# Global Information Assurance Certification Paper

Author: Paul M. Tiedemann
Date: 8/8/00
Firewall and Perimeter Protection

The following 11 sections for this assignment will be demonstrated using a Redhat Linux 6.1 operating system using IPChains as the packet-filtering device. I have made the assumption that eth0 is the external interface. For each section I will begin with a brief explanation why the particular services should be blocked followed by how the services work and how I can block these services using the IPChains rules. I will also try to include some helpful hints I have picked up along the way.

## Exercise I – Blocking spoofed or private (RFC1918) addresses.

There are a few packets we should never see on the firewall. These packets do not occur naturally in the wild. These consist of spoofed packets claiming to be coming from an internal address, unroutable packets destined for your internal network that that should have never made it past the prior router, and various other packets that exist only to cause pain and suffering for the firewall administrator.

You should always block packets from entering your network if the source or destination appears to be a private (RFC1918) address. They never should have been routed to you in the first place. Their very existence means that some other network administrator hasn't done a very good job at egress filtering. However, only the destination address seems to be filtered for private addresses in the real world. What this means is that it is quite likely you will see some packets that seem to be originating from a private address and are addressed to your internal network. These packets only have two origins, the first being malicious spoofing and the second, being a very confused network administrator who doesn't understand network address translation. Both are unacceptable and should be filtered at your perimeter. Here is the rules list for adding the rules to the IPChains rule set. Note: I am using the –b flag indicating that these are bi-directional, which means the rule will match packets either to or from the IP address range specified.

```
#  rules for standard unroutables
ipchains –A input –i eth0 –s 255.255.255.255/32 –b –j  DENY
ipchains –A input –i eth0 –s 127.0.0.0/8 –b –j  DENY

# rules for private (RFC1918) addresses
ipchains –A input –i eth0 –s 10.0.0.0/8 –b –j DENY
ipchains –A input –i eth0 –s 172.16.0.0/12 –b –j DENY
ipchains –A input –i eth0 –s 192.168.0.0/16 –b –j DENY

#rule for reserved addresses
ipchains –A input –i eth0 –s 240.0.0.0/5 –b –j DENY

# rule for protecting internal network from spoofing
ipchains –A input –i eth0 –s (insert internal network here) –j –l DENY
```

Now here is a short explanation of what these rules mean:

ipchains – this is the command to interact with the firewall
-A input– this option means to append a rule to the end of the named chain in this case the input chain
-i eth0 – this the interface to apply the rule against (in this case my external ethernet interface)
-s – this option indicates the source address used for filtering
-b – this option indicates the rule is bi-directional meaning it applies to packets from or to the address specified
-l – this option means to log the packet to the syslog facility
-j DENY – this option indicates the target for the packet in this case it says to silently drop the packet

Note: I have only used the logging option in the last rule in order to avoid logging numerous packets that are not by themselves a cause for concern. If more granular details were needed all of the rules can be logged by simply adding the –l option. Also make sure you have two network interfaces or the spoofing rule will block all outbound client attempts.

IP packets can contain one or more options in the options field in the IP header. These options are supposed to be used by application to get more control over how the packet is delivered across the network. In the wild these options are rarely used and their presence can be an indicator of a crafted packet (a packet created by hand instead of one generated by an application). There are two options that are known to be particularly problematic. They are "lsrr" and "ssrr" which stand for loose source routing and strict source routing. These are bad and should be blocked by the firewall and logged. For more information on source routing see http://www.nationwide.net/~aleph1/FAQ and search on the page for source routing. Now on to blocking them at the firewall, since Linux does not implement this feature in the firewall they are implemented as a kernel option. When compiling the kernel the option CONFIG_IP_NOSR must be turned on and the following line needs to added to the firewall script or typed at the command line.

```
echo 0 > /proc/sys/net/ipv4/conf/all/accept_source_route
and
echo 0 > /proc/sys/net/ipv4/conf/default/accept_source_route
```

The first line sets a general policy and the second sets a default that will be applied to all interfaces after this line is executed. I highly recommend you add these lines to the beginning of your firewall script.

After applying these rules you should always make sure you test them. The easiest way to test them is to use the built in testing rule in IPChains. If you use a C (for check) instead of a A (for append) IPChains will tell you right away whether you are on the right track. An example of this is the following command after applying the rules listed above:

```
ipchains –C input –i eth0 –p tcp –s 192.168.0.1/32 25 –d 0.0.0.0/0 25
```

This should come back with a resounding denied after you enter the command. There are a few simple rules that are common to the –C command. You have to enter a chain, a protocol, a source address, a destination address, and a port for each address. While you may be testing for general rules that don't involve a protocol or port the test should still apply. A good example is that although I haven't denied port 25 in any of the rules above it will still apply because it meets the rule stated above:

```
ipchains –A input –i eth0 –s 192.168.0.0/16 –b –j DENY
```

This rule simply states that a 192.168.0.0 address is to be blocked regardless of protocol or specific port.

Another good tool to use is the tried and tested nmap tool by fyodor. This tool can be found at http://www.insecure.org/nmap/. It is available for most operating systems and is the most popular scanning tool available today. You can direct a number of tests at a network to determine the port status of any machine on the Internet. Also it can be used to predict the type of operating system in use. You can run this test after your firewall is in place to check for open ports that you meant to block. It also allows you to spoof the source address, which will allow you to test for spoofed address protection. There is one caveat with this tool. It should be run from a location outside your network to give the most accurate information. This will require someone to test the network from a position outside the firewall. If you have a laptop you can simply plug in a hub to the incoming network connection on your external router and give the laptop an appropriate IP address. When running the test you are looking to make sure the ports you are testing come back with a blocked status. Also note that you must disable the ping test first or nmap will believe that the IP address in question is not being used. Below we will be blocking incoming ping requests so this will be very important.

One more note is that there are a number of packet crafting tools available to give a tester granular control over all parts of a TCP/UDP/ICMP packet. These tools allow for tests that not only test ports but also options such as the don't fragment flag or other types of IP options. For more information on what can be done when crafting packets please go to http://www.packetfactory.net/libnet/.

## Exercise II – Blocking Login services

Login services are often blocked by firewalls because of their inherent security issues. All of these services allow direct interactions with another unknown or untrusted source. However these issues must be balanced with the requirements of the organization. The telnet protocol is used to gain remote access to another operating system. It uses tcp port 23 for all connection attempts to a server. While there is nothing inherently evil with this protocol if allowed into your network hackers have an easy access point and if they manage to gain a password they can wreak havoc. This protocol also has no means available to encrypt the information passed from device to device and is therefore susceptible to sniffing for passwords. The SSH protocol is an enhanced telnet type protocol. It uses tcp port 22 for all connection attempts to the server. It has built in encryption for the payloads as well as a secure authentification method. It should always be used in place of telnet whenever the organization has a need to allow remote access through the firewall. This program has had a few security problems in the past and an administrator would be wise to stay on top of program updates. FTP is a remote file transfer protocol. It uses tcp port 21 for initial connection attempts and tcp port 20 for all data transfer requests. It allows one remote system to access files located on another system. This protocol has probably had more security problems than any other single application. For this reason if FTP is a requirement for the organization this should be placed on a server located on a different subnet than the internal protected network. Only under protest should a firewall administrator allow ftp access from the outside world into your protected network. NetBIOS is Microsoft's implementation for file and print sharing. It uses tcp ports 135-139 and udp ports 137-138. Windows 2000 uses tcp port 445. It is a non-routable protocol but can be tunneled inside TCP/IP and thus used to access remote files and printers. It is advisable to block incoming connection attempts because of the ability for someone to map your internal network by querying for various information that Microsoft servers are only to happy to provide to anyone. One such problem is unprotected network shares. If just one person in your network has an open share (not protected by a password) it is trivial for someone to gain access to that share and depending on operating system type can either alter or remove files. Rlogin is a Unix service that is intended to be more user friendly than telnet. It uses tcp ports 512-514 and udp port 513. It provides more emulation semantics than telnet and is thus more capable of transporting terminal characteristics across remote connections. There is also another reason that end-user's like the program because it can be set up to avoid having to use password by setting up a trust environment. This last piece of the protocol has allowed malicious people to use various means to break this trust mechanism and get remote access to remote systems. It has fallen out of favor recently due to various problems with the security and is generally being blocked by most firewall administrators. Now onto the firewall rules:

```
# rule to block incoming and outgoing telnet connections
ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 23 –l –j DENY
ipchains –A output –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 23 –l –j DENY

# rule to block incoming and outgoing SSH connections
ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 22 –l –j DENY
ipchains –A output –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 22 –l –j DENY

#rule to block incoming and outgoing FTP connections

ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 21 –l –j DENY
ipchains –A output –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 21 –l –j DENY

# rule to block incoming and outgoing WinNT 4.0 NetBIOS connections

ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 139 –l –j DENY
ipchains –A output –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 135:139 –l –j DENY
ipchains –A input –i eth0 –p udp –s 0.0.0.0/0 –d 0.0.0.0/0 137:138 –l –j DENY
ipchains –A output –i eth0 –p udp –s 0.0.0.0/0 –d 0.0.0.0/0 137:138 –l –j DENY

# rule to block incoming and outgoing Win2000 NetBios connections
```

```
ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 445 –l –j DENY
ipchains –A output –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 445 –l –j DENY

#rule to block incoming and outgoing rlogon connections

ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 512:514 –l –j DENY
ipchains –A output –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 512:514 –l –j DENY
ipchains –A input –i eth0 –p udp –s 0.0.0.0/0 –d 0.0.0.0/0 513 –l –j DENY
ipchains –A output –i eth0 –p udp –s 0.0.0.0/0 –d 0.0.0.0/0 513 –l –j DENY
```

Again here is a short explanation of the rule structure
ipchains – this is the command to interact with the firewall
-A input – this is the option to append a rule to the end of a chain in this case the input chain
-i eth0 – this the interface to apply the rule against (in this case my external ethernet interface)
-s 0.0.0.0/0 – this option indicates the source address used for filtering in this case anywhere
-d 0.0.0.0/0 – this option indicates the destination address used for filtering in this case anywhere
23 – this options indicates which port this rule applies to in the case of tcp and udp, it can follow either the source or destination address
-l – this option means to log the packet to the syslog facility
-j DENY – this option indicates the target for the packet in this case it says to silently drop the packet

Now I should mention some additional information. Some of these rules may seem redundant because input and output are just different names but serve the same purpose. I purposely use an input and an output in these examples to show that it is often better to include additional rules so the administrator behind you can immediately see what you are trying to accomplish. It also allows you to view (ipchains –L input) chains and follow their purpose without having to go from chain to chain. Also it is very common that an organization would like to deny packets coming in for these services while allowing outbound connections to be established. Because this is so often the case I will include an example to allow client connections to servers located outside the company while denying incoming server requests. For this example I will show a telnet ruleset that performs this action. Similar rules can be implemented if any of the other services are required.

```
# rule to allow outgoing telnet client requests

ipchains –A output –i eth0 –p tcp –s (insert internal network here) –d 0.0.0.0/0 23 –j ACCEPT
ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 23 –d (insert internal network here) ! –y –j ACCEPT

# rule to deny incoming telnet server requests

ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d (insert internal network here) 23 -l –j DENY
```

A second observation should also be noted here, IPChains in not a state aware firewall. This simply means that for each rule that allows an outgoing request there must also be a companion rule that allows in incoming response into the network. There is a second problem with stateless firewalls, it only blocks default protocol installations. This means that if I was to install the SSH protocol inside my network on a port other than 22 my rule above won't work. The opposite is also true if I want to prohibit my internal network from using telnet to a remote network if the remote server has telnet listening on a different port my rule will be ignored. This is a real problem if your internal staff is untrusted or technically savvy.

Testing is pretty straightforward and can be done using the same procedures as given above. I will only test for the first port blocked here (telnet) but the others will be very similar with only the ports needing to be changed. Note: I have picked the source address and port using random numbers purely for testing purposes. You can substitute any numbers you want when testing the results should all be the same.

```
ipchains –C input –i eth0 –p tcp –s 100.100.1.1/32 999 –d 0.0.0.0/0 23
```

The testing can also be done using the nmap tool listed above using the same guidelines as given in the first exercise.

## Exercise III – Blocking RPC and NFS

Remote file sharing protocols should never be let through your firewall unless it has been secured (i.e. virtual private network). These protocols can act as a mounting point of an attack and can provide an attacker with more information than you want them to have. These protocols have also been subject to numerous security problems that have resulted in remote root exploits that have compromised many internet servers. For these reasons these should be among the first protocols you consider blocking at your firewall. The protocols are fairly straightforward in how they work. NFS runs on UDP and TCP, the port it runs on is more loosely defined than other protocols. Clients coming in to connect to NFS do so by asking a service called the RPC portmapper to tell them which UDP or TCP ports the NFS server is running on today. The RPC portmapper runs on TCP and UDP port 111. The NFS server usually ends up running on TCP or UDP port 2049 because that is the default port. Lockd is an RPC server that processes NFS file locking requests from the local kernel or from another remote lock daemon. It defaults to UDP port 4045 and also has had it's share of security problems. Perhaps a more general rule is block all RPC and related services at your firewall. Here are the firewall rules to block these services:

```
# rule to block incoming and outgoing connections for Portmap/rpcbind

ipchains –A input –i eth0 –p tcp  -s 0.0.0.0/0  –d 0.0.0.0/0 111  -l –j DENY
ipchains –A input –i eth0 –p udp -s 0.0.0.0/0  –d 0.0.0.0/0 111  -l –j DENY

# rule to block incoming and outgoing connections for NFS (default port)

ipchains –A input –i eth0 –p tcp  -s 0.0.0.0/0  –d 0.0.0.0/0 2049  -l –j DENY
ipchains –A input –i eth0 –p udp -s 0.0.0.0/0  –d 0.0.0.0/0 2049  -l –j DENY

# rule to block incoming and outgoing lockd requests

ipchains –A input –i eth0 –p udp -s 0.0.0.0/0  –d 0.0.0.0/0 4045  -l –j DENY
```

Again here is a short explanation of the rule structure
ipchains – this is the command to interact with the firewall
-A input – this is the option to append a rule to the end of a chain in this case the input chain
-i eth0 – this the interface to apply the rule against (in this case my external ethernet interface)
-s 0.0.0.0/0 – this option indicates the source address used for filtering in this case anywhere
-d 0.0.0.0/0 – this option indicates the destination address used for filtering in this case anywhere
23 – this options indicates which port this rule applies to in the case of tcp and udp, it can follow either the source or destination address
-l – this option means to log the packet to the syslog facility
-j DENY – this option indicates the target for the packet in this case it says to silently drop the packet

Testing is pretty straightforward and can be done using the same procedures as given in the first exercise. I will only test for the first port blocked here (Portmap/rpcbind) but the others will be very similar with only the ports needing to be changed. Note: I have picked the source address and port using random numbers purely for testing purposes. You can substitute any numbers you want when testing the results should all be the same.

```
ipchains –C input –i eth0 –p tcp –s 100.100.1.1/32 999 –d 0.0.0.0/0 111
```

The testing can also be done using the nmap tool listed above using the same guidelines as presented earlier in the first exercise.

## Exercise IV – Blocking NetBIOS in Windows NT

NetBIOS is the Microsoft version of file and print sharing. It is an old protocol dating from Workgroup for Windows era. It is generally a non routable protocol, however Microsoft decided to bind NetBIOS over TCP/IP so you could remotely share files and printers over different subnets. It is at best a very misunderstood protocol with many administrators not understanding what information is available can be gained without any authentification procedure. It is the Microsoft version of NFS and as such should be blocked at the firewall. Most scans in the wild are looking for open shares with no password validation. The protocol consists of a set of set up UDP and TCP connections attempts based on what information is needed by the client. Most recently, problems have arisen with various types of viruses trying to take advantage of open shares. This is one of the few protocols that is recommended get blocked both in and out of your organization. The problem with letting outbound attempts is that it can provide information to your neighbors that you may not want them to see. The final and most significant problem with this protocol is the general lack of security training in the Microsoft training classes. Most Microsoft server administrators receive little or no training in security with the result that they are unable to properly build a bastion host. The firewall administrator has been forced to learn what they can about the protocol in an effort to protect the organization. With this in mind here are the ports that must be blocked both for incoming and outgoing connections:

# rule for blocking inbound and outbound Windows NT 4.0 NetBIOS queries

```
ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 135:139 –l –j DENY
ipchains –A output –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 135:139 –l –j DENY
ipchains –A input –i eth0 –p udp –s 0.0.0.0/0 –d 0.0.0.0/0 137:138 –l –j DENY
ipchains –A output –i eth0 –p udp –s 0.0.0.0/0 –d 0.0.0.0/0 137:138 –l –j DENY
```

# rule for blocking inbound and outbound Windows 2000 NetBIOS queries

```
ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 135:139 –l –j DENY
ipchains –A output –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 135:139 –l –j DENY
ipchains –A input –i eth0 –p udp –s 0.0.0.0/0 –d 0.0.0.0/0 137:138 –l –j DENY
ipchains –A output –i eth0 –p udp –s 0.0.0.0/0 –d 0.0.0.0/0 137:138 –l –j DENY
ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0/0 445 –l –j DENY
ipchains –A output –i eth0 –p udp –s 0.0.0.0/0 –d 0.0.0.0/0 445 –l –j DENY
```

Again here is a short explanation of the rule structure
ipchains – this is the command to interact with the firewall
-A input – this is the option to append a rule to the end of a chain in this case the input chain
-i eth0 – this the interface to apply the rule against (in this case my external ethernet interface)
-s 0.0.0.0/0 – this option indicates the source address used for filtering in this case anywhere
-d 0.0.0.0/0 – this option indicates the destination address used for filtering in this case anywhere
23 – this options indicates which port this rule applies to in the case of tcp and udp, it can follow either the source or destination address
-l – this option means to log the packet to the syslog facility
-j DENY – this option indicates the target for the packet in this case it says to silently drop the packet

Testing is pretty straightforward and can be done using the same procedures as given in the first exercise. I will only test for the first port blocked here (Portmap/rpcbind) but the others will be very similar with only the ports needing to be changed. Note: I have picked the source address and port using random numbers purely for testing purposes. You can substitute any numbers you want when testing the results should all be the same.

```
ipchains –C input –i eth0 –p tcp –s 100.100.1.1/32 999 –d 0.0.0.0/0 139
```

The testing can also be done using the nmap tool listed above using the same guidelines as presented earlier in the first exercise.

For more information about NetBIOS you can read more about it at http://cable-dsl.home.att.net/netbios.htm. Also Robert Graham has a great section about the most of the reasons that you might be seeing so many attempts for these services showing up on your firewall logs. His information can be found at http://www.robertgraham.com/pubs/firewall-seen.html.

## Exercise V – Blocking X Windows

The X Windows protocol is an effort to run graphical application on one computer and view them on another. It was designed for Unix and dates from 1984. There are several problems with running X across your firewall. X is not encrypted, and its authentification system can be compromised. Someone can hijack and X session causing all sorts of problems not the least of which is being allowed access to the file system. Particular versions of X clients and servers have also had numerous security problems in the past. Running an X session across your firewall should leave a sinking pit in the middle of your stomach and should be eliminated if at all possible. This is yet another protocol that is advisable to block in both incoming and outgoing directions. Here are the firewall rules required to block these ports:

# rule to block incoming and outgoing X session establishment

ipchains –A output –i eth0 –p tcp –s (insert internal network here) –d 0.0.0.0/0 6000:6255 –l –j REJECT
ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d (insert internal network here) 6000:6255 –l –j DENY

Again here is a short explanation of the rule structure
ipchains – this is the command to interact with the firewall
-A input – this is the option to append a rule to the end of a chain in this case the input chain
-i eth0 – this the interface to apply the rule against (in this case my external ethernet interface)
-p tcp – this options indicates which protocol to filter on in this case TCP
-s 0.0.0.0/0 – this option indicates the source address used for filtering in this case anywhere
-d 0.0.0.0/0 – this option indicates the destination address used for filtering in this case anywhere
6000:6255 – this options indicates which port or port range this rule applies to in the case of tcp and udp, it can follow either the source or destination address
-l – this option means to log the packet to the syslog facility
-j REJECT - this option indicates the target for the packet in this case it says to send a reset packet back to the client (a nice thing to do when a rule only applies to internal clients going to an external network).

I should probably explain the difference in this rule as to why I have two rules for one protocol. The main reason I would want to use a rule set like this one is if I want my internal clients to quickly be notified if a service is unavailable. If I was to notify someone outside the firewall of my desire to reset their connection they can see that I have responded to their attempt to connect a sure sign that there is something at that IP address for them to check into. My internal clients are just probably unaware that they are violating a security policy and should have their client process know immediately that this is not allowed.

Testing is pretty straightforward and can be done using the same procedures as given in the first exercise. I will only test for the first rule blocked here but the others will be very similar with only the ports needing to be changed. Note: I have picked the addresses and ports using random numbers (except where noted) purely for testing purposes. You can substitute any numbers you want when testing the results should all be the same.

ipchains –C input –i eth0 –p tcp –s 100.100.1.1/32 999 –d (insert internal network here) 6000

The testing can also be done using the nmap tool listed above using the same guidelines as presented earlier in the first exercise.

## Exercise VI – Blocking Naming Services

Domain Name Resolution has been an integral part of the Internet since the beginning of its inception. DNS is a protocol that allows us to request services from names instead of numbers. A DNS server takes requests for names from clients and is responsible for translating them into numbers that the routers can understand. The main problem with DNS is that it is installed as a default service on most Unix boxes even if it is not being used as the organizations name server. DNS also provided many mechanisms to provide useful troubleshooting information to server administrators. This information, in the wrong hands can provide information useful to a hacker when gathering intelligence about a company. It has also had a few major security vulnerabilities in the past few years that have led to many Internet servers becoming compromised. For these reasons this protocol should be restricted to only those servers that are considered the master or slave DNS servers for the organization. LDAP is a directory service protocol that is most frequently used for sharing address books. Services such as Bigfoot or Four 11 user this protocol to provide directory services for its members. This protocol has had some security problems in the past and should not run through the firewall if at all possible. This protocol consists of a server that listens on udp or tcp port 389 by default. Here are the rules for blocking Naming Services:

```
# rule to block incoming dns queries to all but one internal master server (192.168.0.1)

ipchains –A input –i eth0 –p udp –s 0.0.0.0/0 –d  ! 192.168.0.1/32 53 –l –j DENY
ipchains –A output –i eth0 –p udp –s ! 192.168.0.1/32 53 –d 0.0.0.0/0 –l –j DENY

# rule to allow outgoing dns queries from our internal name server (192.168.0.1)

ipchains –A output –i eth0 –p udp –s 192.168.0.1/32 1024:65535 –d 0.0.0.0/0 53  –j ACCEPT
ipchains –A input –i eth0 –p udp –s 0.0.0.0/32 53 –d 192.168.0.1/32 1024:65535 –j ACCEPT

# rule to allow incoming zone transfer requests from our external slave server (192.168.1.1)

ipchains –A input –i eth0 –p tcp –s ! 192.168.1.1/32 –d ! 192.168.0.1/32 53 –l –j DENY
ipchains –A output –i eth0 –p udp –s ! 192.168.0.1/32 53 –d ! 192.168.1.1/32  –l –j DENY

# rule to block incoming and outgoing LDAP service requests

ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0  –d 0.0.0.0/0 389 –l –j DENY
ipchains –A input –i eth0 –p udp –s 0.0.0.0/0 –d 0.0.0.0/0 389 –l –j DENY
```

Again here is a short explanation of the rule structure
ipchains – this is the command to interact with the firewall
-A input – this is the option to append a rule to the end of a chain in this case the input chain
-i  eth0 – this the interface to apply the rule against (in this case my external ethernet interface)
-p tcp – this options indicates which protocol to filter on in this case TCP
-s 0.0.0.0/0 – this option indicates the source address used for filtering in this case anywhere
-d 0.0.0.0/0 –  this option indicates the destination address used for filtering in this case anywhere
389 – this options indicates which port or port range this rule applies to in the case of tcp and udp, it can follow either the source or destination address
! 192.168.0.1/32– this option negates the information or says "anything but" followed the parameter information in this case it means this rule applies to everyone except the IP address 192.168.0.1
-l – this option means to log the packet to the syslog facility
-j DENY – this option indicates the target for the packet in this case it says to silently drop the packet

There are several things you should realize the rules above would do. The first thing is that if a DNS query response is larger than 512 bytes it cannot be handled by UDP and will instead send a truncated flag and try to establish a TCP connection on port 53. The rules above will block these connections. In

real life this is rarely a problem because typical query responses should be under 512 bytes. If this is a major concern then instead you can put limitations on the bind servers to make them not respond to zone transfer attempts from anyone but the slave servers. Also these rules have assumed that you are using bind version greater than eight and not set the query port to be 53. I have assumed that you are using an ephemeral port (greater than 1023) to do queries to other servers. If this is not the case make sure that you change the rules above so the source port will be 53 for DNS client requests. Other things you should keep in mind is that people will often try to gather information from your DNS servers themselves other than zone transfers. In BIND 8 there are many security options that can be setup in the configuration files. For more information you should go to http://www.isc.org/products/BIND/. The last note is that due to the numerous security problems with BIND you should always upgrade your servers to the latest stable versions.

Testing is pretty straightforward and can be done using the same procedures as given in the first exercise. I will only test for the first rule blocked here (restricting dns queries to 192.168.0.1 only) but the others will be very similar with only the ports needing to be changed. Note: I have picked the source address and port using random numbers purely for testing purposes. You can substitute any numbers you want when testing the results should all be the same.

ipchains –C input –i eth0 –p udp –s 100.100.0.1/32 53 –d 192.168.0.2/32 53

The testing can also be done using the nmap tool listed above using the same guidelines as presented earlier in the first exercise.

## Exercise VII – Blocking Mail Access

The Simple Mail Transport Protocol is another of the most important functions of the Internet. This is probably one of the most common services offered by an organization with an internet presence. Because of the prevalence of the protocol it is naturally one of the most commonly scanned for protocols. Although spamming has been the number one concern of late there are still mailer daemons cracked every day. This protocol should be blocked at the firewall except from your external mail relay in to your internal mail server. Much like DNS this service is most likely a requirement for an organization so rather than blocking the protocol most emphasis will be placed on minimizing the exposure of the servers. Only SMTP should be allowed through the firewall. The various client protocols such as POP and IMAP should be blocked by the firewall. Most organizations requiring client access from the internet to the mail servers be done through a secure channel such as a web server using SSL. Following, are the rules required to implement these security policies:

# rule to block incoming SMTP traffic except to the internal mail server 192.168.0.1 from the external mail relay 192.168.1.1

ipchains –A input –i eth0 –p tcp –s ! 192.168.1.1/32 –d ! 192.168.0.1/32 25 –l –j DENY
ipchains –A output –i eth0 –p tcp –s ! 192.168.0.1/32 25 –d ! 192.168.1.1/32 –l –j DENY

# rule to block incoming POP and IMAP traffic

ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0.0/0 109:110 –l –j DENY
ipchains –A input –i eth0 –p tcp –s 0.0.0.0/0 –d 0.0.0.0.0/0 143 –l –j DENY

Again here is a short explanation of the rule structure
ipchains – this is the command to interact with the firewall
-A input – this is the option to append a rule to the end of a chain in this case the input chain
-i eth0 – this the interface to apply the rule against (in this case my external ethernet interface)
-p tcp – this options indicates which protocol to filter on in this case TCP
-s 0.0.0.0/0 – this option indicates the source address used for filtering in this case anywhere
-d 0.0.0.0/0 – this option indicates the destination address used for filtering in this case anywhere
25 – this options indicates which port or port range this rule applies to in the case of tcp and udp, it can follow either the source or destination address
-l – this option means to log the packet to the syslog facility
-j DENY – this option indicates the target for the packet in this case it says to silently drop the packet

The only thing to keep in mind with these rules are that if you have remote users who need to get their mail through the internet these rules will block them. The main problem that you can get into trouble with mail servers is to allow them to relay mail outbound without any spam prevention. This can get you in trouble with the internet community and can lead to your organization being black listed and prevented from sending other servers mail. In order to prevent this you should always make sure that whatever mail server you are using has been implemented with some type of spam filtering mechanism. More information about spam and Sendmail (the most popular SMTP server available) is available from http://www.sendmail.org/. One last note is that this also blocks outgoing POP and IMAP attempts. If this service is desired you can follow my instructions listed earlier on allowing the telnet client through the firewall.

Testing is pretty straightforward and can be done using the same procedures as given in the first exercise. I will only test for the first rule blocked here (restricting dns queries to 192.168.0.1 only) but the others will be very similar with only the ports needing to be changed. Note: I have picked the source address and port using random numbers purely for testing purposes. You can substitute any numbers you want when testing the results should all be the same.

ipchains –C input –i eth0 –p udp –s 100.100.0.1/32  999 –d 192.168.0.2/32 25

The testing can also be done using the nmap tool listed above using the same guidelines as presented earlier in the first exercise.

## Exercise VIII – Blocking World Wide Web Service

Blocking internal web access is important because many current network devices are starting to ship with web service being offered by default. A good example is on current Cisco Switches they are configured by default with a web server running on port 80. This combined with a weak password can lead to disastrous situations where hackers can either change internal network settings or use these devices a launching platforms for attacks on other devices. Another problem with allowing unchecked port 80 access to your network is the prevalence of internal intranets with sensitive information on them. Often companies store information on their internal web servers that only meant for internal company employees such as phone numbers or addresses. Another problem is that web servers are often the primary targets for hackers. This is because they are often misconfigured or are running some type of cgi or java script that is unsafe. By restricting web traffic from the internet many of these problems can be avoided. It is a pretty straightforward protocol with clients contacting a server by address or name on TCP port 80. SSL is a secure version of HTTP that runs on TCP port 443. For the same reasons I block HTTP traffic SSL should also be blocked if unneeded. If a company desires to have a web presence by installing a web server most experts agree that it should be placed outside of firewall as a bastion host or segregated onto it's own subnet if possible. The other related HTTP ports are from people doing port remapping or redirecting. There are several reasons for this including the fact that only root on a Unix server can bind to a port under 1024 so if a user other than root wants to run a web service he must pick a higher numbered one. Another reason is that they will not be picked up in a routing scan for port 80 thus avoiding many of the most common cgi scans going on in the wild. Yet another reason is that most high end ports are not blocked by default. Therefore if an internal user wants to bypass a security policy and run a web server he can attempt to run it on a high end port that is not blocked. The most commonly used port numbers for these are those related to 80 such as 81, 8000, or 8080, or even 8888. These ports should also be blocked more for logging purposes unless you are doing some sort of port redirection or have reason to suspect internal users are up to mischief. Following are the rules to block these ports:

```
# rule to block all incoming HTTP server requests

ipchains –A input –i eth0 –p tcp -s 0.0.0.0/0 –d (insert internal network here) 80  -l –j DENY
ipchains –A input –i eth0 –p tcp -s 0.0.0.0/0 –d (insert internal network here) 443  -l –j DENY

# rule to block all other HTTP server request ports

ipchains –A input –i eth0 –p tcp -s 0.0.0.0/0  –d (insert internal network here) 8000  -l –j DENY
ipchains –A input –i eth0 –p tcp -s 0.0.0.0/0  –d (insert internal network here) 8080  -l –j DENY
ipchains –A input –i eth0 –p tcp -s 0.0.0.0/0  –d (insert internal network here) 8888  -l –j DENY
ipchains –A input –i eth0 –p tcp -s 0.0.0.0/0  –d (insert internal network here) 81  -l –j DENY
```

Again here is a short explanation of the rule structure
ipchains – this is the command to interact with the firewall
-A input – this is the option to append a rule to the end of a chain in this case the input chain
-i eth0 – this the interface to apply the rule against (in this case my external ethernet interface)
-p tcp – this options indicates which protocol to filter on in this case TCP
-s 0.0.0.0/0 – this option indicates the source address used for filtering in this case anywhere
-d 0.0.0.0/0 – this option indicates the destination address used for filtering in this case anywhere
80 – this options indicates which port or port range this rule applies to in the case of tcp and udp, it can follow either the source or destination address
-l – this option means to log the packet to the syslog facility
-j DENY – this option indicates the target for the packet in this case it says to silently drop the packet

Testing is pretty straightforward and can be done using the same procedures as given in the first exercise. I will only test for the first rule blocked here (blocking inbound http server requests) but the

others will be very similar with only the ports needing to be changed. Note: I have picked the source address and port using random numbers purely for testing purposes. You can substitute any numbers you want when testing the results should all be the same.

ipchains –C input –i eth0 –p tcp –s 100.100.0.1/32 999 –d (insert local network here) 80

The testing can also be done using the nmap tool listed above using the same guidelines as presented earlier in the first exercise.

## Exercise IX – Blocking Small Services

Small services are those services that run less than UDP or TCP port 20. The most frequently used ones are echo (UDP port 7), chargen (UDP and TCP port 19), and discard (UDP and TCP port 9). These services especially their UDP versions are rarely used for legitimate purposes and can be used to launch denial of service attacks. Although most of the problems with these ports can be prevented with good anti-spoofing rules these services should be blocked by the firewall. A perfect example is the echo port which will echo back a packet to the originator and can be used as a denial of service if two internet servers can be convinced to send packets to the echo port of the other machine. Since these ports are so rarely used anymore they should be blocked at the firewall and disabled at the border router to avoid any problems that can be caused by malicious hackers looking to tie up bandwidth or processor time. Following are the rules that can be applied at the firewall to block these services:

```
# rules to block small services or those that run < port 20

ipchains –A input –i eth0 –p tcp  -s 0.0.0.0/0  –d 0.0.0.0/0 0:19  -l –j DENY
ipchains –A output –i eth0 –p tcp -s 0.0.0.0/0 0:19 –d 0.0.0.0/0   -l –j DENY
```

Again here is a short explanation of the rule structure
ipchains – this is the command to interact with the firewall
-A input – this is the option to append a rule to the end of a chain in this case the input chain
-i eth0 – this the interface to apply the rule against (in this case my external ethernet interface)
-p tcp – this options indicates which protocol to filter on in this case TCP
-s 0.0.0.0/0 – this option indicates the source address used for filtering in this case anywhere
-d 0.0.0.0/0 – this option indicates the destination address used for filtering in this case anywhere
80 – this options indicates which port or port range this rule applies to in the case of tcp and udp, it can follow either the source or destination address
-l – this option means to log the packet to the syslog facility
-j DENY – this option indicates the target for the packet in this case it says to silently drop the packet

The only thing that should be noted with these rules is that I have blocked outbound ports as well so that my internal clients cannot cause trouble for other servers. These rules are pretty standard on the internet and actually should be blocked on the border router as to avoid filling up your logs with generally useless information. If you need any more information regarding these ports or why to block them I suggest the following Cisco page http://www.cisco.com/warp/public/707/21.html - small.

Testing is pretty straightforward and can be done using the same procedures as given in the first exercise. I will only test for the first rule blocked here (blocking small services requests) but the others will be very similar with only the ports needing to be changed. Note: I have picked the source address and port using random numbers purely for testing purposes. You can substitute any numbers you want when testing the results should all be the same.

```
ipchains –C input –i eth0 –p tcp –s 100.100.0.1/32  999 –d (insert local network here) 1
```

The testing can also be done using the nmap tool listed above using the same guidelines as presented earlier in the first exercise.

## Exercise X – Blocking Miscellaneous Services

Firewalls should guard against certain protocols that could be potential security loopholes. These protocols include TFTP, finger, NNTP, NTP, LPD, syslog, SNMP, BGP, and SOCKS. They can be useful in certain situations but generally the potential problems outweigh any gains and can usually be accomplished in another way. TFTP is a type of connectionless FTP server. It does not guarantee delivery of the file contents and can be used to place files on machines without any user validation. The finger program is used to view the comments field in the user setup file. It could include potentially dangerous information about who is currently logged into the system as well as how they are connected. NNTP is the Network News Transfer Protocol and is used to post, distribute, and retrieve USENET articles. This is a very useful protocol but has had some security problems recently and should not be allowed through the firewall. NTP is the Network Time Protocol and is used to sync the times and dates of internet servers. It is meant to be a useful tool but could be used to alter time and dates of servers in order to confuse or harass technicians reading log files. LPD is the printer protocol meant to transfer remote printer jobs to printers. Allowing servers to send print requests is an obvious problem. If an attacker gains access to the printer he can cause a denial of service by merely printing a lot of useless information or perhaps sending files to print which are offensive in nature. Syslog is a service that provides logging for Unix servers and various network devices. By allowing syslog entries though your firewall you risk a denial of service by filling the logs with worthless information prior to an attempt to gain entrance to your network. By filling the logs an attacker can eventually fill the partition in which the logs are located causing an entire server to become unavailable to clients. SNMP is the Simple Network Management Protocol, which is meant to poll network devices to learn performance statistics. This protocol can be a problem because network devices are often configured out of the box with default passwords. Also information can now be written to these devices causing updates of their configurations, which can lead to various problems. BGP is the Border Gateway Protocol, which is used to update intra network routing tables. It can be hazardous to your network to allow in any protocol updates from untrusted sources. Also most internal networks should not be using this protocol within their network. They should instead use one of the numerous interior routing protocols. The SOCKS protocol is a proxy server that will allow clients to proxy their server requests through the proxy server. They are not troublesome in themselves but they are scanned for frequently on the internet in order to find unblocked SOCKS servers that will allow an attacker to telnet to and establish a connection from that device to another. It can allow a series of servers to be connected together so that an attacker can gain anonymity while performing nefarious tricks. It is often used to cause problems for IRC servers. For these reasons they should be blocked at the firewall. Here are the rules to block these services:

```
# rule to block incoming and outgoing TFTP server requests

 ipchains –A input –i eth0 –p udp  -s 0.0.0.0/0  –d 0.0.0.0/0 69  -l –j DENY

# rule to block incoming and outgoing finger requests

ipchains –A input –i eth0 –p tcp  -s 0.0.0.0/0  –d 0.0.0.0/0 79  -l –j DENY

# rule to block incoming and outgoing NNTP server requests

ipchains –A input –i eth0 –p tcp  -s 0.0.0.0/0  –d 0.0.0.0/0 119  -l –j DENY

# rule to block incoming and outgoing NTP server requests

ipchains –A input –i eth0 –p tcp  -s 0.0.0.0/0  –d 0.0.0.0/0 123  -l –j DENY

# rule to block incoming and outgoing LPD printer jobs

ipchains –A input –i eth0 –p tcp  -s 0.0.0.0/0  –d 0.0.0.0/0 515  -l –j DENY
```

# rule to block incoming and outgoing syslog messages

ipchains –A input –i eth0 –p udp -s 0.0.0.0/0 –d 0.0.0.0/0 514 -l –j DENY

# rules to block incoming and outgoing SNMP polling requests

ipchains –A input –i eth0 –p tcp -s 0.0.0.0/0 –d 0.0.0.0/0 161:162 -l –j DENY
ipchains –A input –i eth0 –p udp -s 0.0.0.0/0 –d 0.0.0.0/0 161:162 -l –j DENY

# rule to block incoming and outgoing BGP route messages

ipchains –A input –i eth0 –p tcp -s 0.0.0.0/0 –d 0.0.0.0/0 179 -l –j DENY

# rule to block incoming and outgoing SOCKS server connections

ipchains –A input –i eth0 –p tcp -s 0.0.0.0/0 –d 0.0.0.0/0 1080 -l –j DENY

Again here is a short explanation of the rule structure
ipchains – this is the command to interact with the firewall
-A input – this is the option to append a rule to the end of a chain in this case the input chain
-i eth0 – this the interface to apply the rule against (in this case my external ethernet interface)
-p tcp – this options indicates which protocol to filter on in this case TCP
-s 0.0.0.0/0 – this option indicates the source address used for filtering in this case anywhere
-d 0.0.0.0/0 – this option indicates the destination address used for filtering in this case anywhere
80 – this options indicates which port or port range this rule applies to in the case of tcp and udp, it can follow either the source or destination address
-l – this option means to log the packet to the syslog facility
-j DENY – this option indicates the target for the packet in this case it says to silently drop the packet

There isn't much to say about these various protocols except that logging these entries can fill up logs as they are frequently scanned for on the internet. They are pretty standard and should cause no problems when blocked. There is one final item worth mentioning. TFTP is still used to update Cisco IOS versions. This could be a problem if you needed to update routers remotely through many different subnets. Be sure when using this rule that you realize that your network administrators may argue the point. Also much like the remote logon protocols blocked earlier you may find a need to allow outbound client requests to remote servers if that is the case you must use the same procedure noted above in exercise II to include the client rules.

Testing is pretty straightforward and can be done using the same procedures as given in the first exercise. I will only test for the first rule blocked here (blocking inbound TFTP server requests) but the others will be very similar with only the ports needing to be changed. Note: I have picked the source address and port using random numbers purely for testing purposes. You can substitute any numbers you want when testing the results should all be the same.

ipchains –C input –i eth0 –p tcp –s 100.100.0.1/32 999 –d (insert local network here) 69

The testing can also be done using the nmap tool listed above using the same guidelines as presented earlier in the first exercise.

## Exercise XI – Blocking Specific ICMP traffic

ICMP has caused more arguments between firewall administrators and network engineers than any other topics. The first topic of discussion is the echo request. This is a basic service meant to assure that a remote machine has a properly functioning TCP/IP stack. When a network device receives this request it sends back a packet to the originator using the ICMP type echo reply. This seemingly innocent tool can be used to map entire IP subnets looking for active hosts that can be further examined for vulnerabilities. They are the most common type of scan on the internet. When configuring a firewall the first step in locking down services is blocking this type of packet. The next topic of discussion is the time exceeded reply. An IP packet has a field that contains a time to live field. This field is decremented as it passes each top on the route to the destination. This field is used to prevent packets from getting misrouted and ending up in a routing loop. The router that receives a packet with the field value of one has the job of sending a packet back to the originator letting them know that the time has been exceeded and is being discarded. A separate program called trace route that is responsible for displaying the route a packet is taking to reach a given destination also commonly uses this type of packet. An attacker can gain valuable information regarding the surrounding networks and the network location with this type of knowledge. A third type of troublesome ICMP packet is the unreachable messages (both destination and host). These packets can be used by an attacker to reverse map a network by noting which addresses the router returns a destination unreachable message. The host unknown message provides the exact opposite information that a ping packet provides. Instead of giving information about what hosts are active it provides information about which hosts are unavailable. Here are the methods for blocking these types of packets:

```
# rule to block incoming ICMP echo requests

ipchains -A input -i eth0 -p icmp  -s 0.0.0.0/0  -d 0.0.0.0/0 8 -l -j DENY

# rule to block outgoing ICMP echo replies

ipchains -A output -i eth0 -p icmp  -s 0.0.0.0/0 0 -d 0.0.0.0/0 -l -j DENY

# rule to block outgoing time exceeded and unreachable messages

ipchains -A output -i eth0 -p icmp  -s (insert local network here) 11 -d 0.0.0.0/0 -l -j DENY
ipchains -A output -i eth0 -p icmp  -s (insert local network here) 3 -d 0.0.0.0/0 -l -j DENY
```

Again here is a short explanation of the rule structure
ipchains – this is the command to interact with the firewall
-A input – this is the option to append a rule to the end of a chain in this case the input chain
-i eth0 – this the interface to apply the rule against (in this case my external ethernet interface)
-p icmp – this options indicates which protocol to filter on in this case ICMP
-s 0.0.0.0/0 – this option indicates the source address used for filtering in this case anywhere
-d 0.0.0.0/0 – this option indicates the destination address used for filtering in this case anywhere
8 - this options indicates which type of ICMP packet the rule applies to , it can follow either the source or destination address in this case it is the echo request icmp type
-l – this option means to log the packet to the syslog facility
-j DENY – this option indicates the target for the packet in this case it says to silently drop the packet

As mentioned earlier this protocol is deemed necessary for network troubleshooting. A specific problem might arise if you disallow the icmp type of destination-unreachable because networks use it to negotiate packet fragment size. Also your ISP may have a need to use the echo-request packet to troubleshoot network outages. For each ICMP rule you apply to your firewall there will be a network administrator arguing that it is necessary for them to do their jobs.

In order to verify that the firewall is blocking these packets you can use the –C flag in the IPChains command to verify you are blocking ICMP traffic. I have included an example for only the first rule

(blocking incoming ICMP echo requests) but similar checks for the other rules will follow by changing the ICMP type codes.

ipchains –C input –i eth0 –p icmp -s 0.0.0.0/0 0 –d 0.0.0.0/0 8

The testing can also be done using the nmap tool listed above using the same guidelines as presented earlier in the first exercise. There will be only one problem if you are testing for udp ports on a firewall box blocking icmp port unreachable will stop nmap from properly functioning so it may not be a valid test for udp when done.

## Final Thoughts

The overriding factor in determining firewall rules is the security policy that each organization maintains. This policy should be examined closely so that all security issues are brought to light before the firewall is ever established. Also functionality versus security should be examined so that important services that the organization may need is not blocked arbitrarily for pure security concerns. When building the firewall it is essential that all factors are considered in the ruleset as carelessness can lead to end users becoming frustrated due to loss of services that they have come to rely on in order to accomplish their jobs. The reverse is also true, your job may hinge on an attackers ability to compromise your network. With this in mind you should always strive to eliminate points of attack that your organization has no need for.

The final section is a complete list of all the rules we have applied to our firewall. I recommend that you write these rules to a script so that they can be changed easily and rearranged if needed. I have included a complete script that accomplishes this as well as some other things particular to Linux and IPChains.

## Sample Script

The following is a script that should accomplish all the previous tasks and block the necessary ports. Notice that I begin by flushing all existing rules then setting the default policy for the chain. Next I complete a couple of protection procedures and then I proceed to the rules listed above. This script should be made executable and then run. If desired this script can be used as part of an init.d script to start firewalling on boot up. If this is desired the script should be run before the networking daemon starts to avoid any potential problems from occurring during the brief time it will take the firewall script to run.

**NOTE**: I have set the default policy for this script to be ACCEPT for all chains. This should never be done in a production environment. A far better solution would be to deny everything but what is explicitly allowed. This would cause the deny rules to be redundant except for logging purposes. I have used the ACCEPT policy to follow the exercise guidelines which wanted to know how to deny problem ports.

```
#!/bin/sh

echo "Starting firewalling....."

#  Flush and set default policy of ACCEPT

        #  Remove all existing rules belonging to this filter
          ipchains -F input
```

```
                ipchains -F output
                ipchains -F forward

               #  Set the default rules belonging to this filter
                ipchains -P input ACCEPT
                ipchains -P output ACCEPT
                ipchains -P forward ACCEPT

#  Enable TCP SYN Cookie Protection
echo 1 >/proc/sys/net/ipv4/tcp_syncookies

#  enable ip spoofing protection
#  turn on source address verification
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
                echo 1 > $f
                done

#  disable ICMP Redirect Acceptance
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
                echo > $f
                done

#  disable source routed packets
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
                echo 0 > $f
                done


#  rules for standard unroutables
ipchains -A input -i eth0 -s 255.255.255.255/32 -b -j  DENY
ipchains -A input -i eth0 -s 127.0.0.0/8 -b -j  DENY

# rules for private (RFC1918) addresses
ipchains -A input -i eth0 -s 10.0.0.0/8 -b -j DENY
ipchains -A input -i eth0 -s 172.16.0.0/12 -b -j DENY
ipchains -A input -i eth0 -s 192.168.0.0/16 -b -j DENY

#rule for reserved addresses
ipchains -A input -i eth0 -s 240.0.0.0/5 -b -j DENY

# rule for protecting internal network from spoofing
ipchains -A input -i eth0 -s 198.148.188.0/24 -l -j DENY

# rule to block incoming and outgoing telnet connections
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 23 -l -j DENY

# rule to block incoming and outgoing ssh connections
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 22 -l -j DENY

#rule to block incoming and outgoing FTP connections

ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 21 -l -j DENY

# rule to block incoming and outgoing WinNT 4.0 NetBIOS connections

ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 139 -l -j DENY
```

```
ipchains -A output -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 135:139 -l -j DENY
ipchains -A input -i eth0 -p udp -s 0.0.0.0/0 -d 0.0.0.0/0 137:138 -l -j DENY
ipchains -A output -i eth0 -p udp -s 0.0.0.0/0 -d 0.0.0.0/0 137:138 -l -j DENY

# rule to block incoming and outgoing Win2000 NetBIOS connections

ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 445 -l -j DENY
ipchains -A output -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 445 -l -j DENY

#rule to block incoming and outgoing rlogon connections

ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 512:514 -l -j DENY
ipchains -A output -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 512:514 -l -j DENY
ipchains -A input -i eth0 -p udp -s 0.0.0.0/0 -d 0.0.0.0/0 513 -l -j DENY
ipchains -A output -i eth0 -p udp -s 0.0.0.0/0 -d 0.0.0.0/0 513 -l -j DENY

# rule to block incoming and outgoing connections for Portmap/rpcbind

ipchains -A input -i eth0 -p tcp  -s 0.0.0.0/0  -d 0.0.0.0/0 111  -l -j DENY
ipchains -A input -i eth0 -p udp -s 0.0.0.0/0  -d 0.0.0.0/0 111  -l -j DENY

# rule to block incoming and outgoing connections for NFS (default port)

ipchains -A input -i eth0 -p tcp  -s 0.0.0.0/0  -d 0.0.0.0/0 2049  -l -j DENY
ipchains -A input -i eth0 -p udp -s 0.0.0.0/0  -d 0.0.0.0/0 2049  -l -j DENY

# rule to block incoming and outgoing lockd requests

ipchains -A input -i eth0 -p udp -s 0.0.0.0/0  -d 0.0.0.0/0 4045  -l -j DENY

# rule for blocking inbound and outbound Windows NT 4.0 NetBIOS queries

ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 135:139 -l -j DENY
ipchains -A output -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 135:139 -l -j DENY
ipchains -A input -i eth0 -p udp -s 0.0.0.0/0 -d 0.0.0.0/0 137:138 -l -j DENY
ipchains -A output -i eth0 -p udp -s 0.0.0.0/0 -d 0.0.0.0/0 137:138 -l -j DENY

# rule for blocking inbound and outbound Windows 2000 NetBIOS queries

ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 135:139 -l -j DENY
ipchains -A output -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 135:139 -l -j DENY
ipchains -A input -i eth0 -p udp -s 0.0.0.0/0 -d 0.0.0.0/0 137:138 -l -j DENY
ipchains -A output -i eth0 -p udp -s 0.0.0.0/0 -d 0.0.0.0/0 137:138 -l -j DENY
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 445 -l -j DENY
ipchains -A output -i eth0 -p udp -s 0.0.0.0/0 -d 0.0.0.0/0 445 -l -j DENY

# rule to block incoming and outgoing X session establishment

ipchains -A output -i eth0 -p tcp -s (insert local network here) -d 0.0.0.0/0 6000:6255 -l -j REJECT
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d (insert local network here) 6000:6255 -l -j DENY

# rule to block incoming dns queries to all but one internal master server (192.168.0.1)

ipchains -A input -i eth0 -p udp -s 0.0.0.0/0 -d ! 192.168.0.1/32 53 -l -j DENY
ipchains -A output -i eth0 -p udp -s ! 192.168.0.1 53 -d 0.0.0.0/0 -l -j DENY
```

```
# rule to allow outgoing dns queries from our internal name server (192.168.0.1)

ipchains -A output -i eth0 -p udp -s 192.168.0.1/32 -d 0.0.0.0/0 53 -j ACCEPT
ipchains -A input -i eth0 -p udp -s 0.0.0.0/0 53 -d 192.168.0.1/32 -j ACCEPT

# rule to allow incoming zone transfer requests from our external slave server (192.168.1.1)

ipchains -A input -i eth0 -p tcp -s ! 192.168.1.1/32 -d ! 192.168.0.1/32 53 -l -j DENY
ipchains -A output -i eth0 -p udp -s ! 192.168.0.1/32 53 -d ! 192.168.1.1/32 -l -j DENY

# rule to block incoming and outgoing LDAP service requests

ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0  -d (insert local network here) 389 -l -j DENY
ipchains -A input -i eth0 -p udp -s 0.0.0.0/0 -d (insert local network here) 389 -l -j DENY

# rule to block incoming SMTP traffic except to the internal mail server (192.168.0.1) from the external
mail relay 192.168.1.1

ipchains -A input -i eth0 -p tcp -s ! 192.168.1.1/32 -d ! 192.168.0.1/32 25 -l -j DENY
ipchains -A output -i eth0 -p tcp -s ! 192.168.0.1/32 25 -d ! 192.168.1.1/32 -l -j DENY


# rule to block incoming POP and IMAP traffic

ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 –d 0.0.0.0/0 109:110  -l -j DENY
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 143  -l -j DENY

# rule to block all incoming HTTP server requests

# ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d (insert local network here) 80  -l -j DENY
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0 -d (insert local network here) 443  -l -j DENY

# rule to block all other HTTP server request ports

ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0  -d (insert local network here) 8000  -l -j DENY
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0  -d (insert local network here) 8080  -l -j DENY
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0  -d (insert local network here) 8888  -l -j DENY
ipchains -A input -i eth0 -p tcp -s 0.0.0.0/0  -d (insert local network here) 81  -l -j DENY

# rule to block incoming and outgoing small services
ipchains -A input -i eth0 -p tcp  -s 0.0.0.0/0  -d 0.0.0.0/0 0:19  -l -j DENY
ipchains -A output -i eth0 -p tcp -s 0.0.0.0/0 0:19 -d 0.0.0.0/0   -l -j DENY
ipchains -A input -i eth0 -p udp  -s 0.0.0.0/0  -d 0.0.0.0/0 0:19  -l -j DENY
ipchains -A output -i eth0 -p udp -s 0.0.0.0/0 0:19 -d 0.0.0.0/0   -l -j DENY

# rule to block incoming and outgoingTFTP server requests

 ipchains -A input -i eth0 -p udp  -s 0.0.0.0/0  -d 0.0.0.0/0 69  -l -j DENY

# rule to block incoming and outgoing finger requests

ipchains -A input -i eth0 -p tcp  -s 0.0.0.0/0  -d 0.0.0.0/0 79  -l -j DENY

# rule to block incoming and outing NNTP server requests

ipchains -A input -i eth0 -p tcp  -s 0.0.0.0/0  -d 0.0.0.0/0 119  -l -j DENY
```

```
# rule to block incoming and outgoing NTP server requests

ipchains -A input -i eth0 -p tcp  -s 0.0.0.0/0  -d 0.0.0.0/0 123  -l -j DENY

# rule to block incoming and outgoing LPD printer jobs

ipchains -A input -i eth0 -p tcp  -s 0.0.0.0/0  -d 0.0.0.0/0 515  -l -j DENY

# rule to block incoming and outgoing syslog messages

ipchains -A input -i eth0 -p udp  -s 0.0.0.0/0  -d 0.0.0.0/0 514  -l -j DENY

# rules to block incoming and outgoing SNMP polling requests

ipchains -A input -i eth0 -p tcp  -s 0.0.0.0/0  -d 0.0.0.0/0 161:162 -l -j DENY
ipchains -A input -i eth0 -p udp  -s 0.0.0.0/0  -d 0.0.0.0/0 161:162 -l -j DENY

# rule to block incoming and outgoing BGP route messages

 ipchains -A input -i eth0 -p tcp  -s 0.0.0.0/0  -d 0.0.0.0/0 179 -l -j DENY

# rule to block incoming and outgoing SOCKS server connections

ipchains -A input -i eth0 -p tcp  -s 0.0.0.0/0  -d 0.0.0.0/0 1080 -l -j DENY

# rule to block incoming ICMP echo requests

ipchains -A input -i eth0 -p icmp  -s 0.0.0.0/0  -d 0.0.0.0/0 8 -l -j DENY

# rule to block outgoing ICMP echo replies

ipchains -A output -i eth0 -p icmp  -s 0.0.0.0/0 0 -d 0.0.0.0/0 -l -j DENY

# rule to block outgoing time exceeded and unreachable messages

ipchains -A output -i eth0 -p icmp  -s (insert local network here) 11 -d 0.0.0.0/0 -l -j DENY
ipchains -A output -i eth0 -p icmp  -s (insert local network here) 3 -d 0.0.0.0/0 -l -j DENY
```