Global Information Assurance Certification Paper

Jason R. Mitchell
SANS DC2000
July 5 - 10, 2000

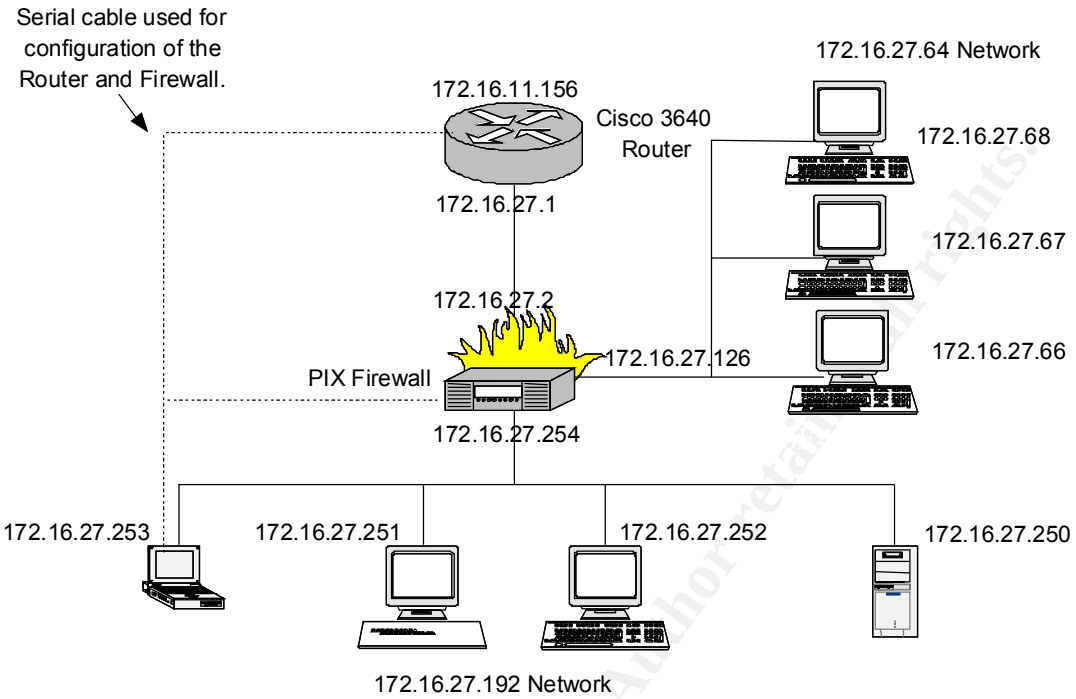
Firewall and Perimeter Protection Practical

This tutorial will explain how to implement a filtering policy on a perimeter defense solution with the following hardware: a Cisco 3640 series router running IOS 12.04(T) and a Cisco PIX 520 firewall appliance software version 5.0(3). Two hardware devices have been chosen to help illustrate a defense-in-depth solution.

By implementing defense in depth, or layering, security of the network can be exponentially increased. There are several reasons for this, one being the elimination of a single point of failure. By securing each layer as though all others have failed, any misconfigurations or exploits "du-jour" will be less likely to penetrate completely through. Defense in depth also gives more time when under attack. The attacker has to penetrate multiple layers to compromise the internal network, which promotes situational awareness. By seeing the attack on the outer edges of the network, the administrator can take necessary action to negate the threat

When using a layered approach it is important to have each device complement and work with the others. For example, the main job of the router is to route packets. Make sure that the router isn't replicating the firewall rule base. With this is mind, care will be taken in selecting which device should be responsible for the filtering of each rule. In general, the router is used to limit the flow of traffic onto the internal network to that which truly belongs there.

The network example that will be used here consists of a private class C address space (172.16.27.xxx) with a 255.255.255.192 netmask, giving us four class C subnets with 62 hosts per subnet. Following is a picture of our network with addresses:



The 172.16.27.156 gives connectivity to the internet via a switch. The interfaces on the router are fastethernet1/0 (address 172.15.11.156) and ethernet2/0 (address 172.16.27.1). The interfaces on the firewall are called outside (address 172.16.27.2), screened (172.16.27.64 network), and inside (172.16.27.192 network). The hostnames of the router and firewall are perimeter and pixfirewall respectively.

Both the router and firewall are command line driven, and the filters are applied to the router and firewall via a serial cable (as shown above).

The rest of the tutorial will explain each rule of the filtering policy to be implemented, why it is important, and show how to apply that filter independently of the other rules. At the end, we will bring it all together and put the entire policy into a ruleset on the router and firewall.

1. To begin, we will discuss the importance of blocking "spoofed" addresses into (ingress filtering) or out of (egress filtering) a network. Also, the importance of blocking source routed packets will be reviewed. These filters will be handled by the router.

Introduction

"IP Address Spoofing" involves changing or disguising your IP address, and is one of the most common types of attack building blocks. Spoofing is used by an attacker who doesn't want to have his or her actions traced. IP Address Spoofing can also be used to undermine applications, especially those that rely only on IP address for authentication. Some examples of tools that support IP address authentication are the

UNIX "r-commands" (e.g., rlogin, rsh, rcp, etc.). By blocking incoming packets that have a source address found in your internal network address space, an attacker doesn't have the chance to exploit any tools that support IP address authentication.

You should also implement a filter that blocks outbound spoofing, or egress filtering, as well as source routed packets. This is one method of reducing the risk of unknowingly participating in a Distributed Denial of service (DDoS) attack. This will not protect your own network from these types of attacks, but by ensuring that your routers forward only IP packets that have the correct Source IP Address for your network, you help prevent your systems from being used to damage other networks

Syntax of the ingress filter

Type the following commands at the prompt shown.

perimeter> **ena**ble
Password:

perimeter# **conf**igure **t**erminal

perimeter (config)# **int**erface **fa**stethernet**1/0**

perimeter (config-if)# ip access-group 10 in
perimeter(config-if)#exit

perimeter(config)# access-list 10 deny 172.16.27.0 0.0.0.255 log
perimeter(config)# access-list 10 deny 10.0.0.0 0.255.255.255 log
perimeter(config)# access-list 10 deny 192.168.0.0 0.0 255.255 log
perimeter(config)# access-list 10 deny 127.0.0.0 0.255.255.255 log
perimeter(config)# access-list 10 permit any

perimeter(config)#^Z

perimeter

Description of each of the parts of the filter

In the above example, we use access-list 10 - a Cisco **standard** access-list. A standard access list is chosen since all we need to look at is the source IP address of each packet. This is applied to interface fastethernet1/0 of the router. This access-list will be used to filter packets as they arrive from systems on the internet destined for the private network.

This access-list is specified as **in**bound. This will perform the filtering immediately at the router's input, which allows the router to drop offending packets as early as possible in the packet stream, avoiding unnecessary processing.

All denied attempts are logged for administrative action.

(Note: This access-list blocks only RFC 1918 private addresses and 127.x.x.x network addresses. For a more complete list of reserved addresses that should be blocked please refer to http://www.iana.org/ipaddress/ip-addresses.htm. Also, the whois service is very useful. The main American server is located at whois.arin.net. Other servers include whois.ripe.net (European IP addresses), whois.apnic.net (Asia-Pacific addresses). You can also query whois.nic.mil and whois.nic.gov to find military and governmental address allocations respectively.)

The access-list is comprised of the following lines:

Line	Description
access-list 10 deny 172.16.27.0 0.0.0.255 log	Specifies that the router will deny any packet whose source address is "172.16.27.x". The wildcard "0.0.0.255" specifies that the first three address octets are to be checked. Logs attempt of private address coming in.
access-list 10 deny 10.0.0.0 0.255.255.255 log	Specifies that the router will deny any packet whose source address is "10.x.x.x". The wildcard "0.255.255.255" specifies that only the first octet is to be checked. Logs attempt of private address coming in.
access-list 10 deny 192.168.0.0 0.0.255.255 log	Specifies that the router will deny any packet whose source address is "192.168.x.x". The wildcard "0.0.255.255" specifies that the first two octets will be checked. Logs attempt of private address coming in.
access-list 10 deny 127.0.0.0 0.255.255.255 log	Specifies that the router will deny any packet whose source address is "127.x.x.x". The wildcard "0.255.255.255" specifies that only the first octet is to be checked. Logs attempt of private address coming in.
access-list 10 permit any	Specifies that any packet not specifically dropped above is permitted through. This line is needed since Cisco routers implicitly deny all when an access-list is created.

Explain how to test the filter

Since this filter logs all dropped packets, testing is only a matter of generating packets on an external machine with a source IP address falls in the private (RFC 1918) or network (127.x.x.x) allocated address space. You could use a freely available tool, like trash (which can be found at http://packetstorm.securify.com), that allows source address spoofing and send these packets to your network. You can check that the router is blocking these packets through the log, or you can run an IDS like SNORT (http://www.snort.org) to see that the packets do not go by the router.

Syntax of the egress filter

perimeter> **ena**ble
Password:

perimeter# **conf**igure **t**erminal

perimeter (config)# **int**erface **eth**ernet**2/0**

perimeter (config-if)# ip access-group 11 in
perimeter(config-if)#exit

perimeter(config)# access-list 11 permit 172.16.27.0 0.0.0.255 log
perimeter(config)# access-list 10 deny any log

perimeter(config)#^Z

perimeter

Description of each of the parts of the filter

In the above example, we use access-list 11 - a Cisco **standard** access-list. A standard access list is chosen since all we need to look at is the source IP address of each packet. This is applied to interface fastethernet2/0 of the router. This access-list will be used to filter packets as they arrive from systems on the private Class C network.

This access-list is specified as **in**bound. This will perform the filtering immediately at the router's input, which allows the router to drop offending packets as early as possible in the packet stream, avoiding unnecessary processing.

(Note: For more information on using routers to protect yourself, and a more complete list of address to block with egress filters, go to http://www.sans.org/dosstep/index.htm.)

The access-list is comprised of the following lines:

Line	Description
access-list 11 permit 172.16.27.0 0.0.0.255	Specifies that the router will pass all packets whose source address is "172.16.27.x". The wildcard "0.0.0.255" specifies that the first three octets will be checked.
access-list 11 deny any log	Specifies that any packet not permitted will be explicitly dropped and logged for administrative review.

Explain how to test the filter

Since this filter logs all dropped packets, testing is only a matter of generating packets on an internal machine with a source IP address that does **not** begin with 172.16.27 and a destination address of an external host. To do this, temporarily change an internal host's ip to one that doesn't begin with 172.16.27, or use a freely available tool that can generate spoofed source addresses for packets.

Syntax of source route filter

perimeter>**ena**ble
Password:

perimeter#no ip source-route

Description of each of the parts of the filter

Source routing allows a user to re-route packets to another system upon arrival to a remote location. This is a method that can be employed to deliver harmful packets to destinations that may not be reached normally due to access lists.

This filter consists of the one line:

Line	Description
no ip source-route	Disables the router's ability to route packets with pre-defined routes.

Explain how to test the filter

To test this, you need to use a tool that allows you to specify the route of a packet. Send this packet to your router and ensure that it is dropped. For testing purposes, you could add the "log" command to the end of the line so you can see in the log that the packet was actually dropped.

Example log

Ingress Anti-spoof

```
04:34:28: %SEC-6-IPACCESSLOGDP: list 110 denied icmp 10.1.1.1 ->
172.16.27.251 (0/0), 1 packet
04:34:45: %SEC-6-IPACCESSLOGDP: list 110 denied icmp 172.16.27.3 ->
172.16.27.251 (0/0), 1 packet
04:34:55: %SEC-6-IPACCESSLOGDP: list 110 denied icmp 192.168.1.1 ->
172.16.27.251 (0/0), 1 packet
04:35:07: %SEC-6-IPACCESSLOGDP: list 110 denied icmp 127.0.3.7 ->
172.16.27.251 (0/0), 1 packet
```

Egress Anti-spoof

```
04:35:19: %SEC-6-IPACCESSLOGDP: list 110 denied icmp 172.10.3.4 ->
172.16.11.152 (0/0), 1 packet
04:35:33: %SEC-6-IPACCESSLOGDP: list 110 denied icmp 172.16.3.4 ->
172.16.11.152 (0/0), 1 packet
04:35:47: %SEC-6-IPACCESSLOGDP: list 110 denied icmp 172.10.27.4 ->
172.16.11.152 (0/0), 1 packet
04:35:59: %SEC-6-IPACCESSLOGDP: list 110 denied icmp 172.16.26.4 ->
172.16.11.152 (0/0), 1 packet
```

(Note: It is worth mentioning here that the PIX 520 firewall has egress anti-spoof filters built in. This came as a bit of a surprise when I started to test my router's access list and got no logs on the router saying the outbound spoofing had been blocked.)

2. The next part of our security policy deals with Login services. The services covered here will be telnet (23/tcp), SSH (22/tcp), FTP (21/tcp), NetBIOS (139/tcp), rlogin et al (512/tcp, 513/tcp, 514/tcp).

Introduction

Login services are a vulnerability to any network. By allowing remote logins to your network, you open the door for attackers to begin. Firstly, you should block all login requests from the internet to your internal network at the router. One exception would be the FTP protocol. If your manage a publicly accessible FTP server, then you will need to allow incoming FTP traffic to your server. In our example, we do not have an FTP server; therefore we will not allow it through.

Also, with the firewall, you should block your internal users from attempting to login to the public servers found on your DMZ/screened subnet. Here also, you should block login attempts from your servers to internal hosts. This is needed to ensure that even if a server were compromised, there is no way to login through the firewall to your internal network.

Syntax of filter for router

```
perimeter>ena
Password:

perimeter#conf t

perimeter(config)#int fa1/0

perimeter(config-if)#ip access-group 110 in
perimeter(config-if)#exit

perimeter(config)#access-list 110 deny tcp any any eq 23 log
perimeter(config)# access-list 110 deny tcp any any eq 22 log
perimeter(config)# access-list 110 deny tcp any any eq 21 log
perimeter(config)# access-list 110 deny tcp any any eq 139 log
perimeter(config)# access-list 110 deny tcp any any eq 512 log
perimeter(config)# access-list 110 deny tcp any any eq 513 log
perimeter(config)# access-list 110 deny tcp any any eq 514 log
perimeter(config)# access-list 110 permit ip any any
perimeter(config)#^Z

perimeter#
```

Description of each of the parts of the filter

This access-list is created and applied inbound to interface fastethernet1/0. This is to ensure packets are dropped as early as possible to avoid unnecessary computation time on them. An **extended** access-list is used so that we may specify port and protocol checking.

Line	Description
access-list 110 deny tcp any any eq 23 log	Specifies that the router will drop all packets using tcp protocol destined for port 23 and logs attempts to do so for review.
access-list 110 deny tcp any any eq 22 log	Specifies that the router will drop all packets using tcp protocol destined for port 22 and logs attempts to do so for review.
access-list 110 deny tcp any any eq 21 log	Specifies that the router will drop all packets using tcp protocol destined for port 21 and logs attempts to do so for review.
access-list 110 deny tcp any any eq 139 log	Specifies that the router will drop all packets using tcp protocol destined for port 139 and logs attempts to do so for review.
access-list 110 deny tcp any any eq 512 log	Specifies that the router will drop all packets using tcp protocol destined for port

	512 and logs attempts to do so for review.
access-list 110 deny tcp any any eq 513 log	Specifies that the router will drop all packets using tcp protocol destined for port 513 and logs attempts to do so for review.
access-list 110 deny tcp any any eq 514 log	Specifies that the router will drop all packets using tcp protocol destined for port 514 and logs attempts to do so for review.
access-list 110 permit ip any any	Specifies that any packet not specifically dropped above is permitted through. This line is needed since Cisco routers implicitly deny all when an access-list is created.

Explain how to test the filter

The above can be tested by attempting to open a connection from outside the router to a machine inside the router on one of the blocked ports. One way to accomplish this would be to attempt to connect to each of the listed ports. Each attempt should be blocked by the router and will show up in the router's logs.

There is a great freeware tool available to test these filters - nmap. You can find nmap at http://www.insecure.org/nmap/. Nmap allows you to attempt tcp connections to specific ports as well as udp port (specific) scans. This is very useful for testing udp specific filters. All the following tests can be done with nmap, and nmap is used in our tests.

Example Log

```
05:22:11: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(12856)
-> 172.16.27.251(23), 1 packet
05:22:25: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(15425)
-> 172.16.27.251(22), 1 packet
05:22:40: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(12387)
-> 172.16.27.251(21), 1 packet
05:22:51: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(11568)
-> 172.16.27.251(139), 1 packet
05:23:07: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(12856)
-> 172.16.27.251(512), 1 packet
05:23:25: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(15425)
-> 172.16.27.251(513), 1 packet
05:23:33: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(12387)
-> 172.16.27.251(514), 1 packet
```

Syntax of the filter on the firewall (screened interface)

The following rules apply to the screened interface (this example places the public Web, Mail and DNS servers off a third interface of the firewall, not in front of it This

makes it a true screened subnet and not just a DMZ). Following these rules will be the ruleset for the internal network.

```
pixfirewall>ena ble
Password:

pixfirewall#conf igure t erminal

pixfirewall(config)# outbound 10 deny 0 0 23 tcp
pixfirewall(config)# outbound 10 deny 0 0 22 tcp
pixfirewall(config)# outbound 10 deny 0 0 21 tcp
pixfirewall(config)# outbound 10 deny 0 0 139 tcp
pixfirewall(config)# outbound 10 deny 0 0 512 tcp
pixfirewall(config)# outbound 10 deny 0 0 513 tcp
pixfirewall(config)# outbound 10 deny 0 0 514 tcp
pixfirewall(config)# apply (screened) 10 outgoing_src
pixfirewall(config)#^Z

pixfirewall#write mem ory
pixfirewall#
```

Description of each of the parts of the filter

The general syntax for outbound filters on the PIX follows this format:

outbound *List_ID* **permit|deny** *ip_address netmask port(range) protocol*

(Note: all following **outbound** *rules follow this syntax notation)*

0 0 can be substituted for 0.0.0.0 0.0.0.0 (ip address and netmask) to mean all hosts on the network.

The syntax for the **apply** command is as follows:

apply (*if_name*) *List_ID* **outgoing_src|outgoing_dest**

(Note: all following **apply** *commands follow this syntax notation)*

Line	Description
outbound 10 deny 0 0 23 tcp	Specifies the firewall will deny packets with tcp protocol on port 23
outbound 10 deny 0 0 22 tcp	Specifies the firewall will deny packets with tcp protocol on port 22
outbound 10 deny 0 0 21 tcp	Specifies the firewall will deny packets with tcp protocol on port 21
outbound 10 deny 0 0 139 tcp	Specifies the firewall will deny packets with tcp protocol on port 139
outbound 10 deny 0 0 512 tcp	Specifies the firewall will deny packets

	with tcp protocol on port 512
outbound 10 deny 0 0 513 tcp	Specifies the firewall will deny packets with tcp protocol on port 513
outbound 10 deny 0 0 514 tcp	Specifies the firewall will deny packets with tcp protocol on port 514
apply (screened) 10 outgoing_src	Specifies the firewall will apply the above rules outbound on the screened interface while checking the *ip_address* and *netmask* against the source address. So, any host on the screened subnet sending to any of the above ports on the tcp protocol will have their packets dropped.

Explain how to test the filter

To test this filter, try to connect to the blocked ports from a machine on your screened subnet to both a) machines on the internet, and b) machines on your internal network. Make sure the firewall is dropping the packets by checking your firewall logs.

Syntax of the filter on the firewall (inside interface)

```
pixfirewall>ena
Password:

pixfirewall#conf t

pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 23 tcp
pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 22 tcp
pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 21 tcp
pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 139 tcp
pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 512 tcp
pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 513 tcp
pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 514 tcp
pixfirewall(config)# apply (inside) 15 outgoing_dest
pixfirewall(config)#^Z

pixfirewall#write mem
pixfirewall#
```

Description of each of the parts of the filter (inside interface)

Line	Description
outbound 15 deny 172.16.27.64	Specifies the firewall will deny packets

255.255.255.192 23 tcp	with tcp protocol on port 23
outbound 15 deny 172.16.27.64 255.255.255.192 22 tcp	Specifies the firewall will deny packets with tcp protocol on port 22
outbound 15 deny 172.16.27.64 255.255.255.192 21 tcp	Specifies the firewall will deny packets with tcp protocol on port 21
outbound 15 deny 172.16.27.64 255.255.255.192 139 tcp	Specifies the firewall will deny packets with tcp protocol on port 139
outbound 15 deny 172.16.27.64 255.255.255.192 512 tcp	Specifies the firewall will deny packets with tcp protocol on port 512
outbound 15 deny 172.16.27.64 255.255.255.192 513 tcp	Specifies the firewall will deny packets with tcp protocol on port 513
outbound 15 deny 172.16.27.64 255.255.255.192 514 tcp	Specifies the firewall will deny packets with tcp protocol on port 514
apply (inside) 15 outgoing_dest	Specifies the firewall will apply the above rules outbound on the inside interface while checking the *ip_address* and *netmask* against the destination address. So, any host on the inside interface sending to any host on the screened interface, on the above ports, on the tcp protocol, will have their packets dropped.

Explain how to test the filter

Again, you can test this filter by attempting a connection on the given ports initiated from the internal network to the screened subnet. Check the firewall logs to see the dropped packets.

3. The next section will deal with RPC and NFS. The following are the ports to be filtered: Portmap/rpcbind (111/tcp and 111/udp), NFS (2049/tcp and 2049/udp) and lockd (4045/tcp and 4045/udp).

Introduction

Portmapper, which provides a service to translate port numbers for RPC services, has many weaknesses. One of which being, it allows remote users to register/unregister services on a remote host by way of forging UDP packets. An attacker can utilize this to gain increased access to the local machine. An example attack involves unregistering a service from the portmapper, and then re-registering the service on a new port, which they have control over. This allows an attacker to impersonate security critical services and gain increased access to the network.

Syntax of the filter for the router

```
perimeter>ena
Password:

perimeter#conf t

perimeter(config)# int fa1/0

perimeter(config-if)# ip access-group 110 in
perimeter(config-if)#exit

perimeter(config)# access-list 110 deny tcp any any eq 111 log
perimeter(config)# access-list 110 deny udp any any eq 111 log
perimeter(config)# access-list 110 deny tcp any any eq 2049 log
perimeter(config)# access-list 110 deny udp any any eq 2049 log
perimeter(config)# access-list 110 deny tcp any any eq 4045 log
perimeter(config)# access-list 110 deny udp any any eq 4045 log
perimeter(config)# access-list 110 permit ip any any
perimeter(config)#^Z

perimeter#
```

Description of each of the parts of the filter

Line	Description
access-list 110 deny tcp any any eq 111 log	Specifies that the router will drop all packets using tcp protocol destined for port 111 and logs attempts to do so for review.
access-list 110 deny udp any any eq 111 log	Specifies that the router will drop all packets using udp protocol destined for port 111 and logs attempts to do so for review.
access-list 110 deny tcp any any eq 2049 log	Specifies that the router will drop all packets using tcp protocol destined for port 2049 and logs attempts to do so for review.
access-list 110 deny udp any any eq 2049 log	Specifies that the router will drop all packets using udp protocol destined for port 2049 and logs attempts to do so for review.
access-list 110 deny tcp any any eq 4045 log	Specifies that the router will drop all packets using tcp protocol destined for port 4045 and logs attempts to do so for

	review.
access-list 110 deny udp any any eq 4045 log	Specifies that the router will drop all packets using udp protocol destined for port 4045 and logs attempts to do so for review.
access-list 110 permit ip any any	Specifies that any packet not specifically dropped above is permitted through. This line is needed since Cisco routers implicitly deny all when an access-list is created.

Explain how to test the filter

This filter can be tested by attempting a connection through the router on the above protocols (tcp **and** udp) and on the listed ports.

Example Log

```
05:24:13: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(35984)
-> 172.16.27.251(111), 1 packet
05:24:22: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(16847) -> 172.16.27.251(111), 1 packet
05:24:34: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(11236)
-> 172.16.27.251(2049), 1 packet
05:24:45: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(21847) -> 172.16.27.251(2049), 1 packet
05:24:57: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(11236)
-> 172.16.27.251(4045), 1 packet
05:25:10: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(21847) -> 172.16.27.251(4045), 1 packet
```

Syntax of the filter for the firewall (screened interface)

```
pixfirewall>ena
Password:

pixfirewall#conf t

pixfirewall(config)# outbound 10 deny 0 0 111 tcp
pixfirewall(config)# outbound 10 deny 0 0 111 udp
pixfirewall(config)# outbound 10 deny 0 0 2049 tcp
pixfirewall(config)# outbound 10 deny 0 0 2049 udp
pixfirewall(config)# outbound 10 deny 0 0 4045 tcp
pixfirewall(config)# outbound 10 deny 0 0 4045 udp
pixfirewall(config)# apply (screened) 10 outgoing_src
pixfirewall(config)#^Z
```

pixfirewall#write mem
pixfirewall#

Description of each of the parts of the filter

Line	Description
outbound 10 deny 0 0 111 tcp	Specifies the firewall will deny packets with tcp protocol on port 111
outbound 10 deny 0 0 111 udp	Specifies the firewall will deny packets with udp protocol on port 111
outbound 10 deny 0 0 2049 tcp	Specifies the firewall will deny packets with tcp protocol on port 2049
outbound 10 deny 0 0 2049 udp	Specifies the firewall will deny packets with udp protocol on port 2049
outbound 10 deny 0 0 4045 tcp	Specifies the firewall will deny packets with tcp protocol on port 4045
outbound 10 deny 0 0 4045 udp	Specifies the firewall will deny packets with udp protocol on port 4045
apply (screened) 10 outgoing_src	Specifies the firewall will apply the above rules outbound on the screened interface while checking the *ip_address* and *netmask* against the source address. So, any host on the screened subnet sending to any of the above ports on the tcp or udp protocol will have their packets dropped.

Explain how to test the filter

This filter is tested by attempting a tcp connection from the screened subnet to addresses in both the internal network and on the internet on the given ports. Also, do a udp port scan on the above ports to the same address. All connections and scan attempts should be blocked by the router and shown in the logs.

Syntax of the filter for the firewall (inside interface)

pixfirewall>ena
Password:

pixfirewall#conf t

pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 111 tcp
pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 111 udp

```
pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 2049 tcp
pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 2049 udp
pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 4045 tcp
pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 4045 udp
pixfirewall(config)# apply (inside) 15 outgoing_dest
pixfirewall(config)#^Z

pixfirewall#write mem
pixfirewall#
```

Description of each of the parts of the filter

Line	Description
outbound 15 deny 172.16.27.64 255.255.255.192 111 tcp	Specifies the firewall will deny packets with tcp protocol on port 111
outbound 15 deny 172.16.27.64 255.255.255.192 111 udp	Specifies the firewall will deny packets with udp protocol on port 111
outbound 15 deny 172.16.27.64 255.255.255.192 2049 tcp	Specifies the firewall will deny packets with tcp protocol on port 2029
outbound 15 deny 172.16.27.64 255.255.255.192 2049 udp	Specifies the firewall will deny packets with udp protocol on port 2029
outbound 15 deny 172.16.27.64 255.255.255.192 4045 tcp	Specifies the firewall will deny packets with tcp protocol on port 4045
outbound 15 deny 172.16.27.64 255.255.255.192 4045 udp	Specifies the firewall will deny packets with udp protocol on port 4045
apply (inside) 15 outgoing_dest	Specifies the firewall will apply the above rules outbound on the inside interface while checking the *ip_address* and *netmask* against the destination address. So, any host on the inside interface sending to any host on the screened interface, on the above ports, on the tcp or udp protocol, will have their packets dropped.

Explain how to test the filter

Test this filter by attempting tcp connections, and udp port scans, initiated from the inside interface destined for the screened subnet on the listed ports. All should be blocked by the firewall. Check the firewall logs to ensure they are.

4. The next section will deal with blocking Windows NT™ and Windows2000™ NetBIOS traffic on the network. The following ports should be blocked:

WindowsNT™ 135 (tcp and udp), 137 (udp), 138 (udp), 139 (tcp), and (for Windows2000™ - all previous ports plus the following) 445 (tcp and udp).

Introduction

Microsoft Windows™ uses NetBIOS as the default networking protocol. Misconfiguration problems are very common in NetBIOS. Users are often unaware that they have left shares unpassworded, or that they are sharing files at all. There are also known circumstances where remote users can access files that are in directories other than those that are intentionally shared.

Syntax of the filter for the router

```
perimeter>ena
Password:

perimeter#conf t

perimeter(config)#int fa1/0

perimeter(config-if)# ip access-group 110 in
perimeter(config-if)#exit

perimeter(config)# access-list 110 deny tcp any any eq 135 log
perimeter(config)# access-list 110 deny udp any any eq 135 log
perimeter(config)# access-list 110 deny udp any any eq 137 log
perimeter(config)# access-list 110 deny udp any any eq 138 log
perimeter(config)# access-list 110 deny tcp any any eq 139 log
perimeter(config)# access-list 110 deny tcp any any eq 445 log
perimeter(config)# access-list 110 deny udp any any eq 445 log
perimeter(config)# access-list 110 permit ip any any
perimeter(config)#^Z

perimeter#
```

Description of each of the parts of the filter

Line	Description
access-list 110 deny tcp any any eq 135 log	Specifies that the router will drop all packets using tcp protocol destined for port 135 and logs attempts to do so for review.
access-list 110 deny udp any any eq 135 log	Specifies that the router will drop all packets using udp protocol destined for port 135 and logs attempts to do so for

	review.
access-list 110 deny udp any any eq 137 log	Specifies that the router will drop all packets using udp protocol destined for port 137 and logs attempts to do so for review.
access-list 110 deny udp any any eq 138 log	Specifies that the router will drop all packets using udp protocol destined for port 138 and logs attempts to do so for review.
access-list 110 deny tcp any any eq 139 log	Specifies that the router will drop all packets using tcp protocol destined for port 139 and logs attempts to do so for review.
access-list 110 deny tcp any any eq 445 log	Specifies that the router will drop all packets using tcp protocol destined for port 445 and logs attempts to do so for review.
access-list 110 deny udp any any eq 445 log	Specifies that the router will drop all packets using udp protocol destined for port 445 and logs attempts to do so for review.
access-list 110 permit ip any any	Specifies that any packet not specifically dropped above is permitted through. This line is needed since Cisco routers implicitly deny all when an access-list is created.

Explain how to test the filter

The above filter can be tested by sending packets from outside the network destined internally, to the above ports on the given protocols. The router will drop these packets and log the attempt. Review the log files to ensure this.

Example Log

```
06:11:14: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(14523)
-> 172.16.27.251(135), 1 packet
06:11:25: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(16874) -> 172.16.27.251(135), 1 packet
06:11:34: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(13582) -> 172.16.27.251(137), 1 packet
06:11:41: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(12589) -> 172.16.27.251(138), 1 packet
06:11:52: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(14932)
-> 172.16.27.251(139), 1 packet
```

06:12:04: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(14687) -> 172.16.27.251(445), 1 packet
06:12:13: %SEC-6-IPACCESSLOGDP: list 110 denied udp 172.16.11.254(16324) -> 172.16.27.251(445), 1 packet

Syntax of the filter for the firewall (screened interface)

```
pixfirewall>ena
Password:

pixfirewall#conf t

pixfirewall(config)# outbound 10 deny 0 0 135 tcp
pixfirewall(config)# outbound 10 deny 0 0 135 udp
pixfirewall(config)# outbound 10 deny 0 0 137 udp
pixfirewall(config)# outbound 10 deny 0 0 138 udp
pixfirewall(config)# outbound 10 deny 0 0 139 tcp
pixfirewall(config)# outbound 10 deny 0 0 445 tcp
pixfirewall(config)# outbound 10 deny 0 0 445 udp
pixfirewall(config)# apply (screened) 10 outgoing_src
pixfirewall(config)#^Z

pixfirewall#write mem
pixfirewall#
```

Description of each of the parts of the filter

Line	Description
outbound 10 deny 0 0 135 tcp	Specifies the firewall will deny packets with tcp protocol on port 135
outbound 10 deny 0 0 135 udp	Specifies the firewall will deny packets with udp protocol on port 135
outbound 10 deny 0 0 137 udp	Specifies the firewall will deny packets with udp protocol on port 137
outbound 10 deny 0 0 138 udp	Specifies the firewall will deny packets with udp protocol on port 138
outbound 10 deny 0 0 139 tcp	Specifies the firewall will deny packets with tcp protocol on port 139
outbound 10 deny 0 0 445 tcp	Specifies the firewall will deny packets with tcp protocol on port 445
outbound 10 deny 0 0 445 udp	Specifies the firewall will deny packets with udp protocol on port 445
apply (screened) 10 outgoing_src	Specifies the firewall will apply the above rules outbound on the screened interface while checking the *ip_address* and *netmask*

	against the source address. So, any host on the screened subnet sending to any of the above ports on the given tcp or udp protocol will have their packets dropped.

Explain how to test the filter

The filter can be tested by trying to establish an outbound connection from the screened subnet on the given protocols and ports. The firewall should drop all these packets. Check the logs to ensure they were dropped.

Syntax of the filter for the firewall (inside interface)

```
pixfirewall>ena
Password:

pixfirewall# conf t

pixfirewall(config)#outbound 15 deny 172.16.27.64 255.255.255.192 135 tcp log
pixfirewall(config)#outbound 15 deny 172.16.27.64 255.255.255.192 135 udp log
pixfirewall(config)#outbound 15 deny 172.16.27.64 255.255.255.192 137 udp log
pixfirewall(config)#outbound 15 deny 172.16.27.64 255.255.255.192 138 udp log
pixfirewall(config)#outbound 15 deny 172.16.27.64 255.255.255.192 139 tcp log
pixfirewall(config)#outbound 15 deny 172.16.27.64 255.255.255.192 445 tcp log
pixfirewall(config)#outbound 15 deny 172.16.27.64 255.255.255.192 445 udp log
pixfirewall(config)#apply (inside) 15 outgoing_dest
pixfirewall(config)#^Z

pixfirewall#write mem
pixfirewall#
```

Description of each of the parts of the filter

Line	Description
outbound 15 deny 172.16.27.64 255.255.255.192 135 tcp log	Specifies the firewall will deny packets with tcp protocol on port 135
outbound 15 deny 172.16.27.64 255.255.255.192 135 udp log	Specifies the firewall will deny packets with udp protocol on port 135
outbound 15 deny 172.16.27.64 255.255.255.192 137 udp log	Specifies the firewall will deny packets with udp protocol on port 137
outbound 15 deny 172.16.27.64 255.255.255.192 138 udp log	Specifies the firewall will deny packets with udp protocol on port 138
outbound 15 deny 172.16.27.64 255.255.255.192 139 tcp log	Specifies the firewall will deny packets with tcp protocol on port 139
outbound 15 deny 172.16.27.64	Specifies the firewall will deny packets

255.255.255.192 445 tcp log	with tcp protocol on port 445
outbound 15 deny 172.16.27.64 255.255.255.192 445 udp log	Specifies the firewall will deny packets with udp protocol on port 445
apply (inside) 15 outgoing_dest	Specifies the firewall will apply the above rules outbound on the inside interface while checking the *ip_address* and *netmask* against the destination address. So, any host on the inside interface sending to any host on the screened interface, on the above ports, on the tcp or udp protocol given, will have their packets dropped.

Explain how to test the filter

To test this filter, have a machine on the internal network attempt a tcp connection or udp port scan on the listed ports to a machine in the screened subnet. The firewall should drop these packets. Check the logs to ensure they were dropped.

5. This section will talk about X Windows. The ports to be blocked are in the following range: 6000/tcp - 6255/tcp.

Introduction

By being able to connect to an X Windows server on a target host, an attacker can monitor all keystrokes and windows on the target server. In addition to monitoring, the attacker can also inject keystrokes into the target X Windows server, allowing them to execute arbitrary commands on the host. Some operating systems are shipped without any access restrictions to the X Windows server.

If you are running X Windows, you should review the security access control in your version of X windows and implement it. Use the following command to determine the current access control setting on any host needing to run X windows:

```
# xhost
access control disabled, clients can connect from any host
```

This setting provides no access control, and any user is allowed to connect to the X server. To enable access control, use this command:

```
# xhost -
access control enabled, only authorized clients can connect
```

Access control is now enabled. To specifically allow other hosts on your network

to utilize your X Windows display, use this command:

xhost +hostname

If you must run an X Windows server, you can find more information about securing X Windows at the following site:

CIAC 2316 - Securing X Windows
http://ciac.llnl.gov/ciac/documents/ciac2316.html

Syntax of the filter for the router

```
perimeter>ena
Password:

perimeter#conf t

perimeter(config)# int fa1/0

perimeter(config-if)# ip access-group 110 in
perimeter(config-if)# exit

perimeter(config)# access-list 110 deny tcp any any range 6000 6255 log
perimeter(config)# access-list 110 permit ip any any
perimeter(config)#^Z

perimeter#
```

Description of each of the parts of the filter

Line	Description
access-list 110 deny tcp any any range 6000 6255 log	Specifies that the router will drop all packets using tcp protocol destined for ports in the range 6000 - 6255 and logs attempts to do so for review.
access-list 110 permit ip any any	Specifies that any packet not specifically dropped above is permitted through. This line is needed since Cisco routers implicitly deny all when an access-list is created.

Explain how to test the filter

To test this filter, use a machine outside the router and do a tcp connection scan with the given range of ports (6000 - 6255). The router should drop all these packets. Check the router logs and ensure it did.

Example Log

```
03:03:21: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(13547)
-> 172.16.27.251(6000), 1 packet
03:03:33: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(21389)
-> 172.16.27.251(6001), 1 packet
03:03:47: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(18432)
-> 172.16.27.251(6002), 1 packet
03:03:52: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(12854)
-> 172.16.27.251(6003), 1 packet
03:04:08: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(11365)
-> 172.16.27.251(6100), 1 packet
03:04:17: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(16217)
-> 172.16.27.251(6200), 1 packet
03:04:30: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(13472)
-> 172.16.27.251(6225), 1 packet
03:04:42: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(11267)
-> 172.16.27.251(6255), 1 packet
```

(Note: For obvious reasons I didn't test the entire range. I sufficed it enough to test the beginning few ports, a few in the middle the the lat one in the range to ensure that they were all covered.)

Syntax of the filter for the firewall (screened interface)

There should never be a reason that any machine on your screened interface should be attempting an X Window connection. Therefore, don't allow traffic from the screened interface on these ports.

```
perimeter>ena
Password:

perimeter#conf t

perimeter(config)# outbound 10 deny 0 0 6000-6255 tcp
perimeter(config)# apply (screened) 10 outgoing_src
perimeter(config)#^Z

perimeter#write mem
perimeter#
```

Description of each of the parts of the filter

Line	Description
outbound 10 deny 0 0 6000-6255 tcp	Specifies the firewall will deny packets with tcp protocol on the range of ports between 6000 - 6255
apply (screened) 10 outgoing_src	Specifies the firewall will apply the above rules outbound on the screened interface while checking the *ip_address* and *netmask* against the source address. So, any host on the screened subnet sending to any of the above ports on the tcp protocol will have their packets dropped.

Explain how to test the filter

To test this filter, use a machine on your dmz and attempt a tcp connection scan on the range of ports listed above. All these packets should be dropped by the firewall and logged. Check the logs to ensure that the packets are being dropped.

Syntax of the filter for the firewall (inside interface)

This may be an issue if your users need access to external X Windows servers. You should check your site's policy on this. In this example, we are only blocking X connections from our internal network to our screened network. Following is the syntax for this:

```
pixfirewall>ena
Password:

pixfirewall#conf t

pixfirewall(config)# outbound 15 deny 172.16.27.64 255.255.255.192 6000-6255
tcp
pixfirewall(config)# apply (inside) 15 outgoing_dest
pixfirewall(config)#^Z

pixifirewall#write mem
pixfirewall#
```

Description of each of the parts of the filter

Line	Description
outbound 15 deny 172.16.27.64 255.255.255.192 6000-6255 tcp	Specifies the firewall will deny packets with tcp protocol on the range of ports

	between 6000 - 6255
apply (inside) 15 outgoing_dest	Specifies the firewall will apply the above rules outbound on the inside interface while checking the *ip_address* and *netmask* against the destination address. So, any host on the inside interface sending to any host on the screened interface, on the above ports, on the tcp protocol, will have their packets dropped.

Explain how to test the filter

To test this filter, do a tcp connection attempt on the range of ports listed above from the internal network to the screened network. The firewall should block all these attempts. Check the logs to make ensure this.

6. This section will deal with naming services. More specifically, we want to block DNS (53/udp) to all machines which are not DNS servers, DNS zone transfers (53/tcp) except from external secondaries, and LDAP (389/tcp and 389/udp).

Introduction

One of the best resources a determined attacker has to use against you are your DNS records. You should limit the amount of information that remote users can gather via DNS. As a rule remote users have no reason to have your zone maps. You should configure DNS not to honor zone transfers. You should also be aware that there are many misconfiguration issues with DNS that can lead to security compromises.

One good security implementation worth mentioning here is "split DNS". This involves having two separate nameservers for the domain. You could place an "external" DNS server in the screened subnet, and an "internal" DNS server in your protected network. The external server would only have a few hosts in its database - the address of the firewall, the external web server, the external mail server, and any other publicly available servers you may have on site. This keeps would be attackers from having access to detailed information about the mapping of your internal, protected network. With split DNS, even if the external DNS server is compromised, an attacker cannot learn your internal mapping.

Your internal DNS is where all internal hosts point. This server contains all internal IP mapping. It may also contain the external database as well, and does all recursive DNS lookups for internet systems.

For information on LDAP, visit http://www.umich.edu/~dirsvcs/ldap/doc/.

Syntax of the filter for the router

```
perimeter>ena
Password:

perimeter#conf t

perimeter(config)#int fa1/0

perimeter(config-if)# ip access-group 110 in
perimeter(config-if)#exit

perimeter(config)# access-list 110 deny tcp any any eq 389 log
perimeter(config)# access-list 110 deny udp any any eq 389 log
perimeter(config)# access-list 110 permit udp any host 172.16.27.66 eq 53
perimeter(config)# access-list 110 deny udp any any eq 53 log
perimeter(config)# access-list 110 permit tcp x.x.x.x host 172.16.27.66 eq 53
perimeter(config)# access-list 110 deny tcp any any eq 53 log
perimeter(config)# access-list permit ip any any
perimeter(config)#^Z

perimeter#
```

(Note: x.x.x.x in the above access-list line should be replaced by **your** *secondary DNS server's address. Check with the network administrator if you don't know this address.)*

Description of each of the parts of the filter

Line	Description
access-list 110 deny tcp any any eq 389 log	Specifies that the router will drop all packets using tcp protocol destined for port 389 and logs attempts to do so for review.
access-list 110 deny udp any any eq 389 log	Specifies that the router will drop all packets using udp protocol destined for port 389 and logs attempts to do so for review.
access-list 110 permit udp any host 172.16.27.66 eq 53	Specifies that the router will permit access from any host to the external DNS server only on port 53 udp
access-list 110 deny udp any any eq 53 log	Specifies that the router will deny any host trying to connect to any internal host on port 53 udp. This **must** follow the previous rule in order to allow DNS requests only to the DNS server. Also logs attempts to access other machines on port 53 udp for review.
access-list 110 permit tcp host x.x.x.x host	Specifies that the router will permit access

172.16.27.66 eq 53	from a specific host (x.x.x.x) to the external DNS server only on port 53 tcp. This would allow for a secondary name server to do a zone transfer from the external DNS server.
access-list 110 deny tcp any any eq 53 log	Specifies that the router will deny any host trying to connect to any internal host on port 53 tcp. This **must** follow the previous rule in order to allow zone transfers only to authorized secondaries. This also logs attempts at others trying to do a zone transfer.
access-list permit ip any any	Specifies that any packet not specifically dropped above is permitted through. This line is needed since Cisco routers implicitly deny all when an access-list is created.

Explain how to test the filter

To test the LDAP filters, attempt a tcp connection scan and a udp port scan on port 389. The router's logs should show you the denied packets. Also, attempt to do DNS lookups and zone transfers from an external machine. The queries should be let through ok, but the zone transfer should be blocked. Make sure your router log shows a denied packet on tcp 53. To attempt a zone transfer, from a UNIX machine outside your network, get type 'nslookup' at the prompt, followed by 'set type=any'. Next type 'ls -d target_network_name'.

Example Log

```
07:09:56: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(15463)
-> 172.16.27.251(389), 1 packet
07:10:05: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(26547) -> 172.16.27.251(389), 1 packet
07:10:18: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(15484) -> 172.16.27.251(53), 1 packet
07:10:33: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(13268)
-> 172.16.27.251(53), 1 packet
```

Syntax of the filter for the firewall (screened interface)

For this filter you need to open the firewall to accept the udp traffic associated with DNS. You need to allow outgoing traffic on both tcp and udp for requests and zone transfers (limited to authorized secondary servers). This introduces a new command on

the PIX we have not talked about yet - the **conduit** command. The **conduit** command is used for sending traffic into the firewall. Here are the commands to implement DNS traffic from the screened interface to the internet.

```
pixfirewall>ena
Password:

pixfirewall#conf t

pixfirewall(config)# conduit permit udp host 172.16.27.66 eq 53 any
pixfirewall(config)# outbound 10 permit 172.16.27.66 255.255.255.192 53 udp
pixfirewall(config)# outbound 11 permit x.x.x.x         xxx.xxx.xxx.xxx 53 tcp
pixfirewall(config)# apply (screened) 10 outgoing_src
pixfirewall(config)# apply (screened) 11 outgoing_dest
pixfirewall(config)#^Z

pixfirewall#write mem
pixfirewall#
```

(Note: your secondary DNS server's address and netmask should replace x.x.x.x and xxx.xxx.xxx.xxx in the above outbound line. Check with the network administrator if you don't know this address.)

Description of each of the parts of the filter

Line	Description
conduit permit udp host 172.16.27.66 eq 53 any	Specifies that the firewall will permit in only traffic for the DNS server (host 172.16.27.66) on port 53 udp
outbound 10 permit 172.16.27.66 255.255.255.192 53 udp	Specifies that the firewall will permit out traffic on port 53 udp
outbound 11 permit x.x.x.x xxx.xxx.xxx.xxx 53 tcp	Specifies that the firewall will permit out traffic on port 53 tcp
apply (screened) 10 outgoing_src	Specifies the firewall will apply rule 10 above outbound on the screened interface while checking the *ip_address* and *netmask* against the source address. So, any external host can query our external DNS server.
apply (screened) 11 outgoing_dest	Specifies the firewall will apply rule 11 above outbound on the screened interface while checking the *ip_address* and *netmask* against the destination address. So, only authorized secondaries are allowed to perform zone transfers.

Explain how to test the filter

Doing simple DNS querys will verify that this rule works. To test the ability to only send your zone map to your external secondary, you could make the ip rule in outbound 11 above be the address of a different machine you have. Attempt the zone transfer from the machine and see if it dumps the map. Also, attempt the zone transfer from a different machine to ensure it is getting dropped to all but the external secondary and logged.

Syntax for the filter for the firewall (inside interface)

The only reason there should be DNS traffic between the inside interface and the screened interface is for internal hosts to look up addresses not on the internal network. Therefore, the only traffic that should be passing through is udp traffic on port 53.

```
pixfirewall>ena
Password:

pixfirewall# conf t

pixfirewall(config)# conduit permit udp host 172.16.27.194 eq 53 host
172.16.27.66 gt 1023
pixfirewall(config)# outbound 15 permit 172.16.27.194 255.255.255.192 53 udp
pixfirewall(config)# apply (inside) 15 outgoing_dest
pixfirewall(config)^Z

pixfirewall#write mem
pixfirewall#
```

(Note: only in the newer versions of bind, 8.x.x, does a requesting DNS server use an ephemeral port, above 1023. In older versions of bind, 4.x.x, both servers would use port 53)

Description of each of the parts of the filter

Line	Description
conduit permit udp host 172.16.27.194 eq 53 host 172.16.27.66 gt 1023	Specifies that the firewall will permit incoming udp traffic on port 53, from a port greater than 1023, **only** between the two DNS servers (external and internal)
outbound 15 permit 172.16.27.194 255.255.255.192 53 udp	Specifies that the firewall will permit outgoing traffic on port 53 udp
apply (inside) 15 outgoing_dest	Specifies the firewall will apply rule 15 above outbound on the inside interface while checking the *ip_address* and *netmask*

	against the destination address. So, the internal DNS server can only talk to the external DNS server when name resolution needs to be done.

Explain how to test the filter

Again, to test this filter, have a machine on your internal network do a DNS query and make sure it goes through. Also, by temporarily setting a test machine to the ip of the internal DNS, you can check to ensure that only udp traffic is allowed on and only on port 53. Check the firewall logs to verify this.

7. This section will deal with mail protocols. Specifically, we will block SMTP (25/tcp) to all machines which are not external mail relays, POP (109/tcp and 110/tcp), and IMAP (143/tcp).

Introduction

The mail protocols are plagued with vulnerabilities. One of the most common SMTP vulnerabilities is the MIME buffer overflow attack which is a weakness that will allow intruders root access. More information on this can be found at
CERT Advisory CA-96.24.sendmail.daemon.mode
http://www.cert.org/advisories/CA-96.24.sendmail.daemon.mode.html

A vulnerability has also been reported in some versions of the Internet Message Access Protocol (IMAP) and Post Office Protocol (POP). Attackers exploiting this vulnerability can obtain unauthorized root access. For more information, visit http://ciac.llnl.gov/ciac/bulletins/h-46a.shtml.

Syntax of the filter for the router

```
perimeter>ena
Password:

perimeter# conf t

perimeter(config)#int fa1/0

perimeter(config-if)# ip access-group 110 in
perimeter(config-if)# exit

perimeter(config)# access-list 110 permit tcp any eq 25 host 172.16.27.67 eq 25
perimeter(config)# access-list 110 deny tcp any any eq 25 log
```

```
perimeter(config)# access-list 110 deny tcp any any eq 109 log
perimeter(config)# access-list 110 deny tcp any any eq 110 log
perimeter(config)# access-list 110 deny tcp any any eq 143 log
perimeter(config)# access-list permit ip any any
perimeter(config)#^Z

perimeter#
```

Description of each of the parts of the filter

Line	Description
access-list 110 permit tcp any eq 25 host 172.16.27.67 eq 25	Specifies that the router will permit traffic from any host to the external mail server on port 25 tcp.
access-list 110 deny tcp any any eq 25 log	Specifies that the router will deny any host trying to connect to any internal host on port 25 tcp. This **must** follow the previous rule in order to allow mail to only external mail relays. This also logs attempts at others trying to hit a machine that does not serve mail, or the mail port on other machines.
access-list 110 deny tcp any any eq 109 log	Specifies that the router will drop all packets destined for any machine on port 109 tcp and logs the attempt for review.
access-list 110 deny tcp any any eq 110 log	Specifies that the router will drop all packets destined for any machine on port 110 tcp and logs the attempt for review.
access-list 110 deny tcp any any eq 143 log	Specifies that the router will drop all packets destined for any machine on port 143 tcp and logs the attempt for review.
access-list permit ip any any	Specifies that any packet not specifically dropped above is permitted through. This line is needed since Cisco routers implicitly deny all when an access-list is created.

Explain how to test the filter

To test this filter, attempt to send tcp traffic to a) the mail server on port 25, and b) any other machine on port 25. You should only see packets dropped that were destined for machines other than the mail server. Also, send packets to any machine on the other ports listed (109, 110, and 143 - all tcp). These should also be blocked. Check the router's logs to ensure this.

Example Log

```
06:56:24: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(12657)
-> 172.16.27.251(25), 1 packet
06:56:40: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(24236)
-> 172.16.27.251(109), 1 packet
06:56:53: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(13527)
-> 172.16.27.251(110), 1 packet
06:57:06: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(19214)
-> 172.16.27.251(143), 1 packet
```

Syntax of the filter for the firewall (screened interface)

```
pixfirewall>ena
Password:

pixfirewall# conf t

pixfirewall(config)# conduit permit tcp host 172.16.27.67 eq 25 any eq 25
pixfirewall(config)# outbound 10 permit 172.16.27.67 255.255.255.192 25 tcp
pixfirewall(config)# apply (screened) 10 outgoing_src
pixfirewall(config)# ^Z

pixfirewall#write mem
pixfirewall#
```

Description of each of the parts of the filter

Line	Description
conduit permit tcp host 172.16.27.67 eq 25 any eq 25	Specifies that the firewall will permit incoming traffic to the mail server on port 25 from any host on port 25
outbound 10 permit 172.16.27.67 255.255.255.192 25 tcp	Specifies that the firewall will permit outgoing traffic on port 25 tcp
apply (screened) 10 outgoing_src	Specifies the firewall will apply rule 10 above outbound on the screened interface while checking the *ip_address* and *netmask* against the source address. So, external mail server can send mail to any other mail server.

Explain how to test the filter

You can test this filter by attempting to send out of the screened subnet on port 25 tcp from a machine other than the mail server. This packet should be blocked and will

appear in the firewall's logs. Also, sending packets to the firewall from the outside to a host other than the mail server will also cause a log trap to be generated.

Syntax of the filter for the firewall (inside interface)

```
pixfirewall>ena
Password:

pixfirewall# conf t

pixfirewall(config)# conduit permit tcp host 172.16.27.195 host 172.16.27.67 eq
25
pixfirewall(config)# outbound 15 permit 172.16.27.67 255.255.255.192 25 tcp
pixfirewall(config)# apply (inside) 15 outgoing_dest
pixfirewall(config)# ^Z

pixfirewall#write mem
pixfirewall#
```

Description of each of the parts of the filter

Line	Description
conduit permit tcp host 172.16.27.195 host 172.16.27.67 eq 25	Specifies that the firewall will permit mail traffic from the external mail relay to the internal mail server on tcp port 25
outbound 15 permit 172.16.27.195 255.255.255.192 25 tcp	Specifies that the firewall will permit outbound tcp traffic on port 25
apply (inside) 15 outgoing_dest	Specifies the firewall will apply rule 15 above outbound on the inside interface while checking the *ip_address* and *netmask* against the destination address. So, the internal mail server can only talk to the external mail relay on port 25 tcp.

Explain how to test the filter

To test this filter, have a machine in the screened subnet (other than the mail relay) attempt the send a packet through the firewall on port 25. Also, have a machine on the internal network attempt the send a packet to a machine on the screened subnet (other than the mail relay) on port 25 tcp.

8. Now we will discuss web traffic. This will include blocking HTTP (80/tcp) and SSL (443/tcp) except to external web servers. Also, high-order port choices should be blocked (e.g. 8000/tcp, 8080/tcp, 8888/tcp).

Introduction

HTTP and SSL traffic for the web are open to many vulnerabilities. Most involve some sort of buffer overflow allowing root compromise. Some of these involve CGI scripts or certain versions of some web servers, from IIS to Apache.

Syntax of the filter for the router

```
perimeter>ena
Password:

perimeter# conf t

perimeter(config)#int fa1/0

perimeter(config-if)# ip access-group 110 in
perimeter(config-if)# exit

perimeter(config)# access-list 110 permit tcp any host 172.16.27.68 eq 80
perimeter(config)# access-list 110 permit tcp any host 172.16.27.68 eq 443
perimeter(config)# access-list 110 deny tcp any any eq 80 log
perimeter(config)# access-list 110 deny tcp any any eq 443 log
perimeter(config)# access-list 110 deny tcp any any eq 8000 log
perimeter(config)# access-list 110 deny tcp any any eq 8080 log
perimeter(config)# access-list 110 deny tcp any any eq 8888 log
perimeter(config)# access-list 110 permit ip any any
perimeter(config)# ^Z

perimeter#
```

Description of each of the parts of the filter

Line	Description
access-list 110 permit tcp any host 172.16.27.68 eq 80	Specifies that the router will permit tcp packets from anywhere to the external web server on port 80
access-list 110 permit tcp any host 172.16.27.68 eq 443	Specifies that the router will permit tcp packets from anywhere to the external web server on port 443
access-list 110 deny tcp any any eq 80 log	Specifies that the router will drop all packets on tcp port 80 for any machine on

	the network. This **must** follow the above rule(s) in order to allow port 80 traffic only to the external web server.
access-list 110 deny tcp any any eq 443 log	Specifies that the router will drop all packets on tcp port 443 for any machine on the network. This **must** follow the above rule(s) in order to allow port 443 traffic only to the external web server.
access-list 110 deny tcp any any eq 8000 log	Specifies that the router will drop all packets destined for tcp port 8000 and logs the attempt for review.
access-list 110 deny tcp any any eq 8080 log	Specifies that the router will drop all packets destined for tcp port 8080 and logs the attempt for review.
access-list 110 deny tcp any any eq 8888 log	Specifies that the router will drop all packets destined for tcp port 8888 and logs the attempt for review.
access-list 110 permit ip any any	Specifies that any packet not specifically dropped above is permitted through. This line is needed since Cisco routers implicitly deny all when an access-list is created.

Explain how to test the filter

You would test this filter by attempting a tcp connection on the above ports from outside your network to the inside. The router should block all packets except those that are destined for port 80 tcp or 443 tcp at the web server's address. Check the router's logs to ensure this.

Example Log

```
04:43:12: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(28501)
-> 172.16.27.251(80), 1 packet
04:43:25: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(23298)
-> 172.16.27.251(443), 1 packet
04:43:38: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(21654)
-> 172.16.27.251(8000), 1 packet
04:43:54: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(13517)
-> 172.16.27.251(8080), 1 packet
04:44:14: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(31265)
-> 172.16.27.251(8888), 1 packet
```

Syntax of the filter for the firewall (screened interface)

```
pixfirewall>ena
Password:
```

```
pixfirewall# conf t

pixfirewall(config)# conduit permit tcp host 172.16.27.68 eq 80 any
pixfirewall(config)# outbound 10 permit 172.16.27.68 255.255.255.192 80
pixfirewall(config)# apply (screened) 10 outgoing_src
pixfirewall(config)# ^Z

pixfirewall#write mem
pixfirewall#
```

Description of each of the parts of the filter

Line	Description
conduit permit tcp host 172.16.27.68 eq 80 any	Specifies that the firewall will permit web traffic from the from the internet to the external mail server on tcp port 80
outbound 10 permit 172.16.27.68 255.255.255.192 80	Specifies that the firewall will permit outbound traffic from the web server on port 80
apply (screened) 10 outgoing_src	Specifies the firewall will apply rule 10 above outbound on the screened interface while checking the *ip_address* and *netmask* against the source address. So, external web server can talk to the internet on tcp port 80.

Explain how to test the filter

To test the filter (independent of the router), place a machine between the firewall and the router. Attempt to connect to the web server on a port other than 80 from this machine. This will ensure the conduit is set up correctly, as long as the firewall blocks these connection attempts. Also, have the web server (or another machine with the web servers address) attempt to send packets out on a port other than 80. This will ensure the outbound rule is set up correctly.

Syntax of the filter for the firewall (inside interface)

This filter does not apply to the inside interface of the firewall. There are internal web servers set up so people never have to use the public web servers. Also, remote logins to the web server are not allowed, i.e. maintaining the public web server is done locally at the machine. This ensures the web server is as secure as possible and not prone to vulnerabilities associated with misconfiguring remote access.

9. Here we will discuss blocking the "small services". This includes blocking ports below 20/tcp and 20/udp, along with time - 37/tcp and 37/udp.

Introduction

There are some services, called small services, that are used for things like echo, character generation and discarding data that are hardly ever used. They exist on both tcp and udp protocols. You should disable these services to prevent an undiscovered vulnerability from being able to penetrate the network.

One example of a vulnerability associated with these small services is the echo/chargen packet flood attack.

There is a service designed to generate a stream of characters called the character generator (chargen), and was designed primarily for testing purposes. Remote users/intruders are able to abuse this service, thereby exhausting system resources.

Spoofed network sessions that appear to come from that local system's echo service can be pointed at the chargen service to form a "loop." This session will cause huge amounts of data to be passed in an endless loop that causes heavy load to the system.

Pointing this spoofed session at a remote system's echo service will cause heavy network traffic/overhead that considerably slows your network down.

It should be noted that an attacker does not need to be on your subnet to achieve this attack. The source addresses for the services can be forged with ease.

If at all possible, you should comment chargen out of /etc/inetd.conf. This was designed for debugging tcp in its earlier stages of development and should not be needed on your hosts.

The lines in the /etc/inetd.conf are:

```
chargen      stream  tcp    nowait  root    internal
chargen      dgram   udp    wait    root    internal
```

To comment these line out, place a # character as the first character of each line and restart the inetd daemon. To ensure that the service is not listening, run the 'netstat' command. You can also find more information about this specific type of attack at http://www.cert.org/advisories/CA-96.01.UDP_service_denial.html , and denial of service attacks in general at the following link - http://www.sans.org/infosecFAQ/dos.htm

This filter will only be applied to the incoming interface of the router. Your security policy may also dictate that users do not need access to these services, but implementing the rules on the firewall is beyond the scope of this perimeter defense solution.

Syntax of the filter for the router

```
perimeter>ena
Password:

perimeter# conf t

perimeter(config)#int fa1/0

perimeter(config-if)# ip access-group 110 in
perimeter(config-if)#exit

perimeter(config)# access-list 110 deny tcp any any lt 20 log
perimeter(config)# access-list 110 deny udp any any lt 20 log
perimeter(config)# access-list 110 deny tcp any any eq 37 log
perimeter(config)# access-list 110 deny udp any any eq 37 log
perimeter(config)# access-list 110 permit ip any any
perimeter(config)# ^Z

perimeter#
```

Description of each of the parts of the filter

Line	Description
access-list 110 deny tcp any any lt 20 log	Specifies that he router will drop all packets on the tcp protocol with a destination port less than 20 and logs for review
access-list 110 deny udp any any lt 20 log	Specifies that the router will drop all packets on the udp protocol with a destination port less than 20 and logs for review
access-list 110 deny tcp any any eq 37 log	Specifies that the router will drop all packets on the tcp protocol with a destination port of 37 and logs for review
access-list 110 deny udp any any eq 37 log	Specifies that the router will drop all packets on the udp protocol with a destination port of 20 and logs for review
access-list 110 permit ip any any	Specifies that any packet not specifically dropped above is permitted through. This line is needed since Cisco routers implicitly deny all when an access-list is created.

Explain how to test the filter

Testing this filter involves running a tcp connection scan on the range of ports under 20 and on port 37. Also, you can do a udp port scan on the same ports. All these packets should be dropped by the router and show up in the logs.

Example Log

```
01:21:11: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(23544)
-> 172.16.27.251(7), 1 packet
01:21:24: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(32157) -> 172.16.27.251(7), 1 packet
01:21:38: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(15487)
-> 172.16.27.251(19), 1 packet
01:21:51: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(12354) -> 172.16.27.251(19), 1 packet
01:22:09: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(20165)
-> 172.16.27.251(37), 1 packet
01:22:17: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(20168) -> 172.16.27.251(37), 1 packet
```

(Note: I only tested the echo and chargen ports below 20 to prove that the rule was working, and since this is the specific vulnerability I discussed above.)

10. Next, we will discuss blocking miscellaneous ports that may pose security problems. These include TFTP (69/udp), finger (79/tcp), NNTP (119/tcp), NTP(123/tcp), LPD (515/tcp), syslog (514/udp), SNMP (161/tcp, 161/udp, 162/tcp, 162/udp), BGP (179/tcp), and SOCKS (1080/tcp).

Introduction

All the items in this list are commonly probed services for an attacker. I will talk about a few of the most common vulnerabilities in this list.

An attacker can create arbitrary files on the target system by utilizing the TFTP service. If not configured properly, TFTP will allow remote users to create, or overwrite critical system files such as the password and shadow password file.

You should review the security settings on your TFTP daemon. TFTP daemons sometimes support a "-s" flag, which allows the specification of a directory to which requests are limited. If you do not need the TFTP service, you should disable it in the /etc/inetd.conf configuration file. For more information about TFPT, you can visit -

CERT Advisory CA-91.18.Active.Internet.tftp.Attacks
http://www.cert.org/advisories/CA-91.18.Active.Internet.tftp.Attacks.html
CERT Advisory CA-91.19.AIX.TFTP.Daemon.vulnerability
http://www.cert.org/advisories/CA-91.19.AIX.TFTP.Daemon.vulnerability.html
CIAC Advisory ciac-05.unix-holes

http://www.ja.net/CERT/CIAC/fy89/ciac-05.unix-holes
CIAC Advisory b-44.ciac-automated-tftp-probes
http://www.ja.net/CERT/CIAC/b-fy91/b-44.ciac-automated-tftp-probes
AUSCERT Advisory AA-93.05.tftp.Attacks
ftp://ftp.auscert.org.au/pub/auscert/advisory/AA-93.05.tftp.Attacks

"Finger" is an online information service that provides data about users on a system. The information provided by "finger" is frequently sensitive, and can be used by an attacker to focus attacks more effectively, by monitoring who uses the system and how they use it. The finger service can be a major vulnerability as it provides quite a lot of information to outsiders such as names and phone numbers of users, user home directory and login shell, time a user has been idle, when a user last read e-mail, and the remote host that a user is logged in from and so on.

Not only can finger be used to reveal possibly private or sensitive information, some of the information may be used by an attacker to make inferences about trust relationships between hosts on your network, collect usernames for password guessing attempts, obtain phone numbers for "social engineering" attacks and monitor the activity on your system.

Unless you require a finger daemon running, you should disable it by editing your /etc/inetd.conf configuration file and commenting out the appropriate line, then restart inetd.

Many Network Time Protocol (NTP) servers offer detailed information about their setup which may include systems which they peer with. Other information that can be obtained via NTP includes the following: system time statistics (uptime), system IO statistics, system memory statistics, and time daemon peer listing.

Ensure that the NTP server only allows authorized users to obtain critical setup information.

An SNMP enabled device can provide an attacker with access to a wide variety of information. This information ranges from the type and model of the device, to active network connections, processes running on the host, and users logged into the host.

Write access on an SNMP device provides an attacker with the ability to alter networking and other device parameters. An attacker can alter the routing and arp tables, bring network interfaces up and down, enable or disable packet forwarding and alter several other networking parameters. Also, vendor extensions may provide other control parameters that an attacker can manipulate. This can lead to denial of service or the compromise of security or confidential information.

These filters will only be handled by the router. There may be circumstances where you will want to block some, or all, of these services internally, but for the purpose of this tutorial they are allowed in the internal network.

Syntax of the filter for the router

```
perimeter>ena
Password:

perimeter# conf t

perimeter(config)#int fa1/0

perimeter(config-if)# ip access-group 110 in
perimeter(config-if)# exit

perimeter(config)# access-list 110 deny udp any any eq 69 log
perimeter(config)# access-list 110 deny tcp any any eq 79 log
perimeter(config)# access-list 110 deny tcp any any eq 119 log
perimeter(config)# access-list 110 deny tcp any any eq 123 log
perimeter(config)# access-list 110 deny tcp any any eq 515 log
perimeter(config)# access-list 110 deny udp any any eq 514 log
perimeter(config)# access-list 110 deny tcp any any eq 161 log
perimeter(config)# access-list 110 deny udp any any eq 161 log
perimeter(config)# access-list 110 deny tcp any any eq 162 log
perimeter(config)# access-list 110 deny udp any any eq 162 log
perimeter(config)# access-list 110 deny tcp any any eq 179 log
perimeter(config)# access-list 110 deny tcp any any eq 1080 log
perimeter(config)# access-list 110 permit ip any any
perimeter(config)# ^Z

perimeter#
```

Description of each of the parts of the filter

Line	Description
access-list 110 deny udp any any eq 69 log	Specifies that the router will drop all packets destined for udp port 69 and logs the attempt for review.
access-list 110 deny tcp any any eq 79 log	Specifies that the router will drop all packets destined for tcp port 79 and logs the attempt for review.
access-list 110 deny tcp any any eq 119 log	Specifies that the router will drop all packets destined for tcp port 119 and logs the attempt for review.
access-list 110 deny tcp any any eq 123 log	Specifies that the router will drop all packets destined for tcp port 123 and logs the attempt for review.
access-list 110 deny tcp any any eq 515 log	Specifies that the router will drop all packets destined for tcp port 515 and logs the attempt for review.
access-list 110 deny udp any any eq 514 log	Specifies that the router will drop all

	packets destined for udp port 514 and logs the attempt for review.
access-list 110 deny tcp any any eq 161 log	Specifies that the router will drop all packets destined for tcp port 161 and logs the attempt for review.
access-list 110 deny udp any any eq 161 log	Specifies that the router will drop all packets destined for udp port 161 and logs the attempt for review.
access-list 110 deny tcp any any eq 162 log	Specifies that the router will drop all packets destined for tcp port 162 and logs the attempt for review.
access-list 110 deny udp any any eq 162 log	Specifies that the router will drop all packets destined for udp port 162 and logs the attempt for review.
access-list 110 deny tcp any any eq 179 log	Specifies that the router will drop all packets destined for tcp port 179 and logs the attempt for review.
access-list 110 deny tcp any any eq 1080 log	Specifies that the router will drop all packets destined for tcp port 1080 and logs the attempt for review.
access-list 110 permit ip any any	Specifies that any packet not specifically dropped above is permitted through. This line is needed since Cisco routers implicitly deny all when an access-list is created.

Explain how to test the filter

Test these filtering rules by attempting tcp connections, and udp port scans, on the specified ports above. All attempts should be blocked and logged. Check the router's log to ensure that it has been.

Example Log

```
02:14:41: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(17546) -> 172.16.27.251(69), 1 packet
02:14:54: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(16217)
-> 172.16.27.251(79), 1 packet
02:15:10: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(23571)
-> 172.16.27.251(119), 1 packet
02:15:25: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(31527)
-> 172.16.27.251(123), 1 packet
02:15:37: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(19521)
-> 172.16.27.251(515), 1 packet
02:15:51: %SEC-6-IPACCESSLOGDP: list 110 denied udp
172.16.11.254(10236) -> 172.16.27.251(514), 1 packet
```

02:16:06: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(19821) -> 172.16.27.251(161), 1 packet
02:16:20: %SEC-6-IPACCESSLOGDP: list 110 denied udp 172.16.11.254(26542) -> 172.16.27.251(161), 1 packet
02:16:36: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(36851) -> 172.16.27.251(162), 1 packet
02:16:51: %SEC-6-IPACCESSLOGDP: list 110 denied udp 172.16.11.254(18542) -> 172.16.27.251(162), 1 packet
02:17:04: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(25410) -> 172.16.27.251(179), 1 packet
02:17:18: %SEC-6-IPACCESSLOGDP: list 110 denied tcp 172.16.11.254(10341) -> 172.16.27.251(1080), 1 packet

11. Lastly, we discuss blocking ICMP messages. This includes blocking incoming echo request (ping and Windows traceroute), outgoing echo replies, time exceeded, and unreachable massages.

Introduction

ICMP gives a would be attacker more information about your network than you want. By pinging you and your address space, the attacker can determine what IP numbers are alive, thus giving him/her a target. Also, by using the ICMP error messages (time to live exceeded and destination unreachable) an attacker can determine open ports through your firewall. This technique is called firewalking. Firewalk allows an attacker to determine what ports are open on a packet filtering firewall. It is based on the traceroute program which utilizes the IP TTL field to determine the path of a packet. For more information about firewalk, go to http://packetstorm.securify.com/UNIX/audit/firewalk

Knowing which ports are open through your firewall is very useful information to an attacker. Each open port offers a possible entry point into your network.

By blocking the ICMP error messages and echo request and echo reply, you severely limit the information an attacker can find out about your site. There is a price for blocking these messages. Your users will not have the ability to ping/traceroute. You should check your security policy to determine if this is acceptable.

Syntax of the filter for the router

perimeter>ena
Password:

perimeter#conf t

```
perimeter(config)#int fa1/0

perimeter(config-if)# ip access-group 110 in
perimeter(config-if)# exit

perimeter(config)# access-list 110 deny icmp any any log
perimeter(config)# no ip unreachables
perimeter(config)# ^Z

perimeter#
```

Description of each of the parts of the filter

Line	Description
access-list 110 deny icmp any any log	Specifies that the router will drop any ICMP traffic coming into the network and log for review.
no ip unreachables	Prevents the router from giving out network information based on ICMP error messages.

Explain how to test the filter

To test this filter, simply try to ping through the router. The router's logs will show evidence of a working filter.

Example Log

```
09:42:28: %SEC-6-IPACCESSLOGDP: list 110 denied icmp 172.16.11.254 ->
172.16.27.251 (0/0), 1 packet
09:42:44: %SEC-6-IPACCESSLOGDP: list 110 denied icmp 172.16.11.254 ->
172.16.27.251 (0/0), 1 packet
```

Bringing it all together

The purpose of this tutorial was to give you an idea of how to implement a filtering policy effectively to block some of the most commonly probed and attacked services and ports. Each filter was discussed separately, and the syntax given to implement that filter independently of any others. In this final section, we will pull all the rules together into a common ruleset for this security policy.

Syntax for the router

```
perimeter>ena
```

Password:

perimeter# conf t

perimeter(config)# access-list 110 deny ip 172.16.27.0 0.0.0.255 any log
perimeter(config)# access-list 110 deny ip 10.0.0.0 0.255.255.255 any log
perimeter(config)# access-list 110 deny ip 192.168.0.0 0.0 255.255 any log
perimeter(config)# access-list 110 deny ip 127.0.0.0 0.255.255.255 any log

perimeter(config)# access-list 110 deny tcp any any eq 23 log
perimeter(config)# access-list 110 deny tcp any any eq 22 log
perimeter(config)# access-list 110 deny tcp any any eq 21 log
perimeter(config)# access-list 110 deny tcp any any eq 139 log
perimeter(config)# access-list 110 deny tcp any any eq 512 log
perimeter(config)# access-list 110 deny tcp any any eq 513 log
perimeter(config)# access-list 110 deny tcp any any eq 514 log

perimeter(config)# access-list 110 deny tcp any any eq 111 log
perimeter(config)# access-list 110 deny udp any any eq 111 log
perimeter(config)# access-list 110 deny tcp any any eq 2049 log
perimeter(config)# access-list 110 deny udp any any eq 2049 log
perimeter(config)# access-list 110 deny tcp any any eq 4045 log
perimeter(config)# access-list 110 deny udp any any eq 4045 log

perimeter(config)# access-list 110 deny tcp any any eq 135 log
perimeter(config)# access-list 110 deny udp any any eq 135 log
perimeter(config)# access-list 110 deny udp any any eq 137 log
perimeter(config)# access-list 110 deny udp any any eq 138 log
perimeter(config)# access-list 110 deny tcp any any eq 445 log
perimeter(config)# access-list 110 deny udp any any eq 445 log

perimeter(config)# access-list 110 deny tcp any any range 6000 6255 log

perimeter(config)# access-list 110 deny tcp any any eq 389 log
perimeter(config)# access-list 110 deny udp any any eq 389 log
perimeter(config)# access-list 110 permit udp any host 172.16.27.66 eq 53
perimeter(config)# access-list 110 deny udp any any eq 53 log
perimeter(config)# access-list 110 permit tcp x.x.x.x host 172.16.27.66 eq 53
perimeter(config)# access-list 110 deny tcp any any eq 53 log

perimeter(config)# access-list 110 permit tcp any eq 25 host 172.16.27.67 eq 25
perimeter(config)# access-list 110 deny tcp any any eq 25 log
perimeter(config)# access-list 110 deny tcp any any eq 109 log
perimeter(config)# access-list 110 deny tcp any any eq 110 log
perimeter(config)# access-list 110 deny tcp any any eq 143 log

```
perimeter(config)# access-list 110 permit tcp any host 172.16.27.68 eq 80
perimeter(config)# access-list 110 permit tcp any host 172.16.27.68 eq 443
perimeter(config)# access-list 110 deny tcp any any eq 80 log
perimeter(config)# access-list 110 deny tcp any any eq 443 log
perimeter(config)# access-list 110 deny tcp any any eq 8000 log
perimeter(config)# access-list 110 deny tcp any any eq 8080 log
perimeter(config)# access-list 110 deny tcp any any eq 8888 log

perimeter(config)# access-list 110 deny tcp any any lt 20 log
perimeter(config)# access-list 110 deny udp any any lt 20 log
perimeter(config)# access-list 110 deny tcp any any eq 37 log
perimeter(config)# access-list 110 deny udp any any eq 37 log

perimeter(config)# access-list 110 deny udp any any eq 69 log
perimeter(config)# access-list 110 deny tcp any any eq 79 log
perimeter(config)# access-list 110 deny tcp any any eq 119 log
perimeter(config)# access-list 110 deny tcp any any eq 123 log
perimeter(config)# access-list 110 deny tcp any any eq 515 log
perimeter(config)# access-list 110 deny udp any any eq 514 log
perimeter(config)# access-list 110 deny tcp any any eq 161 log
perimeter(config)# access-list 110 deny udp any any eq 161 log
perimeter(config)# access-list 110 deny tcp any any eq 162 log
perimeter(config)# access-list 110 deny udp any any eq 162 log
perimeter(config)# access-list 110 deny tcp any any eq 179 log
perimeter(config)# access-list 110 deny tcp any any eq 1080 log

perimeter(config)# access-list 110 deny icmp any any log
perimeter(config)# no ip unreachables

perimeter(config)#access-list 110 permit tcp any any established


perimeter(config)# access-list 11 permit 172.16.27.0 0.0.0.255 log
perimeter(config)# access-list 10 deny any log

perimeter(config)#int fa1/0

perimeter(config-if)#ip access-group 110 in
perimeter(config-if)#exit

perimeter(config)# int eth2/0

perimeter(config-if)#ip access-group 115 in
perimeter(config-if)#exit

perimeter#
```

This above list of commands will implement the entire filtering policy for the router. As you can see there are quite a few rules in this list. The more rules in the list, the more processing must be done on a packet, especially if it is to be permitted and has to pass every rule in the list. A much faster implementation on the router would be to only permit the things to your network that you truly wanted there, and have a default deny at the end - rather than the permit. It would definitely make the ruleset smaller and easier to handle. Then, you have the firewall do whatever filtering you wish on the content of the packets that make it through.

This condensed ruleset would look something like this:

```
perimeter>ena
Password:

perimeter# conf t

perimeter(config)# access-list 110 deny ip 172.16.27.0 0.0.0.255 any log
perimeter(config)# access-list 110 deny ip 10.0.0.0 0.255.255.255 any log
perimeter(config)# access-list 110 deny ip 192.168.0.0 0.0 255.255 any log
perimeter(config)# access-list 110 deny ip 127.0.0.0 0.255.255.255 any log

perimeter(config)# access-list 110 permit udp any host 172.16.27.66 eq 53
perimeter(config)# access-list 110 permit tcp x.x.x.x host 172.16.27.66 eq 53
perimeter(config)# access-list 110 permit tcp any eq 25 host 172.16.27.67 eq 25
perimeter(config)# access-list 110 permit tcp any host 172.16.27.68 eq 80
perimeter(config)# access-list 110 permit tcp any host 172.16.27.68 eq 443
perimeter(config)# access-list 110 permit tcp any any established
perimeter(config)# access-list 110 deny ip any any log

perimeter(config)# access-list 10 deny 172.16.27.0 0.0.0.255 log
perimeter(config)# access-list 10 deny any log

perimeter(config)# int fa1/0

perimeter(config-if)# ip access-group 110 in
perimeter(config-if)# exit

perimeter(config)# int eth2/0

perimeter(config-if)# ip access-group 10 in
perimeter(config-if)# exit
```

Following now are a few things used to harden the router. This is a good idea since the router is sitting out there with no protection.

```
perimeter(config)# service password-encryption
perimeter(config)# enable secret
perimeter(config)# no service tcp-small-servers
perimeter(config)# no service udp-small-servers
perimeter(config)# no service finger
perimeter(config)# no ip unreachables
perimeter(config)# no ip direct-broadcast
perimeter(config)# no ip bootp server
perimeter(config)# no ip http server
perimeter(config)# no ip source-route
perimeter(config)# no snmp
perimeter(config)# logging <log host ip address>
perimeter(config)# banner / WARNING: Authorized Access Only /

perimeter(config)# ^Z

perimeter#
```

(Note: We still leave the anti-spoof rules in so we can log when someone is trying to spoof an address)

(Note: x.x.x.x in the above access-list line should be replaced by **your** *secondary DNS server's address. Check with the network administrator if you don't know this address.)*

(Note: The established command above, in bold, is to have the router allow return connections on the tcp protocol. If this command were not there, we would have to massage the list with rules in order for a tcp connection to hold.)

This list will only allow in the traffic that really belongs on the network and denies all other traffic and logs it for review. The firewall ruleset from above need not be changed because of this. The firewall is keeping traffic between the screened subnet and internal network in check, limiting what can be done internally. The router rules were repeated here merely to illustrate the difference in the lists and show an example of how much simpler the access control lists can be made in the router by reviewing what really needs to be let through. Just by only allowing in the network what services are strictly needed, the list was cut from sixty-three lines to just thirteen.

For more information about configuring and using perimeter routers please go to
http://www.geek-speak.net/paperws/access-lists.html
or
http://pasadena.net/cisco/secure.html