a very good reference that deals with the characteristics of UDP packet loss, which is located at:
http://www.isoc.org/isoc/whatis/conferences/inet/97/proceedings/F3/F3_1.HTM

RFC768 specifies the length of the UDP packet used in the UDP header as a 16bit value, thus the maximum length of a UDP packet is 65535 octets (this however depends on the size of the socket buffer set by the system).

For our purposes we assume that all of the attackers send an 8192 bytes long UDP packets from an arbitrary port to an arbitrary port. Since we mentioned the amount of traffic generated to be 3125Kb/s it is now easy to calculate the number of packets received each second by the firewall.

So having 50 cable modem systems of upstream bandwidth of 1Mbps sending 8Kb packets simultaneously at max. speed, and the efficiency rate being at 50% the number of packets received by the firewall each second will be 390.

Both the firewall and the router should be able to deal with such amount of UDP packets. The problem here is that the amount of traffic generated most likely exceeds the upstream bandwidth of the network described in Alex's practical.

So what is the answer? No certain answer can be given to this question. The following things might or might not happen: nothing will happen, the router will crash, the firewall load will become extremely high, the firewall will crash or the WAN connection will become incredibly slow.

5.3) An attack plan to compromise an internal system

Due to the latest BIND vulnerabilities, and the large amount of BIND servers still being unpatched, we choose to attack the DNS server.

Please note that this attack is purely theoretical, since Alexander specifically states in his practical how to secure the DNS server software.

Rule 12 of the firewall configuration in Alex's practical specifies that the Internet is allowed access

to the external DNS server for DNS lookups.

Our first objective would be to find out the version of the DNS server software used. Assuming the DNS server software is BIND we can achieve this with the following command:

dig @target version.bind. txt chaos

That would work only if the BIND server was configured to return its real version.

After we have retrieved the version we would check if that particular version is vulnerable. That information can be obtained from: **http://www.isc.org/products/BIND/bind-security.html**

After consulting with that page, we would start looking for the exploit for this particular version of BIND. Most of the exploits (buffer overflow / format string) are operating system / hardware dependent (well, the shellcode is anyway). There are many places on the net where exploits can be downloaded. We will mention just one web site here: **http://www.computec.ch/exploits/bind/**

It should be remembered that exploits are an extremely dangerous code to run, especially if you do not understand the source code in question. A good example would be the trojaned BIND8 exploit that was posted to BugTraq (Jan/Feb 2001). Also some of the exploits that are published on the net are broken and require a little tweaking.

Suppose we've found and compiled the appropriate exploit code for the particular version of BIND / hardware / operating system in question and it worked we should now have full access to the DNS server.

Well, not quite. The BIND server in question could've been ran in *chroot()*ed environment, or perhaps it wasn't run under *root*-privileges. It's also possible that the exploit in question didn't provide us with an interactive shell.

For our purposes, however, we assume that now we have a read/write access to the zone information. That means that we could execute multiple malicious attacks like a simple DoS attack or a 'traffic-hijacking-by-trojaning-dns-records' attack to redirect the traffic meant for the web server to our own server for the purposes of seizing the user authentication information.

Also now being in the same segment as the SMTP relay and the web server, we could scan them to see if they are running services that are blocked by the firewall and are visible from the inside. Some of those services might be exploitable.

If we wanted to launch a wider-scale attack against the network in question, we would be using some automated vulnerability scanning tool. Many good scanning tools are available on the net, with Nessus (http://www.nessus.org) being one of the best. By using Nessus with updated vulnerability database we would get a good idea of what might be vulnerable on the network in question fairly quickly.

Attachment I: Firewall configuration

ipf.rules

```
#
# Sergei Ledovskij / 7.Apr.2001
# an example firewall ruleset for GCFW practical
#
# IP Filter 3.3 Syntax
#

#--------------------------------------------------------------
#
# ep0 (.2)  - (world) -> router
# xl0 (.33) - (screened subnet) -> customers web server
# xl1 (.65) - (inside segment) -> inner firewall
#

#--------------------------------------------------------------
# block & log short fragments on all interfaces

block in log quick all with short

#--------------------------------------------------------------
# setup per-interface groups for traffic going in and out
#
# our default policy is to *block* everything,
# but the required traffic
#
# --------------------------------------------------------------
# Group 100 - traffic coming into the world interface
# Group 150 - traffic going out of the world interface
# --
# Group 200 - traffic coming into the screened subnet interface
# Group 250 - traffic going out of the screened subnet interface
# --
# Group 300 - traffic coming into the inside interface
# Group 350 - traffic going out of the inside interface
# --------------------------------------------------------------

block in on ep0 all head 100
block out log on ep0 all head 150

block in log on xl0 from a.b.c.32/27 to any head 200
block out log on xl0 all head 250

block in log on xl1 all head 300
block out log on xl1 all head 350

#--------------------------------------------------------------
# Deny reserved addresses coming into the world interface

block in log quick from 10.0.0.0/8 to any group 100
block in log quick from 192.168.0.0/16 to any group 100
block in log quick from 172.16.0.0/12 to any group 100

# Prevent IP spoofing

block in log quick from a.b.c.0/24 to any group 100

#--------------------------------------------------------
# localnet traffic should only exist on loopback interface
```

```
block in log quick from 127.0.0.0/8 to any group 100
block in log quick from any to 127.0.0.0/8 group 100
block in log quick from 127.0.0.0/8 to any group 200
block in log quick from any to 127.0.0.0/8 group 200
block in log quick from 127.0.0.0/8 to any group 300
block in log quick from any to 127.0.0.0/8 group 300

# we allow packets to traverse the loopback interface

pass in quick on lo0 all
pass out quick on lo0 all

#---------------------------------------------------------
# Allow traffic necessary to the environments functionality

# Allow the world to connect to the web servers

pass in quick proto tcp from any to a.b.c.40 port = 443 flags S keep state group 100
pass in quick proto tcp from any to a.b.c.70 port = 443 flags S keep state group 100

# Allow database requests from the customers' web server to the db server

pass in quick proto tcp from a.b.c.40 to a.b.c.100 port = 3306 flags S keep state group 200

# Allow VPN traffic from/to partners
# we only allow one partner to establish VPN right now
# more should be added here as needed

pass in quick proto udp from d.e.f.30 port = 500 to a.b.c.67 port = 500 keep state group 100
pass in quick proto udp from a.b.c.67 port = 500 to d.e.f.30 port = 500 keep state group 300
pass in quick proto 50 from d.e.f.30 to a.b.c.67 group 100
pass in quick proto 50 from a.b.c.67 to d.e.f.30 group 300

# Allow traffic coming from the maintenance lan

pass in quick proto tcp/udp from a.b.c.128/25 to any keep state group 300
pass in quick proto icmp from a.b.c.128/25 to any keep state group 300
```

Attachment II: VPN Box configuration

isakmpd.conf

```
# /etc/isakmpd/isakmpd.conf

[Phase 1]
d.e.f.30=               partner1-vpn-gw

[Phase 2]
Connections=            partner1-conn

[partner1-vpn-gw]
Phase=                  1
Transport=              udp
Address=                d.e.f.30
Configuration=          Default-main-mode
Authentication=         gcfwpractical

[partner1-conn]
Phase=                  2
ISAKMP-peer=            partner1-vpn-gw
Configuration=          Default-quick-mode
Local-ID=               giac-net
Remote-ID=              partner1-net

[giac-net]
ID-type=                IPV4_ADDR_SUBNET
Network=                a.b.c.64
Netmask=                255.255.255.224

[partner1-net]
ID-type=                IPV4_ADDR_SUBNET
Network=                x.y.z.0
Netmask=                255.255.255.0

[Default-main-mode]
DOI=                    IPSEC
EXCHANGE_TYPE=          ID_PROT
Transforms=             3DES-MD5

[Default-quick-mode]
DOI=                    IPSEC
EXCHANGE_TYPE=          QUICK_MODE
Suites=                 QM-ESP-3DES-MD5-PFS-SUITE
```

isakmpd.policy

```
Authorizer: "POLICY"
Licensees: "passphrase:gcfwpractical"
Conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
              esp_enc_alg != "null" -> "true";
```

Attachment III: Resources

- OpenBSD manual pages
- http://www.google.com
- http://www.isc.org
- http://www.isoc.org
- http://www.securityfocus.com
- http://www.securiteam.com
- http://packetstorm.securify.com
- http://www.attrition.org
- http://www.self-evident.com