Global Information Assurance Certification Paper

SANS GIAC Certified Firewall Analyst (GCFW) Practical Assignment

Version 1.7

GIAC Enterprises

A secure network design for the small to medium sized enterprise.

Kirk Ismay

June 4, 2002

Table of Contents

Introduction

Overview

At GIAC Enterprises, we will implement a secure network infrastructure to support our growing e-commerce business through online sales of original fortune cookie sayings in many languages. We have two revenue streams; online sales directly to consumers, and subscriptions to our fortune sales interface (an ASP-style web based application allowing resellers to sell fortunes from their own sites.)

At GIAC Enterprises, we're focused on using open source software to grow our online business, as Linux, OpenBSD, Apache and other open source programs have a proven track record of security and stability at a lower cost than most other platforms.

Assumptions

- GIAC Enterprises is a small but growing company with approximately 50 employees.
- The GIAC Enterprises programming team is composed of Linux/Apache talented web application programmers.

Part 1: Security Architecture

Access Requirements

Customers

- May view corporate information on our public web site (http – 80).
- Purchase fortunes via our SSL encrypted web site (https- 443).
- Send email to employees (smtp – 25).
- Both require access to our DNS server (dns – 53).

Suppliers

- Will upload fortunes via SSL page using HTTPS (448).
- Authenticate using SSL client certificates.

Partners

From an IT perspective, GIAC enterprises have two distinct types of business partners. We have translators and we have resellers. Therefore, they have different access requirements and service needs:

Translators

- HTTPS access to a web based application to allow the translation of fortunes.

Resellers

- We will provide a branded and customized HTTPS purchasing system for our resellers as an application service provider (ASP).
- They will also have HTTPS access to reports, statistics, and online orders

Both resellers and translators will require valid SSL client certificates to access their portions of the site.

Internal Employees

- General Web & FTP access.
- SMTP, POP & IMAP email service on the internal network.
- IT admin staff require outbound SSH & Telnet access, as well as using ping and traceroute utilities to troubleshoot network problems.
- Programmers and DBA's require access to the database server.
- Internal access to file and print services

Mobile Employees

- IPSec VPN access to the above services.
- All have laptops and 802.11b wireless Ethernet cards, and want access to the network. Rather than rely on WEP for security, we will employ IPSec on the wireless network.

System Administration Staff

At GIAC Enterprises, we don't have a 24x7 onsite system administration presence. We instead rely on programs like Swatch[1] and Big Brother[2] to monitor the health of critical systems, and notify system administration staff by pager or cellular phone in case of emergency. The system administration staff requires remote management capabilities to take care of minor emergencies from home.

To enable remote management, we will allow SSH access to service network systems from the Internet via a single SSH admin system. The SSH admin system will be a hardened OpenBSD server.

The service network systems will also require FTP & HTTP access to the Internet to download software updates and security patches. This access will be provided via a proxy server running on the SSH admin system.

Our ISP

- If providing backup DNS services, their DNS server would need to be allowed TCP access to port 53 of our DNS server and be allowed to do zone transfers.
- If they were a backup email relay they would need TCP access to our SMTP server. Our mail server would need to be configured to allow relays from all backup MX hosts. An alternative option for backup DNS and SMTP service would use a co-located server running DNS and SMTP services with its own onboard firewall.

[1] "SWATCH: The Simple WATCHer": http://www.oit.ucsb.edu/~eta/swatch/

[2] "Big Brother System and Network Monitor": http://bb4.com/

Network Design

Network Diagram

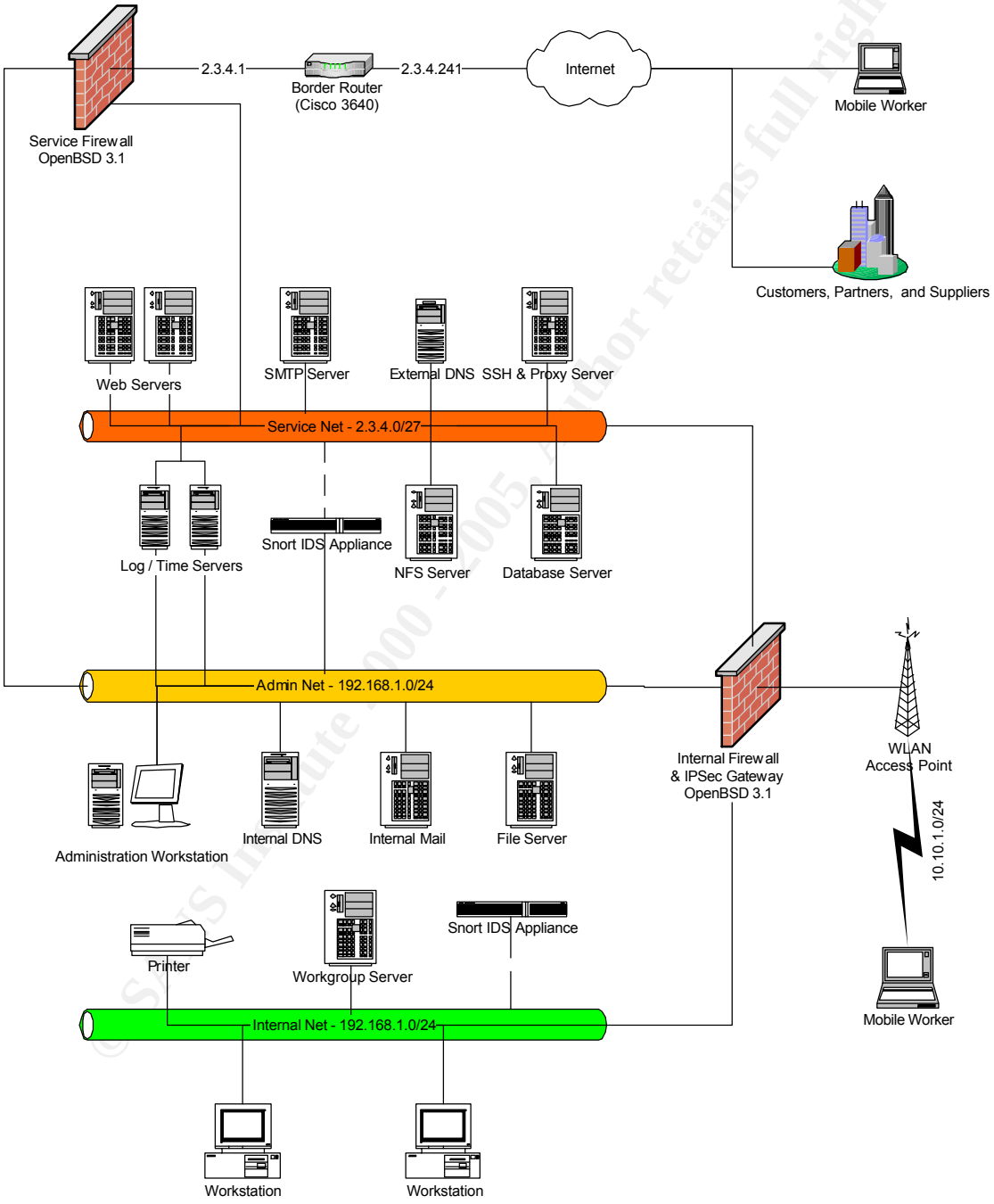

Figure 1: GIAC Enterprises Network Diagram

Security Components

A secure network requires a layered design approach. The last few years have proven that no one system can provide 'ultimate security', regardless of whether we're talking about firewalls, VPN solutions or operating systems. As a result we need to design secure networks using a layered approach, each layer contributing to overall defense in depth.

Each component will require not only initial setup and hardening, but also regular maintenance and observation to ensure the security of the enterprise.

Filtering Router

We will use a Cisco 3640[3] router for our border router. This is a fairly robust modular router, more than adequate for the needs of GIAC enterprises. It includes QOS and IPSec VPN capabilities, and can be managed via SSH.

Firewalls

OpenBSD 3.1 PacketFilter[4] will be used for the network level firewalls, as it has an excellent security track record and is extremely flexible. OpenSSH 3.4 is installed for secure remote administration of the system, and set to listen on internal or administrative network interfaces only.

Linux NetFilter[5] will be used on all Linux servers in the service network for an additional host level firewall solution.

VPNs

HTTPS will be used to create an application level VPN for our translators and business partners. We are using HTTPS instead of IPSec for a number of reasons:

- We have web application programmers on staff already.
- HTTPS enabled browsers are easier and cheaper to deploy than IPSec clients, and require less CPU resources than encrypting the entire network connection.
- We give them access to a web application to make changes to the database, rather than direct access to the database itself.

IPSec will be used to allow employees to telecommute from their home broadband Internet connections, and to encrypt the data on the staff wireless network.

---

[3] "Cisco 3600 Series" URL: http://www.cisco.com/univercd/cc/td/doc/pcat/3600.htm

[4] "OpenBSD" URL: http://www.openbsd.org/faq/faq6.html#PF

[5] "NetFilter/IPTables". URL: http://www.netfilter.org/

IDS & Log Analysis

Intrusion detection is as important as having a firewall for providing in depth security. While a firewall provides the 'locked doors' to our network, an IDS provides an 'alarm system' to let us know when the locks have been breached.

In our network design a Snort[6] IDS system will be attached to a span port on the Catalyst switch, and one on the internal LAN hub. This will allow us to watch traffic that is allowed through the firewall and alert us to suspicious traffic.

Snort was chosen for our IDS product for its comprehensive rule base, and the expertise of the snort developer community in general.

In addition to the Snort network IDS sensors, Swatch will be configured to analyze syslog information on the log hosts for evidence of intrusive behavior, and send alerts to security staff. This would include failed authentication requests, kernel messages, and daemon log entries.

The AIDE[7] (Advanced Intrusion Detection Environment) is used to ensure the integrity of files on our systems. AIDE has a database of cryptographic checksums (including MD5 & SHA-1) of each file, and if any file is modified even slightly, the cryptographic checksum will be different. AIDE can be used to determine which files have changed after an intrusion; it can also detect newly added files, and changes to file attributes. Running AIDE nightly will alert us to an intruder that has successfully gotten through our other layers of security, and provides the last line of defense.

Hubs & Switches

Cisco 2950[8] series Catalyst switches will be used for their management and on board security features on the service networks. Unused ports will be turned off.

Standard hubs will be used on the internal network for office workstations, making it easier to deploy IDS systems to monitor internal users, and because hubs are much less expensive. Any unused physical Ethernet jacks will be disconnected from the hubs at the patch panel in the electrical room.

---

[6] Snort: The Open Source Network Intrusion Detection System http://www.snort.org/

[7] "AIDE – Advanced Intrusion Detection Environment" http://www.cs.tut.fi/~rammer/aide.html

[8] "Cisco Catalyst 2950 Series Switches" http://www.cisco.com/en/US/products/hw/switches/ps628/index.html

Software

Unless otherwise noted, these are the software versions in use at GIAC enterprises:

Operating Systems

- Cisco IOS 12.1
- Debian GNU/Linux 3.0 (2.4.19 series kernel)
- OpenBSD 3.1
- Windows 2000 SP 2

Security Software

- AIDE 0.9 (HIDS)
- Grisoft AVG 6.0 anti-virus
- Zone Alarm Pro 3.1 firewall
- Trend Micro ServerProtect for Linux (Anti-Virus)
- Trophie 1.12& Virge 3.03[9] (email AV solution for Linux)

Applications / Servers

- Apache 1.3.26 & mod_ssl 2.8.7-1
- Big Brother 1.9c
- Bind 8.3.3
- OpenSSH 3.4 (3.4p1 for Linux)
- Postgresql 7.2.1-2
- Postfix 1.1.11 (internal smtp)
- Sendmail 8.12.3.-4 (external smtp)
- Squid 2.4.6
- Syslog-ng 1.5.15
- Snort 1.8.6
- Swatch 3.0.4

Admin staff will typically patch systems for security problems within hours of an announcement of a vulnerability (or patch), or will take services offline if possible.

Internet programs running as daemons are also locked down as much as possible (ie running as non-root, chrooted, and set to listen on specific interfaces/IP addresses)

For example, the SSH daemon on the internal firewall will have the ListenAddress set to the IP of the interface on the Admin network only, in addition to firewall rules restricting SSH connections to the admin network.

System Backup and Recovery

In the unlikely even of a successful intrusion, the most likely remedy will be to restore the system data using backup media. Backups of all core systems (router, firewalls, and servers) will be made nightly, when possible full backups will be made, but for larger file systems, weekly full backups with daily incremental backups will be necessary. Backups will be made using a tape

[9] "vanja.com – Tools". URL: http://www.vanja.com/tools/

drive installed on a system on the Administration network using GNU tar tunneled over SSH. Backups will also be tested periodically to ensure data can be restored.

The procedure for restoring a compromised system will be as follows:

1. Physically disconnect the system from the network.
2. Take a snapshot of the hard disk contents for future analysis and auditing. Recovery of non-executable data would also be permitted. (The Unix dd utility can be used to copy entire partitions to another disk or tape)
3. Wipe the hard disk, and then restore the operating system, applications, and data from backups or original installation media.
4. Install up to date patches or alternative applications to prevent the intrusion from occurring again.
5. Review firewall rules & IDS signatures.
6. Restore the system to active service.
7. If possible, share lessons learned with the security community, sanitizing logs and other data to prevent leaking sensitive information.

Administration Server

Our service network will be well protected by strict rules on the border firewall. By default, systems on the service network will not be able to initiate connections to hosts on the Internet, with the exception the SMTP and DNS servers that need to communicate with their peers to operate.

This will make remote administration and the installation of patches somewhat more difficult. To resolve this issue, we will use a hardened OpenBSD system for administrative purposes. It will provide http & ftp proxy capability to hosts on the service network, using a Squid proxy on running. Authentication will be required to use the proxy.

This system will also be used for off site remote administration of other systems on the service network. System administrators will be able to SSH to the admin system, then use SSH to connect to the other service network hosts. SecureID SSH authentication methods will be implemented for additional security. The SSH daemon will run on a non standard port (2222), to keep it safe from automated attacks probing the standard SSH port.

Swatch will be used to monitor the system logs for signs of unauthorized intrusions, and alert security staff if there is any suspicious activity. Staff will disable SSH access on this system in reaction to any threat if necessary. (Repeated failed login attempts, vulnerability in OpenSSH discovered)

This will allow staff to maintain the service network, and minimize the possibility of the Administration server becoming a threat to other hosts if compromised.

Hardened Systems

There is more to hardening a system than can be covered in this paper. This section is meant to give a quick overview of the GIAC Enterprises hardening process, rather than an exhaustive complete guide.

Debian

We harden Debian systems[10] by first installing only the base operating system. We also add security.debian.org to our apt sources list, so that we get a Debian installation with all the latest security updates from the start. Once that is complete, we use apt (the Debian package installer) to install only the packages required. This approach does require Internet access during installation, but since our external firewall does not allow access to or from unregistered systems, the risk is mitigated. We have to use the proxy server on the Admin server to download packages. Another option is to use a local mirror of the Debian security site on a local server.

This allows fine-grained control of what goes into the final product. The end result is a system running only the services we require and no others.

In addition to running only required services, we also:

1. Mount file systems (ie /tmp, /var) nodev, noexec, nosuid where possible. This measure can defeat some exploits.
2. Enable md5 passwords, to allow passwords longer than 8 characters
3. Static kernel, loadable modules are not allowed, to prevent kernel module root kits.
4. Install the Debian harden package, which automates the removal of known insecure packages.
5. Consider a modified kernel with either ACL's or executable stack protection such as OpenWall, grsecurity, LIDS, or NSA Security Enhanced Linux.
6. Subscribe to the Debian security announcements mailing list.

One thing to note, is that the Debian maintainers will often back port a security patch to the distributed version of a package, rather than upgrade to the latest upstream version. This means that if normally foobar version x.y.z has a new buffer overflow, it may be patched in Debian. It will usually have an additional version tag like foobar x.y.z-4, indicating the Debian patch level.

[10] "Securing Debian Manual"
URL: http://www.debian.org/doc/manuals/securing-debian-howto/index.en.html

OpenBSD

OpenBSD systems are pretty secure “out of the box”, in that most services ship turned off. The ones that are have been extensively audited for security flaws. However, there are a few unnecessary services that should be turned off. The first step is to edit /etc/rc.conf and set the following lines as shown:

```
pf=YES                  # Packet filter / NAT
sendmail_flags=NO
portmap=NO
inetd=NO
ntpd=NO
```

We then edit /etc/ssh/sshd_config to turn off ssh1 support, and listen to our admin IP only:

```
Protocol 2
ListenAddress 192.168.1.10
```

Windows 2000

Installing regular updates including Service Pack 2, is the critical part of the Windows 2000 workstation hardening process. We also remove unneeded services, including IIS and the MSSql Server, and applications (where possible).

Zone Alarm Pro was chosen for our personal firewall, and AVG 6 Professional is our anti-virus program of choice.

Log & Time Servers

For redundancy purposes, our network design calls for 2 log / time servers. They will use identical, though modest, hardware. All they require are reasonably large hard disks. For our operating system, one system will run OpenBSD, the other Debian. Both will be hardened and running internal firewall software. This will make it more difficult to remotely compromise our log hosts, because it would be highly unlikely to have a remotely exploitable hole in both operating systems at the same time.

Each host has a network interface on both the external and admin network for management. SSH access is only permitted from the admin network. IP forwarding and NAT facilities have been disabled or uninstalled from the systems

Physical Security

Physical security is another layer in a good defense in depth strategy. The data center will be physically secure behind locked doors. In addition to locks and alarms, physical key management is important. Employees will be required sign a key log before being given keys to the building and alarm codes. A separate key would be required for access to the data center.

An electronic key card system if business need justifies the expense. Video

surveillance in the data center should also be considered.

Security Policy

Security Policy makes up an important part of the overall security picture. It can be thought of as the "Human Firewall[11]". The humanfirewall.com site has an excellent checklist of 8 essential steps:

1. Get top management buy-in and commitment.
2. Assign and clarify roles and responsibilities.
3. Create an Action Plan with a budget.
4. Develop and/or update information security policies.
5. Develop an organization-wide Security Awareness/Education program.
6. Measure the progress of your Security Awareness/Education efforts.
7. Adapt and improve your Security Awareness/Education programs according to progress/feedback.
8. Develop an information security incident response team and plan.

The development of security policy for GIAC Enterprises will incorporate principles from this list.

[11] "8 Essential Steps to Building a Human Firewall": http://www.humanfirewall.com/

Part 2: Security Policy and Tutorial

IP Addresses

IP addresses and hostnames for the GIAC Enterprises Network:

IP	Name	Notes
2.3.4.241	router-ext.srv.giacent.com	Router external interface.
2.3.4.1	router.srv.giacent.com	Router internal interface.
2.3.4.2	ns1.giacent.com	Primary DNS
2.3.4.3	mail1.giacent.com	Primary MX
2.3.4.4	db1.srv.giacent.com	Database Server
2.3.4.5	nfs1.srv.giacent.com	NFS
2.3.4.10	http1.srv.giacent.com	http/https server
2.3.4.11	http2.srv.giacent.com	http/https server
2.3.4.20	log1.srv.giacent.com	Syslog/time server
2.3.4.21	log2.srv.giacent.com	Syslog/time server
2.3.4.30	admin.srv.giacent.com	SSH Admin & Proxy Server
2.3.4.31	ifw.srv.giacent.com	Internal Firewall/ IPSec Gateway
2.3.4.32	adm.srv.giacent.com	Admin NAT address
2.3.4.33	lan.srv.giacent.com	Lan NAT address
192.168.1.25-49		Admin internal IPs
192.168.1.50-99		Internal server IPs
192.168.1.100+		Internal LAN IPs

DNS Zone Security

For internal use, we have setup the srv.giacent.com zone for GIAC staff, queries and DNS transfers are restricted to the GIAC network. Only systems that need to be accessed externally have generic names in the giacent.com zone (ie mail, www, etc). This reduces the amount of information that can be gathered by an intruder using DNS as a reconnaissance tool.

Border Router

For the border router, we will restrict access to any administration services to the internal interface, allowing connections these services from the administration network only. We will also set up a basic anti spoofing ACL on the router as well. We are using only basic ACL's to reduce complexity and increase overall performance. As with all other hosts we have disabled unnecessary services on the router.

Here is the router configuration:

```
! Set the host and domain name
   hostname GIACRouter

! disable unecessary services/features[12]
   no cdp run
   no service tcp-small-servers
   no service udp-small-servers
   no ip finger
   no ip bootp server
   no ip http server
   no ip source-route
   no snmp
   no ip domain-lookup

! enable ssh[13]
   crypto key generate rsa
   ip ssh timeout 60
   ip ssh authentication retries 3

! enable log timestamps
   service timestamps debug uptime
   service timestamps log uptime
   service password-encryption

! Set log hosts
   logging buffered
   logging 2.3.4.20
   logging 2.3.4.21

! Stern warning banner on login consoles
   banner /
     WARNING: Authorized Access only. Unauthorized users will be
   punished to the fullest extent of the law!/
```

[12] SANS 2.3 - Firewalls 102: Perimeter Protection and Defense in-Depth

[13] "Configuring Secure Shell on Cisco IOS Routers"
http://www.cisco.com/warp/public/707/ssh.shtml

```
! Use password encryption
   enable secret 5 $1$b7aF$TaGNer6M6KiTplS.xmNHu0

! routing options
   ip subnet-zero
   ip routing
   ip classless

!!! ACLs START [14]

!! Telnet/SSH Service ACL
   access-list 1 permit 2.3.4.30   ! Admin Server
   access-list 1 permit 2.3.4.32   ! Admin Workstation NAT IP
   access-list 1 deny any log

!! External Interface Spoofing protection

! Deny rfc 1918 addresses:
   access-list 10 deny 192.168.0.0 0.0.255.255 log
   access-list 10 deny 172.16.0.0 0.15.255.255 log
   access-list 10 deny 10.0.0.0 0.255.255.255 log

! Deny packets with localhost, broadcast and multicast
addresses:
   access-list 10 deny 127.0.0.0 0.255.255.255 log
   access-list 10 deny 255.0.0.0 0.255.255.255 log
   access-list 10 deny 224.0.0.0 7.255.255.255 log

! Deny packets without ip address.
   access-list 10 deny host 0.0.0.0 log

! Prevent spoofing. Deny incoming packets that have
! our internal addresses:
   access-list 10 deny 2.3.4.0 0.0.0.31 log

! More spoofing prevention, external
! router interface ip address:
   access-list 10 deny host 2.3.4.241 log
   access-list 10 permit any

!! Internal Interface - only our network
   access-list 20 permit 2.3.4.0 0.0.0.31
   access-list 20 deny any log

   ! These ACL's were derived from Frank Keeney's
   ! paper: "Screening Router Access List"
   ! http://www.pasadena.net/cisco/secure.html
```

[14] Keeney, Frank. "Screening Router Access List" 1998-12-30.
URL: http://pasadena.net/cisco/secure.html

```
!!! END ACLs
```

```
! Interface configuration

   interface Ethernet 0/0  ! External (Internet) Interface
    no shutdown
    ip address 2.3.4.241 255.255.255.248
    ip access-group 10 in
    no ip unreachables               ! Prevent
   traceroute/firewalking
    no ip directed-broadcast     ! Prevent smurf attacks
    keepalive 10

   interface Ethernet 1/0  ! Internal Interface
    no shutdown
    ip address 2.3.4.1 255.255.255.224
    keepalive 10

   line console 0
    exec-timeout 0 0
    password secret
    login

   line vty 0 4
    access class 1 in
    password moresecret
    transport input ssh         ! Only allow ssh connections
    login

   end

!!! THE END !!!
```

External Firewall

The external firewall at GIAC Enterprises is an OpenBSD 3.1 system set up as an Ethernet bridge[15] rather than as a router. The advantage of this method is that the firewall does not require any IP addresses for the interfaces on the service network. A third network interface is provided for administrative remote access.

We will use the OpenBSD Packet Filter (PF) firewall software. PF is a stateful firewall, and is optimized so that state table lookups are faster than rule evaluations. PF rules are interpreted in a last matching rule wins format, so you can start with a default "block all" rule and then use specific rules to allow access to servers and services. PF also supports shell style variables to make writing and maintaining rules easier.

Before we can use PF, we do have to remember to enable PF in the /etc/rc.conf file, so that PF is enabled when we reboot the system. Edit rc.conf and change the line pf=NO to read pf=YES.

Reading PF rules

- # is a comment character, everything past the # is ignored to the end of the line.
- Pass rules allow traffic through, block rules don't.
- The quick keyword added to either will override rule order (if the condition of the rule matches, further rule interpretation stops, and the packed is passed or blocked immediately).
- A line with `port { 32><36 }` indicates a range of ports, in this case ports 33,34,35 would be affected. (The range doesn't include the range boundaries) It may help to read that as greater than 32 and less than 36.

External Rules

```
# /etc/pf.conf
# PF allows variable declarations.
   intIF="xl0"   # internal interface
   extIF="xl1"   # external interface
   admIF="fxp0"  # admin interface

# Hosts
   webservers="{ 2.3.4.10/32, 2.3.4.11/32 }"
   dnsservers="2.3.4.2/32"
   smtpservers="2.3.4.3/32"
   sshservers="2.3.4.30/32"
   timeservers="{ 2.3.4.20/32, 2.3.4.21/32 }"
```

[15] The Open BSD Packet Filter, section 3 Filtering Bridges: http://www.inebriated.demon.nl/pf-howto/html/node4.html

```
   logservers="{ 2.3.4.20/32, 2.3.4.21/32 }"
   internalfw="2.3.4.31/32"
   externalfw="192.168.1.10/32"
   router="2.3.4.241/32"

# Networks
   admins="2.3.4.32/32"
   servernet="2.3.4.0/27"
   internalnet="2.3.4.33/32"


# Private or reserved network space (Bogus Networks)
# There is an extensive list developed by Rob Thomas at:
#   http://www.cymru.com/Documents/bogon-list.html

# My list just contains the basic private address ranges.
   bogons="{ 0.0.0.0/8, \
           127.0.0.0/8, \
           192.0.2.0/24, \
           10.0.0.0/8, \
           172.16.0.0/12, \
           192.168.0.0/16, \
           169.254.0.0/16}"
# PF Rules

# Pass everything on the internal interface.
# Otherwise we have to write every rule twice for each one
# (because we are bridging).
   pass in  quick on $intIF all
   pass out quick on $intIF all

# block everything by default
   block in  on $extIF all
   block out on $extIF all

# don't accept bogus packets from strangers...
   block in  log quick on $extIF from $bogons to any
# don't let out bogus traffic either
   block out log quick on $extIF from ! $servernet to any

# allow incoming traffic from the router
   pass in on $extIF proto udp from $router to $logservers \
     port 514 keep state
   pass in on $extIF proto { tcp, udp } from $router \
     to $timeservers port 123 keep state

# allow outside access to our servers
   pass in on $extIF proto tcp from any to $webservers \
     port { 80, 443 } flags S/SA keep state
   pass in on $extIF proto tcp from any to $smtpservers \
     port 25 flags S/SA keep state
   pass in on $extIF proto { tcp, udp } from any to $dnsservers
```

```
   \
     port 53 flags S/SA keep state
   pass in on $extIF proto tcp from any to $sshservers \
     port 2222 flags S/SA keep state

# pass in IPSec traffic to our internal firewall
   pass in on $extIF proto esp from any to $internalfw keep
   state
   pass in on $extIF proto udp from any port 500 to $internalfw
   \
     port = 500 keep state

# allow pings to our dns & web servers
   pass in on $extIF proto icmp from any to $webservers \
     icmp-type echoreq keep state
   pass in on $extIF proto icmp from any to $dnsservers \
     icmp-type echoreq keep state

# allow dns queries from our network
   pass out on $extIF proto { tcp, udp } from $dnsservers \
     to any port 53 flags S/SA keep state
   pass out on $extIF proto { tcp, udp } from $admins \
     to any port 53 flags S/SA keep state

# allow limited access from our admin server
   pass out on $extIF proto tcp from $sshservers \
     to any port { 21, 22, 25, 80, 443 } keep state
   pass out on $extIF proto icmp from $sshservers \
     to any icmp-type echoreq keep state

# allow smtp servers to send mail
   pass out on $extIF proto tcp from $smtpservers \
     to any port 25 flags S/SA keep state

# allow time servers to synchronize with internet time servers
   pass out on $extIF proto { tcp, udp } from $timeservers \
     to any port 123 flags S/SA keep state

# allow admins to access the internet.
   pass out on $extIF proto { tcp, udp } from $admins \
     to any keep state
   pass out on $extIF proto icmp from $admins \
     to any icmp-type echoreq keep state

# corporate users can access the internet too.
   pass  out on $extIF proto { tcp, udp } from $internalnet \
     to any keep state
   pass  out on $extIF proto icmp from $internalnet \
     to any icmp-type echoreq keep state

### Egress Filters ###
# Note that port { 160><163 } is the pf range notation, which do
# not include the range boundaries. So this example blocks
```

```
# ports 161 and 162 (snmp & snmptrap).

# block access to the router for every one but the admin
network.
   block out log on $extIF from ! $admins to $router

# block telnet, pop, imap, syslog, lp etc
   block out on $extIF proto tcp from any to any \
     port { 23, 110, 143, 511><516 }

# netbios, portmap
   block out on $extIF proto { tcp, udp } from any to any \
     port { 136><140, 111, 445 }

# tftp, snmp
   block out on $extIF proto udp from any to any \
     port { 69, 160><163 }
```

Conclusions

This rule set leaves us with a fairly strong security stance. We are blocking all traffic by default, including outgoing network traffic. In this case, a new system brought online in the server network would not have access to the Internet, nor would it be accessible from the Internet until the rule set was changed. This is very important, as research by the HoneyNet[16] project indicates that a system can be hacked into within 15 minutes of being plugged into an Internet connection. Our database and NFS servers automatically fall under this umbrella of protection, so only systems on the service network have access. They are not visible to the Internet.

On the internal network, new systems would have access to the internet, but because of the keep state rules and network address translation, only traffic they request will get back to them. We are also blocking access to the most common clear-text authenticated services to prevent passwords from being sniffed on the wire as employees check their ISP's POP account. While slightly weaker than the default deny stance of the service network, the possibility of error increases when IT staff have to add each new workstation to the outer firewall rules.

One way to improve the security posture of the service network would be to add a third network interface to our bridge and place the database and NFS servers on that interface. Then we can create specific rules to allow specific systems to access the support servers. Since the database and NFS servers are running Linux NetFilter, and are hardened systems, it was decided the additional complexity and cost of extra hardware (we would need another switch) was outweighed by the minimal benefit this solution would provide.

If we had an especially vulnerable or difficult to harden system in use, then the

[16] "Know Your Enemy: Statistics" http://project.honeynet.org/papers/stats/

additional interface and layer of security would probably be worth the cost.

Intsernal Firewall & IPSec VPN

Our internal firewall is again using the OpenBSD 3.1 packet filter. We have two interfaces in bridge mode (the admin and internal network), an external interface on the service network and a fourth interface tied in to the wireless network. The wireless network uses a separate private address subnet. Mobile workers use IPSec to tunnel in to the main office LAN when using the wireless network.

Network address translation will be used between the two bridge interfaces and the service network to connect to the Internet.

Before we can use NAT, we have to enable ip forwarding first, by editing /etc/sysctl.conf and changing the net.inet.ip.forwarding line to:

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

The command: `sysctl -w net.inet.ip.forwarding=1` allows you to enable forwarding without a reboot.

Here is the configuration for the internal network:

NAT Configuration

```
#  NAT Configuration from: /etc/nat.conf
#  Administration Stations
   nat on fxp0 from 192.168.1.25/32 to any -> 2.3.4.32
   nat on fxp0 from 192.168.1.26/32 to any -> 2.3.4.32
# Everything Else
   nat on fxp0 from 192.168.1.0/24 to any -> 2.3.4.33


# Redirect rules, these rules forward ports on the
# external interface to systems on the inside, but only if
# the traffic is coming from trusted addresses
   rdr on fxp0 from 2.3.4.3/32 to 2.3.4.33/32 -> \
     192.168.1.53 port 25       # Internal Mail Server
   rdr on fxp0 from 2.3.4.0/27 to 2.3.4.32/32 port 1984 -> \
     192.168.1.55 port 1984          # Big Brother traffic
```

Internal Rules

```
# Interfaces
   extIF="fxp0"       # external interface
   intLanIF="dc0"     # internal LAN interface
   intSvcIF="dc1"     # internal Service/Admin Network interface

## External Interface ##

# block everything by default
   block in  on $extIF all
   block out on $extIF all
```

```
# pass in IPSec traffic
   pass in on $extIF proto esp from any to $internalfw keep
   state
   pass in on $extIF proto udp from any port 500 to $internalfw
   \
     port = 500 keep state

# pass in traffic to our mail server
   pass in on $extIF proto tcp from $extMailServers \
     to $lanNATaddr port 25 flags S/SA keep state

# big brother monitoring port
   pass in on $extIF proto tcp from $servernet to $admNATaddr \
     port 1984 flags S/SA keep state

# allow dns queries from our network
   pass out on $extIF proto { tcp, udp } from any \
     to $dnsservers port 53 flags S/SA keep state

# pass out regular LAN traffic
   pass out on $extIF proto { tcp, udp } from { $lanNATaddr, \
     $admNATaddr } to any keep state
   pass out on $extIF proto icmp from { $lanNATaddr, \
     $admNATaddr } to any icmp-type echoreq keep state

# Egress filters
# block telnet, pop, imap, syslog, lp etc
   block out on $extIF proto tcp from any to any \
     port { 23, 110, 143, 511><516 }

# netbios, portmap
   block out on $extIF proto { tcp, udp } from any to any \
     port { 136><140, 111, 445 }
```

Wireless Interface Security

The wireless interface represents a big challenge, as it extends the reach of the network out into the parking lot. There was some disagreement between the management staff and security staff on whether or not to even deploy it, but in the end the boss got it their way. We did win one concession though, in the event of any intrusive activity from the wireless LAN, security staff is authorized to turn off the wireless network immediately. In addition to having support from management for our security policy, we will also use our firewall and VPN to full advantage to compartmentalize the wireless and Ethernet networks.

The wireless access point, an ORiNOCO AP-2000[17] has its own DHCP server, and MAC address access control. Both have been enabled so that we don't have to allow DHCP traffic into the internal LAN, and so that unregistered cards are not allowed on the network. The AP-2000 also supports 802.1x authentication for further access control.

```
## Wireless LAN ##
   wiIF="fxp1"
   internalfw="10.10.1.1/32"

#block everything by default
   block in log on $wiIF all
   block out log on $wiIF all

# pass only IPSec traffic to our internal firewall
   pass in on $extIF proto esp from any to $internalfw keep
   state
   pass in on $extIF proto udp from any port = 500 to \
     $internalfw port = 500 keep state
```

IPSec VPN Configuration

For the IPSec VPN, we are using the SSH Sentinel IPSec client version 1.4. The internal firewall acts as our security gateway. The SSH Sentinel client is configured to use a virtual IP address on the internal network to communicate with hosts on the internal LAN that are using private IP addresses. Sentinel also has the ability to retrieve its policy configuration via HTTP or LDAP.

The following is the OpenBSD isakmpd configuration used in GIAC enterprises, it is the example configuration provided on the SSH support FAQ. I've modified it to use AES encryption and work with OpenBSD 3.1.

isakmpd.conf[18]

```
[Phase 1]
Default= ISAKMP-clients

[Phase 2]
Passive-Connections= IPsec-clients

[ISAKMP-clients]
Phase= 1
Configuration= Sentinel-main-mode
Authentication= thisisnotthatsecret

[IPsec-clients]
Phase= 2
Configuration= Sentinel-quick-mode
Local-ID= Local-net
```

[17] "ORiNOCO - AP-2000 Access Point" http://www.orinocowireless.com/template.html?section=m58&page=3040&envelope=94

[18] "SSH Support – FAQ" http://www.ssh.com/support/faq/sentinel/qa_2_898.html

```
Remote-ID= Remote-host

[Local-net]
ID-type= IPV4_ADDR_SUBNET
Network= 0.0.0.0
Netmask= 0.0.0.0

[Remote-host]
ID-type= IPV4_ADDR
Address= 0.0.0.0

[Sentinel-main-mode]
EXCHANGE_TYPE= ID_PROT
Transforms= AES-SHA

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

isakmpd.policy

```
Comment: This policy accepts ESP SAs from a remote that uses
the right password.
Authorizer: "POLICY"
Conditions: app_domain == "IPsec policy" &&
esp_present == "yes" &&
esp_enc_alg != "null" -> "true";
```

OpenBSD PF Tutorial

Implementing complex firewall rules using the OpenBSD PF firewall doesn't have to be a difficult task. If we simply organize the rule base into smaller logical sections, kept in separate files, we can easily make changes to the firewall policy without having to grovel through a single large file hundreds of lines long. If we align our logical divisions to parallel the pf rule parser, we can pick up performance benefits as well. The features we will use heavily in our tutorial are variables, lists, and "skip steps", and we will use samples from our external firewall rule set.

> *"For each rule, PF automatically calculates a so-called skip step for each of these parameters, which tells PF how many successive rules have the same value for the parameter…. So if you'd like to maximize your ruleset performance, you should sort your ruleset by interface, by protocol, source address and port, and finally by destination address and port, in that order."*[19]

With that performance tip in mind, here is how to structure a manageable set of pf rules. Our first step is to create a subdirectory of /etc/ called pf.d:

[19] "The OpenBSD Packet Filter HOWTO" (Basic Firewalling, Section 2.9) http://www.inebriated.demon.nl/pf-howto/html/node3.html

```
root:~# mkdir /etc/pf.d 0700
```

Then we create a file for variables. Having all variables declared in a single file allows new hosts to be added without changing the rule declarations. In our example, we could add a new web server to our cluster actually touching the rules.

```
# mg is a compact clone of the emacs text editor.
   root:~# mg /etc/pf.d/pf.vars

# Contents of pf.vars:
# PF allows variable declarations.
   intIF="xl0"   # internal interface
   extIF="xl1"   # external interface
   admIF="fxp0"  # admin interface

# Hosts
   webservers="{ 2.3.4.10/32, 2.3.4.11/32 }"
   dnsservers="2.3.4.2/32"
   smtpservers="2.3.4.3/32"
   adminstn="192.168.1.10/32"

#EOF
```

Now that we've declared some variables, lets make up some rules for each interface, which will be split up into a separate file for each interface:

```
# contents of pf.xl0  (Internal Interface)
# Pass everything on the internal interface.
# Otherwise we have to write every rule twice for each one
# (because we are bridging).
   pass in  quick on $intIF all
   pass out quick on $intIF all
#EOF

# contents of pf.xl1  (External Interface)
# block everything by default
   block in  on $extIF all
   block out on $extIF all
# allow outside access to our servers
   pass in on $extIF proto tcp from any to $webservers \
     port { 80, 443 } flags S/SA keep state
                                ...
#EOF
```

Did you see the last part of the webserver rule? In the clause `flags S/SA` , the part before the slash indicate the flags to match, and the part after the slash is a mask value indicating which flags are allowed to be set. This prevents invalid flag combinations from sneaking through the firewall. The last part, `keep`

state adds the matching packet to the state table, so that the rest of the packets in this connection from both sides are passed through.

```
# contents of pf.fxp0  (Administrative Interface)
# block everything by default
   block in  on $admIF all
# allow ssh access from the admin station
   pass in on $admIF proto tcp from $adminstn to 192.168.1.1/32
   \
     port 22 flags S/SA keep state
                                ...
#EOF
```

Now, in order to automate things a bit, we'll use a Makefile to automatically combine the separate files, check its syntax, and install it. For those that haven't used makefiles before, they are commonly used to automate a series of sequential tasks, often used in programming projects. It checks each target to see if the required files have been changed, and 'compiles' them into the finished product.

```
   root:~# mg /etc/pf.d/Makefile

#Contents of Makefile
   all: pf.conf
           @echo "now run 'make install' or 'make test'"

   test: pf.conf
           pfctl[20] -vnR pf.conf
           date > test.stamp

   install: pf.conf test.stamp
           cp /etc/pf.conf /etc/pf.conf.old
           cp pf.conf /etc
           pfctl -F rules -R /etc/pf.conf

   pf.conf: pf.vars pf.xl0 pf.xl1 pf.fxp0
           cat pf.vars pf.xl0 pf.xl1 pf.fxp0 > pf.conf
```

Once we have the Makefile, we now have a simple process to maintain our rules:

1. Edit the file(s) we need to change.
2. Run 'make test' to test our changes.
3. Run 'make install' to flush our existing rules, and install the new rule set into pf.

[20] "OpenBSD pfctl man page"
http://www.openbsd.org/cgi-bin/man.cgi?query=pfctl&manpath=OpenBSD+3.1&arch=i386

Part 3: Verify the Security Policy

Audit Plan

To audit policies in place on the GIAC Enterprises primary firewall we will use deploy two auditors and their systems on either side of the firewall. One will be the 'attacker', the other the 'detector' to see if the attack is detectable. After completing one set of tests, they will reverse roles to test egress filtering. The auditors will use various tools to attempt to bypass the firewall, with the detector station using tcpdump to watch for traces of the attack making it through. Figure 2 shows the logical layout of this process.

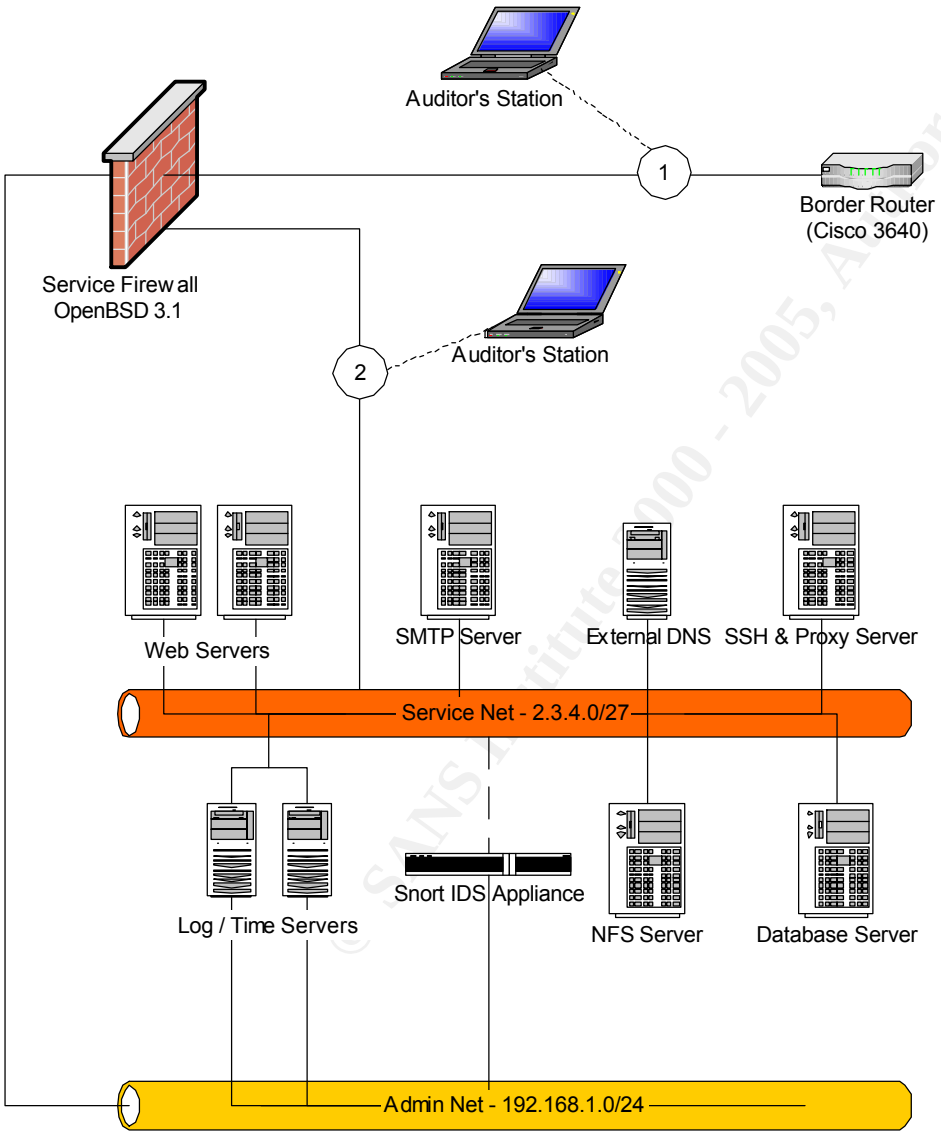

Figure 2: GIAC Enterprises Audit Plan

Tools

Our Auditors will use the following tools to test the firewall policy:

Nmap[21]: Nmap is an open source utility for network mapping and security auditing. It can do port scanning, remote OS fingerprinting, forge packets, and perform port scans with weird TCP flags (FIN, Null, Xmas) packets.

Netcat[22]: Netcat allows you to construct TCP or UDP connections to any source or destination p

Tcpdump/Windump[23]: This tool is a powerful packet sniffer and can be used to monitor network packets on the wire.

Pfctl: The pfctl utility comes with OpenBSD, and is used to manage and monitor the firewall rules. We can use it to get a hit count on our rule base.

Ping / Traceroute: Sure, they're not fancy, but they will tell us if a host is up or down, and how routers are in between us. Both the Windows and Unix variants will be used.

Ingress Tests

During ingress tests, the auditor at station 2 monitors network traffic with tcpdump, while auditor 1 conducts each test.

1. Use nmap ping scan to map the network. Only the DNS & Web servers should respond.
2. Use nmap to port scan the service network to detect unwanted open ports on the firewall or target hosts.
3. Use nmap's decoy feature to send spoofed packets from private networks. These packets should be dropped.
4. Repeat step 4 using the nmap FIN, Null, and Xmas scan options to see if the firewall can be fooled.
5. Use fragrouter and netcat to attempt to flood the firewall with a lot of TCP / UDP fragments.

---

[21] "Nmap: Network Mapper" http://www.insecure.org/nmap/

[22] Armstrong, Tom. "Netcat – The TCP/IP Swiss Army Knife". 2001-02-15. URL: http://rr.sans.org/audit/netcat.php

[23] http://www.tcpdump.org/

Egress Tests

During egress testing, the auditor at station 2 performs the tests while the auditor at station 1 watches the wire using tcpdump, looking for anomalies.

1. Test the default deny rule. The auditor will attempt to connect to hosts on the Internet.
2. Use nmap to attempt to bypass the firewall with FIN, Null or Xmas port scans directed at other auditor's system.
3. Use nmap's decoy feature to see if bogus addresses get through. Include an IP from a host on the service network.

For most of these tests, network performance will not be affected.

For the duration of the tests, the first audit host's IP address is 2.3.4.51, and the second's is 2.3.4.52.

Schedule & Budget

The network audit will be scheduled Sunday morning, between 12 and 4 am. This will prevent testing from interfering with normal operations.

Since we've scheduled the tests to last 4 hours, and require 2 auditors we will need to budget for 2 x 4 man-hours at time and a half. The results of the tests will require an estimated additional 8 man-hours of the senior security manager's time to analyze the data and compile the report. This can be scheduled for the next business day, so we don't have to pay overtime.

Resource	Total Time	Unit Cost	Total Cost
2 Auditors	2 x 4 = 12 hours	$40 * 1.5 ($60)	$720
1 Analyst	8 hours	$60	$480
Total			$1,200

Egress Test Results

nmap ping scan

 # nmap –sP 2.3.4.0/27

```
Starting nmap V. 2.30BETA18 by fyodor@insecure.org (
www.insecure.org/nmap/ )
Host ns1.giacent.com (2.3.4.2) appears to be up.
Host http1.srv.giacent.com (2.3.4.10) appears to be up.
Host http2.srv.giacent.com (2.3.4.11) appears to be up.
Host  (2.3.4.51) appears to be up.
Nmap run completed -- 32 IP addresses (4 hosts up) scanned in
1 second
```

Isn't that odd, there's an extra host here, and it doesn't have a DNS entry. It's not the wily hacker though, that's auditor 1's system. On auditor 2's system all we can see in the tcpdump log are the pings that were allowed through:

```
01:13:15.368070 2.3.4.51 > 2.3.4.2: icmp: echo request
01:13:14.884442 2.3.4.2 > 2.3.4.51: icmp: echo reply
```

nmap open ports

 # nmap –n 2.3.4.0/27

```
Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com,
www.insecure.org/nmap/)
Interesting ports on  (2.3.4.2):
(Not showing ports in state: filtered)
Port    State       Protocol  Service
53      open        tcp        domain

Interesting ports on  (2.3.4.10):
(Not showing ports in state: filtered)
Port    State       Protocol  Service
80      open        tcp        http
443     open        tcp        https

Interesting ports on  (2.3.4.11):
(Not showing ports in state: filtered)
Ports   State       Protocol  Service
80      open        tcp        http
443     open        tcp        https

Interesting ports on  (2.3.4.51):
Port    State       Protocol  Service
22      open        tcp        ssh
Nmap run completed -- 32 IP addresses (4 hosts up) scanned in
191 seconds
```

This scan shows almost exactly what we expected. Only the ports we allow through the firewall show up. Again we see the auditor system again, but that's

not the weird thing here. We don't see the mail server (smtp) and admin system (ssh), even though those ports are open. A little investigation revealed that nmap pings a host to see if its up first before performing the port scan. There is an option that turns this feature off, so lets try another scan (some results have been omitted):

```
# nmap -P0 2.3.4.0/32 -p22,25,53,80
   Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com,
   www.insecure.org/nmap/)
   Interesting ports on  (2.3.4.2):
   Port    State       Protocol  Service
   22      filtered    tcp        ssh
   25      filtered    tcp        smtp
   53      open        tcp        domain
   80      filtered    tcp        http

   Interesting ports on  (2.3.4.3):
   Port    State       Protocol  Service
   22      filtered    tcp        ssh
   25      open        tcp        smtp
   53      filtered    tcp        domain
   80      filtered    tcp        http

   Interesting ports on  (2.3.4.30):
   Port    State       Protocol  Service
   22      open        tcp        ssh
   25      filtered    tcp        smtp
   53      filtered    tcp        domain
   80      filtered    tcp        http

   Nmap run completed -- 32 IP addresses (6 hosts up) scanned in
   2 seconds
```

The tcpdump logs show only the standard 3 way tcp handshake sequence in the case of an open port, with nmap sending Fin-Ack to tear down the connection right afterwards. If the host isn't listening to a given port, it responds with Ack – Reset packet. Since nmap doesn't see the ack-reset sequence, nmap concludes that the port is filtered.

nmap decoy scan

```
# nmap –N -sS -D 192.0.2.2, 10.10.1.4,ME, \
  127.0.0.5, 172.16.0.23, 169.254.2.5
```

This showed the same ports open as the open port scan, so the output won't be repeated. Our tcpdump logs showed only the traffic from our host network. To ensure the packets were actually sent, we check the firewall hit count:

```
# pfctl -vs rules

   @6 block in quick on dc1 inet from 169.254.0.0/16 to any
   [ Evaluations: 80927      Packets: 1492         Bytes: 59392
   ]

   @7 block in quick on dc1 inet from 172.16.0.0/12 to any
   [ Evaluations: 79126       Packets: 1492        Bytes: 59392
   ]

   @8 block in quick on dc1 inet from 10.0.0.0/8 to any
   [ Evaluations: 77634       Packets: 1492        Bytes: 59392
   ]

   @9 block in quick on dc1 inet from 192.0.2.0/24 to any
   [ Evaluations: 76022       Packets: 1492        Bytes: 59392
   ]

   @10 block in quick on dc1 inet from 127.0.0.0/8 to any
   [ Evaluations: 74530       Packets: 1492        Bytes: 59392
   ]
```

This clearly shows the bogus packets from the decoy scan being discarded.

nmap FIN, Null, and Xmas scans

With each of the Fin, Null, and Xmas scans, the result was the same. All ports scanned were listed as open, at least on hosts that responded to ICMP echo requests. This was unexpected to say the least! The scan also took a very long time. Our tcpdump logs only showed ICMP and ARP requests. After some investigation by running tcpdump on the firewall's external interface showed what was going on; there was an initial Arp lookup followed by ICMP echo request. If the ICMP request was successful, packets were sent to each port, with no response from the target host. It seems that nmap interprets this lack of response as a sign of an open port:

```
03:27:51.752950 2.3.4.41 > 2.3.4.10: icmp: echo request
03:27:51.753061 2.3.4.41.33171 > 2.3.4.10.80: . ack 0
03:27:51.754017 2.3.4.10 > 2.3.4.41: icmp: echo reply
03:27:51.831117 2.3.4.41.33151 > 2.3.4.10.8: F 0:0(0)
03:27:51.842878 2.3.4.41.33151 > 2.3.4.10.78: F 0:0(0)
03:27:51.854682 2.3.4.41.33151 > 2.3.4.10.87: F 0:0(0)
```

We compared this against a Fin scan done in the lab:

```
12:39:26.529085 2.3.4.41 > 2.3.4.42: icmp: echo request
12:39:26.529129 2.3.4.42 > 2.3.4.41: icmp: echo reply
12:39:26.529491 2.3.4.41.48740 > 2.3.4.42.80: . ack 2535 96
12:39:26.529531 2.3.4.42.80 > 2.3.4.41.48740: R 2535:2535 (0)
12:39:26.846623 2.3.4.41.48720 > 2.3.4.42.265: F 0:0(0)
12:39:26.846656 2.3.4.42.265 > 2.3.4.41.48720: R 0:0(0) ack 0
12:39:26.847087 2.3.4.41.48720 > 2.3.4.42.911: F 0:0(0)
```

```
12:39:26.847116 2.3.4.42.911 > 2.3.4.41.48720: R 0:0(0) ack 0
```

The Xmas scan showed similar traces, only instead of just the FIN flag set, we have the FIN, PUSH, and URG flags set. The null scan has no flags set. In all cases, ports that are really open are sent an ACK as well, at least on UNIX hosts. I have been unable to find the cause of this behavior so far. It is possible to set up PF to filter on these flag combinations with the following addition to pf.conf:

```
block in log quick on $extIF inet proto tcp from any \
  to any flags FUP/FUP
block in log quick on $extIF inet proto tcp from any \
  to any flags SF/SFRA
block in log quick on $extIF inet proto tcp from any \
  to any flags /SFRA
```

It is recommended that these rules to the external firewall. Since the traffic is being blocked in any case, our service network is only marginally more secure. The real benefit of adding this rule is the fact that it any traffic logged by these rules will be coming from an intruder, and proactive measures can be taken.

nmap ACK scan

nmap -PT 192.168.100.40-43

```
Starting nmap V. 2.12 by Fyodor fyodor@dhp.com,
www.insecure.org/nmap/)
Nmap run completed -- 32 IP addresses (0 hosts up) scanned in
30 seconds
```

The ACK scan was completely bounced by the firewall, which was confirmed by the tcpdump logs.

Ingress Test Results

The ingress test results were almost boring. Nothing got through; and nothing was seen in our tcpdump logs. The following tests were performed:

1. Ping the router. (Unsuccessful)
2. Connect to http://www.sans.org/ with a web browser. (Unsuccessful)
3. Attempt to retrieve mail on the auditor's home ISP POP account. This was unsuccessful and blocked by our default egress block rule:

```
@40 block out on dc1 proto tcp from any to any port = imap
[ Evaluations: 4           Packets: 3           Bytes: 192   ]

@41 block out on dc1 proto tcp from any to any port = pop3
[ Evaluations: 4           Packets: 3           Bytes: 192   ]
```

4. Use nmap Fin or Null scan on the other auditor. This was blocked by the default 'block out all' rule.

Part 4: Design Under Fire

In this section, I've chosen to attack the network design by Adrian Hobbs, which is shown below. His practical can be found at:

http://www.giac.org/practical/Adria_Hobbs_GCFW.zip

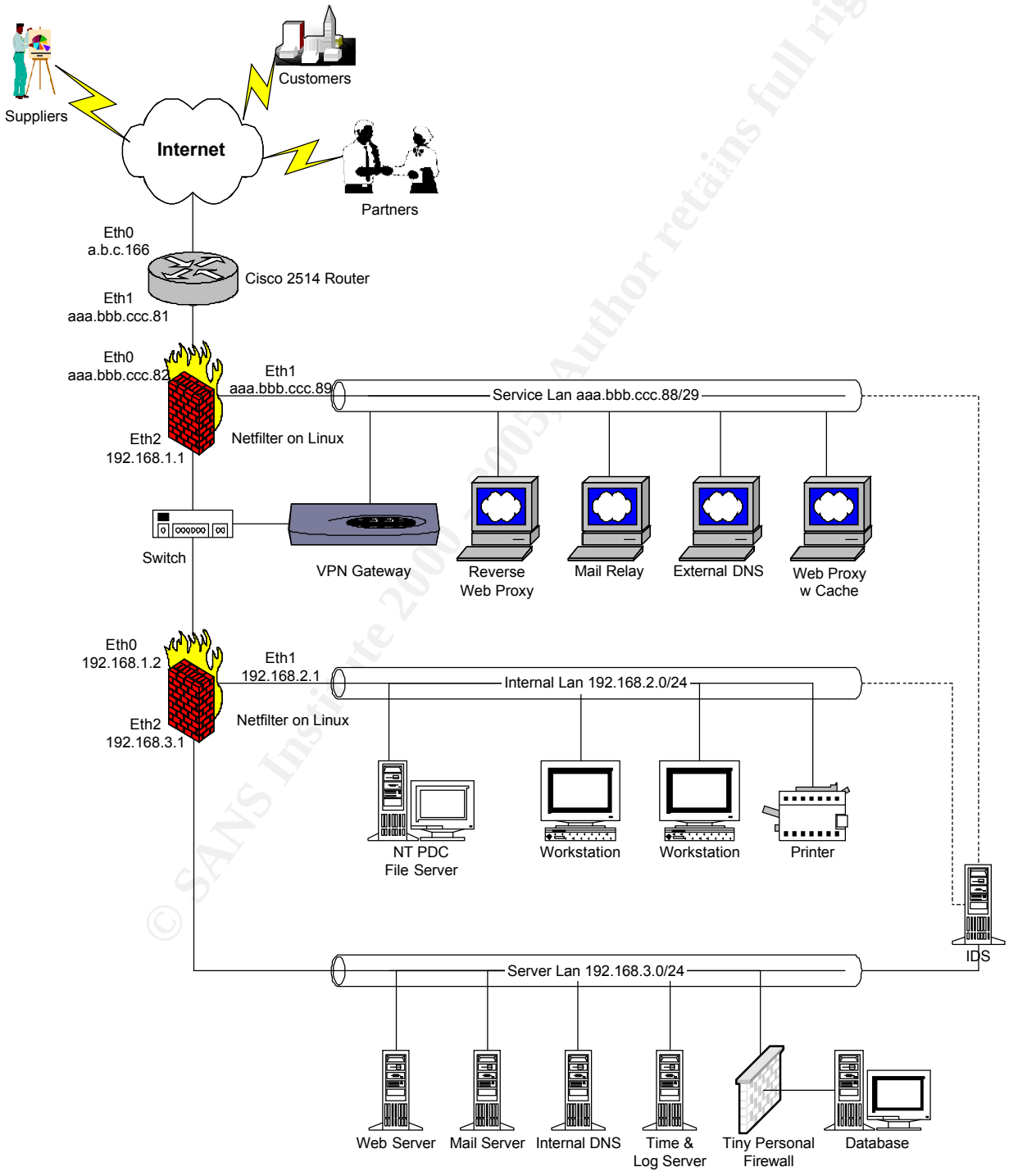

Figure 3: Design Under Fire - Adrian Hobbs

Attacking the Firewall

One option is to saturate their network connection with packets somehow and fill up the state table. Stateful firewalls keep track of the state of each TCP connection, which can be in a number of different states corresponding to the different phases of a TCP connection (NEW, ESTABLISHED, FIN_WAIT 1 & 2, and TIME_WAIT). Entries in the Established state table are kept around the longest (5 Days)[24], while the other states are only kept around for a few minutes.

Netfilter will start to drop packets when the maximum is reached. This is depends on the amount of system memory[25], and for a machine with only 256MB of RAM (like Adrian's), would be 16384 connections. If we could generate that many ESTABLISHED connections, it could cause the firewall to start dropping legitimate connections and adversely affect network performance.

Experimentation with tools like Naptha[26] and Netkill[27] were successful in creating thousands of state table entries, but these entries didn't stay ESTABLISHED, rather, they went into TIME_WAIT, and quickly expired from the state table

I couldn't find any tools that seem to do the job. However, it's possible that with 50 compromised cable modems at my disposal, it could do the job. I had upwards of 6000 TIME_WAIT entries created at a given time, so if you were to multiply that by 50, it could easily fill the conntrack table. However, the effect would disappear within minutes once the attacker shut down. This sort of attack would probably be mitigated by rate limiting on the router level.

As an attack against the firewall, this is would probably be more of an annoyance than anything. It would, however, make an effective smokescreen for a more subtle attack. If carried on long enough, it would probably make a good DoS against the log server as the disk would soon begin to fill up with firewall, router, and snort IDS logs, which would be annoying in itself.

---

[24] "IPTables: Connection Tracking". URL:
http://www.sns.ias.edu/~jns/security/iptables/iptables_conntrack.html

[25] "netfilter/iptables FAQ: Problems at runtime". URL:
http://www.netfilter.org/documentation/FAQ/netfilter-faq-3.html#ss3.16

[26] "The Naptha DoS Vulnerabilities". URL:
http://razor.bindview.com/publish/advisories/adv_NAPTHA.html

[27] "Netkill: Generic remote DoS attack tool" URL:
http://www.securiteam.com/tools/5QR0B000AU.html

The only other weakness in Netfilter up to version 1.2.6 is the NAT/ICMP code information leak, described in the Security Bulleting released May 8, 2002:

http://www.netfilter.org/security/2002-04-02-icmp-dnat.html

It's the most recent and only applicable flaw in NetFilter, but it is more of a reconnaissance tool than an attack on the firewall. If Adrian's design wasn't patched against this vulnerability, it would allow an intruder to discover the real IP addresses of the Destination NAT mapped hosts on eth1. (The internal Mail, Web, and Time servers can be detected in this fashion). As stated in the bulletin, a modified version of nmap exists that makes it very easy to map networks with this vulnerability. The output of that on this design might look like this:

```
# ./nmap -sS -P0 aaa.bbb.ccc.93 -p 80,443 -t 9

   Starting nmap V. 2.54BETA32 ( www.insecure.org/nmap/ )
   Interesting ports on xxx.xxx.xxx.xxx:
   Port       State       Service
   80/tcp    UNfiltered  unknown
     DNAT to 192.168.3.11:80
```

This is shows the reserved IP address of Adrian's internal webserver. Now we know what the internal address space looks like which may help in future attacks.

Distributed Denial of Service Attack

A distributed denial of service (DDoS) attack requires the control of a number of 'Zombie' hosts; these zombie systems are controlled by master systems in the manner shown in Figure 4:

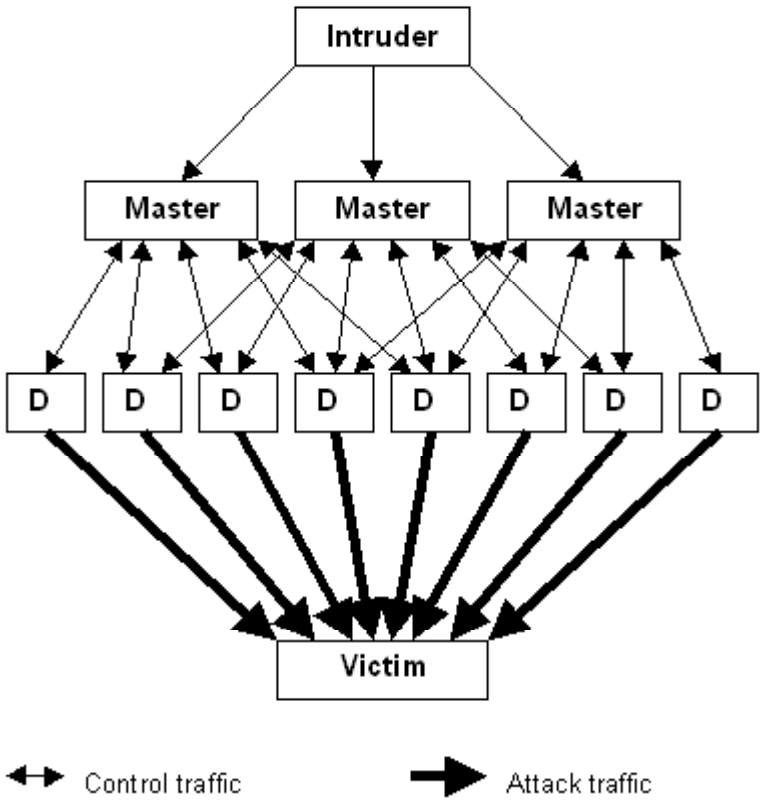

Figure 4: CERT Distributed Systems Attack[28]

The control channels may be encrypted, and possibly one-way UDP connections that allow the intruder to broadcast commands to the zombie systems to perform specific actions. Most DDoS attacks are like this, including the Trinoo, Tribe Flood Network (TFN), stacheldraht, TFN2K, shaft, mstream, all use this command structure.

In our attack, the TFN2K[29] daemon has been installed on our weakly secured cable modem subscribers, with a few systems acting as masters. Cable modem users were targeted because of the availability of broadband

[28] CERT. "Results of the Distributed-Systems Intruder Tools Workshop".
URL: http://www.cert.org/reports/dsit_workshop-final.html

[29] http://packetstormsecurity.nl/distributed/TFN2k_Analysis-1.3.txt

connections, and because most cable Internet providers recommend disabling personal firewalls when their users have connection problems. The network address space of several cable providers was scanned looking for exploitable remote vulnerabilities. Since TFN2K is known to work on Unix, Solaris, and Windows NT platforms, this gave us a wide variety of exploits to explore.

Once this network was setup, it was trivial to issue have the master control systems initiate the attack. This brings us to the other reason for choosing TFN2K – the variety of attacks we can use. TFN2K can be instructed to launch TCP SYN, UDP, ICMP/Ping, and Broadcast ICMP (Smurf) attacks. Or randomly alternate between all of them. This type of attack will certainly cause chaos and confusion at GIAC Enterprises!

To launch a mixed attack[30] we would issue the following command from the master(s):

```
./tfn -h zombien.cableco.net -c 8 -i aaa.bbb.ccc.93@
```

This could be automated on a Unix system by using a simple shell script and added to run via at:

```
for z in `cat mylistofzombies.txt`
 do ./tfn -h $z -c 8 -i aaa.bbb.ccc.93@
done
```

This attack would probably cause a serious performance impact on Adrian's primary web server and firewall given the sheer variety and number of incoming packets. It would also generate a lot of log entries from the firewall and IDS systems.

The best way to prevent this sort of attack is education and awareness.

1. System operators, including home users, need to secure their systems from remote compromise. The Sans/ FBI Top List is a good place to start. (http://www.sans.org/top20/)
2. Network operators, and ISP's especially should be required to implement egress filtering on their networks, to prevent spoofed traffic from getting past their routers.
3. Disable ICMP directed broadcasts.
4. Sharing log information through organizations like DShield, (dshield.org), so that daemon systems can be found before they are used in an attack.

The "Results of the Distributed-Systems Intruder Tools Workshop", published by CERT, is an source excellent of additional resources for Managers, System Administrators, ISP's and Network operators, and Incident Response teams:

http://www.cert.org/reports/dsit_workshop-final.html

[30] "A TFN2K Analysis" URL: http://personal.ie.cuhk.edu.hk/~shlam/ssem/is/ddos.txt

Attack an internal host

After looking over Adrian's design and access requirements, one thing caught my eye; his design allows direct access to his database server from his business partner's network. Even though there are three firewalls in between, they all have the same policy: pass all traffic to the database server from our business partners. The weakness here, of course, is now we must rely on the perimeter security of our business partners to maintain control of our data. Personally, I'd be happier with a middleware application layer of some sort in between.

In this attack we will assume:

1. The business partner network security is weaker than that of GIAC enterprises.
2. Only the MSSql database server on the partner side is allowed to access the GIAC database server.
3. Neither database has the latest patch for the vulnerability in this bulletin: http://www.microsoft.com/technet/security/bulletin/MS02-056.asp
4. Both are running SQL server 2000.
5. Dave Aitel's proof of exploit code[31] actually installs a remote backdoor, such as a simple netcat UDP connection that makes a connection to another netcat listening on port 53. This will bypass NAT, as the connection is made from the inside out, and uses the DNS port.

So we have a vulnerability that can get us access, but how would we get this into Adrian's network in the first place?

First, I would have to get inside the business partner's perimeter security. In this case, we'll say that it was through a minimally secured wireless LAN. After getting into the network, I was able to install a network sniffer by installing a root kit on a shared network drive. It was simple to force the windows 2000 system to reboot using the SMBDie DOS tool. This automatically installed the root kit on startup. Once I had my foot in the door, I watched the network using a packet sniffer. I found a lot of traffic flowing between both the local database server, and the database server at GIAC Enterprises.

I wanted to find out more, so patiently I waited for my opportunity. Then, when the time was right, I downloaded this new exploit from the immunitysec site, and prepared to break into both database servers. I waited for the dead of night, then I had my 'associate' launch a series of TFN2K attacks against the main GIAC site to keep their people busy. Between DDoS storms, I fired up my modified Nessus 'sploit and now I own two MSSql database servers. Now I can sort through their files whenever I feel like it. Maybe I can find some credit card

[31] Dave Aitel. Nessus test to Exploit MSSql You had me at Hello buffer overflow. URL: http://www.immunitysec.com/vulnerabilities/mssql_hello_overflow.nasl

numbers or system accounts I can trade for exploits with other 'l33t haX0r's…

Conclusion

I chose Adrian's design because the database server was too exposed, not because it was easy to hack his firewall. Despite the three firewalls, access from the partner site was pretty much unrestricted. Even if my scenario was a bit contrived, it is possible. If not by a bored hacker, then insiders on the partner network could do it.

The next SQLSnake worm could be a blended threat, and like Nimda, get behind firewalls through email and spread through the internal network from SQL server to SQL server, right over that encrypted IPSec connection to the server on the other side. I don't think that database servers are 'hard' enough to be deployed that close to the edge of the network, and until they are middleware applications containing rigorous data integrity testing should be put in place between your network and other people's networks.

References

"SWATCH: The Simple WATCHer"
URL: http://www.oit.ucsb.edu/~eta/swatch/

"Big Brother System and Network Monitor": URL: http://bb4.com/

"Cisco 3600 Series": URL:
http://www.cisco.com/univercd/cc/td/doc/pcat/3600.htm

"OpenBSD": URL: http://www.openbsd.org/faq/faq6.html#PF

"NetFilter/IPTables": URL: http://netfilter.samba.org/

"Snort: The Open Source Network Intrusion Detection System"
URL: http://www.snort.org/

"AIDE – Advanced Intrusion Detection Environment"
URL: http://www.cs.tut.fi/~rammer/aide.html

"Cisco Catalyst 2950 Series Switches" URL:
http://www.cisco.com/en/US/products/hw/switches/ps628/index.html

"vanja.com - Tools" URL: http://www.vanja.com/tools/

"Securing Debian Manual" 2002-09-18.

URL: http://www.debian.org/doc/manuals/securing-debian-howto/index.en.html

"8 Essential Steps to Building a Human Firewall"
URL: http://www.humanfirewall.com/

SANS 2.3 - Firewalls 102: Perimeter Protection and Defense in-Depth.

"Configuring Secure Shell on Cisco IOS Routers" 2002-09-15.
URL: http://www.cisco.com/warp/public/707/ssh.shtml

Keeney, Frank. "Screening Router Access List" 1998-12-30.
URL: http://pasadena.net/cisco/secure.html

Coene, Wouter. "The Open BSD Packet Filter HOWTO", section 3 Filtering Bridges. 2002-04-05.
URL: http://www.inebriated.demon.nl/pf-howto/html/node4.html

The Honeynet Project. "Know Your Enemy: Statistics" 2001-07-22. URL: http://project.honeynet.org/papers/stats/

ORiNOCO Wireless Networks. "AP-2000 Access Point" URL: http://www.orinocowireless.com/template.html?section=m58&page=3040&envelope=94

"SSH Support – FAQ"
http://www.ssh.com/support/faq/sentinel/qa_2_898.html

"OpenBSD pfctl man page"
http://www.openbsd.org/cgi-bin/man.cgi?query=pfctl

"Nmap: Network Mapper" http://www.insecure.org/nmap/

Armstrong, Tom. "Netcat – The TCP/IP Swiss Army Knife". 2001-02-15.
URL: http://rr.sans.org/audit/netcat.php

"TCPDump public repository" 2002-05-13.
URL: http://www.tcpdump.org/

Hobbs, Adrian. "GFCW Practical Assignment". URL: http://www.giac.org/practical/Adrian_Hobbs_GCFW.zip

"IPTables: Connection Tracking". URL:
http://www.sns.ias.edu/~jns/security/iptables/iptables_conntrack.html

"netfilter/iptables FAQ: Problems at runtime". URL:
http://www.netfilter.org/documentation/FAQ/netfilter-faq-3.html#ss3.16

"The Naptha DoS Vulnerabilities". URL:
http://razor.bindview.com/publish/advisories/adv_NAPTHA.html

"Netkill: Generic remote DoS attack tool" URL:
http://www.securiteam.com/tools/5QR0B000AU.html

"NAT/ICMP code information leak" 2002-05-08. URL:
http://www.netfilter.org/security/2002-04-02-icmp-dnat.html

CERT. "Results of the Distributed-Systems Intruder Tools Workshop".
URL: http://www.cert.org/reports/dsit_workshop-final.html

Barlow, Jason and Thrower, Woody. "TFN2K - An Analysis" Version 1.3.
2000-02-10. URL:
http://packetstormsecurity.nl/distributed/TFN2k_Analysis-1.3.txt

The SANS Institute. "SANS / FBI Top Twenty list". Version 3.2.
2002-08-17. URL: http://www.sans.org/top20/

"A TFN2K Analysis" URL:
http://personal.ie.cuhk.edu.hk/~shlam/ssem/is/ddos.txt

Aitel, Dave. "You had me at Hello (Nessus test script)" URL:
http://www.immunitysec.com/vulnerabilities/mssql_hello_overflow.nasl

Other References

Hoang Q. Tran. "OpenBSD firewall using pf". 2002-07-17. URL:
http://www.unixcircle.com/features/openpfnat.php

Northcutt, Stephen and Novak, Judy. "Network Intrusion Detection,
Second Edition" Indianapolis: New Riders, 2000.

Preston, W Curtis. "Unix Backup & Recover" Sebastopol: O'Reilly &
Associates, 1999.

O'Reilly & Associates Staff. “The Networking CD Bookshelf” Sebastopol: O’Reilly & Associates, 1998.

Frisch, Æleen. “Essential Systems Administration, 2nd Edition” Sebastopol: O’Reilly & Associates, 1995.