Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

GIAC Certified Firewall Analyst (GCFW) Practical Assignment

© SANS Institute 2003, Author retains full rights.

By: Penny Hermann-Seton
GIAC GCFW Practical Assignment v1.9

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Table of Contents

List of Figures

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Abstract

This paper will present a network security architecture for GIAC Enterprises, an e-business which sells online fortune cookie sayings. The business operations will be explained for dealing with customers, suppliers, business partners, internal GIAC employees, and a mobile work force. A Security Policy for the Border Router, Primary Firewall, and Virtual Private Network (VPN) will be provided. A Tutorial of how to implement the Security Policy for the chosen Firewall will be given. Details for planning, conducting and evaluating an audit of the GIAC's Firewall Security Policy as implemented by the chosen Firewall will be provided. Next, three different types of attacks against a GCFW student's network design will be explained. This will include an attack against the chosen Firewall, a Denial of Service attack using the TFN2K DDoS attack tool, and an attempt to compromise the Web Server through the perimeter system.

Assignment 1 – Security Architecture

GIAC Enterprises is an e-business, which sells online fortune cookie sayings.

Business operations

Customers:

- Need to be able to access GIAC Enterprise's web-based ordering for purchasing bulk online fortunes. Customers will need to access our Web Server. The protocols used will be http and https (SSL is needed for encryption of private information, such as credit cards, etc on the ordering forms.) After the customer has specified the type of online fortunes he/she is interested in purchasing, the web-server is responsible for accessing GIAC's Internal Database and retrieving the requested information for downloading by the customer.
- Need to be able to send e-mail to GIAC Enterprises. They will need access to our Mail Server (using SMTP).
- Need to be able to access our DNS Server (using DNS) to find the location of our website.
- Customers do not need to have access to any other internal hosts.

Suppliers:

- Need to send their fortune cookie sayings in a secure manner to prevent others from intercepting this valuable information. GIAC will provide secure access via a VPN, where the fortune cookie sayings will be encrypted. The suppliers will use FTP (namely the put command) to transfer the files with their fortune cookie sayings to GIAC Enterprises over the VPN.
- After the supplier has used FTP (via the VPN) to send their file to GIAC, they will access the GIAC website to fill out a form detailing what has been supplied (file name, etc.). This form notifies GIAC that they have just supplied

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

a new file. GIAC will then retrieve and process the file for placement into GIAC's Internal Database. The Suppliers will access GIAC's website over the regular Internet (using HTTP and HTTPS); no VPN is needed.
- Need to be able to send e-mail to GIAC Enterprises for basic communication needs.
- Need to be able to access our DNS and Mail Server as detailed in the above section.
- Suppliers need to have access to GIAC's FTP Server over the VPN.
- Suppliers do not need to have access to any other internal hosts.

Partners:
- Need to access GIAC Enterprise's website to place an order for the type of sayings that they are interested in reselling. GIAC will process this information and retrieve the specified fortune cookie sayings from the GIAC Internal Database. The file(s) will be placed on the FTP server in the pre-designated location for that particular business partner. GIAC will then send an e-mail notifying the business partner the file/files is/are ready to be retrieved. The Partners will access GIAC's website over the regular Internet (using HTTP and HTTPS); no VPN is needed.
- Need to retrieve file(s) containing fortune cookie sayings in a secure manner to prevent others from intercepting this vital information. GIAC will provide secure access via a VPN, where the fortune cookie sayings will be encrypted. The business partners will use FTP (namely the get command) to retrieve the requested file(s) over the VPN. Our business partners will then translate these fortune cookie sayings and resell them.
- Need to be able to send e-mail to GIAC Enterprises for basic communication needs.
- Need to be able to access our DNS and Mail Server as detailed in the above section.
- Partners need to have access to GIAC's FTP Server over the VPN.
- Partners do not need to have access to any other internal hosts.

GIAC Enterprises employees located on GIAC Enterprise's internal network:

- Need to be able to receive and send out e-mail (SMTP) and browse the Internet (using HTTP, HTTPS, FTP, DNS)

GIAC Enterprises mobile sales force and teleworkers:
- Need a secure channel to be able to access the internal network. From the internal network, they can access their e-mail and other internal hosts. The secure channel will be provided by a VPN.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Network Security Architecture for GIAC Enterprises

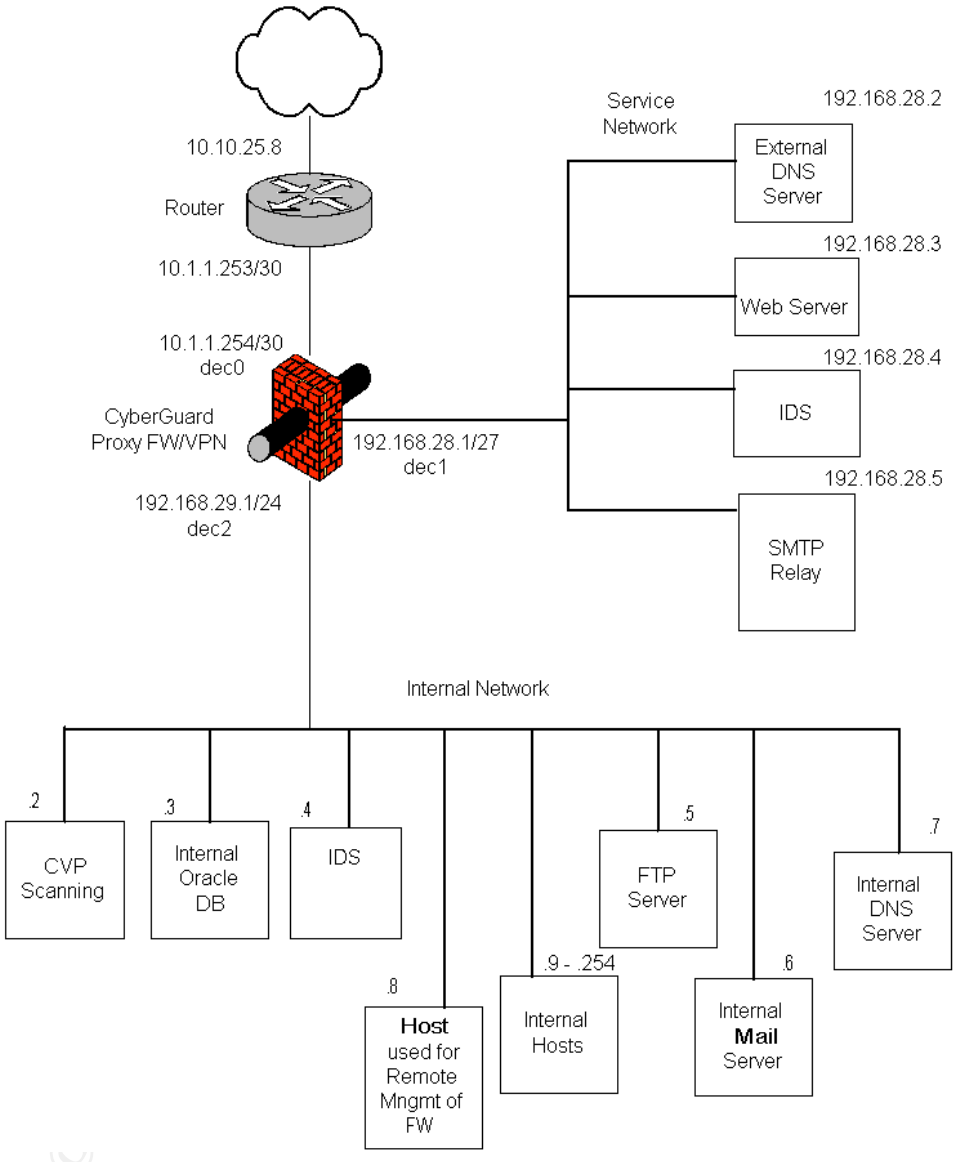


Figure 1 GIAC Enterprise's Network

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Filtering Router:

GIAC Enterprises will be using a CISCO 2621 router, with CISCO IOS version 12.2, as its filtering router. This router is placed between the Internet and our primary firewall. This router provides the first line of defense. We will be using the packet filtering capabilities of the router to filter out obvious unwanted packets from entering our network; packets from non-routable addresses, the loopback address, unallocated addresses, etc. By having the router perform this filtering, it will also lesson the load on our proxy firewall. We will not be using any stateful ACLs since we do not want to bog down the router. We will leave this type of processing to our high-powered proxy FW. Connection to the router will be via a console cable.

Firewall/VPN:

GIAC Enterprise will be using the CyberGuard Firewall Release 5.0 (model KS1500) as the proxy Firewall/VPN component. The firewall is placed after the filtering router, but before our service network and our Internal Network, please refer to Figure 1. The Firewall, placed in this location, protects our internal network from both the Internet and our service network. The firewall also protects our service network from the Internet. GIAC will be utilizing the VPN component of the CyberGuard Firewall. The Firewall will also be utilized to protect our Internal network from our business partners and suppliers who will be utilizing the VPN. GIAC is using the high-powered proxy FW to provide high security. The proxy firewall will provide the capability to inspect the data being sent via the different protocols (HTTP, FTP, etc.). We will also be using the NAT capabilities of the FW to protect our internal private addresses. The VPN is used to provide confidentiality of sensitive information exchanged between GIAC and our business partners and suppliers. The VPN will also be used to extend our corporate network to our mobile sales force and teleworkers.

We will utilize the following proxies on the CyberGuard FW:

- HTTP proxy for outbound HTTP connections. In our next round of funding, we will be investigating the use of the HTTP Proxy for inbound connections, also referred to as “a reverse proxy”. This will allow us to filter on POST requests in order to prevent attacks against our Web Server.
- SSL Protocol Proxy for outbound SSL protocol connections.
- FTP Proxy for both inbound and outbound FTP connections. The FTP Proxy will be used to authenticate our business suppliers and partners and restrict their FTP requests. We will restrict business suppliers to using “put” and business partners to using “get”.
- SQL*Net Proxy to provide a secure connection between the Web Server and the Internal Oracle Database server for retrieving online fortunes for our customers.

Due to GIAC’s high volume of e-mails, we will not be using the SMTP Proxy on the FW. SMTP messages will be relayed to the SMTP relay server on the

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Service Network. We will be utilizing CyberGuard's Content Vector Protocol (CVP) to perform content scanning for the FTP and HTTP proxies. The proxy will transfer the file to be scanned to the Internal CVP scanning server for processing. This functionality will scan files received from our Suppliers as well as files retrieved by our internal users from the Internet for viruses. Also, the CyberGuard Firewall allows for alerting of suspicious events including: land attacks, ping of death attacks, TCP SYN flood attacks, network port scan attempts and many other security related events that we would like to be alerted on.

Service Network:
The service network consists of the following components:

- External DNS Server – for access by external users and contains public information.
- Web Server –hosts the GIAC Enterprises website
- Intrusion Detection Server - Network based IDS system is used for defense-in-depth. In case there is a compromise of the Service Network, this IDS system is watching packets for known attack patterns. We are going to save money here and use Snort, version 1.8.7 (it's free).
- SMTP Relay server – to act as a relay between the Internet and the Internal mail server.

The Internal Network:
The internal network consists of the following components:

- Oracle Database Server – contains the GIAC Database of Fortune Cookie Sayings. We will be relying on the security mechanisms of the Oracle Database server for authentication and the Firewall (SQL*NET proxy) for protection of this vital information.
- Intrusion Detection Server - IDS system for defense-in-depth. In case there is a compromise of the Internal Network, this IDS system could detect it. (Snort, version 1.8.7)
- Internal mail Server
- Internal DNS server – for access by internal users and contains private information. This information is stored on the internal network and protected by the firewall from outside security probes.
- CVP Scanning server – performs content scanning for the FTP and HTTP Proxies.
- FTP Server - server to hold files for our business partners/suppliers
- Various other Internal hosts

Remote Users:

- We will require that the mobile sales force and teleworkers have a Personal Firewall installed on their computers to avoid hacker tunnelling since they will be running a VPN client. Recommend Zone Labs Integrity Desktop, version 2.0.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

- We will require our business partners/suppliers to have firewalls installed on their networks to protect our VPN channel. The particular brand is left up to them.
- There will be no modems in the GIAC network.

The current design of GIAC is both technically and financially feasible.
We have placed a large portion of the perimeter security budget in our Firewall. We need a highly secure FW and decided to use a proxy firewall with it's many available proxies (HTTP, SSL, FTP, and SQL*NET). Also, quite a few of our employees are already experienced with the CyberGuard Firewall. We do not need to spend additional money retraining these personnel on a different Firewall product. We are also saving money by using the free Snort IDS.

We will be defining a subnet between the internal interface of the router and the external Interface of the Firewall. Namely, 10.1.1.253/30 and 10.1.1.254/30, as seen in figure 1.

We are using the Public IP addresses 192.168.28.0/27 for the Service Network and Private IP addresses 192.168.29.0/24 for the Internal Network. Given the defined Service Network subnet, the Service Network subnet address is 192.168.28.0, the usable host addresses are 192.168.28.1-30, and the broadcast address is 192.168.28.31. Given the defined Internal Network subnet, the Internal Network subnet address is 192.168.29.0, the usable addresses are 192.168.29.1 -254, and the broadcast address is 192.168.29.255. This should allow for adequate IP addresses on the Internal and Service networks, which should be sufficient for the current company IP needs and allow for extra capacity for future growth.

Note: All IP addresses defined for GIAC Enterprises, whether specified as Public or Private, will be from the non-routable address space.

Assignment 2 – Security Policy and Tutorial

Security Policy for Border Router

We will use Extended ACL's for each of the interfaces.

Interface from the Internet to the Router

Rule: deny ip host 10.10.25.8 any log
Rule: deny ip 10.1.1.253 0.0.0.3 any log
Rule: deny ip 192.168.28.1 0.0.0.31 any log
Purpose: Filter out traffic with a source IP address from our internal network
Why: Since there should be no traffic originating from GIAC's internal network coming through the Internet interface, it would be wise to drop these packets. If we do receive packets with these source IP addresses, it is due to some error or

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

mal intent. Note: filtering out GIAC's private internal address space (192.168.29.0 0.0.0.255) is covered in the rules to filter out all private addresses.

Rule: deny ip 127.0.0.0 0.255.255.255 any
Purpose: Filter out loopback address
Why: Traffic originating from the loopback address should never be on any interface - it is processed internally. This can be used as a Denial of Service (DoS) attack.

Rule: deny ip 224.0.0.0 15.255.255.255 any
Purpose: Filter out multicast traffic from Multicast Addresses 224.0.0.0 to 239.255.255.255
Why: Filter out unwanted IP packets.

Rule: deny ip 10.0.0.0 0.255.255.255 any
Rule: deny ip 172.16.0.0 0.15.255.255 any (172.16.0.0 - 172.31.255.255)
Rule: deny ip 192.168.0.0 0.0.255.255 any
Purpose: Filter out traffic with a source IP address that is considered to be a private address space (As defined in RFC 1918 - Private Network Allocations).
Why: There should be no traffic originating from a private address space. If we receive an IP packet with this configuration, it is due to some error or mal intent. We also won't be able to successfully send a response to this address. This type of traffic may be used in a DoS attack.

Rule: deny ip 0.0.0.0 0.255.255.255 any
Rule: deny ip 1.0.0.0 0.255.255.255 any
Rule: deny ip 2.0.0.0 0.255.255.255 any
Rule: deny ip 5.0.0.0 0.255.255.255 any
Rule: deny ip 7.0.0.0 0.255.255.255 any
Rule: deny ip 23.0.0.0 0.255.255.255 any
Rule: deny ip 27.0.0.0 0.255.255.255 any
Rule: deny ip 31.0.0.0 0.255.255.255 any
... etc.
Purpose: Filter out traffic with a source IP address that is not allocated.
WHY: We shouldn't be receiving traffic from these addresses. If we are, it most likely is a crafted packet with a spoofed IP address.

Rule: deny ip host 217.160.110.151 any log
Rule: deny ip host 200.30.203.134 any log
Rule: deny ip host 200.30.203.160 any log
Rule: deny ip host 66.250.32.211 any log
Rule: deny ip host 205.251.79.36 any log
Rule: deny ip host 61.135.148.70 any log
Rule: deny ip host 195.18.123.148 any log
Rule: deny ip host 195.191.15.238 any log
Rule: deny ip host 207.112.194.18 any log

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Rule: deny ip host 213.33.158.35 any log
Purpose: Block source IP address from www.incidents.org list of the top 10 source IP addresses that are launching attacks.
Why: Since they are known to launch attacks, the probability is high that they'll launch an attack on our network.

Rule: deny TCP any any range 135 139 log
Rule: deny UDP any any range 135 139 log
Rule: deny TCP any any 445 log
Rule: deny UDP any any 445 log
Rule: deny TCP any any 23 log
Rule: deny TCP any any range 512 514 log
Rule: deny TCP any any 111 log
Rule: deny UDP any any 111 log
Rule: deny TCP any any 2049 log
Rule: deny UDP any any 2049 log
Rule: deny TCP any any range 6000 6255 log
Rule: deny UDP any any range 6000 6255 log
Rule: deny TCP any any 69 log
Rule: deny UDP any any 69 log
Rule: deny TCP any any range 161 162 log
Rule: deny UDP any any range 161 162 log
Rule: deny UDP any any 514 log
Purpose: Block Critical Services from entering our network. We have both Windows and Unix computers on our network, so we will be blocking the following critical services:
Windows environments: TCP & UDP 135-139 and 445 (NetBIOS)
UNIX environments: TCP 23 (telnet), TCP 512-514 (rexec, rlogin, and rshell), TCP & UDP 111 (SUN Remote Procedure Call), TCP & UDP 2049 (NFS), TCP & UDP 6000-6255 (X Windows), TCP & UDP 69 (TFTP), TCP & UDP 161,162 (SNMP), UDP 514 (syslogd)
Why: These ports will also be blocked at the firewall; but just as a precaution, we are also blocking these packets at the router since we really don't want these connections to get through.

Rule: permit any ip 10.1.1.254 0.0.0.0
Rule: permit any ip 192.168.28.0 0.0.0.31
Purpose: Only permit traffic to our Public address space.
Why: Allows external users access to GIAC enterprises.

Last Rule: deny any any log
Purpose: Deny and log everything else.
Why: Deny traffic not destined for our Public address space. We don't want to waste any more time on these packets or allow these type of packets to create any kind of malicious effects.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Interface from the FW/VPN to the Router

Rule: deny ip any 205.188.7.0 0.0.1.255 log-input
Purpose: Block "firewall unfriendly" services AOL-Instant Messenger
Why: The AOL-IM service will try to find an open port in our firewall.

Rule: deny ip any host 63.251.224.169 log-input
Purpose: Block access to gotomypc.com
Why: gotomypc.com is designed to go around firewall rules.

Rule: deny TCP any any range 135 139 log-input
Rule: deny UDP any any range 135 139 log-input
Rule: deny TCP any any 445 log-input
Rule: deny UDP any any 445 log-input
Rule: deny TCP any any 23 log-input
Rule: deny TCP any any range 512 514 log-input
Rule: deny TCP any any 111 log-input
Rule: deny UDP any any 111 log-input
Rule: deny TCP any any 2049 log
Rule: deny UDP any any 2049 log
Rule: deny TCP any any range 6000 6255 log
Rule: deny UDP any any range 6000 6255 log
Rule: deny TCP any any 69 log-input
Rule: deny UDP any any 69 log-input
Rule: deny TCP any any range 161 162 log-input
Rule: deny UDP any any range 161 162 log-input
Rule: deny UDP any any 514 log-input
Purpose: Block Critical Services from leaving our network. We have both Windows and Unix computers on our network, so we will be blocking the following critical services:
Windows environments: TCP & UDP 135-139 and 445 (NetBIOS)
UNIX Environments: TCP 23 (telnet), TCP 512-514 (rexec, rlogin, and rshell), TCP & UDP 111 (SUN Remote Procedure Call), TCP & UDP 2049 (NFS) TCP & UDP 6000-6255 (X Windows), TCP & UDP 69 (TFTP), TCP & UDP 161,162 (SNMP), UDP 514 (syslogd)
Why: Prevents outbound connections on these ports in case one of our internal hosts have been compromised.

Rule: permit ip 192.168.28.1 0.0.0.31 any (Public Service Network)
Rule: permit ip 10.1.1.254 0.0.0.3 any (subnet of Firewall)
Purpose: Since we are going to be a good Internet neighbor, only permit traffic with a source IP address from the GIAC Enterprises.
Why: Allow IP connections from GIAC Enterprises.

Last Rule: deny ip any any log-input
Purpose: Deny and log everything else.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Why: This rule along with the above rule prevents any spoofed IP packets from originating inside our network. Also, this can alert us to a security problem that needs to be investigated and addressed. These two rules also prevent our FW (our NAT device) from leaking out GIAC's internal address space, which would be very useful information to an attacker.

The ordering of the ACL list is extremely important. Processing of the ACL list starts from the top. Once a match is made, for either permit or deny, the specified action is taken and processing is stopped.

Additional traffic controls for the router

Rule: access-list 10 deny any
line vty 0 4
access-class 10
login
Purpose: Set up access list to deny remote logins to the router.
Why: Our security policy states that we are only allowing access via the console.

Rule: no snmp
Purpose: disable SNMP
Why: We aren't using SNMP in our current configuration.

Rule: no cdp run
Purpose: disable Cisco Discovery Protocol (CDP)
Why: We don't need CDP.

Rule: no ip source-route
Purpose: Filter out all packets that have the source routing options set.
Why: Prevents attackers from entering through a back door and can also prevent spoofing.

Rule: no service tcp-small-servers
Rule: no service udp-small-servers
Purpose: Disable echo, discard, chargen, and daytime services.
Why: Disable unneeded services that may turn into a potential vulnerability.

Rule: no service finger
Purpose: Disable finger services
Why: Prevent revealing information that could be used to attack our network.

Rule: no ip http
Rule: no ip bootp
Purpose: Disable unneeded server services.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Why: To prevent a potential attack vector into our network.

Rule: no ip redirects
Purpose: Filter out all ICMP redirect messages.
Why: Since we have only a single connection to the Internet, we need to protect ourselves from a potential DoS attack that might use the ICMP redirect message

Rule: no ip direct-broadcast
Purpose: Prevent IP broadcast packets from entering the router.
Why: Prevents broadcasts from being used to perform a Denial of Service attack on our network.

Rule: no ip unreachable
Purpose: Prevents router from returning information about our network based on ICMP error messages.
Why: This network information could be used in launching an attack against our network.

Security Policy for Primary Firewall

Ordering of firewall rules is important; the first rule that matches is processed. If no rule matches, the packet is dropped. Rules are processed from top to bottom. Also, anything that is not specifically allowed is prohibited.

For GIAC Enterprises, we will authorize firewall administration by providing remote management of the firewall from a single IP address from the internal network.

The syntax from the "Firewalls, Perimeter Protection and VPNs" class will be used to specify the firewall rules, denoted by "**Rule:**" (where, -i = interface, -p = protocol (TCP, UDP, etc.), -s =source IP, -d =destination IP). Following each rule of this type will be a rule defined using the syntax for the CyberGuard Firewall, denoted by "**CG Rule:**" which is explained in the tutorial at the end of this section. As seen in Figure 1, dec0 denotes the interface to the router (Internet), dec1 denotes the interface to the Service Network, and dec2 denotes the interface to the Internal Network. For each of the firewall rules, an explanation of the purpose of the rule and why the rule is important will be given, followed by the actual firewall rule.

Following are the Firewall rules for the GIAC Primary Firewall:

Place HTTP access rules at the top for efficiency:

Purpose and Why: Allow HTTP/HTTPS to our Web server. This allows access to the GIAC website.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Rule: permit –i dec0 –p tcp –s 0/0 -d 192.168.28.3 - -dport 80 log
CG Rule: permit http/tcp dec0 192.168.28.3

Rule: permit –i dec0–p tcp –s 0/0 -d 192.168.28.3 - -dport 443 log
CG Rule: permit https/tcp dec0 192.168.28.3

#Outbound to Internet
Purpose and Why: Allow internal hosts to use HTTP/HTTPS/FTP/DNS to Internet. This allows GIAC employees' access to the Internet. We will utilize the HTTP, SSL, and FTP Proxies to protect our internal network.
Rule: permit –i dec2 –p tcp –s 192.168.29.0/24 –d 0/0 - -dport 80 – use HTTP proxy
CG Rule: proxy http/tcp dec2 EVERYONE

Rule: permit –i dec2 –p tcp –s 192.168.29.0/24 –d 0/0 - -dport 443 – use SSL proxy
CG Rule: proxy https/tcp dec2 EVERYONE

Rule: permit –i dec2 –p tcp –s 192.168.29.0/24 –d 0/0 - -dport ftp – use FTP proxy
CG Rule: proxy ftp/tcp dec2 EVERYONE

Rule: permit –i dec2 –p tcp –s 192.168.29.0/24 –d 0/0 - -dport 53
CG Rule: permit domain/tcp dec2 EVERYONE

Rule: permit –i dec2 –p udp –s 192.168.29.0/24 –d 0/0 - -dport 53
CG Rule: permit domain/udp dec2 EVERYONE - make sure "enable replies" is set

Purpose and Why: Allow an internal host access to create a ssh session with the firewall's internal interface to allow for remote management of the firewall
Rule: permit –i dec2 -p tcp –s 192.168.29.8 –d 192.168.29.1 - -dport 22 log
CG Rule: permit 22/tcp 192.168.29.8 192.168.29.1

Purpose and Why: Allow internal hosts to send Echo-Request packets to the firewall for troubleshooting.
Rule: permit –i dec2 –p icmp –s 192.168.29.0/24 -d 192.168.29.1 echo-request
CG Rule: permit echo/icmp dec2 192.168.29.1 - make sure "enable replies" is set

#Inbound Access rules
Purpose and Why: Allow DNS to the External DNS server - Only open UDP 53 inbound, since we can control the size of our records to make sure no replies exceed 492 bytes (which would prompt the use of TCP 53). This allows access to our external DNS Server, which is needed to find out the location of GIAC's website.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Rule: permit –i dec0 -p udp –s 0/0 –d 192.168.28.2 - -dport 53 log
CG Rule: permit domain/udp EVERYONE 192.168.28.2 1 - make sure "enable replies" is set

Purpose and Why: Allow SMTP to our SMTP mail relay. This is needed so that email may be sent to GIAC Enterprises.
Rule: permit –i dec0 –p tcp -s 0/0 –d 192.168.28.5 - -dport 25 log
CG Rule: permit smtp/tcp dec0 192.168.28.5

Service Network Access rules
Purpose and Why: Allow SMTP from the SMTP mail relay to our internal mail server for email services.
Rule: permit –i dec1 –p tcp -s 192.168.28.5 -d 192.168.29.6 - -dport 25 log
CG Rule: permit smtp/tcp 192.168.28.5 192.168.29.6

Purpose and Why: Allow SMTP from the internal mail server to the SMTP mail relay for email services.
Rule: permit –i dec2 –p tcp -s 192.168.29.6 -d 192.168.28.5 6 - -dport 25 log
CG Rule: permit smtp/tcp 192.168.29.6 192.168.28.5

Purpose and Why: Allow SMTP out from the SMTP mail relay so that it can deliver SMTP messages – only out to Internet and not to any hosts on our internal subnet.
Rule: permit –i dec1 –p tcp –s 192.168.28.5 ! -d 192.168.29.0/24 - -dport 25 log
CG Rule: permit smtp/tcp 192.168.28.5 dec0

Purpose and Why: Allow SQL from the Web Server to the Internal Database to allow the Web Server to retrieve fortune cookie sayings for the customer. Use the SQL*Net proxy to further protect this connection.
Rule: permit –i dec1 –p tcp -s 192.168.28.3 -d 192.168.29.3 - -dport 1521 (SQL) log – use SQL*Net Proxy
CG Rule: proxy sqlnet/tcp 192.168.28.3 192.168.29.3

Purpose and Why: Allow DNS out from the External DNS server – only out to Internet and not to any hosts on our internal subnet, to provide the capability for our DNS server to make DNS queries from other name servers on the Internet.
Rule: permit –i dec1 –p tcp - s 192.168.28.2 ! -d 192.168.29.0/24 - -dport 53 log
CG Rule: permit domain/tcp 192.168.28.2 dec0

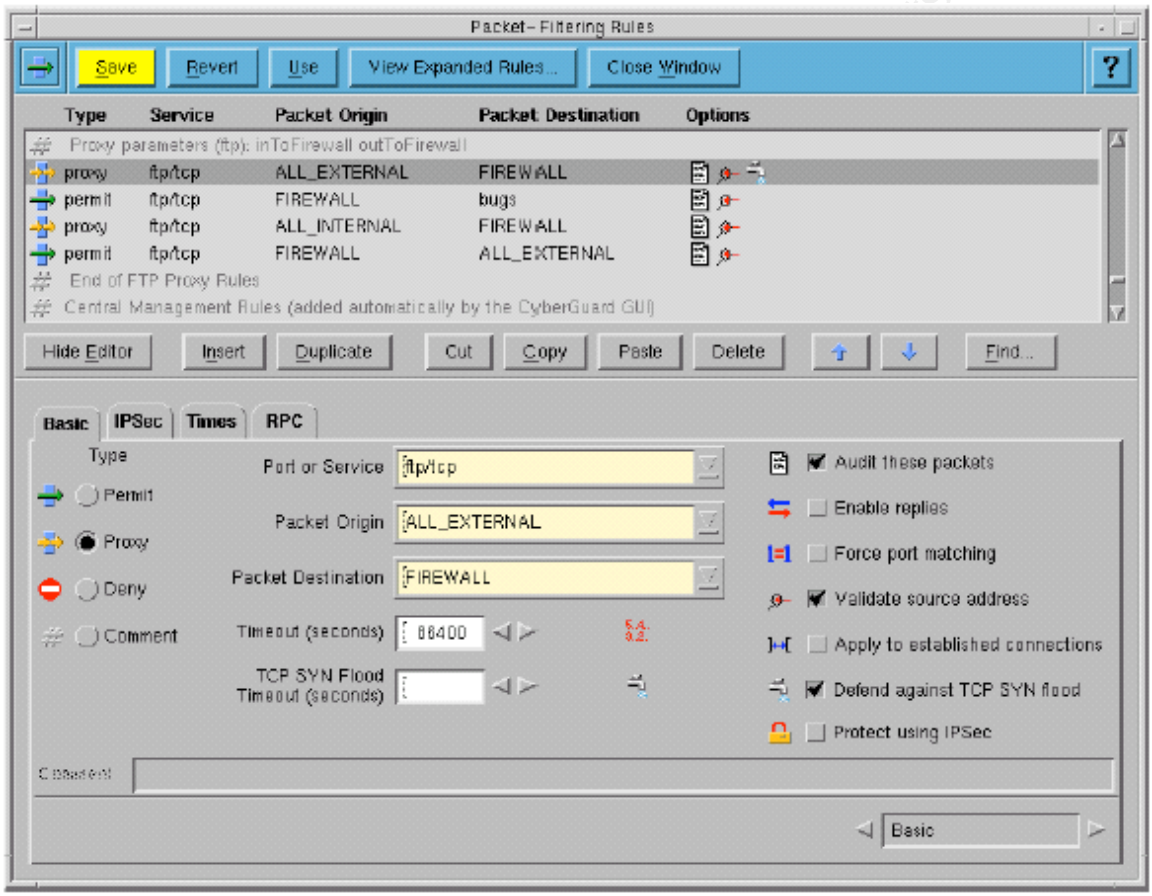
Rule: permit –i dec1 -p udp –s 192.168.28.2 ! -d 192.168.29.0/24 - -dport 53 log
CG Rule: permit domain/udp 192.168.28.2 dec0 1 - make sure "enable replies" is set

The last rule is the implicit deny all rule, we what to drop and log all packets that we didn't specifically allow.
CG Rule: deny ALL EVERYONE EVERYONE

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Tutorial for implementing Firewall rules on the CyberGuard Firewall

This section is a tutorial for implementing the Firewall rules, as defined in the Security Policy for the Primary Firewall section, on the CyberGuard. Following is the "Packet-Filtering Rules" screen from the CyberGuard Firewall (CyberGuard, "Configuring the CyberGuard Firewall", p. II-22).



Make sure that you have defined the IP addresses and Sub-Network Masks in the Network Interface for each of the interfaces, dec0, dec1, and dec2. Otherwise, you will get an error trying to save a firewall rule containing an undefined IP address.

Use the "SmartProxies" screen to set up the following proxies: FTP, HTTP, SQLNET (SQL*Net) and SSL (HTTPS) as defined in assignment #1.

As seen in the screen shot, CyberGuard firewall rules are in the following format: Type, Service, Packet Origin, Packet Destination, and Options.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

For Type, select the appropriate type of rule: permit, proxy (for those permit rules that use a proxy), deny, or comment.

For Service, select the appropriate port or service from the pulldown menu – or in the case of ssh, there is no corresponding menu item, so type in “22/tcp”.

For Packet Origin or Packet Destination, type in the specific IP address, or select an interface if the rule applies to all hosts on that interface. Use EVERYONE for those cases, where the packet origin/destination can by any host on all interfaces.

Use the default value for “Timeout (seconds)”, which is the amount of time a particular connection is waiting for a response.

“TCP SYN Flood Timeout (seconds)” – leave alone, since this is only used with TCP SYN Flood Attack defense.

For those rules that we would like logged, make sure the “Audit these packets” button is checked.

The “Enable replies” button allows returning packets through the firewall. The default for UDP is disabled. For TCP connections this is always enabled. So for all of our UDP connections, make sure this button is checked so that we will get replies to our requests (i.e. DNS).

Only check the “Force port matching” button for DNS requests. This forces the source and destination port to be identical.

Check “Validate source address” (default). This checks the source address against the interface that the packet arrived, to prevent interface spoofing.

Leave the “Apply to established connections” button unchecked, this was added to solve a unique problem specific to a particular company.

The “Defend against TCP SYN Flood” button – leave this unchecked, since a proxy is a better defense mechanism. We’ll be looking at this issue in more detail during our next round of venture capitol funding.

The “Protect using IPSec” button will be used for the firewall rules pertaining to the VPN.

Leave the default time that these rules are applied to all times as seen by selecting the “Times” tab, about halfway down the screen.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

After all the rules have been entered successfully, select the “save” button for the changes to take effect at the next system reboot. Select the “use” button for the changes to take effect immediately.

Security Policy for VPN

We’ll use the following IPSec Security Policy defined for the CyberGuard VPN: the pre-loaded “High Security” settings for the IPSec Protection Strategy. This setting uses both the Encapsulating Security Payload (ESP) and Authentication Header (AH) IPSec protocols. For ESP, the following Encryption Algorithms are used: 3des-cbc, aes-cbc, twofish-cbc. For AH, the following Authentication Algorithm is used: hmac-sha1-96. The “Duration of the Security Association (SA)” is 30 minutes. For our VPN, we will establish keys using IKE, using certificates as authentication. We’ll use the option to compress the IP Payload to counteract overhead introduced by the IPSec protocol. We’ll use the “HighSecurity” setting for IKE Protection Strategy. This setting uses 3des-cbc, aes-cbc, and twofish-cbc for Encryption Algorithms, sha1 for the Hash Algorithm, and Diffe-Hellman Group 2 & 5, SA Lifetime of 3 hours. We will use “Main Mode” as the method for use during IKE Phase 1 negotiations. This is the more secure method since it hides the identity of the IKE negotiators. We are taking a high security stance on all of these options, since files containing our fortune cookie sayings will be transmitted over the VPN. These fortune cookie sayings are the life-blood of our business, so we will protect them the best we can.

The CyberGuard firewall will add rules similar to the following as needed to permit IKE traffic for our VPN channel:

We will use the CyberGuard firewall rule syntax described in the previous section. For this example, we will use a Supplier host of UUU.13.128.36

permit ike/udp FIREWALL UUU.13.128.36
permit ike/udp UUU.13.128.36 FIREWALL

We will also use the FTP proxy on the firewall for our communications with our Suppliers and business Partners to limit their access to only what is needed. Define Suppliers as only having the capability to use FTP to store (put) a file and read the directory, via the CyberGuard FTP proxy. Define Partners as only having the capability to retrieve (get) a file and read the directory, via the FTP proxy.

Firewall rules will be added for each of our Partners and Suppliers, to allow VPN access to our internal FTP server. Note: either a particular IP address or subnet can be specified. Also, the “protect using IPSec” option will be used for these rules.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

GIAC Enterprises mobile sales force and teleworkers will use the CyberGuard Passport One client installed on their PC's, which will connect to our FIREWALL to allow access to the internal and service networks.

Assignment 3 – Verify the Firewall Policy

Planning the Audit

First, we need to obtain proper authorization in writing for performing the audit. Since we will need exclusive access to the Internal and Service Network, we will perform the Audit during the weekend. We will start at 6:00 p.m. on Friday with the goal of having both the Service and Internal network up and running no later than 6:00 a.m. on Monday morning. This will provide the maximum amount of time to perform the audit and fix any problems that come up during our testing, without impacting the traditional work week.

The risks involved with performing the audit include:

- Risk of bringing down the Internal network or Service network. The use of nmap may crash computers or networks. We need to be prepared to reboot servers or hosts and restart processes on those crashed systems. We will be allowing extra time in our audit for these activities.
- Since nmap will be producing a log of traffic, we also run the risk of filling up log space on the Firewall. Use the Alert capability of the Firewall to notify when the disk partition containing the log files is getting full.

In performing our audit, we will be following the steps/tips provided by Lance Spitzner in his paper, "Auditing Your Firewall Setup". (Spitzner)
1) First, audit the firewall. Lance Spitzner recommends port scanning your firewall from the Internet, Internal Network, and Service Network to determine what ports are open on the firewall (Spitzner, p.1).

2) Second, audit the Firewall Rules. Lance Spitzner recommends placing the auditing host on one side of the firewall to scan another system on the other side of the firewall (Spitzner, p.2). This will determine which types of packets can get through the firewall. We will use tcpdump on the other side to verify what packets actually made it through. Our goal is to test every subnet and IP address specified as a source or target IP in our Firewall rules.

Port Scanning consists of a TCP scan, UDP scan and ping (echo request). We will be using nmap and tcpdump to perform our audits. We will use nmap to run a series of scans and tcpdump on the other side of the firewall to verify which packets actually made it through.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

In order to perform a technical audit of GIAC's primary firewall defined in assignment #1 and #2, we need to perform the following scans.

Scanning from the dec0 interface, representing the Internet:

1) Place auditing host on the dec0 interface.
2) Scan the firewall:
   - Verify there are no ports listening.
   - Verify that you can't ping the firewall.
3) Scan the Service Network:
   - Scan the Web server
     - Verify that the Web Server can be accessed only with HTTP/HTTPS traffic.
   - Scan External DNS Server.
     - Verify that the DNS Server can be accessed only with UDP/DNS traffic.
   - Scan SMTP Mail Relay.
     - Verify that the SMTP Mail Relay can be accessed only with SMTP traffic.
   - Scan IDS host.
     - Verity that nothing is accessible.
4) Scan the Internal Network:
   - Scan the Internal FTP Server, Internal Oracle Database, Internal DNS Server, Internal Mail Server, Internal IDS host, internal host used for remote management of the Firewall, CVP Scanning host, another internal host picked at random.
     - Verify that none of our Internal Network hosts can be accessed.

Scanning from the dec1 interface, representing the Service Network:

1. Place auditing host on the dec1 interface.
2. Scan the firewall.
   - Verify that only the SQL*NET proxy has an open listening port.
   - Verify that you can't ping the firewall.
3. Take the Web Server off line and replace it with the auditing host.
   - Verify that the Web Server can only access the internal Database Server (only for SQL traffic).
   - Verify that it cannot access any other host on the Internal Network or a host on the Internet.
4. Take the External DNS Server off line and replace it with the auditing host.
   - Verify that the External DNS Server can only send out DNS traffic to the Internet (UDP and TCP/53) and that it can't access any host on the Internal Network.
5. Take the SMTP Mail Relay off line and replace it with the auditing host.
   - Verify that the SMTP Mail Relay can access only the internal Mail Server (only with SMTP traffic) and no other host on the Internal network.
   - Verify that it can send out SMTP to the Internet.
6. Take the IDS Host off line and replace it with the auditing host

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

- Verify that the IDS Host can't access any host on the Internal network or Internet.

Scanning from the dec2 interface, representing the Internal Network:

1. Place auditing host on the dec2 interface
2. Scan the firewall.
   - Verify that the only listening ports are for the FTP, HTTP, and SSL proxies.
3. Verify that only the following traffic is let through to the Internet from any internal host. (HTTP/HTTPS/DNS/FTP)
4. Verify that HTTP/HTTPS is let through to the Web Server on the Service Network.
5. Verify that the internal Mail Server can access the SMTP Mail Relay on the Service Network (and only with SMTP traffic)
6. Verify that only the internal host 192.168.29.8 can access the firewall's internal interface with ssh.
7. Verify that any internal host is allowed to ping to the firewall.

Cost and level of effort

According to Lance's paper, each scan should take 30-60 minutes (Spitzner, p.2). We will average this to 45 minutes per scan.

Audit from Internet: 6 different nmap scan's performed * 45 minutes = 4.5 hours
Audit from Service Network: 9 different nmap scan's performed * 45 minutes = 6.75 hours
Audit from Internal Network: 6 different nmap scan's performed * 45 minutes = 4.5 hours

Total time to perform scans: 16 hours (rounding up)
Allow for additional time for reboots or bringing the network back up: 4 hours

Note: The number of scans in the above estimate takes into account the ability to use one nmap command to scan the entire Internal or Service network. For the audit from the Internal Network, we will perform scans of the FW, Service Network, and a host on the Internet from 2 different internal hosts.

Estimate: Total of 20 hours for 2 experienced personnel to perform the Audit

Estimate for Planning the Audit: 16 person hours
Estimate for Performing the Audit: 40 person hours
Estimate for Evaluating the Audit: 16 person hours

Audit Total: 72 person hours * $40 per hour = $2880

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Conducting the Audit

For TCP SYN scans, we will use the following nmap command:
nmap –v –sS -sR –P0 –p1-65535 IPAddressToScan
-v - verbose option
–sS for TCP SYN scan – also referred to the "half-open" scan where nmap sends a SYN packet, just like in a valid TCP connection.
-sR – rpc detection (RPC scan)
-P0 – do not try to ping hosts before scanning them
-p option – scan port 1-65535 (all ports)

We will use the –g option to set the source port to 53 for TCP and UDP scans when testing for DNS packets, since we have the rule to force port matching on all of our DNS firewall rules.

We will use the following command to test out an ICMP echo request:
ping IPAddressToPing (ICMP echo request)

For UDP scans, we will use the following nmap command:
nmap –v –sU –sR –P0 –p1-65535 IPAddressToScan

We will use the following tcpdump command on other side to determine which packets actually got through: tcpdump.

Following are a sample of the nmap commands used to conduct the audit.

To perform scans from dec0, I re-placed the router with the auditing host. I changed the IP of the auditing host to the internal interface of the router (10.1.1.253). I issued the following commands to perform a TCP scan and UDP scan of the Firewall, respectively.

```
[root@mdess2-linux penny]# !!
nmap -v -sS -sR -P0 -p1-65535 10.1.1.254

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (10.1.1.254) appears to be up ... good.
Initiating SYN Stealth Scan against  (10.1.1.254)
The SYN Stealth Scan took 6 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (10.1.1.254)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (10.1.1.254) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 16 seconds


[root@mdess2-linux penny]# !!
nmap -v -sU -sR -P0 -p1-65535 10.1.1.254
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (10.1.1.254) appears to be up ... good.
Initiating UDP Scan against  (10.1.1.254)
The UDP Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (10.1.1.254)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (10.1.1.254) are: closed

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

I also issued a ping command to the firewall (i.e. ping 10.1.1.254), from which I received no replies.

As can be seen from the scan results, there were no open ports found on the firewall. Also, by examining the Activity Report of Permitted Packets on the Firewall, no packets were let through. A ping to the firewall did not work.

I issued the following commands to perform a TCP scan and UDP scan of the Web Server:

```
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.28.3

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.28.3) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.28.3)
The SYN Stealth Scan took 9 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.28.3)
The RPCGrind Scan took 0 seconds to scan 0 ports.
Interesting ports on  (192.168.28.3):
(The 65533 ports scanned but not shown below are in state: closed)
Port       State       Service (RPC)
80/tcp     filtered    http
443/tcp    filtered    https

Nmap run completed -- 1 IP address (1 host up) scanned in 19 seconds

[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.28.3

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.28.3) appears to be up ... good.
Initiating UDP Scan against  (192.168.28.3)
The UDP Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.28.3)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.28.3) are: filtered

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

I also issued a ping command to the Web Server (i.e. ping 192.168.28.3), from which I received no replies.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Following is the tcpdump output from the Web Server host:

```
[root@mdess66-linux root]# tcpdump
tcpdump: listening on eth0
09:36:23.479129 10.1.1.253.42399 > 192.168.28.3.https: S
3802527025:3802527025(0) win 2048
09:36:23.479202 192.168.28.3.https > 10.1.1.253.42399: R 0:0(0) ack
3802527026 win 0 (DF)
09:36:23.481297 192.168.28.3.32773 > 192.168.16.1.domain:  44548+ PTR?
253.1.1.10.in-addr.arpa. (41) (DF)
09:36:24.655303 10.1.1.253.42399 > 192.168.28.3.http: S
3802527025:3802527025(0) win 2048
09:36:24.655381 192.168.28.3.http > 10.1.1.253.42399: R 0:0(0) ack
3802527026 win 0 (DF)
```

As can be seen from the tcpdump output, in line 1 our auditing host (10.1.1.253) issued a TCP SYN connection to the https port on the Web Server. In line 4, our auditing host issued a TCP SYN connection to the http port on the Web Server. Based on these results, we know that TCP packets destined to ports 80 and 443 are allowed to the Web Server.

As can be seen from the scan results, the only open ports found on the Web Server were TCP ports 80 and 443, just as we expected. A ping to the Web Server did not work.

I issued the following commands to perform a TCP scan and UDP scan of the External DNS Server:

```
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.28.2

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.28.2) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.28.2)
The SYN Stealth Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.28.2)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.28.2) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.28.2

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.28.2) appears to be up ... good.
Initiating UDP Scan against  (192.168.28.2)
The UDP Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.28.2)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.28.2) are: filtered

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


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

I also issued a ping command to the External DNS Server (i.e. ping 192.168.28.2), from which I received no replies.

Notice that UDP port 53 wasn't seen as being open. Since we selected to "Force port matching" on all of our DNS rules, these crafted UDP packets, whose source ports were not 53, did not make it past the Firewall. We need to re-issue the UDP scan using the –g option, to force the source port to be 53.

```
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p53 -g53 192.168.28.2

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.28.2) appears to be up ... good.
Initiating UDP Scan against  (192.168.28.2)
The UDP Scan took 12 seconds to scan 1 ports.
Adding open port 53/udp
Initiating RPCGrind Scan against  (192.168.28.2)
The RPCGrind Scan took 36 seconds to scan 0 ports.
Interesting ports on  (192.168.28.2):
Port       State       Service (RPC)
53/udp     open        domain

Nmap run completed -- 1 IP address (1 host up) scanned in 58 seconds
```

Following is the tcpdump output from the External DNS Server:

```
[root@mdess66-linux root]# tcpdump
tcpdump: listening on eth0
09:55:47.160451 arp who-has 192.168.28.2 tell 192.168.28.1
09:55:47.160525 arp reply 192.168.28.2 is-at 0:50:8b:d:d9:33
09:55:47.160618 10.1.1.253.domain > 192.168.28.2.domain:  0 [0q] (0)
09:55:47.161989 192.168.28.2 > 10.1.1.253: icmp: 192.168.28.2 udp port
domain unreachable [tos 0xc0]
```

As can be seen from the tcpdump output, in line 3 our auditing host (10.1.1.253) issued a UDP packet from source port 53 (domain) to the domain port on the External DNS Server. Based on these results, we know that UDP packets with a source destination port of 53 are allowed to the External DNS Server.

As can be seen from the scan results, the only open port found on the External DNS Server was UDP port 53, just as we expected. A ping to the External DNS Server did not work.

I issued the following commands to perform a TCP scan and UDP scan of the SMTP Mail Relay:

```
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.28.5
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.28.5) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.28.5)
The SYN Stealth Scan took 8 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.28.5)
The RPCGrind Scan took 0 seconds to scan 0 ports.
Interesting ports on  (192.168.28.5):
(The 65534 ports scanned but not shown below are in state: closed)
Port       State       Service (RPC)
25/tcp     filtered    smtp

Nmap run completed -- 1 IP address (1 host up) scanned in 19 seconds

[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.28.5

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.28.5) appears to be up ... good.
Initiating UDP Scan against  (192.168.28.5)
The UDP Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.28.5)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.28.5) are: filtered

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

I also issued a ping command to the SMTP Mail Relay (i.e. ping 192.168.28.5), from which I received no replies.

As can be seen from the scan results, the only open port found on the SMTP Mail Relay was UDP port 25, just as we expected.

I issued the following commands to perform a TCP scan and UDP scan of the External IDS Host:

```
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.28.4

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.28.4) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.28.4)
The SYN Stealth Scan took 6 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.28.4)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.28.4) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.28.4

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.28.4) appears to be up ... good.
Initiating UDP Scan against  (192.168.28.4)
The UDP Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.28.4)
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.28.4) are: filtered

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

I also issued a ping command to the SMTP Mail Relay (i.e. ping 192.168.28.4), from which I received no replies.

As can be seen from the scan results, there were no open ports found on the External IDS Host, just as we expected.

Refer to Appendix A for the rest of the scans performed from Dec0 to the hosts on the Internal Network. As can be seen from the scan results, there were no open ports found on any of the Internal Network Hosts, which is just as we expected.

Refer to Appendix B for the nmap scans performed from Dec1 to the Firewall, hosts on the Service Network and hosts on the Internal Network.

Refer to Appendix C for the nmap scans performed from Dec2 to the Firewall, hosts on the Service Network and hosts on the Internal Network.

To save on space in Appendix B and C, if the nmap command didn't show any open ports, only the nmap command is given.

Refer to Appendix D for a report of the packets permitted by the Firewall. There were instances where the nmap scan did not show a port as being open, but the actual packet was permitted by the firewall. One such case was performing a TCP scan from a host on the Internal Network to the Internet. The DNS port wasn't shown as open, but by looking at the Firewall logs, the packet was permitted.

Evaluating the Audit

To evaluate the Audit, compare the open ports as seen by the various TCP and UDP scans, along with the results of issuing ping commands, and sometimes even reviewing the "Permitted packets" report on the Firewall, with what was expected. All was as expected with the following exceptions:

- There should have an open listening port on the firewall interface to the Service Network for the SQL*NET Proxy, but the TCP scan did not show an open port. Investigate why there wasn't an open port for the SQL*NET Proxy. Double-check that the SQL*NET Proxy is set up correctly.
- According to the scans, tcp ports 21(ftp), 80 (http) and 443 (https) can be accessed on every host of the Service Network from a host on the Internal Network. The same applies to the tcp and udp port 53(dns). This is not what was intended by the security policy.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The following Firewall rules should be changed:

#Outbound to Internet
CG Rule: proxy http/tcp dec2 EVERYONE
CG Rule: proxy https/tcp dec2 EVERYONE
CG Rule: proxy ftp/tcp dec2 EVERYONE
CG Rule: permit domain/tcp dec2 EVERYONE
CG Rule: permit domain/udp dec2 EVERYONE

These rules should not include EVERYONE since this encompasses too many hosts. Most of these rules should be split into two rules: one to permit access from the Internal Network to the Internet and another rule to specifically permit access to the desired host on the Service network.

Split the rule “proxy http/tcp dec2 EVERYONE” into the following 2 rules:
proxy http/tcp dec2 dec0
proxy http/tcp dec2 192.168.28.3 (Web Server on the Service Network)

Split the rule “proxy https/tcp dec2 EVERYONE” into the following 2 rules:
proxy https/tcp dec2 dec0
proxy https/tcp dec2 192.168.28.3 (Web Server on the Service Network)

Replace the rule “proxy ftp/tcp dec2 EVERYONE” with the more specific rule:
proxy ftp/tcp dec2 dec0
Note: We do not have an ftp server on our Service Network, so no further rule is needed.

Replace the rule “permit domain/tcp dec2 EVERYONE” with the more specific rule:
permit domain/tcp dec2 192.168.28.2 (External DNS server on the Service Network)

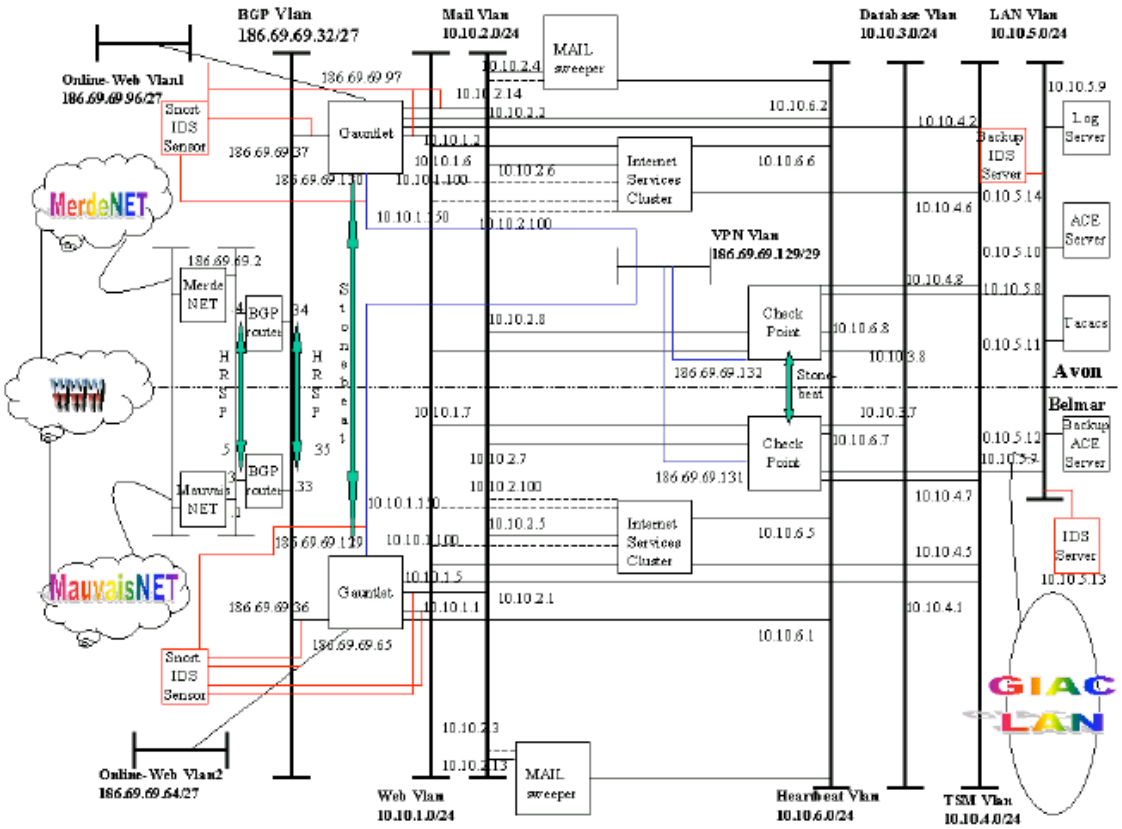
Replace the rule “permit domain/udp dec2 EVERYONE” with the more specific rule:
permit domain/udp dec2 192.168.28.2 (External DNS server on the Service Network)

© SANS Institute 2003, Author retains full rights.

Assignment 4 – Design Under Fire

I have selected the network design from Mark Hillick’s GCFW Practical, Analyst number: 0358, practical assignment URL: http://www.giac.org/practical/GCFW/Mark_Hillick_GCFW.pdf.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.



An attack against the firewall itself

For this type of attack, I will exploit the vulnerability described by the CERT Advisory CA-2002-19 Buffer Overflows in Multiple DNS Resolver Libraries, initial release date: June 28, 2002. Mark Hillick's design uses the Gauntlet 6.0 Firewall with a DNS server running on it (the platform is a Solaris box), which fits this vulnerability.

Basically, the buffer overflow vulnerability resides in the DNS resolver libraries, which are used by the Gauntlet firewall to perform DNS queries. The resolver libraries are provided by the Solaris operating system. There are two different DNS responses that could cause a buffer overflow to occur: host names or addresses and network names or addresses.

To correct this vulnerability, an upgrade to an updated resolver library needs to be applied. There currently is one available for the Solaris operating system (libresolv.so).

Attack:

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

I need to obtain a couple of e-mail addresses of GIAC employees either via social engineering or maybe GIAC's web pages. I also need to have a primary DNS server that I am in control of. The attack will consist of sending an email enticing the user to click on an included URL to a domain whose primary DNS server is under my control. I will then have control over the contents of the DNS response to the DNS query that will be issued by the Gauntlet firewall. When the internal GIAC employee clicks on the URL, it will get forwarded to the Gauntlet firewall to perform the DNS query. When the firewall tries to read the DNS response from my DNS server, it will invoke the vulnerable DNS resolver library, which will have a buffer overflow.

According to Mark's design "all Internal Services that require external DNS services should forward queries to the second external firewall", which is a Gauntlet firewall (Hillick, p.17).

Results of attack:
The extra bytes that I placed in the DNS response to cause the buffer overflow will cause the Solaris box to crash, causing a denial of service of the firewall.

From looking at Mark's Practical assignment, there is no logging performed by the Border Router for our DNS response. It is not explicitly stated that logging is performed for DNS responses received by the Gauntlet Firewall. But it is hard to imagine that through thorough analysis of figuring out what was going on in the system during the crash that our IP address wouldn't show up somewhere. In which case, it would be better to find some exploit code that would give us root access to the firewall, where we would then be able to cover up our tracks (by deleting log files, etc.)

Denial of Service Attack

Perform reconnaissance to obtain IP addresses of Mark's GIAC's network. Obtain the IP address of GIAC's web site (via the nslookup command on a Windows box i.e. "nslookup www.giac.com"). Use nmap to search for a valid list of hosts that make up GIAC's internal network. Make sure to use the option to spoof our IP address, so GIAC won't know where the scans are actually coming from. Use the -S <IP_Address>option, or the decoy option to specify other valid hosts as the initiator of the scans against the target so that Mark's IDS will pick up more than 1 scan taking place. One can also use the –randomize_hosts option, which will randomize the hosts before the scan is performed which will help avoid detection. For each valid host, use nmap to scan for open TCP and UDP ports. Use these open ports as targets for our attack. During this phase, it is highly likely that I will find all the Web Servers (by seeing which hosts have port 80 open), and the SMTP host (by seeing which host has TCP/25 port open). If we are lucky, we will even find the DNS server (which is the Firewall).

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Install the Tribe Flood Network 200 (TFN2K) attack tool on the 50 compromised cable modem/DSL systems. The source code for TFN2K can be found at the following URL: http://1337.tsx.org/. TFN2K is a distributed Denial of Service (DDoS) attack tool. TFN2K consists of a master (containing the client which issues commands to the agents) and agents (programs that actual perform the attack). TFN2K allows both the master and agents to spoof their IP address. Using TFN2K, targets can be attacked with a TCP/SYN, UDP, ICMP/ping, or broadcast ping packet flood.

Using the attack tool, I will instruct the agents to perform the following attacks on GIAC's network:

- udp flood of the entire network (This should affect the DNS Server, since it will be listening on UDP port 53.)
- syn flood ports 80 and 443 to the Web servers, and port 25 to the SMTP host (syn flood port 53 to the DNS Server if the DNS Server can be determined during the Reconnaissance phase.)
- icmp echo flood of the entire network
- icmp broadcast (smurf) flood of the entire network
- targa3 attack of the entire network

In the targa3 attack, invalid IP packets of different protocols are sent to the target host. The result of this attack depends on how robust the IP stack is on the various hosts. The host under attack might just discard the packets or crash.

Mark's network design includes a log server and an IDS server. Our DDoS attack will surely be noticed.

Countermeasures to mitigate attack:

- Block ICMP messages – Mark already blocks ICMP at the Firewall.
- The GIAC's service-based proxies should prevent the syn floods to port 80 and 443 from the TFN2K attack tool. However, since there are no service-specific proxies for DNS, the DNS server should get hit. Mark does have a CSMAP daemon listening for SMTP on the external Gauntlet Firewalls, which will wait until the full SMTP packet is received before relaying it on (Hillick, p. 69).

In the following CISCO white paper, "Strategies to protect against Distributed Denial of Service (DDoS) Attacks", CISCO specifies the following prevention measures against DDoS attacks (CISCO, p.3-5).

- Filter all non-routable IP addresses – which Mark is doing (and logging)
- Block one's own IP address space – which Mark is doing (and logging)
- Use committed access rate (CAR) feature of IOS 12.1 to rate limit SYN packets. Mark is using a router with IOS 12.1, which supports CAR.
- Rely on ISP's to prevent spoofed IP addresses from entering the Internet.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Compromising an internal system through the perimeter system

For this type of attack, I will choose the Web Server since it is an easy target. There appears to be numerous vulnerabilities against Web Servers on the Internet. First, I will perform reconnaissance to obtain the IP address of GIAC's web site (via the nslookup command) and try to obtain a list of all Web Servers in the GIAC Network (as described in the previous attack). I will also use the -O option for the nmap command, to perform OS fingerprinting of the Web Servers.

Once I have found the list of GIAC's Web Servers, I will run the nessus command against them, to detect the type of Web Server running on them and any known vulnerabilities. After performing the reconnaissance, I should know that GIAC's Web Servers are running the Apache software (version 1.3.24) on a Solaris 8 UNIX System.

I will exploit the "Apache HTTP Server Chunk Encoding Vulnerability", which affects the Apache web server version 1.3.24. described by the following CIAC Information Bulletin. URL http://www.ciac.org/ciac/bulletins/m-093.shtml, dated June 2002.

This version of the Apache Web Server contains a bug in the routines that process HTTP requests that are encoded using chunked encoding. A child process is started to process this request. The exploit code sends an invalid request, which causes a stack overflow. This stack overflow problem could be exploited to run code on the Web Server with the same permissions as an Apache Web Server child process.

As described in Robert Crook's GCIH Practical, titled "Port 80: Apache HTTP Daemon Exploit", there is exploit code available for BSD systems (FreeBSD, OpenBSD, and NetBSD). The exploit code also supports a "brute force" mode to try the exploit code on a different OS, Solaris 8 in our case. The apache-nosejob.c exploit code can be found at the following URL: http://packetstorm.decepticons.org/0206-exploits/apache-nosejob.c (Crooks, p.14).

I will try the "brute force" mode of the exploit code to see if it will work with the Solaris 8 platform and allow me access to the Web Server. If this doesn't succeed, I can try to study the exploit code, and with my vast knowledge of the Solaris OS and Apache code, I can try to port this code to the Solaris 8 platform.

If I do succeed in getting access to GIAC's Web Server, my abilities will be limited, since Mark has hardened the Web-Server using TITAN.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

List of References:

Barlow, Jason and Woody Thrower. "TFN2K – An Analysis". 7 March 2000. URL: http://packetstormsecurity.nl/distributed/TFN2k_Analysis-1.3.txt (21 March 2003).

CIAC. "M-093: Apache HTTP Server Chunk Encoding Vulnerability". 19 June 2002. URL: http://www.ciac.org/ciac/bulletins/m-093.shtml (31 March 2003).

CERT. "CERT Advisory CA-2002-19 Buffer Overflows in Multiple DNS Resolver Libraries". 9 September 2002. URL: http://www.cert.org/advisories/CA-2002-19.html (1 March 2003).

Cisco. "Strategies to Protect Against Distributed Denial of Service (DDoS) Attacks". 17 February 2000. URL: http://www.cisco.com/warp/public/707/newsflash.html (21 March 2003).

Crooks, Robert. "Port 80: Apache HTTP Daemon Exploit". September 2002. URL: www.giac.org/practical/Robert_Crooks_GCIH.doc (21 March 2003).

CyberGuard Corporation. Administering the CyberGuard Firewall. February 2002.

CyberGuard Corporation. Configuring Smart Proxies for the CyberGuard Firewall. February 2002.

CyberGuard Corporation. Configuring the CyberGuard Firewall. February 2002.

Forristal, Jeff. "Fireproofing Against DoS Attacks". Network Computing. 10 December 2001. URL: http://www.networkcomputing.com/1225/1225f38.html (21 March 2003).

Fyodor. "The Art of Port Scanning." 6 September 1997. URL: http://www.insecure.org/nmap/nmap_doc.html (1 March 2003).

Fyodor. "Nmap network security scanner man page". URL: http://www.insecure.org/nmap/data/nmap_manpage.html (1 March 2003).

Hillick, Mark. "GCFW 1.7 Assignment". May 2002. URL: http://www.giac.org/practical/GCFW/Mark_Hillick_GCFW.pdf (1 March 2003).

Internet Storm Center. "Top10 Source IPs". 8 February 2003. URL: http://isc.incidents.org/top10.html (9 February 2003).

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Mixter Security. URL: http://1337.tsx.org/ (21 March 2003).

Packet Storm. “distributed attack tools”. URL: http://packetstormsecurity.nl/distributed/ (21 March 2003).

Spitzner, Lance. “Auditing Your Firewall Setup”. 12 December 2000. URL: http://www.spitzner.net/audit.html (1 March 2003).

Tcpdump.org. “tcpdump - dump traffic on a network”. URL: http://www.tcpdump.org/tcpdump_man.html (1 March 2003).

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Appendix A - Auditing from Dec0

```
#TCP Scan Internal FTP Server
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.29.5

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.5) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.5)
The SYN Stealth Scan took 6 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.5)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.5) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#UDP Scan Internal FTP Server
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.29.5

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.5) appears to be up ... good.
Initiating UDP Scan against  (192.168.29.5)
The UDP Scan took 10 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.5)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.5) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 20 seconds

#TCP Scan Internal Oracle Database
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.29.3

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.3) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.3)
The SYN Stealth Scan took 6 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.3)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.3) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#UDP Scan Internal Oracle Database
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.29.3

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.3) appears to be up ... good.
Initiating UDP Scan against  (192.168.29.3)
The UDP Scan took 8 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.3)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.3) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 18 seconds

#TCP Scan Internal DNS Server
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.29.7

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.7) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.7)
The SYN Stealth Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.7)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.7) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#UDP Scan Internal DNS Server
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.29.7

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.7) appears to be up ... good.
Initiating UDP Scan against  (192.168.29.7)
The UDP Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.7)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.7) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#TCP Scan Internal Mail Server
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.29.6

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.6) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.6)
The SYN Stealth Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.6)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.6) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#UDP Scan Internal Mail Server
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.29.6

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.6) appears to be up ... good.
Initiating UDP Scan against  (192.168.29.6)
The UDP Scan took 9 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.6)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.6) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 19 seconds

#TCP Scan Internal IDS Host

[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.29.4

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.4) appears to be up ... good.
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
Initiating SYN Stealth Scan against  (192.168.29.4)
The SYN Stealth Scan took 6 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.4)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.4) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 16 seconds

#UDP Scan Internal IDS Host
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.29.4

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.4) appears to be up ... good.
Initiating UDP Scan against  (192.168.29.4)
The UDP Scan took 9 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.4)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.4) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 20 seconds

#TCP Scan Internal Host used for remote management of FW
root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.29.8

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.8) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.8)
The SYN Stealth Scan took 6 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.8)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.8) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 16 seconds

#UDP Scan Internal Host used for remote management of FW
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.29.8

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.8) appears to be up ... good.
Initiating UDP Scan against  (192.168.29.8)
The UDP Scan took 10 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.8)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.8) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 20 seconds

#TCP Scan Internal CVP Scanning Host
root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.29.2

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.2) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.2)
The SYN Stealth Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.2)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.2) are: closed
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

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

#UDP Scan Internal CVP Scanning Host
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.29.2

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.2) appears to be up ... good.
Initiating UDP Scan against  (192.168.29.2)
The UDP Scan took 10 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.2)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.2) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 20 seconds

#TCP Scan random Internal Host
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.29.20

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.20) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.20)
The SYN Stealth Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.20)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.20) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#UDP Scan random Internal Host
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.29.20

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.20) appears to be up ... good.
Initiating UDP Scan against  (192.168.29.20)
The UDP Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.20)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.20) are: filtered

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

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Appendix B - Auditing from Dec1

```
#TCP scan of FW
[root@mdess2-linux penny]# !!
nmap -v -sS -sR -P0 -p1-65535 192.168.28.1

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.28.1) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.28.1)
The SYN Stealth Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.28.1)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.28.1) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#UDP scan of FW
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.28.1

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.28.1) appears to be up ... good.
Initiating UDP Scan against  (192.168.28.1)
The UDP Scan took 8 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.28.1)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.28.1) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 18 seconds

#TCP Scan WS - Internal Oracle DB
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.29.3

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.3) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.3)
Adding open port 1521/tcp
The SYN Stealth Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.3)
The RPCGrind Scan took 0 seconds to scan 0 ports.
Interesting ports on  (192.168.29.3):
(The 65534 ports scanned but not shown below are in state: closed)
Port       State       Service (RPC)
1521/tcp   open        oracle

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#UDP Scan WS - Internal Oracle DB
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 192.168.29.3

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.3) appears to be up ... good.
Initiating UDP Scan against  (192.168.29.3)
The UDP Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.3)
The RPCGrind Scan took 0 seconds to scan 0 ports.
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
All 65535 scanned ports on  (192.168.29.3) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#TCP Scan WS - Host on Internet
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 10.1.1.253

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (10.1.1.253) appears to be up ... good.
Initiating SYN Stealth Scan against  (10.1.1.253)
The SYN Stealth Scan took 8 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (10.1.1.253)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (10.1.1.253) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 18 seconds

#UDP Scan WS - Host on Internet
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -p1-65535 10.1.1.253

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (10.1.1.253) appears to be up ... good.
Initiating UDP Scan against  (10.1.1.253)
The UDP Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (10.1.1.253)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (10.1.1.253) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#TCP Scan WS - Host on Internal Network
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.29.20

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.20) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.20)
The SYN Stealth Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.20)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.20) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#UDP Scan WS - Host on Internal Network
nmap -v -sU -sR -P0 -p1-65535 192.168.29.20

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.20) appears to be up ... good.
Initiating UDP Scan against  (192.168.29.20)
The UDP Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.20)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.20) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#TCP Scan External DNS - Host on Internet
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -g53 -p1-65535
10.1.1.253

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (10.1.1.253) appears to be up ... good.
Initiating SYN Stealth Scan against  (10.1.1.253)
The SYN Stealth Scan took 6 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (10.1.1.253)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (10.1.1.253) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 16 seconds

#UDP Scan External DNS - Host on Internet
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -g53 -p1-65535
10.1.1.253

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (10.1.1.253) appears to be up ... good.
Initiating UDP Scan against  (10.1.1.253)
The UDP Scan took 9 seconds to scan 65535 ports.
Adding open port 53/udp
Initiating RPCGrind Scan against  (10.1.1.253)
The RPCGrind Scan took 0 seconds to scan 0 ports.
Interesting ports on  (10.1.1.253):
(The 65534 ports scanned but not shown below are in state: filtered)
Port       State       Service (RPC)
53/udp     open        domain

Nmap run completed -- 1 IP address (1 host up) scanned in 19 seconds

#TCP Scan External DNS - Host on Internal Network
nmap -v -sS -sR -P0 -g53 -p1-65535 192.168.29.5/24

[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -g53 -p1-65535
192.168.29.5

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.5) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.5)
The SYN Stealth Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.5)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.5) are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 17 seconds

#UDP Scan External DNS - Host on Internal Network
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -g53 -p1-65535
192.168.29.5

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.5) appears to be up ... good.
Initiating UDP Scan against  (192.168.29.5)
The UDP Scan took 8 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.5)
The RPCGrind Scan took 0 seconds to scan 0 ports.
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
All 65535 scanned ports on  (192.168.29.5) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 18 seconds

#nmap command used to scan a few of the hosts on the Internal Network
nmap -v -sU -sR -P0 -g53 -p1-65535 192.168.29.1-9

#TCP Scan SMTP Relay - Internal Network
[root@mdess2-linux penny]# !!
nmap -v -sS -sR -P0 -p1-65535 192.168.29.1-9

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.1) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.1)
The SYN Stealth Scan took 8 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.1)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.1) are: closed

Host  (192.168.29.2) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.2)
The SYN Stealth Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.2)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.2) are: closed

Host  (192.168.29.3) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.3)
The SYN Stealth Scan took 8 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.3)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.3) are: closed

Host  (192.168.29.4) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.4)
The SYN Stealth Scan took 12 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.4)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.4) are: closed

Host  (192.168.29.5) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.5)
The SYN Stealth Scan took 8 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.5)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.5) are: closed

Host  (192.168.29.6) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.6)
The SYN Stealth Scan took 11 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.6)
The RPCGrind Scan took 0 seconds to scan 0 ports.
Interesting ports on  (192.168.29.6):
(The 65534 ports scanned but not shown below are in state: closed)
Port       State       Service (RPC)
25/tcp     filtered    smtp
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
Host  (192.168.29.7) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.7)
The SYN Stealth Scan took 9 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.7)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.7) are: closed

Host  (192.168.29.8) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.8)
The SYN Stealth Scan took 9 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.8)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.8) are: closed

Host  (192.168.29.9) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.9)
The SYN Stealth Scan took 10 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.9)
The RPCGrind Scan took 0 seconds to scan 0 ports.
All 65535 scanned ports on  (192.168.29.9) are: closed

Nmap run completed -- 9 IP addresses (9 hosts up) scanned in 173
seconds

#UDP Scan SMTP Relay - Host on Internal Network
nmap -v -sU -sR -P0 -p1-65535 192.168.29.1-9

#TCP Scan SMTP Relay - Host on Internet
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 10.1.1.253

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
nmap -v -sS -sR -P0 -p1-65535 10.1.1.253
Host  (10.1.1.253) appears to be up ... good.
Initiating SYN Stealth Scan against  (10.1.1.253)
The SYN Stealth Scan took 9 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (10.1.1.253)
The RPCGrind Scan took 0 seconds to scan 0 ports.
Interesting ports on  (10.1.1.253):
(The 65534 ports scanned but not shown below are in state: closed)
Port       State      Service (RPC)
25/tcp     filtered   smtp

Nmap run completed -- 1 IP address (1 host up) scanned in 19 seconds

#UDP Scan SMTP Relay - Host on Internet
nmap -v -sU -sR -P0 -p1-65535 10.1.1.253

#TCP Scan External IDS - Host on Internal Network
nmap -v -sS -sR -P0 -p1-65535 192.168.29.1-9

#UDP Scan External IDS - Host on Internal Network
nmap -v -sU -sR -P0 -p1-65535 192.168.29.1-9
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Appendix C - Auditing from Dec2

```
#TCP scan the FW
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.29.1

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.1) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.1)
Adding open port 443/tcp
Adding open port 21/tcp
Adding open port 80/tcp
The SYN Stealth Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.1)
The RPCGrind Scan took 1 second to scan 0 ports.
Interesting ports on  (192.168.29.1):
(The 65532 ports scanned but not shown below are in state: closed)
Port       State       Service (RPC)
21/tcp     open        ftp
80/tcp     open        http
443/tcp    open        https

Nmap run completed -- 1 IP address (1 host up) scanned in 18 seconds

#UDP scan the FW
nmap -v -sU -sR -P0 -p1-65535 192.168.29.1

#ping the firewall interface
[root@mdess2-linux penny]# ping 192.168.29.1
PING 192.168.29.1 (192.168.29.1) from 192.168.29.10 : 56(84) bytes of
data.
64 bytes from 192.168.29.1: icmp_seq=1 ttl=64 time=0.266 ms
64 bytes from 192.168.29.1: icmp_seq=2 ttl=64 time=0.197 ms

#TCP Scan - internal host - host on Internet

[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 10.1.1.253

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (10.1.1.253) appears to be up ... good.
Initiating SYN Stealth Scan against  (10.1.1.253)
Adding open port 443/tcp
Adding open port 80/tcp
Adding open port 21/tcp
The SYN Stealth Scan took 7 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (10.1.1.253)
The RPCGrind Scan took 1 second to scan 0 ports.
Interesting ports on  (10.1.1.253):
(The 65532 ports scanned but not shown below are in state: closed)
Port       State       Service (RPC)
21/tcp     open        ftp
80/tcp     open        http
443/tcp    open        https

Nmap run completed -- 1 IP address (1 host up) scanned in 18 seconds
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
# Problems with seeing TCP DNS
#Note: DNS doesn't show to be open, however it is in the FW logs
root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p53 -g53 10.1.1.253

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (10.1.1.253) appears to be up ... good.
Initiating SYN Stealth Scan against  (10.1.1.253)
The SYN Stealth Scan took 6 seconds to scan 1 ports.
Initiating RPCGrind Scan against  (10.1.1.253)
The RPCGrind Scan took 0 seconds to scan 0 ports.
The 1 scanned port on  (10.1.1.253) is: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 16 seconds

#UDP Scan - internal host - host on Internet
# using -g53 option
[root@mdess2-linux penny]# nmap -v -sU -sR -P0 -g53 -p1-65535
10.1.1.253

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (10.1.1.253) appears to be up ... good.
Initiating UDP Scan against  (10.1.1.253)
The UDP Scan took 8 seconds to scan 65535 ports.
Adding open port 53/udp
Initiating RPCGrind Scan against  (10.1.1.253)
The RPCGrind Scan took 1 second to scan 0 ports.
Interesting ports on  (10.1.1.253):
(The 65534 ports scanned but not shown below are in state: filtered)
Port       State       Service (RPC)
53/udp     open        domain

Nmap run completed -- 1 IP address (1 host up) scanned in 19 seconds

#TCP Scan Internal Mail Server - Service Network
nmap -v -sS -sR -P0 -p1-65535 192.168.28.1-5
#Note: ports 21, 80 and 443 were open on all the hosts on the Service
Network


Host  (192.168.28.5) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.28.5)
Adding open port 443/tcp
Adding open port 21/tcp
Adding open port 80/tcp
The SYN Stealth Scan took 9 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.28.5)
The RPCGrind Scan took 1 second to scan 0 ports.
Interesting ports on  (192.168.28.5):
(The 65531 ports scanned but not shown below are in state: closed)
Port       State       Service (RPC)
21/tcp     open        ftp
25/tcp     filtered    smtp
80/tcp     open        http
443/tcp    open        https

#UDP Scan Internal Mail Server - Service Network
nmap -v -sU -sR -P0 -p1-65535 192.168.28.1-5
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
 - no UDP traffic was allowed

#TCP Scan 192.168.29.8 - FW
#Note:  ssh doesn't show up here as an open port - however, it shows up
in the
#Firewall permitted logs
[root@mdess2-linux penny]# nmap -v -sS -sR -P0 -p1-65535 192.168.29.1

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.29.1) appears to be up ... good.
Initiating SYN Stealth Scan against  (192.168.29.1)
Adding open port 80/tcp
Adding open port 21/tcp
Adding open port 443/tcp
The SYN Stealth Scan took 6 seconds to scan 65535 ports.
Initiating RPCGrind Scan against  (192.168.29.1)
The RPCGrind Scan took 2 seconds to scan 0 ports.
Interesting ports on  (192.168.29.1):
(The 65532 ports scanned but not shown below are in state: closed)
Port       State       Service (RPC)
21/tcp     open        ftp
80/tcp     open        http
443/tcp    open        https

Nmap run completed -- 1 IP address (1 host up) scanned in 19 seconds

#UDP Scan 192.168.29.8 - FW

nmap -v -sU -sR -P0 -p1-65535 192.168.29.1
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Appendix D - Firewall's Packets Permitted Report

Note: Following is an excerpt from the Packets Permitted Report from the CyberGuard Firewall. For formatting purposes, I removed the Date/Time stamps from each entry. The columns are Source Interface, Destination Interface, Source IP, Destination IP, protocol (tcp, udp, icmp), Source Port and Destination Port.

```
Packets permitted
Wed Mar 26 04:00:52 2003

dec0  dec1  10.1.1.253     192.168.28.3   tcp    52199      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    52200      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    52201      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    52199      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    52200      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    52201      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    52202      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    52202      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    52203      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    52203      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    52204      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    52204      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    34085      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    34086      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    34087      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    34085      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    34086      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    34087      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    34088      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    34088      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    34089      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    34089      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    34090      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    34090      https
dec0  dec1  10.1.1.253     192.168.28.2   udp    nameserver nameserver
dec0  dec1  10.1.1.253     192.168.28.2   udp    nameserver nameserver
dec0  dec1  10.1.1.253     192.168.28.5   tcp    41390      smtp
dec0  dec1  10.1.1.253     192.168.28.5   tcp    41391      smtp
dec0  dec1  10.1.1.253     192.168.28.5   tcp    41392      smtp
dec0  dec1  10.1.1.253     192.168.28.5   tcp    41393      smtp
dec0  dec1  10.1.1.253     192.168.28.5   tcp    41394      smtp
dec0  dec1  10.1.1.253     192.168.28.5   tcp    41395      smtp
dec1  dec2  192.168.28.3   192.168.29.3   tcp    53251      sqlnet
dec1  dec2  192.168.28.3   192.168.29.3   tcp    32782      sqlnet
dec1  dec0  192.168.28.2   10.1.1.253     tcp    nameserver nameserver
dec1  dec0  192.168.28.2   10.1.1.253     tcp    nameserver nameserver
dec1  dec0  192.168.28.2   10.1.1.253     udp    nameserver nameserver
dec1  dec0  192.168.28.2   10.1.1.253     tcp    nameserver nameserver
dec1  dec0  192.168.28.2   10.1.1.253     udp    nameserver nameserver
dec1  dec0  192.168.28.2   10.1.1.253     udp    nameserver nameserver
dec1  dec2  192.168.28.5   192.168.29.6   tcp    47991      smtp
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
dec1  dec2  192.168.28.5   192.168.29.6   tcp    47992      smtp
dec1  dec2  192.168.28.5   192.168.29.6   tcp    47993      smtp
dec1  dec2  192.168.28.5   192.168.29.6   tcp    47994      smtp
dec1  dec2  192.168.28.5   192.168.29.6   tcp    47995      smtp
dec1  dec2  192.168.28.5   192.168.29.6   tcp    47996      smtp
dec1  dec0  192.168.28.5   10.1.1.253     tcp    43049      smtp
dec1  dec0  192.168.28.5   10.1.1.253     tcp    43050      smtp
dec1  dec0  192.168.28.5   10.1.1.253     tcp    43051      smtp
dec1  dec0  192.168.28.5   10.1.1.253     tcp    43052      smtp
dec1  dec0  192.168.28.5   10.1.1.253     tcp    43053      smtp
dec1  dec0  192.168.28.5   10.1.1.253     tcp    43054      smtp
dec2  lo0   192.168.29.10  192.168.29.1   tcp    35480      https
dec2  lo0   192.168.29.10  192.168.29.1   tcp    35480      ftp
dec2  lo0   192.168.29.10  192.168.29.1   tcp    35480      www-http
dec2  lo0   192.168.29.10  192.168.29.1   tcp    32783      ftp
dec2  lo0   192.168.29.10  192.168.29.1   tcp    32784      www-http
dec2  lo0   192.168.29.10  192.168.29.1   tcp    32785      https
dec2  lo0   192.168.29.10  192.168.29.1   tcp    1030       https
dec2  lo0   192.168.29.10  192.168.29.1   icmp   ECHO
dec2  dec0  192.168.29.10  10.1.1.253     tcp    60001      https
dec2  dec0  192.168.29.10  10.1.1.253     tcp    60001      www-http
dec2  dec0  192.168.29.10  10.1.1.253     tcp    60001      ftp
dec2  dec0  192.168.29.10  10.1.1.253     tcp    32786      ftp
dec2  dec0  192.168.29.10  10.1.1.253     tcp    32787      www-http
dec2  dec0  192.168.29.10  10.1.1.253     tcp    32788      https
lo0   dec0  10.1.1.254     10.1.1.253     tcp    1035       https
dec2  dec0  192.168.29.10  10.1.1.253     udp    nameserver nameserver
dec2  dec0  192.168.29.8   10.1.1.253     tcp    57771      www-http
dec2  dec0  192.168.29.8   10.1.1.253     tcp    57771      ftp
dec2  dec0  192.168.29.8   10.1.1.253     tcp    57771      https
dec2  dec0  192.168.29.8   10.1.1.253     tcp    ftp-proxy  ftp
dec2  dec0  192.168.29.8   10.1.1.253     tcp    32790      www-http
dec2  dec0  192.168.29.8   10.1.1.253     tcp    telnet-pro https
lo0   dec0  10.1.1.254     10.1.1.253     tcp    1040       https
dec2  lo0   192.168.29.8   192.168.29.1   tcp    55232      ftp
dec2  lo0   192.168.29.8   192.168.29.1   tcp    55232      https
dec2  lo0   192.168.29.8   192.168.29.1   tcp    55232      ssh
dec2  lo0   192.168.29.8   192.168.29.1   tcp    55232      www-http
dec2  lo0   192.168.29.8   192.168.29.1   tcp    32792      ftp
dec2  lo0   192.168.29.8   192.168.29.1   tcp    smtp-proxy www-http
dec2  lo0   192.168.29.8   192.168.29.1   tcp    32794      https
dec2  lo0   192.168.29.8   192.168.29.1   tcp    43275      ssh
dec2  lo0   192.168.29.8   192.168.29.1   tcp    1045       https
dec2  lo0   192.168.29.8   192.168.29.1   tcp    48252      ssh
dec2  lo0   192.168.29.8   192.168.29.1   tcp    55708      ssh
dec2  lo0   192.168.29.8   192.168.29.1   tcp    55708      www-http
dec2  lo0   192.168.29.8   192.168.29.1   tcp    55708      ftp
dec2  lo0   192.168.29.8   192.168.29.1   tcp    55708      https
dec2  lo0   192.168.29.8   192.168.29.1   tcp    32795      ftp
dec2  lo0   192.168.29.8   192.168.29.1   tcp    32796      www-http
dec2  lo0   192.168.29.8   192.168.29.1   tcp    32797      https
dec2  lo0   192.168.29.8   192.168.29.1   tcp    1050       https
dec2  lo0   192.168.29.8   192.168.29.1   tcp    39343      ftp
dec2  lo0   192.168.29.8   192.168.29.1   tcp    39343      https
dec2  lo0   192.168.29.8   192.168.29.1   tcp    39343      www-http
dec2  lo0   192.168.29.8   192.168.29.1   tcp    39343      ssh
dec2  lo0   192.168.29.8   192.168.29.1   tcp    32798      ftp
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
dec2  lo0   192.168.29.8   192.168.29.1   tcp    32799      www-http
dec2  lo0   192.168.29.8   192.168.29.1   tcp    32800      https
dec2  dec0  192.168.29.8   10.1.1.253     tcp    56821      ftp
dec2  dec0  192.168.29.8   10.1.1.253     tcp    56821      www-http
dec2  dec0  192.168.29.8   10.1.1.253     tcp    56821      https
dec2  dec0  192.168.29.8   10.1.1.253     tcp    32801      ftp
dec2  dec0  192.168.29.8   10.1.1.253     tcp    32802      www-http
dec2  dec0  192.168.29.8   10.1.1.253     tcp    32803      https
dec2  lo0   192.168.29.8   192.168.29.1   tcp    1058       https
dec2  dec0  192.168.29.8   10.1.1.253     tcp    nameserver nameserver
dec2  lo0   192.168.29.6   192.168.28.1   tcp    44137      https
dec2  lo0   192.168.29.6   192.168.28.1   tcp    44137      ftp
dec2  lo0   192.168.29.6   192.168.28.1   tcp    44137      www-http
dec2  lo0   192.168.29.6   192.168.28.1   tcp    32804      ftp
dec2  lo0   192.168.29.6   192.168.28.1   tcp    32805      www-http
dec2  lo0   192.168.29.6   192.168.28.1   tcp    32806      https
dec2  dec1  192.168.29.6   192.168.28.2   tcp    44137      https
dec2  dec1  192.168.29.6   192.168.28.2   tcp    44137      ftp
dec2  dec1  192.168.29.6   192.168.28.2   tcp    44137      www-http
dec2  dec1  192.168.29.6   192.168.28.2   tcp    32807      ftp
dec2  dec1  192.168.29.6   192.168.28.2   tcp    32808      www-http
dec2  dec1  192.168.29.6   192.168.28.2   tcp    32809      https
dec2  dec1  192.168.29.6   192.168.28.3   tcp    44137      https
dec2  dec1  192.168.29.6   192.168.28.3   tcp    44137      ftp
dec2  dec1  192.168.29.6   192.168.28.3   tcp    44137      www-http
dec2  dec1  192.168.29.6   192.168.28.3   tcp    32810      ftp
dec2  dec1  192.168.29.6   192.168.28.3   tcp    32811      www-http
dec2  dec1  192.168.29.6   192.168.28.3   tcp    32812      https
dec2  dec1  192.168.29.6   192.168.28.4   tcp    44137      https
dec2  dec1  192.168.29.6   192.168.28.4   tcp    44137      ftp
dec2  dec1  192.168.29.6   192.168.28.4   tcp    44137      www-http
dec2  dec1  192.168.29.6   192.168.28.4   tcp    32813      ftp
dec2  dec1  192.168.29.6   192.168.28.4   tcp    32814      www-http
dec2  dec1  192.168.29.6   192.168.28.4   tcp    32815      https
dec2  dec1  192.168.29.6   192.168.28.5   tcp    44137      https
dec2  dec1  192.168.29.6   192.168.28.5   tcp    44137      smtp
dec2  dec1  192.168.29.6   192.168.28.5   tcp    44138      smtp
dec2  dec1  192.168.29.6   192.168.28.5   tcp    44139      smtp
dec2  lo0   192.168.29.6   192.168.28.1   tcp    1073       https
dec2  dec1  192.168.29.6   192.168.28.5   tcp    44137      ftp
dec2  dec1  192.168.29.6   192.168.28.5   tcp    44137      www-http
dec2  dec1  192.168.29.6   192.168.28.5   tcp    44140      smtp
dec2  dec1  192.168.29.6   192.168.28.5   tcp    44141      smtp
dec2  dec1  192.168.29.6   192.168.28.5   tcp    44142      smtp
dec2  dec1  192.168.29.6   192.168.28.5   tcp    32816      ftp
dec2  dec1  192.168.29.6   192.168.28.5   tcp    32817      www-http
dec2  dec1  192.168.29.6   192.168.28.5   tcp    32818      https
dec2  lo0   192.168.29.6   192.168.28.1   icmp   ECHO
dec2  lo0   192.168.29.6   192.168.29.1   icmp   ECHO
dec2  lo0   192.168.29.6   192.168.29.1   icmp   ECHO
dec2  lo0   192.168.29.6   192.168.28.1   icmp   ECHO
dec2  lo0   192.168.29.6   10.1.1.254     icmp   ECHO
dec2  dec0  192.168.29.6   10.1.1.253     icmp   ECHO
dec2  dec0  192.168.29.6   10.1.1.253     icmp   ECHO
dec2  lo0   192.168.29.6   10.1.1.254     tcp    40155      www-http
dec2  lo0   192.168.29.6   10.1.1.254     tcp    40155      https
dec2  lo0   192.168.29.6   10.1.1.254     tcp    40155      ftp
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
dec2  lo0   192.168.29.6   10.1.1.254     tcp    32819      www-http
dec2  lo0   192.168.29.6   10.1.1.254     tcp    32820      https
dec2  lo0   192.168.29.6   10.1.1.254     tcp    1084       https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    48016      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    60759      www-http
dec0  dec1  10.1.1.253     192.168.28.3   tcp    60759      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    42399      https
dec0  dec1  10.1.1.253     192.168.28.3   tcp    42399      www-http
dec0  dec1  10.1.1.253     192.168.28.2   udp    nameserver nameserver
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.