Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

GIAC Certified Firewall Analyst (GCFW)
Practical Assignment Version 1.7
Track 2c

May 6-12, 2002
Capital SANS, Washington, D.C.

Created by:
Scott A. Petinga

October 17, 2002

© SANS Institute 2000 - 2005 Author retains full rights.

Table of Contents

© SANS Institute 2000 - 2005, Author retains full rights.

© SANS Institute 2000 - 2005, Author retains full rights.

Practical Assignment Goals

Design and develop a network security infrastructure for GIAC Enterprises, a fictitious e-business which deals in the online sale of fortune cookie sayings. Describe in detail the security policy of the various components with in the design. Also, perform a security audit on the firewalls to verify that the policies are correctly enforced based on your network design and policy.

Finally, select a network design from previously posted GCFW practicals and orchestrate an attack against their architecture.

Company Overview

GIAC Enterprises is a global leader in the online sale of fortune cookie sayings. Over the past 5 years they have seen their business grow substantially as e-commerce on the internet has matured. GIAC Enterprises has also been looking to expand their fortune cookie saying empire into other areas, on-line fortune telling and calendars (taking on the Dilbert desktop calander monopoly). Having just moved into their new facility, they now can centrally manage their e-business/internet infrastructure, as well as control all corporate business functions.

Assignment 1 – Security Architecture

We have been asked to design a network security architecture for GIAC Enterprises, an e-business which deals in the online sale of fortune cookie sayings. The first step is to understand GIAC Enterprise's current and future business and network requirements. After extensive research, the following requirements were defined:

- **Customers** – Companies or individuals that purchase bulk online fortunes.
  These customers (either account customers or individuals) will be accessing our external website portal.
- **Suppliers** – Companies that supply GIAC Enterprises with their fortune cookie sayings.
  GIAC has an in-house writing team but majority of their fortune saying come from outside partners businesses. These businesses need "secure" access to various resources within the GIAC Enterprise network.
- **Partners** – International companies that translate and resell fortunes
  Everyone loves fortunes, and GIAC Enterprises has started to work on expanding their global presence with other companies around the world. These partner businesses will need "secure"

© SANS Institute 2000 - 2005 Author retains full rights.

access to various resources within the GIAC Enterprise network.

- **Employees** – Located on the GIAC Enterprise's internal network
  GIAC Enterprises employs approximately 500 full-time employees. Majority are located at the HQ facility.
- **Mobile users** – sales and teleworkers
  There are approximately 25 full time sales people who travel endlessly both domestic and internationally. Good help is hard to find and GIAC has worked hard to help it's employees work from home (approval is required with business justification). At most, at any one time, no more the 100 users would need external access into GIAC Enterprises networks.

Business Design Considerations

Management has a number items they would like met:

- Security – Above all, the fortunes must not be comprised!!
- Reliability – When all possible, redundancy should be incorporated into the design.
- Manageability – The design must be manageable. GIAC Enterprises has three excellent Security Specialists on staff. There are no "contractor" dollars available and everything will have to be developed and maintained by this team.
- Scalability – The design should be flexible enough to meet GIAC's needs as the business grows without a major redesign. This means being able to accommodate additional internet traffic as new customers are brought in and being able to implement new technologies (either security/networking or business applications) to continually service the customers.
- Cost effective solutions – Security is expensive and more often then not this is the main reason for lax security. All areas that need securing will be secured but when a viable low-cost open-source solution is available, it will be evaluated and considered.

© SANS Institute 2000 - 2005 Author retains full rights.

GIAC Enterprises

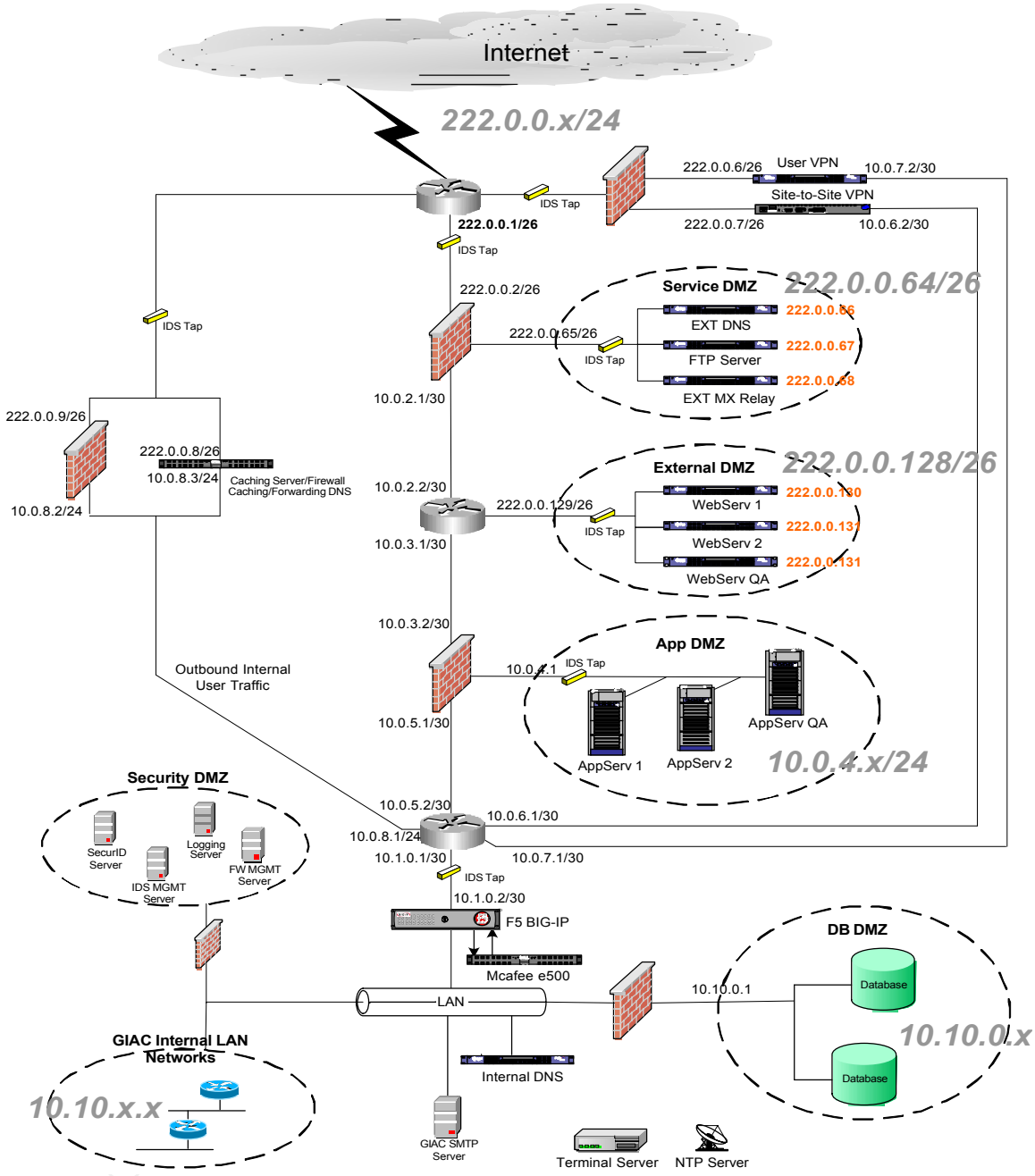


Figure 1 – GIAC Enterprise Network Design

© SANS Institute 2000 - 2005 Author retains full rights.

Security Architecture

Figure 1 shows a detailed technical drawing outlining GIAC Enterprise's DMZ architecture. Externally, GIAC Enterprises owns the class C address space of 222.0.0.0 (GIAC intentionally did not lease this address space from the ISP. This way, if we ever change ISPs, we do not need to re-address). Internally, all devices are based on RFC 1918 addressing *(http://www.ietf.org/rfc/rfc1918.txt)* using the class A address space of 10.0.0.0 and broken into class C segments within the GIAC internal network.

Through out GIAC Enterprises, a number of internal "Security Zones" have been established. We firmly believe that servers should be segregated by function a much as possible. This allows for a logically structured design, yet at the same time minimizes our risks if a server were to be compromised. "According to InterGov (*http://www.intergov.org*), insiders commit nearly 80 percent of all computer and Internet-related crime, causing an average loss of approximately $110,000 per corporate victim"[1]. Every measure is taken to ensure that all systems are equally protected, from internal and external threats. There are a number of internal DMZ zones – a database DMZ, Security Systems DMZ and a Development DMZ. Each one of these is separated from the LAN network by a dedicated firewall. Only external DMZ zones will be discussed in this paper.

Hardware considerations

GIAC Enterprises has chosen to standardize on select hardware platforms for majority of it's networking devices and production servers. Overall, this practice has been shown to be beneficial for GIAC Enterprises. This has proven to be cost effective by centralizing our hardware/software maintenance support contracts, reusability of hardware parts, and allows our IT staff to offer better support by specializing and becoming experts on a few select platforms. This eliminates trying to keep up with many different types of hardware and software support issues and stops us from wasting time trying to understand a product when there is a problem. We are then able to leverage our staff as a whole to support the environment and do not have to depend on only one individual who knows a specific technology. By working as a team and proactive cross training, no one person becomes a "silo".

Cisco routers and switches (being the de facto network equipment vender) were deployed in most cases, including site-to-site VPN. Cisco products are well supported through out the industry and our staff is familiar with their technologies. In some cases, were specialty "nitch" product solutions fit best, alternative venders were chosen (i.e. F5, Checkpoint User VPN, etc).

---

[1] Carr, p 42

© SANS Institute 2000 - 2005 Author retains full rights.

Sun Microsystems (http://www.sun.com) was chosen for our production server hardware platform, all running Solaris 8. For smaller systems (mail relay, DNS, etc), Netra T1s were deployed. For larger application and database servers, Enterprise Starfire class servers were deployed. The UNIX staff (in co-operation with the Security Team) has setup and configured a Solaris Jumpstart server to completely automate the Solaris OS install process. Jumpstart allows us to "script" the entire server build process – we are able to control what packages are installed (i.e. install only the minimum packages needed), patch the system with the latest cluster patches, and do all GIAC specific configuration and hardening automatically. Using Jumpstart gives us a repeatable, consistent build process that ensures that our systems are configured correctly. This process also lends well for D/R purposes and greatly improves our time getting new or replacement systems into production.

As for Intel based hardware, Compaq was chosen. Within our environment, we have a number of open-source OS solutions. And for nearly all of these, the Compaq hardware was a excellent solution. Their hardware has proven to be well supported (i.e. driver support) and is a reliable and cost effective solution.

Security Staff

The Network Security team consists of 3 Security Specialists and one very tenacious Desktop Support Analyst who is always there to help with any miscellaneous tasks. Our staff has a strong IT infrastructure background, having worked as application and/or server administrators for a number of years before moving into the network security field. This gives our team solid real life, hands-on insight into the inner workings of the applications and servers we are being asked to protect and support. We are all "UNIX nuts" and collectively have experience on nearly all UNIX variants and hardware architectures. Lucky, management has given the staff the freedom to choose the technologies we use and are given an active role in GIAC's technology direction.

Routing

The design of the network was kept simple for the use of static routing. We were fortunate enough to have been able to build the network more or less from scratch. GIAC did not have to worry about being forced into bad network design in order to maintain and support legacy network segments and equipment. For all routing in the DMZ, static routes are assumed.

IDS

- Though the deployment and configuration of IDS is beyond the scope of

© SANS Institute 2000 - 2005, Author retains full rights.

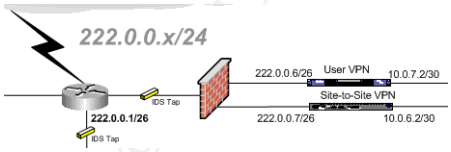
this paper, serious consideration was given in the design as to where IDS sensors would be placed. We wanted to make IDS an integral part of the network design. The primary goal was to make each network segment easy to monitor for both incoming and outgoing traffic. Fortunately this forces us to keep our design simple and straight forward. For each DMZ zone there is only one link for traffic to flow through. The IDS sensors should not in any way hinder performance. In the past, all IDS sensors were simple, "dumb" 10/100Mb hubs (not switches, not switching hubs, just dumb hubs!!!!). This solution, though simple and cost effective was not ideal in full duplex environments where performance is crucial. Port mirroring on the switches proved to be too CPU intensive in areas with large amounts of traffic.

The best network tap solution was the single port Finisar UTP Tap IL/1 *(http://www.finistar-systems.com – formally Shomiti)*. The IL/1 In-Line tap is able to monitor both sides of a full duplex link by "tapping" each directional communication individually. From an IDS perspective, you then have 2 communication streams that are to be monitored, in bound and out bound. This device is also passive, preventing any disruption in network traffic in the event the IL/1 were to fail (for example, a power supply failure).

Snort *(http://www.snort.org)* is being used as the IDS backend software. Snort uses a number of other support applications to log and process it's data. For storing the "alerts" and events collected, MySQL *(http://www.mysql.com)* an open-source SQL database, is used. For viewing the data, ACID *(http://acidlab.sourceforge.net)*, a web based tool for viewing the data and events collected will be used. This combination was able to offer us a low cost, robust, scalable network IDS solution. All IDS collectors are running OpenBSD 3.1 OS software on Compaq DL360s with 1GB memory.

Architecture Overview

Border Router/External Link



A T3 (45Mbps) is used for the main external link. One option was to use 2 external links (2 T1 or T3s for example) in combination with either BGP or some hardware link load-balancing solution (Radware or F5) to allow for high availability and network redundancy. Due to the cost of adding a second link and the past reliability of our ISP (as hard as it is to believe), this was not considered practical at this time. It should be noted that this network design was created to allow for future expansion (multiple links, BGP, etc) and this option will be revisited as are our requirement change. By logically addressing the DMZ zones and with GIAC Enterprises owning their own external class C address range (222.0.0.0/24), we should easily be able to connect up with a second service provider and use BGP for redundant network connectivity.

© SANS Institute 2000 - 2005 Author retains full rights.

Our border router is a Cisco 7206VXR running Cisco IOS 12.1.14. We have configured our router with a NPE-400 processor, 256MB or RAM and 128MB of flash. The 7206 connects to the external link via a HSSI interface. The 7206's internal interface is a GIG Ethernet Adaptor connected to the Gig Fibre uplink port of a Cisco 3524 switch, where all externally connected devices on the 222.0.0.x/24 network connect into.

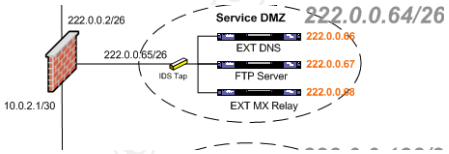
With the Border Router being the first point of contact to the outside world, extensive ACLs will be used to filter out extraneous traffic before it enters our DMZ zones. Being the central point for internal user internet traffic, user and site-to-site VPN and DMZ access, we wanted to eliminate any contention at this point. The 7206 also has enough expandability to accommodate future growth. By having a heavier router at our parameter, we will be able to implement a very tight security policy via ACLs and fend off resource intensive attacks such as DDoS attacks.

Directly off of this router are our VPN gateways. For security and performance reasons, each VPN link is segregated off based on it's role. We will have 2 VPN gateways:

- User VPN (remote sales and telecommuters access)
- Site-to-Site/Extranet connections (business partners)

Each one of these VPN links has different access requirements and this allows us to individually configure and secure these links as needed. For site-to-site VPN access we have a dedicated Cisco 3621 with an AIM/VPN Accelerator Card. Each one of the remote sites also has a Cisco router configured for VPN. User VPN support will be offered through Checkpoint NG running on a hardened Solaris 8 system equipped with a VPN-1 ACII card.

External Firewall/Service DMZ



Our firewall of choice is OpenBSD 3.1 running pf *(http://www.openbsd.org)*. There were a number of reasons for this choice. First - security. OpenBSD is considered one of the most stable and secure UNIXs (or OSs) available, commercial or open source. Second – our staff is very familiar with the OS and firewall software. Their strong UNIX background makes it easy to maintain and configure the OS and firewall. The open nature of the product allows us to use a number of unique features that are not found in other commercial products. Third, OpenBSD and pf are ported to a number of different hardware architectures, preventing us from getting tied into expensive proprietary hardware/software solutions. We can deploy our firewalls on the hardware that best suits our needs (cost and performance). Finally – it's free. At that price, we can deploy

© SANS Institute 2000 - 2005, Author retains full rights.

firewalls to our hearts content ;-)

As for hardware, all firewalls are Compaq DL360s with 2x18GB drives, 1x1.2Mhz Pentium III processor and 1 GB RAM. The DL360 has 2 integrated interfaces (one of which is attached to the management network) and two 32/64 bit PCI slots. Each firewall is configured with 1 Znyx ZX370 Quad ethernet card, which will be used for the firewall interfaces.

OpenBSD's firewall software, pf, is a true stateful inspection packet filtering firewall. This means that it maintains a record, or "the state", of the connections it allows through. By tracking it's allowed connections, the firewall can intelligently decide which packets are legitimate responses to packets that were initiated and prevent uninitiated packets from entering.

This firewall also serves as the entry point into our Service DMZ, where our External Service servers reside (FTP, SMTP, and External DNS). GIAC Enterprises has standardized on Solaris 8 for all production servers when ever possible (though when justified and tested, exceptions are made). The configurations of the servers are:

- DNS server – Solaris 8 (Hardened), Netra T1, BIND 9.2.1*
- External Mail Relay – Solaris 8 (Hardened), Netra T1, Sendmail 8.12.5*
- External FTP server – Solaris 8 (Hardened), Netra T1, Proftpd 1.2.6*

GIAC Enterprises runs a "split DNS" configuration, where one server is solely dedicated to external DNS and another server is dedicated only to internal DNS. The secure configuration of the DNS server entails allowing only responses from outside users. This server is not used by internal users or systems, so it's only purpose is to server out external GIAC DNS information. Also, recursive queries have been disabled and no zone transfers are allowed. GIAC has established a secondary external DNS server with a local ISP. Since our external DNS changes are made infrequently, we can easily manage our external zones manually. We also have disabled the "fetching" of glue records. This will ensure that our external DNS server does not cache any information.

Our External MX server is merely a SMTP mail relay, used for handling inbound and outbound mail in between the outside world and our internal mail server.

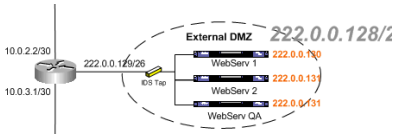
Finally our external FTP Server is used for the transferring of large files between customers and business partners. This is not an anonymous FTP server and a unique user account is required. We have chosen Proftpd for it's security and flexibility. Support was limited to only passive FTP client support.

* Latest release at the time of this writing

© SANS Institute 2000 - 2005 Author retains full rights.

The configuration of the FTP server was to keep the ftp data port range as narrow as possible. Proftpd supports this with the “PassivePorts” directive.

Central Router



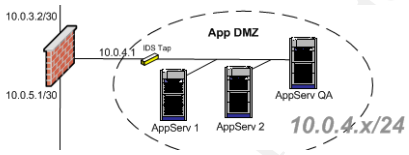
The Central Router is a Cisco 3621. This outer serves as the main route into our xternal DMZ where our web servers reside. The choice for web services was Apache 1.3.27. The server configurations for these servers are:

- Webserv1, 2 and Q/A – Solaris 8 (Hardened), Sun Netra AC200

Our security policy clearly states that external internet users never directly access applications or databases – all transactions must be handled through the application servers. To meet this requirement, externally accessible applications are design in a multi-tiered architecture.
Here once again we will setup ACLs to prevent any unwanted traffic from coming further into our DMZ area or LAN. At this point we should only see two types of traffic coming from “above”:

- Packets destine to the External DMZ zone originating from the External Router
- Packets originating from the Service DMZ servers going to their respective destinations internally.

Internal Firewall

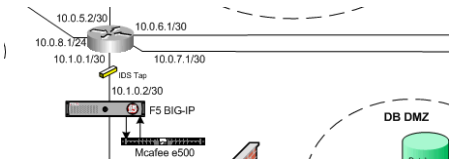


The Internal DMZ Firewall is also OpenBSD 3.1 running pf. This firewall also servers as the link to our App DMZ zone. The servers in this DMZ zone typically handle the majority of the processing load for the transactions. As part of our security policy, servers in the application DMZ **are not** directly accessed from the internet nor are they permitted to initiate traffic to the internet.

The App Servers configurations are:

- Sun Sunfire 280R
- Solaris 8 (Hardened)
- Hitatchi 7700 disk array system (connected via fibre SAN)

Internal Router



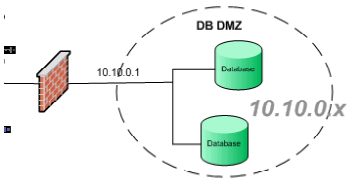
The Internal Router is also a Cisco 7206, and

© SANS Institute 2000 - 2005 Author retains full rights.

with respect to hardware resources (memory, cpu, etc) is configured identically to the external router. Interface wise, it is configured with 2 4port FastEthernet cards. This router is the main connection for internal users accessing the internet (via the dedicated "Outbound Traffic Link"), as well as the point between our DMZ space and our internal LAN network. This router has also been configured with ACLs to prevent any extraneous traffic from entering our LAN environment.

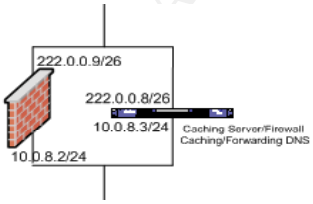
E-mail and web connectivity is critical for our business. The IT department has taken e-mail and web traffic security very seriously as spamming, e-mail borne viruses, and malware threats (included in JavaScript, Java and ActiveX content) are now rampant on the internet. To address this problem we have configured a sophisticated anti-virus and anti-spamming solution. By using the F5's Application Switch's content switching capabilities, we are able direct our inbound and outbound HTTP, SMTP, FTP, and POP3 traffic to the Mcafee e500 Webshield Appliance for virus scanning and anti-e-mail spamming before it gets into our network. Though the e500 supports screening for POP3, this will not be used since we do not support POP3.

Internal DB DMZ



The internal DB DMZ is a DMZ zone located inside GIAC Enterprise's LAN. This is where all GIAC databases reside. This area is accessed by both internal employees (either located locally or via individual user or partner VPN connections) and external customers **indirectly** through applications in the External DMZ, App DMZ or the lan. By design, all applications are tiered and do not require individual access to the databases (except for administration by admins). The firewall is configured with only explicit static host rules. These systems are Sun v880s running Solaris 8 with 8GB of memory. These databases are also attached via the SAN to the Hitachi 7700 disk array.

Outbound/External Default Route



All traffic initiated from GIAC Enterprise's LAN (i.e. internal users) destine to the internet is routed through the Outbound Internal User Traffic segment. This "bypass" connection prevents outbound traffic from having to pass through the DMZ network space and helps to keep our DMZ hardware resources dedicated to DMZ/production network traffic. Isolating just the Outbound network traffic also allows us to make use of a caching/proxing server for http, ftp, and other URL requests.

In our Outbound configuration, we have configured a Squid Caching Server -

© SANS Institute 2000 - 2005 Author retains full rights.

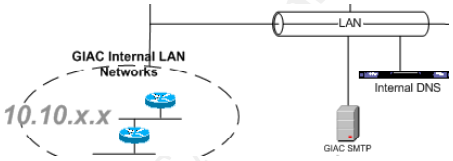
version 2.4* (http://www.squid-cache.org) running on OpenBSD 3.1 with pf. In parallel, we have also configured a stand alone “GIAC issue” firewall. Basically, all internal traffic destine to the internet first comes through the Internal Router and is routed to the Outbound Bypass. We then route all http, ftp (in HTTP), and gopher traffic through the caching device. All other “non-web” traffic is routed out the parallel firewall to the internet. Both firewalls are configured to “allow only authorized traffic out and deny all in”. This will be complimented by the Border and Internal Router ACLs.

The Squid Cache Server offers a number of benefits. First, by caching users web requests, we can improve internet performance for commonly visited site by using locally stored copies of data verses retrieving the same content repeatedly from the actual site. Squid also offers us added security by acting as a “proxy” for the clients.

The Cashing Server also serves as our external DNS caching/forwarding server. Internal users use the Internal DNS server and all external DNS requests are forwarded to the External Caching DNS server, which in turn is forwarded out to the internet for resolution. As for software, it is running BIND 9.2.1. Since this server is only a caching/forwarding server, it contains no DNS zone data, only cached lookups. It is configured to perform recursive lookups only from authorized internal networks.

To support our outbound user traffic, we will be using PAT (hide NAT) on the standalone firewall and the caching server/firewall. pf supports a number of NAT configurations and features. This will allow us to maintain our private addressing internally and provide our users with a valid publicly routable address.

Internal Services and LAN



Our internal services consist primarily of a internal DNS server, running BIND 9.2.1* on a Sun Netra T1 running Solaris 8 (Hardened), 2x18GB Drives (mirrored) with 1GB of memory. As mentioned before, all internal users use this server as their primary DNS server, and all external DNS requests are forwarded on to the DNS Caching server for resolution. The SMTP mail server is Solaris 8 (Hardened) using Sendmail 8.12.5* running on a Sun V480.

Security DMZ Zone

A serious concern to GIAC is the security of the security servers themselves. This consists of the management servers, logging servers and user authentication servers that support our security devices within GIAC

© SANS Institute 2000 - 2005 Author retains full rights.

Enterprises. These servers are kept in a isolated zone protected by a dedicated firewall to prevent unauthorized access.

Terminal Server

GIAC Enterprises has taken a very strict stance on how network devices are accessed for administration. The policy states that all devices have secure terminal/console access for administration. Our solution was to use Lightwave's SCS3230 Secure Console Server *(http://www.lantronix.com)*. The SCS3230 is a 32 port serial console server that allows remote management of nearly any serial console/CLI device (Linux/*BSD systems, Sun, HP-UX, SGI, Nokia, Cisco, etc) through the remote device's console serial port. The device also has fully redundant power supplies and advanced user management allowing specific users access only to the devices they have been authorized to use. Also, the console activity for each systems connection can be logged to a remote syslog server to audit user activity or system messaging.

The key benefit for the SCS3230 is the support for SSH. This allows us to securely login to a remote device's console using SSH. This has allowed us to take our security policy a step further and for all routers and switches, we have disabled all remote access. Management is only allowed through the console over SSH.

SecurID Server

Secure authentication is a requirement for all server access. Our solution here is RSA's SecurID 5.0. SecurID is a two-factor authentication scheme where each user has a pre-defined 4 digit PIN and a "token" device (either software based or a small standalone hardware device). The token generates a new 6 or more digit pseudo-random value every 60 seconds. The users combines their PIN and the current token value to create a "passcode" with which they can use to authenticate against the SecurID ACE Server. Once the user successfully authenticates with the SecurID server, the user is allowed access to the resource *(http://www.rsasecurity.com/rsalabs/faq/5-2-5.html)*.

Two-factor has proven to be the most secure yet manageable solution. RSA's SecurID is well respected through out the industry and many applications support authentication through SecurID (either directly, by proxy, or can be easily integrated into custom applications through it's API). Our SecurID server is a Sun Netra T1 running Solaris 8.

Logging Server

We have configured a logging server for all of our devices to send their logging information via syslog. The firewall is able to send statistics via symon (*http://www.xs4all.nl/~wpd/symon*). Firewall logs are able to monitored with

© SANS Institute 2000 - 2005 Author retains full rights.

swatch and fwanalog (*http://tud.at/programm/fwanalog*). And when ever possible, the devices are configured to transmit the logs securely (either via ssh, stunnel, etc). It is our practice to not store any logs directly on the device them selves (except in the event that communications with the logging server is lost, some devices will hold the logs until communication is restored). Our server is a Sun Enterprise 450 with 4 UltraSparcII Processors, 2GB RAM and 500GB of disk storage. At the present time, we are retaining all logs on disk for 90 days. Once a month, the oldest months logs are tarred and archived on to tape.

Assignment 2 – Security Policies and Tutorial

Border Router - Tutorial

Our Border router is our first line of defense from the internet. As such, it has been designed to filter out and shield the network from scans and network type attacks. Our policy states:

- The Border Router should not be remotely accessible (i.e. telnet, ssh, etc), either internally or externally.
- As much as possible, the router should be configured not to respond to any types of network reconnaissance (pings or traceroute).
- Access to GIAC's NTP server is permitted though the Border Router's internal interface and explicit host ACL rules only.
- The router should **only** allow traffic in destine to: the 222.0.0.64/26 (Service DMZ), the 222.0.0.128/26 (External DMZ) or the Site-to-Site (222.0.0.7) and User VPN (222.0.0.8) router/gateways.
- Access Control Lists:
    - When all possible, use specific host and protocol ACLs.
    - Always use specific protocol to zone ACLs.
- Allow traffic out from internal users. Do not allow any connections to initiate in from the internet to the Outbound traffic link.
- The Border Router is not required to provide stateful inspection for traffic (responsibility of the firewall). But the access control should be designed to compliment the firewall configuration.

For the Border Router (and all routers in the DMZ), we have chosen to harden them by disabling all unnecessary services and disabling all remote access. It should be noted that all switches have intentionally not been assigned an IP address to disable network access (remotely accessible by Console Server). Being that our Border Router is so exposed, we have adopted a very simple policy:

Border Router = No Access

© SANS Institute 2000 - 2005 Author retains full rights.

This basically means that all externally facing interfaces are configured to drop all IP packets destine directly to them. Access to this router is only allowed through the console port (via the Console Server). Though this is considered secure, we have implemented a number of router hardening best practices to disable all unneeded and insecure services, encrypt the passwords, etc. Disabling the unused services also allows us to conserve CPU and memory resources. We have written a simple Router Hardening Procedure/Checklist that is followed for the initial configuration of each router that is deployed. This ensures that our router environment is consistent. Then each router is configured accordingly (ACLs, routes, etc) respective to it's routing responsibilities. It should be noted that we have explicitly disabled functions and services that are "reportedly" disabled by default for completeness.

GIAC Cisco Hardening Procedure

For all commands below, working knowledge of Cisco devices is required. All commands below are to be entered in config mode:

```
Rt001#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Rt001(config)#
```

User Login and Password

- ❑ Assign the router a hostname:

```
hostname <name>
```

Obviously, the router should have a name. It is good practice to not use a name that is too revealing as to it's owner or function.

- ❑ Disable Virtual Terminal Access

```
line vty 0 4
password <password>
transport input none
```

Even though we are not allowing telnet access to our routers, we have chosen to implement a password for our virtual terminals for completeness.

- ❑ Disable aux port

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

From a physical security standpoint, the auxiliary port should be disabled since we are not using them.

- ❑ Assign privileged/enable password

```
service password-encryption
enable secret <new_password>
```

Here the "service password-encryption" encrypts all user passwords. "enable secret" encrypts the enable password with MD5 (Cisco Type 5).

- ❑ Enable logging to our logging server

```
logging on
logging 10.10.10.112
logging buffered 10240 debug
```

© SANS Institute 2000 - 2005, Author retains full rights.

```
no logging console
```

We send all logging information to our log server. The buffered debug command ensures that we are capturing enough log information. Though logging to the console is helpful when working on the system (sometimes), it is best to leave it off to stop console port overruns and can be enabled manually if needed.

© SANS Institute 2000 - 2005, Author retains full rights.

- Configure logging timestamps

```
Service timestamps debug datetime localtime msec
Service timestamps log datetime localtime msec
```

Make sure that our logging timestamps have a real and accurate timestamp. Also, configure the ntp service to ensure we are in sync (see below).

- Configure the login banner

```
banner # Warning: Authorized personnel only! #
```

For all systems with logins we have added a banner informing unauthorized "evildoers" to stop and desist!

Disabling of Services

- Disable SNMP

```
no snmp-server
no snmp-server community public RO
no snmp-server community admin RW
```

Though SNMP is a valuable tool for collecting system statistics, it is widely considered insecure. Though you can use SNMPv2's MD5 authentication, all data is transmitted in clear text. SNMP is disabled by default, but this ensures it is not accidentally configured.

- Disable tcp and udp small services

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

On Cisco routers, a number of diagnostic services are enabled. These include Echo (port 7), Discard (port 9),and Chargen (port 19) (both UDP and TCP). These services can be used in in a DoS attack. It's best to disable them.

- Disable finger

```
no ip finger
```

The finger service can be used to view information about User activity on the router. There is nothing good about this, so this is service is turned off.

- Do not allow attempts at downloading config from remote tftp server

```
no service config
```

Cisco's have the ability to download configs from remote tftp servers at reboot. This could cause serious problems if there were a rogue tftp server on the network and someone wanted to load their own config into the router. This should be disabled.

- Disable pad

```
no service pad
```

We are not running X25, so there is no need for the Packet Assembler/Deassembler.

- Disable DHCP

```
no service dhcp
```

We do not want our router to send out dhcp requests.

- Disable cdp

```
no cdp run
```

CDP (Cisco Discovery Protocol) allows Cisco devices to share basic information with other Cisco devices on the same network segment regarding device type, model, etc. We have no real need for this so it is disabled.

© SANS Institute 2000 - 2005, Author retains full rights.

- Disable http config service

```
no ip http server
```

Since our policy is to only allow access through the serial Console port, the http web interface service is disabled

IP and Miscellaneous Commands

- Disable DNS lookups

```
no ip domain-lookup
```

Disable the automatic name lookups. This quite honestly is an annoying "feature".

- Set the time and zone

```
clock timezone CST -6
clock summer-time CDT recurring
```

Here we have set our timezone to Central Standard Time and made it aware of the "summer-time" daylight savings time.

- Configure ntp client on internal interface

```
ntp server 10.10.10.4 source eth0/0
```

GIAC has it's own Stratum 1 satellite receiver. This allows us to have time synced for all devices. Our router will only be allowed to us it's internal interface for time sync.

- Specify boot location and configuration

```
boot system disk0:<image_name>.bin
no boot network
```

Boot system insures that the correct image is loaded. And for completeness, disable all network configuration file command.

Networking/Routing

- Allow for 0 subnet routing to Internet

```
ip subnet-zero
```

This tells Cisco routers to allow the use of 0 subnet networks.

- Disable multicast routing

```
No ip multicast-routing
```

As a safeguard, ensure that this is disabled.

- Disable IP source routing

```
no ip source-route
```

Source routed packets should never be allowed on the network. More then likely, a packet that has explicit routing is being used for malicious purposes.

- Disable bootp

```
no ip bootp server
```

This service is to allow other routers to boot from this router. Since we are not a bootp server, this will be disabled.

Interface Specific

All interface specific commands are to be entered in on a per interface basis by entering into config mode of that specific interface. For example to enter config mode of the primary serial interface, from config mode you would type:

© SANS Institute 2000 - 2005, Author retains full rights.

```
RtO01(config)#interface serial0/0
RtO01(config-if)#
```

- Configure ntp on internal interface only, disable on external interfaces

```
ntp disable
```

This pertains more to the border router, where it should not see look off it's external interfaces for ntp, only internal trusted networks.

- Disable arp proxy

```
no ip proxy-arp
```

Cisco routers can "proxy" arp requests across networks. This really isn't useful and by disabling this, helps to keep our networks isolated.

- Disable cdp on individual interfaces

```
No cdp run
```

Though we disabled this in global config mode, as a best practice, disable it on each interface individually also.

- Disable route cache

```
No ip route-cache
```

At one time, there was a know issue with using route-caching and extended ACLs[3]. This has since been resolved but should verify it is disabled since we are not using it.

- Disable ip directed-broadcast (individually, on each interface)

```
no ip directed-broadcast
```

Prevent direct broadcasts through each interface which could cause a DoS attack.

- Disable ICMP messages

```
No ip redirect
No ip mask-reply
No ip unreachable
```

ICMP is an excellent network mapping tool, and each one of the above lend to it very nicely. ICPM redirects, mask-reply, and unreachable should be disabled.

Border Router ACLs

Once all services on the router have been configured, some generic network specific ACLs (Access Control Lists) must be applied. ACLs give us the ability to packet filter based on source, destination and port. As each packet enters the interface, it is evaluated against the access lists. When there is a match, the specific access list action is taken: that is either to "deny" and drop the packet or "permit" it through to it's destination. It is important to remember that ACLs on Cisco devices are handled on a first match basis. That is, the router will process the packet based on the first match within the ACL, starting from

[3] **Cert Advisory CA-1992-20 Cisco Access List Vulnerability - http://www.cert.org/advisories/CA-1992-20.html**

© SANS Institute 2000 - 2005 Author retains full rights.

the top. This means that the list must properly be ordered to ensure that packets are not accidentally passed before they are matched to their “deny” rule. Likewise, packets you want to see passed could be prematurely dropped. It should be noted that rule order is also important if for efficiency. Without exception, each packet will be compared against every entry in the ACL until a match is found. You should make sure that the most frequently used rules are at the top so as not to waist system resources continually checking rules that don’t apply. Since the packet is either accepted or dropped once it successfully matches the first applicable rule, keeping the most frequently used rules first help processing performance.

There are two types of ACLs on Cisco devices: standard and extended. Standard access lists only match based on the source IP address of the packet. Extended access lists are more granular, matching rules based on source, destination, port and protocol. Standard access lists, though limiting, are useful for the simple permitting and denying of specific host traffic, where extended ACLs offer much more control. Extended access lists also offer a primitive form of stateful inspection when used with the “established” or “reflexive” switch. This is not the most secure method of stateful inspection as it only evaluates whether it is a new packet or a packet part of an existing connection based on the SYN bit setting. There are a number of attacks designed specifically to exploit packet filters that match only on the SYN bit setting. Stateful inspection firewalls use a much more sophisticated method for checking state. Because of this, it was decided not to use the reflexive option and let the firewall handle the stateful connections, the routers primary goal will be filtering and routing.

The basic syntax for Extended ACLs are:

```
access-list <acl_number> <action> [protocol] [source] <port> [destination] <port>
```

Where:

```
Acl_number – 100-199 (for extended ACLs)
Action     - permit | deny
Protocol   - tcp | ip | udp | esp | gre
Source     - network (with bitmask) or hostname
Port       - port with matching qualifiers
Destination- network (with bitmask) or hostname
Port       - port with matching qualifiers
```

Example – To create a rule in access list 110 to allow tcp packets from any IP source with destination address of 222.0.0.130 using only port 80 and log the transaction would look like:

```
access-list 110 permit tcp any host 222.0.0.130 port 80 log
```

Finally, on the interface that the ACL is to be applied, you add the ACL and state if it is an ACL for inbound traffic on that interface or outbound. You can only have one ACL group for in and one ACL group for out per interface. To

© SANS Institute 2000 - 2005 Author retains full rights.

apply access-list 110 to serial0/0 for all inbound traffic, you can enter config mode for that interface and add the ACL:

```
Rt001(config)#interface serial0/0
Rt001(config-if)#access-list 110 in
```

Filtering can take place two ways, inbound and outbound. This gives you the option of controlling not only what comes in the router but also deciding what can leave the device. Eitherbound filtering, as its known, can cause performance problems. As stated above (assuming an ACL is applied and configured), every packet that comes into the router is first evaluated against the ACL for that interface, one line at a time. If the packet matches a "permit" statement, it is then allowed to proceed on for routing, natting, encryption, etc.[4] But in order for the packet to leave the router, it must go though the outbound ACL and be evaluated once again. Using both inbound and outbound filtering can be very resource intensive. For this reason, I will implement only basic outbound filtering.

For GIAC's Border Router, we have implemented two ACLs, access-list 110 for the external interface and access-list 120 for the internal interface. As for logging, denys will be logged and permits will not. We are interested in seeing what we are not letting in verses traffic we have already allowed out.

```
!-----------------------------------------------------------------
!--- External Interface - External to Internal
!-----------------------------------------------------------------
! Block all traffic from reserved and unregistered addresses. This list is
! based on the reserved IP addresses found at IANA:
! http://www.iana.org/assignments/ipv4-address-space
! 222.0.0.0/24 was used for our example for external IP range
! Another option would be to route these to null but it would be nice to see what kind
! of traffic we are dropping with logging.
 access-list 110 deny ip 1.0.0.0 0.255.255.255 any log
 access-list 110 deny ip 2.0.0.0 0.255.255.255 any log
 access-list 110 deny ip 5.0.0.0 0.255.255.255 any log
 access-list 110 deny ip 7.0.0.0 0.255.255.255 any log
 access-list 110 deny ip 23.0.0.0 0.255.255.255 any log
 access-list 110 deny ip 27.0.0.0 0.255.255.255 any log
 . . .
 . . .
 access-list 110 deny ip 197.0.0.0 0.255.255.255 any log
 access-list 110 deny ip 221.0.0.0 0.255.255.255 any log
 access-list 110 deny ip 223.0.0.0 0.255.255.255 any log
!
! Deny any packets without a source address
!
 access-list 110 deny ip host 0.0.0.0 any log
!
! Drop packets from the RFC1918 address ranges, the standard multicast addresses and
! the default Microsoft DHCP address range.
!
 access-list 110 deny ip 10.0.0.0 0.255.255.255 any log
 access-list 110 deny ip 172.16.0.0 0.15.255.255 any log
 access-list 110 deny ip 127.0.0.0 0.255.255.255 any log
 access-list 110 deny ip 192.168.0.0 0.0.255.255 any log
```

[4] An interesting article I found (actually dealing with NAT) describes Cisco's Order of Operation – http://www.cisco.com/warp/public/556/5.html

© SANS Institute 2000 - 2005 Author retains full rights.

```
 access-list 110 deny ip 169.254.0.0 0.0.255.255 any log
 access-list 110 deny ip 240.0.0.0 15.255.255.255 any log
 access-list 110 deny ip 224.0.0.0 7.255.255.255 any log
!
! Do not allow for any direct access to the router interfaces or the outside interface
! of the firewall. Telnet has been disabled on the routers and console login is
! available. This ensures that the router or firewall is not the target and
! destination of an attack.
!
 access-list 110  deny ip any host 1.2.3.4 log
 access-list 110  deny ip any host 222.0.0.2 log
 access-list 110  deny ip any host 222.0.0.1 log
!
! Protect our internal network from spoof attacks and same src and dest packets. We
! should not allow any direct traffic to our network and broadcast addresses, for each
! network. Also, drop any traffic that looks to be initiated from our address space of
! 222.0.0.0
!
 access-list 110 deny ip 222.0.0.0 0.255.255.255 any log
!
 access-list 110 deny ip any host 222.0.0.0 log
 access-list 110 deny ip any host 222.0.0.63 log
!
 access-list 110 deny ip any host 222.0.0.64 log
 access-list 110 deny ip any host 222.0.0.127 log
!
 access-list 110 deny ip any host 222.0.0.128 log
 access-list 110 deny ip any host 222.0.0.191 log
!
 access-list 110 deny ip any host 222.0.0.192 log
 access-list 110 deny ip any host 222.0.0.255 log
!
! Block ICMP probing. Here we prevent a number of reconnaissance type scans for
! traceroute and ping
!
 access-list 110 deny icmp any any host-unreachable log
 access-list 110 deny icmp any any time-exceeded log
 access-list 110 deny icmp any any echo-reply log
!
! Always allow established traffic initiated from the inside back through from the
! internet.
!
 access-list 110 permit tcp any any established
!
! The User VPN and Site-to-Site VPN gateways require access from the internet
! for AH, ESP, and IKE.
!
 access-list 110 permit 50 any host 222.0.0.7
 access-list 110 permit 51 any host 222.0.0.7
 access-list 110 permit udp any host 222.0.0.7 eq 500
 access-list 110 permit 50 any host 222.0.0.8
 access-list 110 permit 51 any host 222.0.0.8
 access-list 110 permit udp any host 222.0.0.8 eq 500
!
! DNS access. This allows requests TO our external DNS server, but only using UDP.
!
 access-list 110 permit udp any host 222.0.0.66 eq 53
!
! Permit HTTP and HTTPS in to their respective webservers.
!
 access-list 110 permit TCP any host 222.0.0.130 eq 80
 access-list 110 permit TCP any host 222.0.0.131 eq 443
!
! Allow access to smtp-relay on port 25.
!
 access-list 110 permit tcp any host 222.0.0.68 eq 25
!
```

© SANS Institute 2000 - 2005 Author retains full rights.

```
! For FTP, only passive mode will be supported and there will be no support for
! anonymous FTP access, only specific user account access will be allowed. To
! minimize our exposure with passive FTP (requiring client access to ports above
! 1024) we have used the PassivePorts directive on the ftp server (running
! Proftpd 1.2.6). This tells the server exactly what ports are allowed to be used
! for the data connection. Likewise, on the access list, only those specific ports are
! allowed in and only to the ftp server itself.
!
 access-list 110 permit tcp any host 222.0.0.67 eq ftp
 access-list 110 permit tcp any host 222.0.0.67 range 55000 55999
!
! Finally, we want to explicitly deny and log anything that didn't match the above
! rules.
!
 access-list 110 deny ip any any log

!-----------------------------------------------------------------
!--- Internal Interface - Internal to External
!-----------------------------------------------------------------
!
! Make sure we don't accept packets with no source IP address. We should never see
this,
! so log them.
 access-list 120 deny ip host 0.0.0.0 any log
!
! We should never see packets "leaking" with source address of 10.x.x.x. If we
do see
! them, log them.
!
 access-list 120 deny ip 10.0.0.0 0.0.0.255 any log
!
! Only allow out packets with a source IP from our valid IP address range. This
should
! be allowed so there is no need for logging.
!
 access-list 120 permit ip 222.0.0.0 0.0.0.255 any
!
! We do not allow pings in and we drop any ICMP pings replies back out. We
should log
! this since we should not see any traffic here. We should also allow "packet-
too-big"
! packets for our internal servers
!
 access-list 120 deny icmp any any echo-reply
!
! Time-exeeded ICMP messages are the basis for the traceroute program, which can
be
! used to map our network. We want to prevent traceroute responses from leaving
our
! network, but still allow valid ICMP messaging.
!
 access-list 120 deny icmp any any time-exceeded
!
! We have only one permit rule for IP traffic. We will drop and log all other
traffic.
!
 access-list 120 deny ip any any log
```

External Firewall

The External Firewall serves as the link (or barrier if you will) between our Border Router and our externally accessible DMZ zones and LAN. "All too often, the de facto security policy is whatever was implemented at the router/or firewall. I should be the other way around: the organization's policies should dictate the implementation at the firewall"[5] The placement of this firewall offers

© SANS Institute 2000 - 2005 Author retains full rights.

us defense-in-depth by offering a second type of security. Our firewalls are OpenBSD 3.1 using pf. Each firewall node is attached to the Management Network via a dedicated interface. This same network also servers as the logging network, which goes back to the Management DMZ. The firewalls and servers themselves can only be accessed via this network (or secure console access via the Console Servers). Our External Firewall policy states:

- All traffic to and from the firewall should be controlled though a stateful inspection mechanism and filtered.
- All packets originating from the internet coming to the firewall should only be destine to either the Service DMZ or External DMZ only. This policy should also be reinforced at the Center router.
- No traffic should be allowed to pass to a 10.x.x.x network address originating from the Internet.
- Access to GIAC's NTP server is permitted though the internal interface and explicit host ACL rules.
- Service DMZ Zone
    - All rules governing traffic to and from the Service DMZ are to be host specific whenever possible. Access from the internet to Service DMZ servers is allowed.
    - FTP will only be allowed via Passive Mode or secure SSH. No anonymous ftp account or server will exist.
    - Internal access to the FTP server will be allowed based on business justification, only via SSH/SFTP.
    - Only traffic from the mxrelay to the GIAC internal mail server is allowed bi-directionally via a host specific rule (this is to allow out going smtp mail to be relayed out to the internet).
    - Access for administration to the servers will be allowed either through the Secure Console or the Management Network only using SSH and SecurID authentication.
- External DMZ Zone
    - All packets sent from the internet to the External DMZ should be filtered based on port - only port 80 and 443 are allowed.
    - Access for administration to the servers will be allowed via the Management network only, through SSH and SecurID authentication.
    - Only the External DMZ web servers will have access to the App DMZ servers. (i.e. the External DMZ web servers will go though the App DMZ servers to communicate with the DB DMZ servers).

- All transactions will be logged via the Management/Logging Network.

[5] Brenton, pg 489

© SANS Institute 2000 - 2005 Author retains full rights.

pf

pf (*http://www.benzedrine.cx/pf.html*) is the OpenBSD stateful inspection packet filtering firewall that is now part of the OpenBSD 3.x kernel. Previously, OpenBSD's primary firewall software was ipf. Due to either a "misunderstanding" or a change in the licensing of ipf, ipf was found to in violation of the current BSD license and was removed. The OpenBSD group chose to do a complete rewrite and integrate it into the OS. This allowed them to better integrate it into the OS and kernel, incorporate a number of new features and work around some design limitations within ipf. There are a number of similarities to ipf syntax wise, though rule bases are not 100% reverse compatible (i.e. some ipf commands are not applicable to pf and new commands have been added to replace or simplify similar ipf functions).

pf is command line based. The firewall rules are configured in the pf.conf file (and naturally NAT rules are located in the nat.conf file). All configuration directives for pf are configured in these two files. For our Border Firewall, NAT was not needed since only the External DMZ and Service DMZ are allowed direct access to the internet.

Rule Order

A tool included with pf allows the administrator to view statistics of the firewalls. The pfctl command is a control program for pf which allow you to monitor various types of statistics from the firewall, control the loading and unloading of rulebases and set various tcp, udp, and icmp characteristics (i.e. timeout values, stateful connection properties, etc.). This can help to monitor traffic on a production day and understand how much each rule is being used. By monitoring the traffic through each rule, we can organize our rulebase in a more efficient manor. This allows for more efficient processing of the packets and overall better performance.

A unique feature of pf called "skip steps" is used to automatically optimize it's rule set evaluation. Skip steps basically means that pf will see the rules grouped together based "parameters": interface, protocol, source and destination address, etc. pf will then calculate the rules (or "steps") matching each of these parameters and figure how many steps there are for each parameter. If a packet comes in on interface fxp1, and there are 12 rules that match "inbound on fxp0", pf will skip these rules. At the very least, if there were no parameter associations made for a particular packet, all rules are evaluated and performance remains the same as if all rules were evaluated as normal. This helps to ensure that traffic most commonly used on a particular interface, ip address source and destination IP addresses and port are evaluated quickly.

pf has a unique rule matching mechanism where as the packets enter the

© SANS Institute 2000 - 2005 Author retains full rights.

firewall (or exit), they are evaluated in sequential order. The last matching rule dictates the packets fate, either to pass or deny (this is opposite of Checkpoint and Cisco ACLs). The one exception is when a rule uses the “quick” option, in which case all other matching is stopped and the packet is processes according to that rule.

Firewall Rule Policy

```
#
# GIAC Enterprises pf.conf - borderFW
# $Id: pf.conf,v 1.4 2002/09/29 13:33:33 root Exp root $

#____________________________________________________________
#
# Define network objects. pf allows for variable expansion which allows us to create
# network objects and groups of either networks or individual hosts.
#____________________________________________________________
extIF="fxp0"
srvDmzIF="fxp1"
srvDmzIPs="222.0.0.64/26"
extDmzIF="fxp2"
extDmzIPs="222.0.0.128/26"
extBadIPs="{ 222.0.0.0/24, 192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12, 127.0.0.0/8,
0.0.0.0/8, 255.255.255.255/32, 224.0.0.0/3, 0.0.0.0/32, 169.254.0.0/16 }"
intBadIPs="{ 192.168.0.0/16, 172.16.0.0/12, 127.0.0.0/8, 0.0.0.0/8, 255.255.255.255/32,
224.0.0.0/3, 0.0.0.0/32, 169.254.0.0/16 }"
intGiacNet="10.0.0.0/8"
srvDmzNet="222.0.0.64/26"
bdrRtrIntIF="222.0.0.1"
dns_Server="222.0.0.66"
ftp_Server="222.0.0.67"
extMX_Server="222.0.0.68"
extDmzNet="222.0.0.128/26"
www_Server="222.0.0.130"
intMX_Server="10.10.10.2"
intDns_Server="10.10.10.3"
ntp_Server="10.10.10.4"
log_Server="10.10.10.5"
rsa_Server="10.10.10.6"

#____________________________________________________________
#
# pf supports a function called normalizing. This makes sure that all incoming traffic #
is defragmented (performing full packet reassembly) and that they contain no
# abnormalities. This is a good practice, though it can be resource intensive.
#____________________________________________________________
scrub in all label "Scrub_in"

#____________________________________________________________
#
# Allow all loopback traffic – the firewall is working at the kernel level and all
# traffic must pass through the packet filter. We should allow all packets through
# the system’s internal network.
#____________________________________________________________
pass in quick on lo0 all label "loopbk_in"
pass out quick on lo0 all label "loopbk_out"

#____________________________________________________________
#
# Anti-spoofing rule – We want to drop packets with suspicious sources. We have already
# done extensive filtering on the Border Router using ACLs to block inbound traffic.
# Here we have included a segment of addresses more relevant to our internal network to
```

© SANS Institute 2000 - 2005 Author retains full rights.

```
# prevent spoofing on interfaces. A rule was also added so we don't send thing out.†
#_____________________________________________________________
block in log quick on $extIF from $extBadIPs to any label "dropBadAddrsIn"
block out log quick on $extIF from any to $intBadIPs label "dropBadAddrsOut"
block in log quick on $extDmzIF from ! $intGiacNet to any label "noSpoofDmz"
block in log quick on $srvDmzIF from ! $srvDmzNet to any label "noSpoofSrv"

#_____________________________________________________________
#
# This is our drop all policy. We want to allow only explicitly defined packets in.
# pf can return resets to tcp connections and return-icmp port unreachable messages to
# the source host of packets it is dropping. This helps to be more stealthy and look
# more like a host without services running vs. a firewall that is intentionally
# blocking something. For all connections, the "keep state" word is used to perform
# stateful inspection. Another feature of pf for TCP packets is "modulate state". Pf
# will randomized the ISN for better protection (keep state is implied when modulate is
# used). And for sanity keep state and allow all connections out.
#_____________________________________________________________
block return-rst in log all label "block_in_all"
block return-icmp in log on $extIF proto udp all label "blockInUdpAll"
block return-rst in log on $extIF proto tcp all label "blockInTcpAll"
Pass out log inet proto tcp all flags S/SA modulate state label "alltcp_int->out"
Pass out log inet proto { udp, icmp } all keep state label "allU/I_int->out"

#_____________________________________________________________
#
# Allow ICMP to be initiated from the Service and External DMZ but don't let
# anyone ping us ;-) Note: After hours of extensive research, it appears that pf does
# not send return-icmp messages back for ICMP, only TCP/UDP. None the less, I have
# left the return-icmp statement in with the block ICMP rule. It might start working
# someday.
#_____________________________________________________________
pass in log on { $extDmzIF, $srvDmzIF } inet proto icmp all icmp-type 8 code 0 keep
state label "icmpPingFromInside"
block return-icmp(port-unr) in log quick on $extIF inet proto icmp all icmp-type 8 code
0 keep state label "returnIcmpPing"

#_____________________________________________________________
#
# GIAC has a split DNS and this server is for outside user GIAC DNS resolution only.
# Allow all DMZ zones to query this server for reverse lookup authentication if needed.
# Only accept UDP traffic (no zone transfers).
#_____________________________________________________________
pass in log quick proto udp from any to $dns_Server port 53 keep state label "p53u:any-
>dns_Server"

#_____________________________________________________________
#
# Secure authentication is absolutely required for any server in the DMZ that requires
# remote access (though console only access is strongly encouraged). GIAC Enterprises
# has a SecurID ACE Server located in the Security DMZ and SecurID Agent communication
# over UDP 5500 is allowed.
#_____________________________________________________________
pass in log quick proto udp from $srvDmzNet to $rsa_Server port 5500 keep state label
"rsa5500:srvDmz->rsaSrv"

#_____________________________________________________________
#
# Time synchronization is critical for logging. GIAC requires that all network devices
# be synced with our NTP device. NTP access is only allowed for the Border Router via
# it's internal interface and for servers in the Service DMZ to the NTP Server.
#_____________________________________________________________
pass in log quick on $srvDmzIF proto udp from { $bdrRtrIntIF, $srvDmzNet } to
$ntp_Server port 123 keep state label "p123:srvDmz->ntpSrv"
```

† 222.0.0.0/24, an IANA reserved address range, is the external address range used for this paper. I have applied the rules as if it were a publicly routable address range.

© SANS Institute 2000 - 2005 Author retains full rights.

```
#_____________________________________________________________
#
# For our web servers, allow HTTP and HTTPS to our web server DMZ. This rule allows for
# "any" source which allows for both external users and internal GIAC employees to
# access our websites. This was chosen as apposed to routing the internal users though
# the Outbound connection then back in through the firewall. For TCP packets, pf is
# able to filter based on the flag settings. We will only allow TCP packets with the
# SYN flag to create a state connection.
#_____________________________________________________________
pass in log quick proto tcp from any to $extDmzNet port { 80, 443 } flags S/SA keep
state label "p80:any->wwwSrv"

#_____________________________________________________________
#
# Our external mail relay has a number of communication paths. It needs to be able
# receive incoming mail from the internet and then forward on to the internal GIAC
# mail server. It must also server as a mail relay for SMTP mail traffic coming from
# the internal LAN to the internet.
#_____________________________________________________________
pass in log quick proto tcp from any to $extMX_Server port 25 flags S/SA keep state
label "p25:any->extMxSrv"
pass in log quick on $extDmzIF proto { tcp, udp } from $intMX_Server to $extMX_Server
port 25 flags S/SA keep state label "p25:intMxSrv->extMxSrv"

#_____________________________________________________________
#
# For FTP, only passive will be supported. Our FTP server is using ProFTPd 1.2.6 which
# has an option "PassivePorts", where you can control exactly what hi-ports the ftp
# data channel will use. The rule below states that we will "allow from any source to
# the ftp server only ports 55000 to 55999", and the ftp server is configured only to
# use these specific ports.
#_____________________________________________________________
ftpPorts="{ 55000 >< 55999 }" # Ports allocated for passive sessions
pass in log quick on $extIF proto tcp from any to $ftp_Server port 21 flags S/SA keep
state label "p21:ftpPassive"
pass in log quick on $extIF proto tcp from any to $ftp_Server port $ftpPorts flags S/SA
keep state label "ftp-hiports"
#_____________________________________________________________
#
# Only SSH is allowed into the DMZ for logins to DMZ servers (and all authentication is
# done with SecurID). Traffic is not allowed in from the internet with SSH but it is
# allowed from any device within our LAN to systems in the DMZ zones through the
# internal firewall interface. Technically, SSH should not be initiated from the DMZ
# environment out to the LAN (or outward), but occasionally there is a need to create
# an automated process to "push" files from DMZ servers to support servers on the LAN.
# This rule will be adjusted accordingly.
#_____________________________________________________________
pass in quick proto tcp from $extDmzIF to any port 22 flags S/SA keep state label
"p22:any->any"

#
# END
#_____________________________________________________________
```

VPN Security Policy

As stated in the Architecture Overview, GIAC has two VPN user groups – remote user VPN and remote site-to-site VPN. For each of these we chose to use a Cisco solution. By choosing Cisco, we were able to stay with a product line that is already widely used in the GIAC environment. Though our first choice would have gladly been an open source solution, there are a number of

© SANS Institute 2000 - 2005 Author retains full rights.

issues that come with that. First and for most is compatibility. VPN (actually the underlying technology, IPSEC) is quite complex and each vender and/or open source solution seems to implement VPN slightly different. Integrating remote sites with different VPN gateway solutions is a difficult task. Cisco equipment is very common and for the most part, nearly all modern Cisco routers and IOS support VPN, making it very easy to implement. Though this solution can be expensive, it is a proven technology that works very well and is reliable.

In the case of User VPN, Checkpoint NG running on Solaris was chosen. Checkpoint offers a number of unique and powerful options for individual user VPN control. It's most attractive feature is the ability to "push" specific user policies to individual's desktops via the SecuRemote and SecureClient software. This allows for very granular control over what VPN users are allowed to access. Also, Checkpoint has addresses a number of issues within 4.x authentication allowing for full support for SecurID authentication and further control of authenticated topology downloads. For the remainder of this paper, GIAC's remote site-to-site VPN will be discussed.

All inbound connections to the VPN links are protected by a dedicated VPN Firewall. This firewall is once again a standard issue GIAC OpenBSD 3.1 system running pf. This particular firewall is unique in that it is functioning as a bridging firewall. This allows us to place the firewall on a single network segment and the firewall will filter (via the pf rules) and forward the packets through to their destination out the firewall's respective interface. This firewall is essentially the perfect firewall - it is invisible and completely immune to network attacks or penetration. It is also useful by allowing you to place this firewall in the middle of a network segment with out having to re-address the segment behind the firewall.

Site to Site VPN Configuration

At GIAC, we are using a 3620 router with an AIM VPN Accelerator Card. All remote sites and partners will be using some Cisco VPN capable device. We must also secure and control even the valid incoming traffic and only allow them to access specific resources. For our remote facilities, this is not a major issue (being that they are true GIAC employees, entitled to the same access as those in Head Quarters), but the partner companies/sites should only be allowed access to certain resources on the internal network. To control this traffic, ACLs will be placed on the VPN gateway router's internal facing interface to allow traffic only from specific sources to specific destination.

The following are the VPN specific portions of the Cisco 3620 site-to-site config:

First we must configure IKE (Internet Key Exchange) to handle the key

© SANS Institute 2000 - 2005 Author retains full rights.

management. IKE (in conjunction with supporting protocols ISAKMP, Oakley, and SKEME) is responsible for key management within an IPSec session. IKE differs from ISAKMP in that it actually defines a key exchange, where ISAKMP merely provides a generic framework for key exchange that can be used by any key exchange protocol[7]

The “crypto isakmp” command will define various aspects of the key policy. The first policy states that we will be using pre-shared keys. Typically, we will have a separate policy for each site created. For each remote site that we establish a VPN connection with, we have a pre-assigned secret that will be used. A more scalable option would be to use a CA to exchange keys. We will define a DH group for 768 bit. Though 1024 bit keys are available, they can be resource intensive.

```
crypto isakmp policy 1
 authentication pre-share
 group 1
 lifetime 3600
```

To finish the ISAKMP configuration, we enter in the initial password itself, and tell it what host we will be sharing this information with.

```
crypto isakmp key xxxxxxxxxxxxxx address x.x.x.x
```

Next is to configure IPSec. IPSec is actually a group of protocols which handle the encryption and transmission of data. There are two parts to IPSec, AH (Authentication Header) and ESP (Encapsulated Security Protocol). AH, as it’s name implies, is a packet header that is used to ensure that the packets have not been altered and validates the source of the packet while ESP performs the encryption and tunneling of the data.

Here we define what the transform-set name is and what policies we want to use. We will always configure both AH (Authentication Header) and ESP (Encapsulated Security Protocol). Since we have control as to what each side of the VPN will be configured as, we do not set any additional options to fall back to. AH can safely be used because we are not using any NAT for our connection. For sites where NAT would be required (overlapping networks for example), we can define a separate transform-set which does not use AH, and use some of ESP’s authentication support:

```
crypto ipsec transform-set vpn_1-sec-set ah-md5-hmac esp-des esp-md5-hmac
```

For each VPN connection, we also create a access list that defines what network addresses and/or segments are to be encrypted and decrypted. This ACL actually states “encrypt anything coming from 192.168.0.0/24 going to 172.16.1.0/24”, which is different then the traditional Cisco ACLs:

```
access-list 120 permit ip 192.168.0.0 255.255.255.0 172.16.1.0 255.255.255.0
```

Create a map called “giac2site_1”, defining the “peer” that this connections will

[7] RSA Press – IPSEC Securing VPNs, Pg.256

© SANS Institute 2000 - 2005 Author retains full rights.

be made with and what ipsec transform-set will be used. Finally, per the access-list I want any traffic coming from 192.168.0.0/24 going to 172.16.1.0/24 to be encrypted:

```
crypto map giac2site_1 1 ipsec-isakmp
  set peer x.x.x.x
  set transform-set vpn_1-sec-set
  match address 120
```

Add the “crypto map” to the interface that this map applies to:

```
Interface ethernet 0/0
   ip address 192.168.0.1 255.255.255.0
   crypto map giac2site_1
```

This same process is then repeated on the remote VPN gateway.

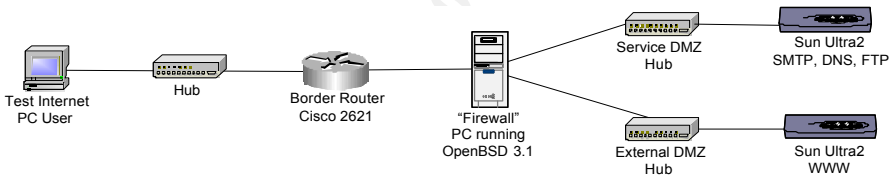
© SANS Institute 2000 - 2005, Author retains full rights.

Assignment 3 – Verification of Firewall Policy

Audit Plan

Auditing of the network environment should be an ongoing process, starting from the time the systems are initially built and continue on over time with periodic verification of the production systems and network. This ensures that as time goes on and production changes take place, that those changes don't adversely affect the existing security policy, possibly leaving them vulnerable without our knowing. GIAC has constructed a small network lab for ongoing testing to repeatedly and rigorously analyze configurations without effecting production. This is also where we build and stage our production servers, verifying and auditing them to ensure that they are properly secured. Our testlab is completely isolated from the network, which allows us to use the exact same configurations (config, IP address ranges, etc) as our production environment

With the test equipment we have on hand, we are able to simulate a simple configuration as follows:



With little cost, we were able to create a realistic version of our current Border DMZ infrastructure. A simple PC or laptop can simulate an internet user. Each "zone" is setup with a simple 8 port hub in place of a switch. A router is normally in place to allow for the testing of various router functions such as NAT, ACLs, VPN, routing, etc. For the Firewall Validation Test, the router will be removed so as not to affect the test. Our firewall is a spare Intel PC system and two older Sun Ultra 2 workstations will serve as our test production servers. Using this configuration allows for the testing of applications on the exact same OS architecture as they would run in production. These servers are built and configured identically to our production systems using the Solaris Jumpstart install (OS, patch levels, etc).

As for our testing system, we have a Dell 840C Laptop running FreeBSD 4.6. FreeBSD was chosen due to it's "laptop" friendly nature. This gave us access to an endless number of open-source freeware tools. Among the tools and techniques used in this assessment are two of the de facto standard tools - nmap (http://www.insecure.org) and Nessus (http://www.nessus.org). Nmap is a portscanning reconnaissance tools. Nmap is able to detect hosts on a

© SANS Institute 2000 - 2005 Author retains full rights.

particular network segment and report back which ports are open, identify OS type, and may other interesting statistics. It's true strength is it's ability to perform various types of "stealth" scans which are intended to go undetected by IDS systems and even penetrate firewalls. Such a powerful tool should definitely be used to gain the same understanding of our firewall as evil intruders would use on our environment.

Nesses is another tool which is invaluable for application vulnerability assessments. Nesses is able probe a system and identify not only what ports are listening and active, but is able to identify what applications are listening on those ports and what exploits and vulnerabilities are associated with them. Though this does not directly "test" the firewall itself (Nessus can actually uses Nmap for it's portscanning functionality) this tool can show us what hackers on the outside are able to learn about our applications, but more importantly, their vulnerabilities. To be fair, we should have the first opportunity to break into our systems.

Another point of interest for our firewall is the systems OS hardening. The firewall's OS itself should be as secure as possible. Being the main point of traffic and a main defense mechanism, this will make an interesting target for would be hackers. As with our production servers, all unnecessary services should be disabled and all unused packages and software should be removed. As with the Solaris systems, a "automated" build process was created to ensure that all of our Firewalls are build and configured the same. As apposed to the Solaris Jumpstart process which is network based, our OpenBSD install process is a GIAC custom CD with a core install and a number of post install scripts to remove unwanted packages and services, then configure the OS per GIAC's standards. For the audit, we will build a firewall using this process.

Finally, physical security and the OS environment (i.e, user and file permissions, host IDS, etc) will be looked at. We will insure that the system is secured from physical logins and tampering.

Scheduling and Costs

To complete a full environment audit would require additional contracting resources, money and approximately 3 weeks time. Not to mention, there would have to be scheduled production time for the majority of the scanning to take place so as not to interfere with business transactions or cause instability in the network. Our time can be more effectively used in a lab environment where we can repeatedly test areas of concern without having to schedule production outage time. Since we will only be doing an audit of the perimeter firewall, time and resources will be considerably less. There will be two technicians working together to expedite the process.

© SANS Institute 2000 - 2005 Author retains full rights.

Action Item	Personnel Time/Resources
Configuration of Lab	8 hours - 2 technicians
Obtain Testing Software	
System hardware	
Configure systems	
Develop test plan	
System Security Audit	8 hours – 2 technicians
Validate physical security	
Validate OS level security	
Verify new patch levels	
Firewall Rulebase Validation	8 hours – 2 technicians
Test servers from outside IF	
Test servers from SRV DMZ	
Test servers from EXT DMZ	
Misc testing	
Audit Results	4 hours – 1 technician
Prepare Audit Report	

For auditing of the Firewall policy, a copy of the configuration file from our production firewall will be used in the above lab configuration.

Audit Process

There are two areas of the firewall that will be investigated:

1. System Security
   - Ensure that the system is physically protected
   - Verify OS overall is secure.
2. Firewall Rulebase Validation
   - Validate that the rules that are in place truly do what they are supposed to do.
   - Ensure that we are able to properly monitor (through tools, logging, etc) all traffic going through the firewall.

System Security

The first basic requirement here is that the server physically is secure. Seeing as how the system is a rack mount device, this is an easy task. The server is mounted in a full size rack with lock doors on the front and the back. All power and network cables are tied down to prevent accidental (or intentional) cable connectivity issues. Finally, there is no physical video terminal or keyboard attached to the server itself. Access to the server is accomplished one of two ways: either via SSH to the Secure Console/Terminal Server, then to the firewall server through the serial console port or using SSH over Management/Logging Network from authorized hosts. The firewall is attached to the management interface by a dedicated ethernet port on the server. User authentication to the system is accomplished through the initial system

© SANS Institute 2000 - 2005, Author retains full rights.

password login and then SecurID authentication. Console login is permitted with via console without SecurID authentication to ensure we are not locked out of the system in the event the SecurID server was to become unreachable.

The OS, OpenBSD 3.1, is inherently very secure. The default installation is relatively secure, but there are a number of things that need to be done to better secure the system on the network. Detailing GIACs system hardening procedure and a complete system audit are beyond the scope of this paper. For the purpose of a Firewall audit, a few specific items that directly relate to the system being secure on the network will be looked at. As stated previously, all systems go though a hardening process that includes the removal of all un-needed services. During the investigation of the firewall, we were able to verify that only the minimum services are enabled:

```
# netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp        0      0  svrdmz-fxp1.44846      ns1.giac-ent.com.domai TIME_WAIT
tcp        0     48  mgmt-fxp4.ssh           titanic.giac-ent..1207  ESTABLISHED
tcp        0      0  localhost.submissi     *.*                    LISTEN
tcp        0      0  localhost.smtp         *.*                    LISTEN
tcp        0      0  mgmt-fxp4.ssh          *.*                     LISTEN
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
udp        0      0  *.syslog               *.*
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp6       0      0  localhost.giac.c.submi *.*                    LISTEN
tcp6       0      0  localhost.giac.c.smtp  *.*                    LISTEN
tcp6       0      0  mgmt-fxp4.ssh          *.*                    LISTEN
Active UNIX domain sockets
Address    Type   Recv-Q Send-Q      Inode       Conn       Refs    Nextref Addr
0xe09a9420 dgram       0      0        0x0 0xe097a440        0x0        0x0
0xe09a9370 dgram       0      0        0x0 0xe097a440        0x0 0xe09bb400
0xe09a9000 dgram       0      0 0xea86ce74        0x0 0xe09bb3c0        0x0 /dev/log
#
```

Here we can see the netstat –a output from the firewall console. SSH is enabled, but only on the fxp4, the system's management interface. All inetd, RPC and other unused services have been disabled. There are interesting anomalies to point out. First, one may notice SMTP (port 53) and the sendmail submission service (port 587) running. Ordinarily, when security is a major concern, Sendmail is not run as a daemon, but setup to queue the mail and deliver on some interval. The version of Sendmail that is installed with OpenBSD 3.1 requires it to be run as a daemon at all times, but the Sendmail default (as shown here) is to only listening on the local internal interface. It was decided to disable Sendmail all together and run it periodically to send out mail from the queue.

Also syslog is shown as listening. I spent sometime looking in to how to disable the "remote listen" function on the daemon. As it turns out, according to the syslogd manpage (it should be noted that I am NOT running syslogd with the –u switch):

© SANS Institute 2000 - 2005 Author retains full rights.

```
-u      Select the historical ``insecure'' mode, in which syslogd
        will accept input from the UDP port.  Some software wants
        this, but you can be subjected to a variety of attacks over
        the network, including attackers remotely filling logs.
```

"syslogd opens an Internet domain socket as specified in /etc/services. Normally syslogd will only use this socket to send messages outwards, but in ``insecure'' mode it will also read messages from this socket. sys-logd also opens and reads messages from the UNIX domain socket /dev/log, and from the special device /dev/klog (to read kernel messages)."

"syslogd opens the above described socket whether or not it is running in secure mode. If syslogd is running in secure mode, all incoming data on this socket is discarded. The socket is required for sending forwarded messages."

OpenBSD System Manager's Manual SYSLOGD(8)[8]

Also, just to verify that the system is not running any insecure (network and non-network) services:

```
# ps auxw
USER       PID %CPU %MEM   VSZ   RSS TT   STAT STARTED       TIME COMMAND
root     16252  0.0  0.2   280   204 p0   R+     1:17AM    0:00.00 ps -auxw
root     17084  0.0  0.3   100   360 ??   Ss    12:22AM    0:00.12 syslogd
root      5507  0.0  0.7   372   820 ??   Is    12:22AM    0:01.00 /usr/sbin/sshd
root     11020  0.0  0.4   224   440 ??   Is    12:22AM    0:00.16 cron
root      1349  0.0  0.2   380   308 C0   Is+   12:22AM    0:00.23 ksh
root     23391  0.0  0.3    48   408 C1   Is+   12:22AM    0:00.03 /usr/libexec/getty
Pc ttyC1
root     31502  0.0  0.3    48   408 C2   Is+   12:22AM    0:00.02 /usr/libexec/getty
Pc ttyC2
root     23855  0.0  0.3    48   408 C3   Is+   12:22AM    0:00.12 /usr/libexec/getty
Pc ttyC3
root      3572  0.0  0.3    48   408 C5   Is+   12:22AM    0:00.02 /usr/libexec/getty
Pc ttyC5
root     25033  0.0  1.0   424  1196 ??   S     12:44AM    0:00.72 sshd:
petinga@ttyp0 (sshd)
petinga  28124  0.0  0.2   376   288 p0   Is    12:44AM    0:00.06 -ksh (ksh)
root     23817  0.0  0.2   376   296 p0   S     12:44AM    0:00.18 ksh
root         1  0.0  0.2   336   196 ??   Ss    12:22AM    0:00.05 /sbin/init
#
```

Finally, as a cross reference, I run fstat to find any mysterious or unexpected programs or open files:

```
# fstat -v
USER     CMD          PID   FD MOUNT      INUM MODE       R/W    DV|SZ
root     fstat      23200   wd /        569856 drwx------   r      512
root     fstat      23200    0 /        488430 crw--w----  rw    ttyp0
. . .
root     fstat      23200    8 /        488375 crw-r-----   r     drum
root     fstat      23200    9 /        455181 -rw-r--r--   r   638976
root     fstat      23200   10 /        641855 -rw-------   r    40960
petinga  ksh         6035   wd /        112897 drwxr-xr-x   r      512
petinga  ksh         6035    0 /        488430 crw--w----  rw    ttyp0
petinga  ksh         6035    1 /        488430 crw--w----  rw    ttyp0
petinga  ksh         6035    2 /        488430 crw--w----  rw    ttyp0
petinga  ksh         6035   10 /        488544 crw-rw-rw-  rw      tty
root     sshd        6934   wd /             2 drwxr-xr-x   r      512
```

[8] syslogd(8) - http://www.openbsd.org/cgi-bin/man.cgi?query=syslogd&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=html

© SANS Institute 2000 - 2005, Author retains full rights.

```
. . .
root     sshd        6934    0 /        488545 crw-rw-rw-  rw     null
root     sshd        6934    6 /        488431 crw-rw-rw-  rw    ptyp0
root     sshd        6934    7 /        488431 crw-rw-rw-  rw    ptyp0
root     sshd        6934    8 /        488431 crw-rw-rw-  rw    ptyp0
root     pftop      26234   wd /             2 drwxr-xr-x   r      512
root     pftop      26234    0 /        488407 crw-------  rw    ttyC1
root     pftop      26234    1 /        488407 crw-------  rw    ttyC1
root     pftop      26234    2 /        488407 crw-------  rw    ttyC1
root     pftop      26234    3 /        488426 crw-------   r       pf
root     tcpdump    12974   wd /        569856 drwx------   r      512
. . .
root     tcpdump    12974    0 /        488409 crw-------  rw    ttyC3
root     tcpdump    12974    5 /        488409 crw-------  rw    ttyC3
root     tcpdump    12974    6 /        488384 crw-------  rw     bpf0
root     vi          8239   wd /             2 drwxr-xr-x   r      512
root     vi          8239    0 /        488406 crw-------  rw    ttyC0
. . .
root     vi          8239    1 /        488406 crw-------  rw    ttyC0
root     vi          8239    8 /        808193 -rw-------  rw        0
root     vi          8239    9 /        464138 -rw-------  rw      442
root     syslogd    17084   wd /             2 drwxr-xr-x   r      512
. . .
root     syslogd    17084    0 /        488545 crw-rw-rw-  rw     null
root     syslogd    17084   15 /        456973 -rw-r-----   w        0
root     syslogd    17084   16 /        456978 -rw-------   w    87995
root     init           1   wd /             2 drwxr-xr-x   r      512
. . .
```

The output has been trimmed but we do see some processes (which can all be explained by some of the other activities going on with the audit – the vi session, tcpdump, etc). It appears that the system is running only valid services.

Results: Physical/OS Security

Overall, the Firewall system itself is secure. Physical access to the server has been minimized. The OS has been hardened and only a minimum number of services are available and running. There are no further recommendations for improvement but further understanding of the Sendmail and syslogd "anomalies" should be further researched and documented.

Firewall Rulebase Validation

The second part of the audit is to verify that the policies put into place on the firewall are working correctly – passing and dropping as designed. I have already obtained a current working copy of the pf.conf from the production firewall and loaded onto the firewall. Next I have configured the network in the above diagram, simulating the publicly addressed areas and configuring servers with the appropriate services. The private network address spaces will be monitored in the log to make sure packets don't pass through. First we will do a simple nmap scan to see what ports are open on the firewall. We are also interested in seeing what other information can be obtained about the firewall from the scans. The first scan is a simple SYN scan (-n disables

© SANS Institute 2000 - 2005 Author retains full rights.

name resolution to speed things up, we of course would never register our firewall in DNS):

```
root@marvin # nmap -n -sS -O 222.0.0.2

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Warning:  OS detection will be MUCH less reliable because we did not find
at least 1 open and 1 closed TCP port
All 1601 scanned ports on  (222.0.0.2) are: closed
Too many fingerprints match this host for me to give an accurate OS guess

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds
root@marvin #
```

Nothing turned up port wise and the OS detection was vague. Next is the TCP scan. Our first point of interest is to see what ports are responding on the external interface of the firewall. First we run the TCP scan against all 65535 available ports:

```
root@marvin # nmap -n -O -P0 -sT -p 1-65535 -v 222.0.0.2

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (222.0.0.2) appears to be up ... good.
Initiating Connect() Scan against  (222.0.0.2)
The Connect() Scan took 39 seconds to scan 65535 ports.
Warning:  OS detection will be MUCH less reliable because we did not find
at least 1 open and 1 closed TCP port
All 65535 scanned ports on  (222.0.0.2) are: closed
Too many fingerprints match this host for me to give an accurate OS guess

TCP/IP fingerprint:
SInfo(V=3.00%P=i386-unknown-freebsd4.6%D=9/30%Time=3D988045%O=-1%C=1)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=N)
PU(Resp=Y%DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)


Nmap run completed -- 1 IP address (1 host up) scanned in 40 seconds
root@marvin #
```

Still no ports are open but the OS finger printing came back with a vague result. It's best guess was actually some version of FreeBSD (in fact, our firewall is OpenBSD). Next a UDP scan against all 65535 ports:

```
root@marvin # nmap -v -sU -p 1-65535 222.0.0.2

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (222.0.0.2) appears to be up ... good.
Initiating UDP Scan against  (222.0.0.2)
The UDP Scan took 761 seconds to scan 65535 ports.
All 65535 scanned ports on  (222.0.0.2) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 761 seconds
root@marvin #
```

It would appear that the firewall has no "open" ports, which is what we were

© SANS Institute 2000 - 2005 Author retains full rights.

looking for. pf can return messages back in response to TCP requests it is not servicing. This is accomplished though the “return-rst” command:

```
block return-rst in log on $extIF proto tcp all label "blockInTcpAll"
```

The above rule states “block any tcp packet coming in on the external interface, and return a reset to the host”. Normally, when a client tries to connect to a host that isn’t running a particular service, the server should respond back with a reset. Firewalls (packet filter firewalls in this case) don’t do this, and the connection appears to stall:

```
root@marvin # telnet 222.0.0.2 80
 Trying 222.0.0.2...
```

Here we used telnet to send a TCP connect() to a specific port. We should receive a reset, but nothing is returned. This is a good indication of a packet filter. Where as with the return-rst command, the firewall is able to act like a regular “host” with an immediate and predictable response:

```
root@marvin # telnet 222.0.0.2 80
Trying 222.0.0.2...
telnet: connect to address 222.0.0.2: Connection refused
telnet: Unable to connect to remote host
root@marvin #
```

An interesting scan can be run with nmap to show this. The “-sA” switch can be used to determine if a firewall is a simple packet filter that blocks incoming SYN packets or is a stateful firewall.

```
root@marvin # nmap -v -n -P0 -sA 222.0.0.2

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (222.0.0.2) appears to be up ... good.
Initiating ACK Scan against  (222.0.0.2)
The ACK Scan took 0 seconds to scan 1601 ports.
All 1601 scanned ports on  (222.0.0.2) are: UNfiltered

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
root@marvin #
```

According to the nmap manpage (*http://www.insecure.org/nmap/data/nmap_manpage.html*):

> “Unfiltered means that the port is known by nmap to be closed and no firewall/filter seems to be interfering with nmap’s attempt to determine this.”

Nmap is apparently seeing the firewall as something other then a packet

© SANS Institute 2000 - 2005 Author retains full rights.

filtering firewall, hopefully due to the resets that were being sent in response ACK packets sent to the ports? Being the curious type, I tried to verify this by disabling the return-rst command in the firewall. After a number of attempts with the default number of ports (nmap consistently stopped after about 5 minutes with "`Skipping host (222.0.0.2) due to host timeout`"), I selected a few ports that I knew were permitted within the firewall's rules and attempted another scan:

```
root@marvin # nmap -v -n -P0 -sA -p 21,23,25,80 222.0.0.2

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (222.0.0.2) appears to be up ... good.
Initiating ACK Scan against  (222.0.0.2)
The ACK Scan took 6 seconds to scan 4 ports.
Interesting ports on  (222.0.0.2):
Port       State       Service
21/tcp     filtered    ftp
23/tcp     filtered    telnet
25/tcp     filtered    smtp
80/tcp     filtered    http

Nmap run completed -- 1 IP address (1 host up) scanned in 6 seconds
root@marvin #
```

It appears that the reset responses are helping to conceal the firewall against basic scans. But our fun will only last so long. When performing a stealth FIN, Xmas Tree and Null scan, we end up with very different results:

```
root@marvin # nmap -n -r -O -sX -v 222.0.0.2

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (222.0.0.2) appears to be up ... good.
Initiating XMAS Scan against  (222.0.0.2)
The XMAS Scan took 100 seconds to scan 1601 ports.
Adding open port 65301/tcp
Adding open port 61441/tcp
Adding open port 61440/tcp
…
Adding open port 4/tcp
Adding open port 3/tcp
Adding open port 2/tcp
Adding open port 1/tcp
(no tcp responses received -- assuming all ports filtered)
Warning:  OS detection will be MUCH less reliable because we did not find
at least 1 open and 1 closed TCP port
All 1601 scanned ports on  (222.0.0.2) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
SInfo(V=3.00%P=i386-unknown-freebsd4.6%D=9/30%Time=3D98869D%O=-1%C=-1)

T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=N)
PU(Resp=Y%DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)

Nmap run completed -- 1 IP address (1 host up) scanned in 116 seconds
root@mavin #
```

© SANS Institute 2000 - 2005

Author retains full rights.

For all three stealth scans, we see the exact same results – all ports are considered "filtered" (though still, no ports were found open). Of course, after looking at the issue further, I discovered everything is working exactly as designed. We have addressed SYN packet scans but done nothing with other "flag" type scans. pf has the ability to filter based on specific TCP flag setting and patterns. One way that we might add to the "stealth ness" or our firewall would be to add something like the following:

```
block return-rst in quick on $extIF inet proto tcp all flags F/SFRA # FIN
block return-rst in quick on $extIF inet proto tcp all flags FUP    # Xmas
block return-rst in quick on $extIF inet proto tcp all flags /SFRA  # Null
```

For each rule, I have addressed the flags that are turned on for each respective type of scan. The issue here isn't that the firewall is insecure, but that it is detectable though some advanced scanning. This is something that should be further tested but would help in concealing our firewall from probing.

A final test for scanning is to see what the firewall rules are actually doing for our systems behind the firewall. We will try a simple scan against the webserver to see what ports can be identified from the outside. For this test we have intentionally opened the following ports on the actual web server itself:

```
telnet  23
ssh     22
ftp     21
http    80
```

The firewall rulebase is configured to only allow port 80 through to the system:

```
root@mavin # nmap -v -sS 222.0.0.130

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host www.giac-ent.com (222.0.0.130) appears to be up ... good.
Initiating SYN Stealth Scan against www.giac-ent.com (222.0.0.130)
Adding open port 80/tcp
The SYN Stealth Scan took 6 seconds to scan 1601 ports.
Interesting ports on www.giac-ent.com (222.0.0.130):
(The 1600 ports scanned but not shown below are in state: closed)
Port       State       Service
80/tcp     open        http

Nmap run completed -- 1 IP address (1 host up) scanned in 6 seconds
root@mavin #
```

The firewall only permitted port 80 through to the web server. The firewall logs also show the packets being blocked by rule 20 which is the "block all incoming tcp traffic" that does not explicitly match a permit rule:

```
Sep 30 14:57:03.091766 rule 20/0(match): block in on fxp0: 222.0.0.2.56731 >
222.0.0.130.555: S 114207588:114207588(0) win 1024
Sep 30 14:57:03.091931 rule 20/0(match): block in on fxp0: 222.0.0.2.56731 >
222.0.0.130.20: S 114207588:114207588(0) win 1024
Sep 30 14:57:03.180100 rule 20/0(match): block in on fxp0: 222.0.0.2.56731 >
222.0.0.130.1421: S 114207588:114207588(0) win 1024
Sep 30 14:57:03.180767 rule 20/0(match): block in on fxp0: 222.0.0.2.56731 >
```

© SANS Institute 2000 - 2005, Author retains full rights.

```
222.0.0.130.2038: S 114207588:114207588(0) win 1024
Sep 30 14:57:03.181606 rule 20/0(match): block in on fxp0: 222.0.0.2.56731 >
222.0.0.130.730: S 114207588:114207588(0) win 1024
Sep 30 14:57:03.181731 rule 20/0(match): block in on fxp0: 222.0.0.2.56731 >
222.0.0.130.1400: S 114207588:114207588(0) win 1024
Sep 30 14:57:03.182030 rule 20/0(match): block in on fxp0: 222.0.0.2.56731 >
222.0.0.130.4000: S 114207588:114207588(0) win 1024
```

For each server that is externally accessible, this test was performed. In the case of our DNS server, which only has UDP port 53 open to the external interface, the TCP scan should how nothing while UDP scan should show only UDP port 53:

```
root@marvin # nmap -v -sS 222.0.0.66

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host ns1.giac-ent.com (222.0.0.66) appears to be up ... good.
Initiating SYN Stealth Scan against ns1.giac-ent.com (222.0.0.66)
The SYN Stealth Scan took 6 seconds to scan 1601 ports.
All 1601 scanned ports on ns1.giac-ent.com (222.0.0.66) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 6 seconds
[root@gaspra /root]# nmap -v -sU 222.0.0.66

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host ns1.giac-ent.com (222.0.0.66) appears to be up ... good.
Initiating UDP Scan against ns1.giac-ent.com (222.0.0.66)
The UDP Scan took 20 seconds to scan 1468 ports.
Adding open port 53/udp
Interesting ports on ns1.giac-ent.com (222.0.0.66):
(The 1467 ports scanned but not shown below are in state: closed)
Port       State       Service
53/udp     open        domain

Nmap run completed -- 1 IP address (1 host up) scanned in 20 seconds
root@marvin #
```

And once again, the firewall logs show the blocked packets:

```
Sep 30 14:57:04.143737 rule 20/0(match): block in on fxp0: 222.0.0.2.56731 >
222.0.0.130.698: S 114207588:114207588(0) win 1024
Sep 30 14:57:04.143906 rule 20/0(match): block in on fxp0: 222.0.0.2.56731 >
222.0.0.130.528: S 114207588:114207588(0) win 1024
Sep 30 14:57:04.144976 rule 20/0(match): block in on fxp0: 222.0.0.2.56731 >
222.0.0.130.703: S 114207588:114207588(0) win 1024
```

Logging

pf allows us to use a number of different logging methods. Logging is done on a per rule basis using the “log” command in the respective rule. The actual log information is then output in tcpdump format. This information can either be saved off to a log file via the pflog(8)[9] daemon or viewed in real time using the pflog interface[10]. For each method, any of the standard tcpdump options can be used for viewing the information (i.e. timestamp information, printing each

[9] pflogd(8) - http://www.openbsd.org/cgibin/man.cgi?query=pflogd&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=html

[10] All pf log traces show in the audit were taken real time using the command tcpdump –n –e –ttt –l pflog0

© SANS Institute 2000 - 2005 Author retains full rights.

packets information, verbose packet information, etc). During the audit, we were able to monitor all traffic through the firewall in real time, and by using standard tcpdump expressions, we can filter on specific traffic when needed. A very nice feature of pf's logging is the ability to capture an entire session of only specific traffic between hosts:

```
pflogd -s 1600 -f webserver.log port 80 and host marvin
```

Here we are able to start a separate pflogd process and log to a separate file all traffic using port 80 from host marvin (our FreeBSD laptop test system). This can be done while all other normal logging continues. The size of the snaplen was also increased to capture the entire packet(s). During the audit (and in production), this allowed us to analyze all packet information of a specific communication. This has proven invaluable in production when tracking suspicious traffic.

Rule and State Monitoring

To manage firewalls effectively, one needs to be able to monitor the traffic and rulebase in real time to ensure that the rules are functioning properly. As mentioned previously, pfctl can be used to view statistics pertaining to firewall traffic A new utility called pftop has come out that allows for the real time monitoring of these statistics in a UNIX "top" like format:

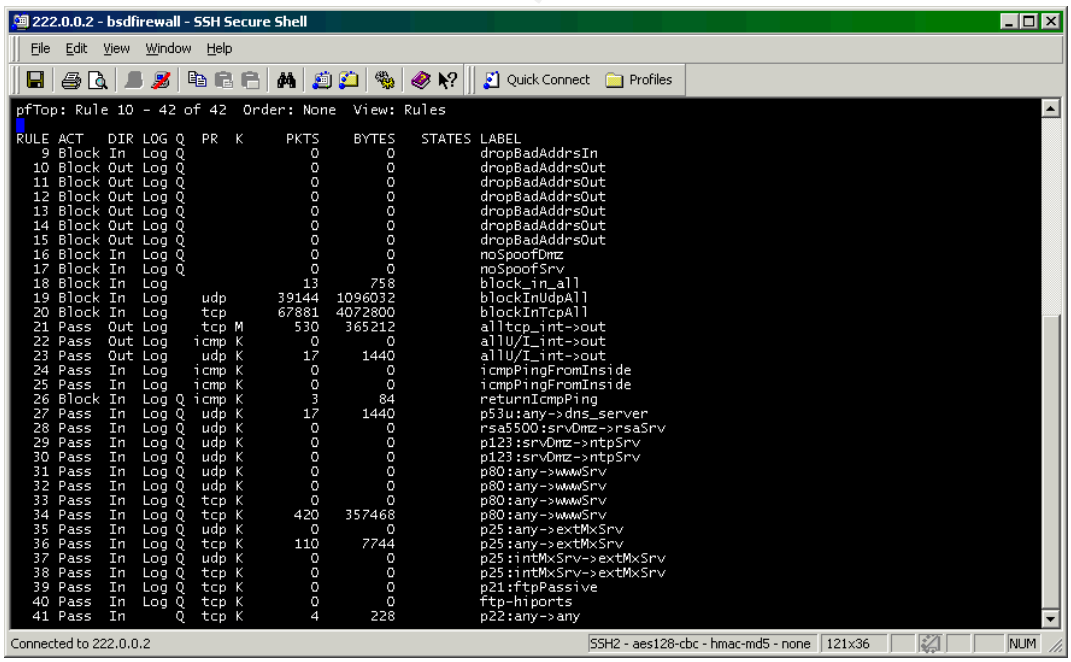


Figure 2 - pftop rules view

We are able to view every rule and various statistics associated with that rule. Each rule is identified by the "label" option used in the pf.conf for each rule. We can see weather it is a blocking rule, passing rule, in bound or outbound,

© SANS Institute 2000 - 2005 Author retains full rights.

and a label describing what the rule is used for, etc. Our tests have been working our block of all TCP and UDP packets very nicely. We can also see a number of other rules with activity as testing has gone on[†]. pftop also allow us to see what the current state table looks like:

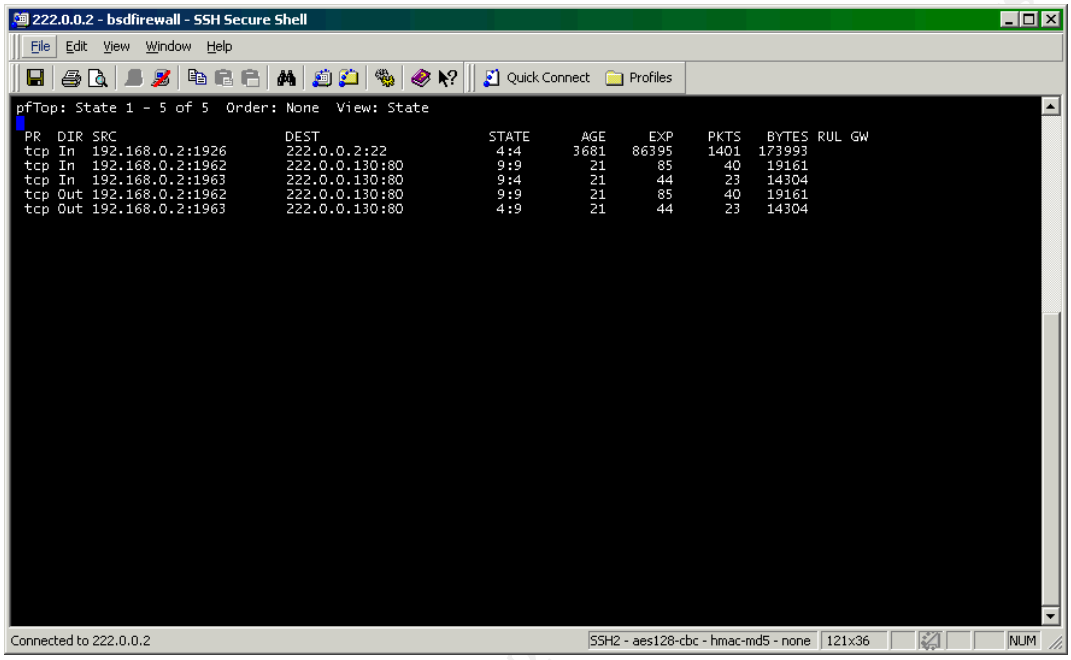


Figure 3 - pftop and State Table entries

Figure 3 shows a number of state table entries showing direction, protocol, source and destination address (including ports). Within the state table view, we can also see the age, expire time, number of bytes, number of packets and the rule that has permitted the traffic through. We can also sort based on any one of these values. Not only during testing but in everyday productions environments, this tool can give us a very clear, real time view as to what the firewall is doing. Once again, during the audit, we were able to monitor the state table and ensure that state entries were being created and expired correctly.

Results: Firewall Rules Validation

The table below shows the results of scan attempts from the direct attached networks of the firewall to various systems within their respective DMZ spaces. For example, when the nmap system was connected to the external network (i.e. the network the External Interface is connected to) and a full port scan was performed against each target system, only the supported port for that server was reported open. Further more, per our security policy, no system in the Service DMZ or External DMZ should be able to initiate traffic

[†] One may notice rule 41 which allows ssh from any to any. This was added after the scanning only to obtain screenshots and done for testing purposes only. This IS NOT allowed in production.

© SANS Institute 2000 - 2005 Author retains full rights.

from their respective networks to any other network zone (of course, returning traffic is allowed through for rules we are keeping state on). The audit tests have shown that all rules are working properly.

© SANS Institute 2000 - 2005, Author retains full rights.

	Service DMZ Servers			External DMZ Servers			Internal
Authorized Port	port 53	port 21	port 25	port 80	port 443	port 81/444	*N/A*
Outside Interface	✓	✓	✓	✓	✓	✓	×
Service DMZ Interface				×	×	×	×
External DMZ Interface	×	×	×				

Port Access Results

Once again, no recommendations for improvement can be seen at this time, though further research into the firewalls visibility to scanning should be investigated.

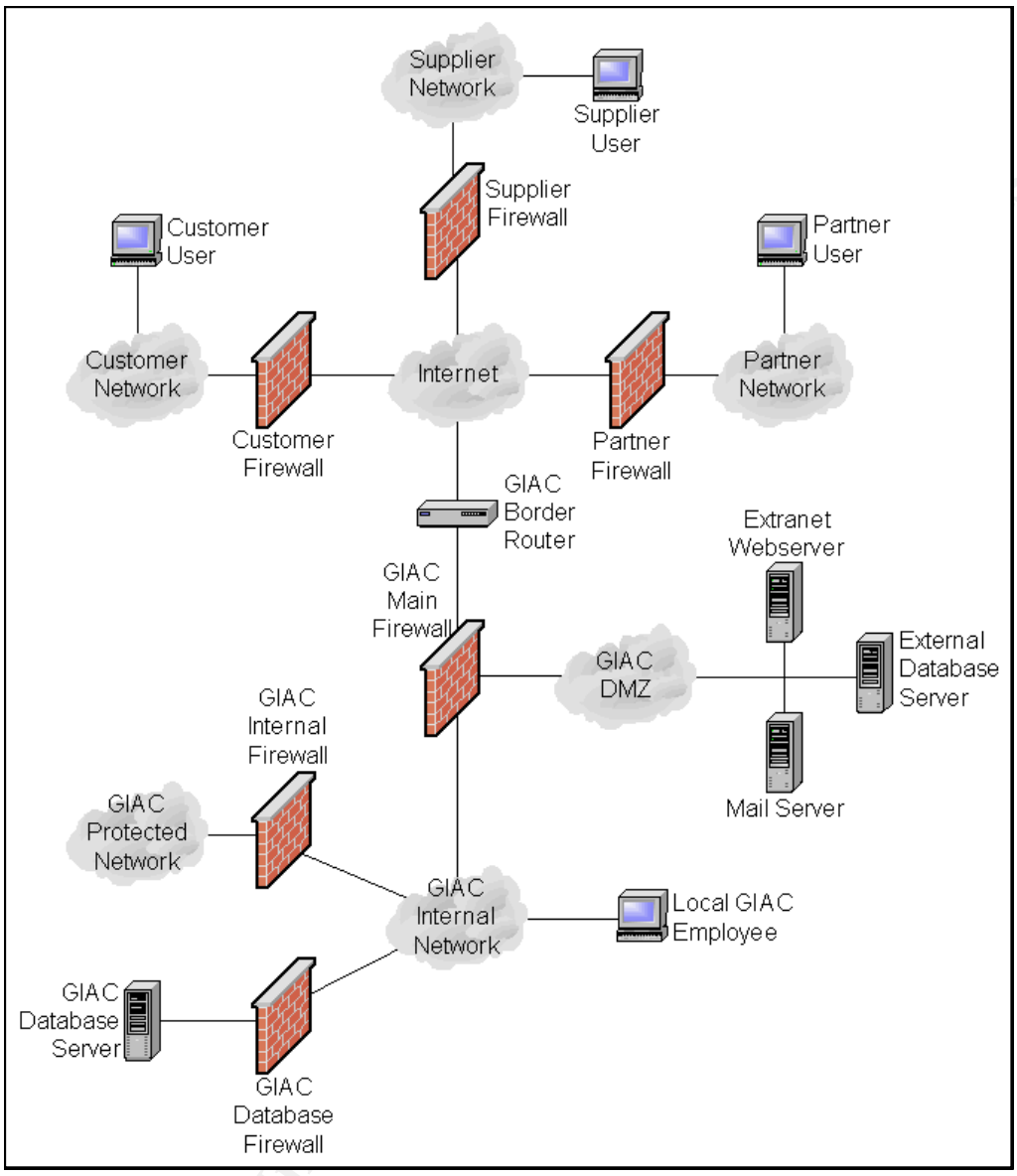
The firewall performed as designed. A more detailed validation of the firewall would include a throughput load test. It is important to understand what the firewall's "limit" is for performance no only for normal traffic throughput but also to test the firewall's stateful inspection performance is under heavy loads. This would also give us an indication of what the system's stability would be not only during normal traffic but also under DOS type attack conditions or stateful inspection overload attacks. The OS and pf offer a number of parameters that can be modified to improve performance[12].

Assignment 4 – Design Under Fire

The final assignment is to create an "attack" on a previously submitted practical assignment. My chosen target is the design of Vince Kornacki, Analyst Number: 0314 (*http://www.giac.org/practical/Vince_Kornacki_GCFW.zip*). His design is deceptively simple on the surface, but further reading into the practical shows a number of safeguards that make penetration into his network somewhat tricky. All externally facing systems within the DMZ require SecurID authenticated logins at the firewall. These GIAC DMZ systems are only accessible to business Partners and Suppliers via VPN and are not open to "public viewing". It should be noted that Vince's GIAC corporate "publicly accessible" web server is actually not in the DMZ, but hosted by an outside ISP. I don't feel that this is practical for an e-business dealing in the on-line sale of fortune cookie sayings but…

[12] Design and Performance of the OpenBSD Stateful Packet Filter (pf) - http://www.benzedrine.cx/pf-slides.pdf

© SANS Institute 2000 - 2005 Author retains full rights.



Compromise Internal System

Attacking or breaking into a network or system isn't as pre-meditated as one might think. More often then not, finding the right exploit is usually by chance. I happen to know where Vince's company is located (it's a very small town) but more importantly, I know where a number of the employees go for lunch – a cyber café called Conference Room C. Many of them come to get out of the office for a while, bring in their laptops and VPN in back to the office. The café happens to have a simple hub network which allows someone like me to watch everything that is going on (this exercise could just have easily been done monitoring cable modem users in the neighborhood). I hear some talk here and there and I have a decent feel for what their network might look like. I learn that they are using Checkpoint VPN to log into the network. So, I start doing some research[13]…

[13] Based on the information given in Vince's document, I don't believe that this design would actually work. First, his router configuration shows very few anti-spoofing ACLs which opens the network up to a number of spoof attacks. But, more importantly, his anti-spoofing ACL for 192.168.0.0/16 is blocking his firewall's 192.168.0.1 address. Technically, nothing would ever make it to the firewall to establish a VPN connection. I will present this assignment assuming the firewall is accessible.

© SANS Institute 2000 - 2005

Author retains full rights.

Lucky for me, there was no shortage of well documented vulnerabilities on the internet directed at Checkpoint firewalls. Here is interesting list of programs that were readily available, all in once place at *http://www.theheap.org/archives/advisories/os/hardware/firewalls/firewall-1/* :

Index of /archives/advisories/os/hardware/firewalls/firewall-1

Name	Last modified	Size	Description
Parent Directory	29-Apr-2002 20:59	-	
cpd.c	29-Apr-2002 21:21	5k	
fm.c	29-Apr-2002 21:21	16k	
ftp-ozone.c	29-Apr-2002 21:21	4k	
fwsa.sh	29-Apr-2002 21:21	12k	
jolt2.c	29-Apr-2002 21:21	4k	

And of course searches on *http://www.securityfocus.com*, *http://www.cert.org*, and mailing list archives like those found at *http://www.netsys.com* and *http://archives.neohapsis.com* turned up many more.

Having done some preliminary probing by visiting GIAC's websites, I realized the public web servers weren't actually located on the GIAC premises, but hosted by an external ISP.
So I decide to do a little sniffing around the café looking for traffic to somewhere with VPN protocols (UDP 500 for IKE, and 50 and 51 for AH and ESP). I can see a number of connections going to address 192.168.0.1. So decide to investigate this with nmap:

```
- port 264/tcp was open
- port 265/tcp was open
```

This is a sure fire indication that the system is a Checkpoint Firewall using User/Session Authentication (*http:/www.phoneboy.com/faq/0105.html*). These are services that are run on a Checkpoint Firewall-1/VPN system for SecuRemote VPN connectivity. So we know for sure we have a Checkpoint VPN system. This is good this for us. It just so happens that I know from my experience if Checkpoint's VPN is configured using hybrid IKE authentication[14], there is a way that any outside user can download a copy of the VPN topology. This "topology" is essentially a configuration file the SecuRemote client downloads identifying what networks the client will have access to. It also lists the VPN gateway's interfaces, and a number of other

[14] http://www.ietf.org/proceedings/00dec/I-D/draft-ietf-ipsec-isakmp-hybrid-auth-05.txt

© SANS Institute 2000 - 2005 Author retains full rights.

pieces of information needed for VPN connectivity. This can be done one of two ways. The first is by using a Checkpoint SecuRemote client and creating a site using the 192.168.0.1 gateway. Once the site is created, the topology is automatically downloaded, no login necessary. Once the site is created, this information is stored in a file called the "userc.C". Or, you can use the sr.pl perl script (*http://packetstormsecurity.nl/0107-exploits/sr.pl*) which allows any IP address to download this information without having to use the client software. The output of sr.pl run against the GIAC VPN gateway address of 192.168.0.1 is shown here[15]:

```
Testing  on port 256
(
     :val (
          :reply (
               : (-SensePost-dotcom-.vpn.giac.com
                    :type (gateway)
                    :supports_tcp_ike ()
                    :is_isakmp (true)
                    :ISAKMP_hybrid_support (true)
                    :certificates (
                         : ("O=chi,C=us"
                              : ("CN=vpn.giac.com,O=chi,C=us")
                         )
                    )
                    :is_subnet_support (true)
                    :uencapport (2746)
                    :fwver (4.1)
                    :ipaddr (192.168.0.1)
                    :ipmask (255.255.255.255)
                    :resolve_multiple_interfaces ()
                    :ifaddrs (
                         : (192.168.0.1)
                         : (10.0.1.10)
                         : (10.3.0.10)
                    )
                    :firewall (installed)
                    :location (external)
                    :keyloc (remote)
                    :userc_crypt_ver (1)
                    :keymanager (
                         :type (refobj)
                         :refname ("#_-SensePost-dotcom-")
                    )
                    :encdomain (Topology
                         :type (topology)
                         :color ("Dark Green")
                         :icon (inbound)
                    )
                    :topology (
                         : (
                              :name (-SensePost-dotcom-.vpn.giac.com)
                              :type (gateway)
                              :ipaddr (192.168.0.1)
                              :ipmask (255.255.255.255)
                         )
                         : (
                              :name (-SensePost-dotcom-.vpn.giac.com)
                              :type (gateway)
                              :ipaddr (10.1.0.10)
                              :ipmask (255.255.255.255)
                         )
                         : (
                              :name (-SensePost-dotcom-.vpn.giac.com)
                              :type (network)
                              :ipaddr (10.3.0.10)
                              :ipmask (255.255.255.255)
                         )
                         : (
                              :name (-SensePost-dotcom-.GIAC_10.1)
                              :type (network)
                              :ipaddr (10.4.0.0)
```

[15] This entire topology in bogus. The fields and structure are real but all data has been designed to match Vince's network design. Due to a lack of addresses in Vince's practical, all missing addresses have been embellished at my discretion.

© SANS Institute 2000 - 2005, Author retains full rights.

```
                    :ipmask (255.255.255.0)
               )
          )
          :BackupGws ()
     )
)
:servers_reply (
     : (-SensePost-dotcom-.GIAC_PolicyServer
          :type (lms_server)
          :ipaddr (192.168.0.1)
          :keymanager (
               :type (refobj)


:dnsinfo (
     :dns_servers (
          : (dns.giac.com
               :obj (
                    : (x.x.x.x)
               )
               :topology (
                    : (
                         :ipaddr (10.1.0.0)
                         :ipmask (255.255.255.0)
                         :ipaddr (10.2.0.0)
                         :ipmask (255.255.255.0)
                         :ipaddr (10.3.0.0)
                         :ipmask (255.255.255.0)
                         :ipaddr (10.4.0.0)
                         :ipmask (255.255.255.0)
                    )
               )
               :domain (
                    : (
                         :dns_label_count (10)
                         :domain (.giac.com)
                    )
```

A very nice start. Vince is using Checkpoint 4.1. Checkpoint NG has the ability to disable unauthorized topology downloads. We now know that there are a number of 10.x.x.x internal address ranges, what their DNS server is, domains available, etc. Technically, this could have been done by chit-chatting with any GIAC employee (i.e. social engineering) but this was just as easy.

The next step is to some how get into the network. Apparently, all access is by Client VPN for external users. There are a number of interesting exploits that may help me with this task. Two of these are based around something known as the "Checkpoint FW-1 VPN Security Flaw" (*http://www.securiteam.com/securitynews/5TP040U8AW.html*), originally discovered by NTA Monitor (*http://www.nta-monitor.com*). The first "flaw" allows for VPN username guessing and the second allows for VPN username sniffing. The username guessing issue allows remote users to literally "validate" usernames against a Checkpoint VPN gateway using a brute force password attack. Basically, you can create a script that repeatedly attempts an IKE user authentication against a Checkpoint VPN gateway (using TCP port 246) and the Checkpoint gateway will respond back with either a "valid" or "non-valid" user response. There is also no limit to the number of requests that can be sent. So, one system could send 10,000 account name guesses and is only limited (time wise) by the speed of the firewall and the system sending the requests. This technically would be possible since there is no mention of any IDS or logging in Vince's design. Username and eventually password guessing could take place undetected. Though, by default, failed attempted

© SANS Institute 2000 - 2005, Author retains full rights.

logins are logged by Checkpoint.

The second issue discovered in the Checkpoint VPN Flaw was that usernames are passed clear text initially in a Checkpoint VPN communication (to be exact, the first packet contains the username). SecuRemote requires the identity in the first packet, before any key are exchanged to allow for encryption. As pointed out by NTA, this is not entirely Checkpoint's fault, it is partly due to IKE aggressive mode behavior. Seeing as how I am able to quietly sit and sniff, this is definitely the way to go. But we will still have to do a password guess on any valid account names we find.

After patiently watching traffic I was able to capture an initial connection with the sniffer and view a valid username. From this point, it is simply a matter of taking the username and running it against a password guessing program, which is conveniently built right into the fwsa.sh[16] script. A simple modification would allow me to continually run against the firewall over a period of time trying new passwords. Though this would be futile because all authentications is done though SecurID. We are unable to directly access any system without SecurID authentication.

Attack on Firewall

An attack on the firewall should be relatively easy based on the fact that the Firewall and VPN are on the same physical system. As previously mentioned, this forces the Firewall to have running services (thus open ports) for accepting VPN connections. The port that we are looking for is TCP 264. This is used by SecurRemote clients for VPN connectivity. This can be a liability for the firewall due to the fact that the Firewall is a stateful inspection firewall and we could easily send thousands of small TCP SYN packets and Checkpoint will continue to hold state on these open connections, thus flooding the state table and potentially causing the system to crash. Here is how it can be done…

First we would have to verify that the firewall has open ports on the system. This is done with nmap:

```
Nmap -n -P0 -sS 192.168.0.1
```

With this scan, we see that there are actually two ports open for attack:

```
264/tcp     open
265/tcp     open
```

Also knowing what we know about their partner network we can avoid detection by spoofing our source address. Based on the router configuration supplied in his practical, there are **very few** anti-spoofing rules, so we could choose almost anything. If we needed to bypass any anti-spoof rules they may

[16] fwsa.sh – see link above from http://www.theheap.org

© SANS Institute 2000 - 2005 Author retains full rights.

have, we could easily spoof one of the partner or customer site VPN IP addresses. This might work in our favor to cause disruption in a number of areas. If we were to send SYN packets (i.e., attempts to open a three-way handshake) with the spoofed source address of ACME's external VPN server, the GIAC server would then send SYN-ACKs back to ACME. These should naturally be dropped by ACME but this will have to force them to look at what ACME is doing on if this is detected (though IDS does not seem to be installed anywhere).

We will use a simple script and program to run against port TCP 264. This technique was outlined in a Bugtraq posting, http://archives.neohapsis.com/archives/bugtraq/2000-07/0085.html. Briefly, it states that you can flood port 264 causing the CPU to reach 100%. As recommended by the author, this can be accomplished with ippacket:

```
Dest IP: Firewall (external interface)
Src IP: no existent IP
Src port: 1000
Dest Port: 264
Data: qwertyuiop101010101010
Number of packets: -1 (continuous mode)
```

Another option would be an attack internally and to again use the fwsa.sh script, which actually is a collection of exploits specifically for Session Authentication. This script contains 2 different DOS exploits and 2 username/password guesser functions. In this design, Session Authentication is actually being used for internal users going to the DMZ systems, who are once again authenticated with SecurID. The SecurID authentication definitely protects against brute-force guess attacks, though the DOS type attacks against the services are still very difficult to defend against.

Denial of Service Attack

Denial of Service attacks are attacks against a remote system (or systems, or even network) in the hopes of "denying" use of certain resources or services. DoS attacks can be[17]:

- an attempt to "flood" a network, thereby preventing legitimate network traffic
- an attempt to disrupt connections between two machines, thereby preventing access to a service
- an attempt to prevent a particular individual from accessing a service
- an attempt to disrupt service to a specific system or person

[17] http://www.cert.org/tech_tips/denial_of_service.html

© SANS Institute 2000 - 2005 Author retains full rights.

Though one “attack” system (i.e. a simple PC connected to a network or the internet) can cause a disruption in service, often times attackers are looking to disable a website or an entire network. Production networks and system hardware are designed to handle large volumes of traffic, so one “attack” PC may not be enough to disrupt service. Over the years, a more sophisticated type of DoS attack was developed which employs many “attack” hosts (also know as agents) attacking a single target. These are know as Distributed Denial of Service Attacks, where you have one “master client” who essentially controls many remote agents and coordinates and controls the attack from the remote agents. The number of agents that can be deployed is more or less limitless, thus making a large scale DDoS attack devastating.

Some of the more popular DDoS attack tools available are:

- TFN (Tribal Flood Network)
- TFN2K (Tribal Flood Network 2000)
- Mstream
- Trinoo
- Stacheldraht

All of these are based on the same principal – at least one or more clients (or master) controlling many agent hosts actually doing the attack. Each one of the above implements this slightly different, offering different types of attacks. Each can send different types of TCP, SYN, FIN, ICMP, or UDP packets. Of the above, TFN2K (a updated version of TFN) is able to send a number of different flood attacks – ICMP, UDP, SYN and Smurf. Both TFN and TFN2K support these types of flood attacks. But TFN2K took a leap forward in offering more stealth (the program agent is difficult to detect), and blowfish encryption for it’s communication between the master client(s) and the agents (the key association between the client and agents are defined at compile time).

For our attack on the GIAC network, we will be using 50 systems that we have comprised that have internet access via cable/DSL. These will be our agents. These “sleeper” agents have been installed over the last month on home desktop that were easily compromised (mostly Redhat Linux systems, 6.2 preferably and Windows XP). These agents are in place on PCs that are typically left on all the time by the home user. So we will be able to launch an attack at any time.

The attack itself is controlled by the client, using the tfn2k client CLI. TFN2K takes security very seriously and is able to encrypt the communication between the master and agent. This traffic can be very difficult to detect because the client-agent communication may actually randomly switch between TCP, UDP, or ICMP. Once the client and agents are compiled, one can simply run the master client and initiate any number of different attacks:

© SANS Institute 2000 - 2005 Author retains full rights.

```
root@mavin # ./tfn --help
    Protocol      : random
    Source IP     : random
_[1;34musage: ./tfn <options>
[-P protocol]   Protocol for server communication. Can be ICMP, UDP or TCP.
                Uses a random protocol as default
[-D n]          Send out n bogus requests for each real one to decoy targets
[-S host/ip]    Specify your source IP. Randomly spoofed by default, you need
                to use your real IP if you are behind spoof-filtering routers
[-f hostlist]   Filename containing a list of hosts with TFN servers to contact
[-h hostname]   To contact only a single host running a TFN server
[-i target string] Contains options/targets separated by '@', see below
[-p port]          A TCP destination port can be specified for SYN floods
<-c command ID>0 - Halt all current floods on server(s) immediately
        1 - Change IP antispoof-level (evade rfc2267 filtering)
            usage: -i 0 (fully spoofed) to -i 3 (/24 host bytes spoofed)
        2 - Change Packet size, usage: -i <packet size in bytes>
        3 - Bind root shell to a port, usage: -i <remote port>
        4 - UDP flood, usage: -i victim@victim2@victim3@...
        5 - TCP/SYN flood, usage: -i victim@... [-p destination port]
        6 - ICMP/PING flood, usage: -i victim@...
        7 - ICMP/SMURF flood, usage: -i victim@broadcast@broadcast2@...
        8 - MIX flood (UDP/TCP/ICMP interchanged), usage: -i victim@...
        9 - TARGA3 flood (IP stack penetration), usage: -i victim@...
        10 - Blindly execute remote shell command, usage -i command
root@mavin #
```

We have a number of interesting choices. Since we are unable to directly access his systems without authentication, it would be just as easy to take out the entire firewall via it's open ports. Basically, take our list of agents (agents.txt) and give it options and command ID to do a MIX flood of UDP, TCP, and ICMP packets to the firewall:

```
tfn -f agents.txt -c 8 -port 264 -i 192.168.0.1
```

Countermeasures

There is no real defense from an attack like this. SANS has put together a best practice as to how to defend against an attack, http://www.sans.org/ddow_roadmap.html. This is geared more towards the internet community as a whole lessoning the chances an attack being launched (i.e. not giving attackers the freedom to carry out such an attack) then it is about how to defend against it. Though a few good tips are given in this analysis of TFN2K (*http://packetstorm.decepticons.org/distributed/TFN2k_Analysis-1.3.txt*):

- Configure your router to do egress filtering, preventing spoofed traffic from exiting your network. Refer to http://www.sans.org/y2k/egress.htm for more information.
- Ask your ISP to configure their router to do ingress filtering on your network, preventing spoofed traffic reaching the Internet from your network. Refer them to RFC 2267.
- Use a firewall that exclusively employs application proxies. This should effectively block all TFN2K traffic. Exclusive use of application proxies is

© SANS Institute 2000 - 2005 Author retains full rights.

often impractical, in which case the allowed non-proxy services should be kept to a minimum.

- Disallow unnecessary ICMP, TCP, and UDP traffic. Typically only ICMP type 3 (destination unreachable) packets should be allowed.
- If ICMP cannot be blocked, disallow unsolicited (or all) ICMP_ECHOREPLY packets.
- Disallow UDP and TCP, except on a specific list of ports.
- Spoofing can be limited by configuring the firewall to disallow any outgoing packet whose source address does not reside on the protected network.
- Take measures to ensure that your systems are not vulnerable to attacks that would allow intruders to install TFN2K.

References

SANS Institute. Track 2 – Firewalls, Perimeter Protection and VPNs. Conference Material 2002

NSA. Cisco Security Configuration Guide. http://nsa1.www.conxion.com/cisco/guides/cis-2.pdf.
March 25, 2002

Brenton, Chris. Mastering Cisco Routers. SYBEX Inc., 2000

Davis, Carlton R. IPSec, Securing VPNs. RSA Press, McGraw-Hill, 2001

Zwicky, Elizabeth D., Cooper, Simon, Chapman, Brent D. Building Internet Firewalls. O'Reilly A Associates, 2000

Albitz, Paul, Lui, Cricket. DNS and BIND, 4th Edition. O'Reilly & Associates, 2001

Carr, Jim. "Thwarting Insider Attacks.", Network Magazine. September 2002: 42-46

URL Links

RFC References - http://www.ietf.org/rfc/rfc
OpenBSD – http://www.openbsd.org
pf homepage - http://www.benzedrine.cx/pf.html
Cisco Connection Documentation - http://www.cisco.com/univercd/home/home.htm
Nmap – http://www.insecure.org/nmap
Sun Microsystems – http://www.sun.com
F5 - http://www.f5.com/f5products/bigip/520-540/index.html
McAfee - http://corporate.mcafee.com/content/software_products/avd_webshield_e500.asp
Nessus – http://www.nessus.org
CERT Overview of Denial of Service Attacks - http://www.cert.org/tech_tips/denial_of_service.html
Design and Performance of the OpenBSD Stateful Packet Filter (pf) - http://www.benzedrine.cx/pf-slides.pdf
Packetstorm – http://www.packetstormsecurity.nl
Cisco's Order of Operation – http://www.cisco.com/warp/public/556/5.html

© SANS Institute 2000 - 2005, Author retains full rights.

Internet Assigned Numbers Authority - http://www.iana.org/assignments/ipv4-address-space
Checkpoint FW-1 Resource – http://www.phoneboy.com

© SANS Institute 2000 - 2005, Author retains full rights.