Global Information Assurance Certification Paper

SANS Security DC 2000 --- Practical Assignment

How to implement Security Policy on Border Routers

Dr. Thomas P. Braun

We are presented with a 'security policy' and asked to implement it in our local environment. This policy mainly defines if and how each network services should accessible form the external network. Our task is to find the best way to implement these requirements.

While the security policy can be rather generic, the implementation has to be specific. It has to fit into the existing technical environment, especially into the local network architecture.

This tutorial describes a solution that implements an entire security policy on the border routers. While this is a reflection of our network architecture, it is not at all limited to this particular environment. Every organization that is large enough to require (or afford) a border router can apply this implementation.

Specific network topology

As a major educational institution with an entire class B address our network architecture is rather complex. Throughout this tutorial I will use the *fictitious* address space 123.123.0.0/16.

Our local network consists of a dual backbone, where two 'Gigabit' switches (Cisco 5500) provide fast connectivity between a number of internal routers (mostly Cisco 8540) for internal traffic, and to two border routers (Cisco 7513) for connections with the public network. The subnets of individual departments or administrative units (either switched or shared ethernet) are connected to the internal routers through an additional layer of switches. For performance reasons (load balancing and redundancy) all traffic on our backbone (between the border routers and the 'subnet switches') is dynamically routed through one of two possible internal routers, one of two gigabit switches and either one of the two border routers (external traffic only).

Because of this complex architecture, there is no *central* firewall system installed. But even if we cannot know how the individual packets are routed (because of the dual backbone) we know that all traffic to and from the public network has to be routed through one of the two border routers.

Therefore, these two border routers are the only points of defense (that protects all machines) for this network architecture. We can (and do) provide additional security for individual hosts or entire subnets with increased security requirements by local firewall systems and/or host-based protection.

The rules of the given security policy affect all hosts within our local area network. The policy will therefore be implemented entirely on the border routers.

Implementation of the security policy

The security policy attempts to address the SANS Top Ten list of the *"Most Critical Internet Security Threats"*. It contains a number of services and connections that should not be accessible from the public network.

Since most requirements of the security policy are related to incoming traffic (from the public network), we can implement it by defining an access list for incoming packets (INGRESS ACL) on the external interfaces

of our border routers. There is, however one additional requirement in the security policy ('ICMP') that is related to traffic leaving the local network. To block these packets we need to install an outbound filter (EGRESS ACL) as well.

The implementation of the entire security policy is described at the end of this tutorial ('Putting it all together'). First we will go through the list one item at a time to understand why these filters are necessary, how they are implemented and what side effects we have to consider.

Security Policy

1. **Block spoofed addresses and source routing**

Spoofing

Packets with forged ('spoofed') source addresses cause major problems on the network. They are used as 'decoys' in host and port scans and for many kinds of Denial of Service (DoS) attacks. By spoofing the source addresses the attackers can hide their location, making it more difficult for us to uncover their true identity.
We cannot defend our networks against these attacks if the attacker is forging legitimate addresses. However, attackers frequently use the reserved address space as source addresses for their spoofed packets.

RFC 1918 defines the ranges

```
    10.0.0.0   -  10.255.255.255  (10.0.0.0/8)
  172.16.0.0   -  172.31.255.255  (172.16.0.0/12)
 192.168.0.0   - 192.168.255.255  (192.168.0.0/16)
```

as 'private addresses'. These packets (as well the 'loopback' network 127.0.0.0/8) should not be routed in the public network. They can only originate from malicious or misconfigured sources. In either case there is no need to accept these packets with 'invalid' source addresses. And since we are only blocking illegitimate traffic, there are no adversary effects of this measure.

A filter rule to block incoming traffic from 'invalid' source addresses at the border routers could be implemented by a simple STANDARD access list (ACL) on the external interface. The format of STANDARD ACL's is quite simple:

```
access-list xx permit/deny source [wild card]
```

The wild card consists of four bytes and looks like a netmask. And like a netmask it determines which logic bits are 'true bits'. But the syntax for wild cards specifies that the true bits represent the bits that are **not** to be examined. So the filter 10.0.0.0 0.255.255.255 tests only the first field ('10'), while 192.168.0.0 0.0.255.255 test the first two fields ('192.168'). The wild card 255.255.255.255 doesn't test any bits while 0.0.0.0 tests all bits. (This is the default if no wild card is given.)

The filter rule for a STANDARD ACL to block 'invalid' source addresses looks like this:

here we define the blocked subnets

```
access-list 12 deny 10.0.0.0 0.255.255.255
access-list 12 deny 172.16.0.0 0.15.255.255
access-list 12 deny 192.168.0.0 0.0.255.255
access-list 12 deny 127.0.0.0 0.255.255.255
```

```
access-list 12 deny 123.123.0.0 0.0.255.255
```
block packets pretending to come from within our domain

```
access-list 12 permit any
```
"any" is a 'special keyword' for "0.0.0.0 255.255.255.255"
without his rule the **implicit deny rule** will block all packets!

This approach would only work if all blocks could be implemented by a STANDARD ACL, or if the more advanced filters could be placed on a separate firewall.

But since we have to block all unwanted incoming traffic (as defined by the security policy) with one single INGRESS ACL on our border routers, we have to use an EXTENDED ACL instead of a STANDOARD ACL (only one ACL can be used for any interface in any direction at the same time).

While STANDARD ACL's test only the source address, EXTENDED ACL's can examine both the source and destination IP addresses, port number and protocol. The general syntax for an EXTENDED ACL is:

```
access-list xxx permit/deny protocol source [wild card] \
                destination [wild card] [options] [log]
```

I will explain details of the syntax as we use it for some of the more complex filters.

For the implementation of the 'anti spoofing' blocks we only need to change the change the access-list number, add the protocol (IP) and specify the destination addresses (to all our hosts):

```
access-list 112 deny ip 10.0.0.0 0.255.255.255 any
access-list 112 deny ip 172.16.0.0 0.15.255.255 any
access-list 112 deny ip 192.168.0.0 0.0.255.255 any
access-list 112 deny ip 127.0.0.0 0.255.255.255 any
access-list 112 deny ip 123.123.0.0 0.0.255.255 any
```

Note that EXTENDED ACL's have a number in the range 100-199
We block all **ip** traffic from these source addresses to **any** of our hosts

```
access-list 112 permit ip any any
```

'implicit deny' also works with EXTENDED ACL's

Source routing

Normally, the routers between the source and destination determine the route a packet takes from its source to its destination. But with 'source routing' the sender of a packet can include information in the packet that determines the route the packet should take to reach its destination. There are many ways in which this function can be (ab) used by an attacker. For example, the packet can be routed through a network we trust and exploit this trust relationship.

In practice, source routing is used very little. The main legitimate use is for debugging network problems or routing traffic over specific links for congestion control in special situations. It is unlikely that we will have to debug the external network so turning off source routing should not cause any problems.

The actual implementation is rather simple. We just execute the following command on our border routers (from privileged or 'enable' mode):

```
no ip source-route
```

2. Login services

According to the security policy, login services through the network should be limited to machines on the internal network. Remote login services are very powerful since they extend full user privileges to the remote user. In addition, many of these services either transmit passwords in clear text or even allow unauthenticated access (from 'trusted' hosts). Since we have no control over the external (public) network, we should prevent this information from ever leaving our local network. Clear text passwords can easily be intercepted ('sniffed') and trust relationships are often used to gain access to restricted systems.

The impact of this aspect of the security policy is rather drastic. Users will not be able to access their machines from remote locations. I would anticipate major protests and prepare the helpdesk for a flood of complaints!

Since we are restricting these services to the internal network, we can simply block access (for these services) to and from all machines at the border routers (internal traffic is not routed through the border routers and are therefore still permitted).

This filter requires and EXTENDED ACL since we want to filter based on tcp ports. Hence, we have to specify the protocol ('tcp') and port number ('eq xx'). For some services we could use the name instead of the port number; I am old fashioned and my ACL reads:

	Disable (from **`any`** host outside to **`any`** host inside)
`access-list 112 deny tcp any any eq 21`	ftp
`access-list 112 deny tcp any any eq 22`	ssh
`access-list 112 deny tcp any any eq 23`	telnet
`access-list 112 deny tcp any any eq 139`	NetBIOS
`access-list 112 deny tcp any any eq 512`	rexec
`access-list 112 deny tcp any any eq 513`	rlogin
`access-list 112 deny tcp any any eq 514`	rshell

3. RPC and NFS

The Remote Procedure Calls combine a whole collection of services can be accessed remotely. Many services use a portmapper (portmap or rpcbind) to bind the service to a particular port.
When an RPC server is started, it will tell portmap what port number it is listening to and what RPC program numbers it is prepared to serve. Portmap then intercepts all client requests and forward the packets (locally) to the appropriate server. Some of these services are powerful in their own right, but the most important reason for shutting them down is the large number of buffer overflows that have been found in these programs. Buffer overflows allow users to gain root access; and we don't want to

hand over the control over our machines.

The portmapper actually simplifies the blocking of these powerful services, since we only have to block the ports that the portmapper is listening to: 111/tcp and 111/udp.

The Network File System is an elegant way to share files between different computers. Whole directories can be mounted to different machines with full privileges. The access rights are given without explicit authentication, a simple configuration file lists the directories are accessible to certain 'trusted' hosts. This leaves many open doors for intruders, so we want to limit access to our internal network.

For this service, we have to block two more ports for both tcp and udp: 2049 and 4045.

Note that these services run on **tcp** and **udp**

portmap/rpcbind

```
access-list 112 deny tcp any any eq 111
access-list 112 deny udp any any eq 111
```

NFS

```
access-list 112 deny tcp any any eq 2049
access-list 112 deny udp any any eq 2049
```

lockd

```
access-list 112 deny tcp any any eq 4045
access-list 112 deny udp any any eq 4045
```

4. NetBIOS in Windows NT and Windows 2000

The Network Basic Input/Output System is a suite of programs that allows applications on different computers to communicate within a local area network (LAN). Originally developed by IBM it is now the networking standard implemented by Microsoft.

As the definition implies, no NetBIOS packets should leave the local network. And since many Windows machines are poorly configured (e.g. allowing anonymous access by default) exploiting unprotected shares has become a major pastime of attackers. There is absolutely no need to share local files and printers with users on the public network.

NetBIOS uses a number of ports, which we disable from all hosts outside our network:

```
access-list 112 deny tcp any any eq 135
access-list 112 deny udp any any eq 135
access-list 112 deny udp any any eq 137
access-list 112 deny udp any any eq 138
access-list 112 deny tcp any any eq 139
access-list 112 deny tcp any any eq 445
access-list 112 deny udp any any eq 445
```

5. X-Window

The X-Window server is a very useful system, but unfortunately it has some major security flaws. In the typical application it allows remote users to write to the local display. This is very convenient e.g. if you have remote access to a machine and want to use two X-terminals on the same display.

But if remote users can gain access to a workstations' X-display they can also monitor keystrokes that a user enters, download copies of the contents of their windows, etc.

The X-Window server listens to ports number 6000/tcp, 6001/tcp, ... depending on the number of X-displays available at the host system. Typically this number is small (<10) but in principle the server is capable of listening to all ports up to 6255/tcp. In order to protect all possible X-Window connections, we have to exclude the entire range of port numbers.

Luckily, the syntax for EXTENDED ACL's allows this with one single command:

```
access-list 112 deny tcp any any range 6000 6255
```

6. Naming and directory services

The Domain Name Service is an essential service for the functionality of the Internet. It translates host names such as www.sans.org into their 'real' address (167.216.133.33). In order for the world to resolve our own addresses we have to offer this service. And with a large number of local users we should also provide a name server to resolve external addresses.

Unfortunately, the program running on most name servers is notorious for the buffer overflows that allow immediate root access to the attacker. It is for this reason that the Domain Name Service has made it to the top of the SANS Top Ten list.

One of the most dangerous features of name servers is their ability to share all their information with a secondary ('backup') name server. This is a necessary and very useful feature since it ensures that our public addresses can be resolve even if we have a local problem with our name server. However, if we don't configure our name server properly, it might give this information not only to our trusted secondary name server but also to any hosts that requests this information. There it is a good idea to restrict this particular service (call a 'zone transfer') to only our secondary name server(s).

The Lightweight Directory Access Protocol (LDAP) allows remote users to access directory services. It is an other fast method to provide a lot of information about your organization. In addition to this problem, there are buffer overflows known for LDAP servers. Since we cannot expect that all servers are running secure, patched versions, we better disable the service entirely. (Of course this service is still available within the local network).

ACL's operate on a 'first match' principle: if the conditions of an ACL entry are fulfilled the required action (deny or permit) is applied to the current packet. If we first *permit* access to all allowed hosts and then *deny* access to 'everybody' (else), the allowed hosts are not affected by the subsequent 'deny'.

Assume that we have two name servers with the addresses 123.123.10.10 and 123.123.10.11 and that our external secondary name server is 167.216.133.33 (a very trusted system :-)

The ACL for restricting DNS access would then look like this:

```
access-list 112 permit udp any host 123.123.10.10 eq 53
access-list 112 permit udp any host 123.123.10.11 eq 53
```

allow dns client requests to our name

```
                                                                                servers
                                                                                but to no
access-list 112 deny udp any any eq 53                                          other host

                                                                                Allow
access-list 112 permit tcp host 167.216.133.33 host 123.123.10.10 eq 53         zone
access-list 112 permit tcp host 167.216.133.33 host 123.123.10.11 eq 53         transfers
                                                                                from the
                                                                                secondary
                                                                                name
                                                                                server
                                                                                ...only!
access-list 112 deny tcp any any eq 53

                                                                                Disable
access-list 112 deny tcp any any eq 389                                         LDAP
access-list 112 deny udp any any eq 389                                         from
                                                                                outside
```

7. Mail

It is somewhat surprising that a service as old and important as email (can anybody even remember a life before email?) still suffers from poorly written programs. But this is exactly the situation with 'sendmail', the most commonly used mail server program. 'Poorly written' is probably too strong a criticism the problem is that sendmail is often just 'too nice'. It will do everything possible to make sure the mail gets to the intended destination. And in doing so it will often use your resources for other purposes than just accepting and delivering your mail.

While patches for e.g. sendmail are made available frequently, there are still many servers running old versions that have known exploits, again often utilizing buffer overflows. But this is by no means the only risk of running a mail server: often an 'open' configuration allows relaying of any mail that reaches the host; in this case you can be sure that your site will soon be known as a 'spam amplifier' sending thousands of unwanted messages to annoyed users all over the globe (with your address for in the header).

The best strategy is to limit the number of mail servers within your organization and make sure that they are patched and configured 'up to date'.

Mail servers have two serve to masters: on the one hand they receive mail from all possible hosts form both your local and the public network; on the other end, they have to deliver them to the local recipients. POP and IMAP are two different protocols that allow (local) users to access their emails on the mail server. While it might be convenient to be able to read emails from abroad, our security policy establishes that these services should only be available from within our local network.

To implement this policy we use the same approach as with the name servers: first we explicitly permit access to the dedicated hosts, and then we deny access to all (i.e. the other) hosts. POP and IMAP are disabled for all external hosts.

Assume we have only three external mail servers with the addresses 123.123.10.20, 123.123.10.21 and 123.123.10.22. The ACL for this setup reads:

```
access-list 112 permit tcp any host 123.123.10.20 eq 25
access-list 112 permit tcp any host 123.123.10.21 eq 25
access-list 112 permit tcp any host 123.123.10.22 eq 25
access-list 112 deny tcp any any eq 25
```

only to our external mail servers

```
access-list 112 deny tcp any any eq 109
access-list 112 deny tcp any any eq 110
access-list 112 deny tcp any any eq 143
```

Disable POP (109 and 110/tcp) and IMAP (143/tcp) from external hosts

8. Webserver

The only Internet application that is more popular than email is surfing the web. Since webservers nowadays come preinstalled with (or are easily available for) many operating systems it is tempting to just let everybody put up their own webserver. However, there is a magnitude of vulnerabilities associated with the operation of a webserver. Again, buffer overflows are known for some servers, but more importantly, there is a million ways to configure a webserver insecurely. This begins with improper permissions on directories (allowing the surprised surfer to browse through all your files) and goes on to vulnerable cgi scripts. The number of known exploits is countless and constantly increasing.

As with electronic mail, the best advise is to centralize the service and (try to) make sure that these webservers (and their content) are as secure as necessary.

While the Hypertext Transfer Protocol (http) is used for most web-based applications, SSL (Secure Socket Layer) is being used for encrypted communications. We implement the restrictions for both protocols and also block access to some common proxy ports.

Let's say our external web servers have the addresses 123.123.10.30, 123.123.10.31, ... to 123.123.10.37. Then we can restrict access with the following ACL:

```
access-list 112 permit tcp any host 123.123.10.30 eq 80
access-list 112 permit tcp any host 123.123.10.31 eq 80
access-list 112 permit tcp any host 123.123.10.32 eq 80
access-list 112 permit tcp any host 123.123.10.33 eq 80
access-list 112 permit tcp any host 123.123.10.34 eq 80
access-list 112 permit tcp any host 123.123.10.35 eq 80
access-list 112 permit tcp any host 123.123.10.36 eq 80
access-list 112 permit tcp any host 123.123.10.37 eq 80
access-list 112 deny tcp any any eq 80
```

allow http only to our (external) web servers

```
access-list 112 permit tcp any host 123.123.10.30 eq 443
access-list 112 permit tcp any host 123.123.10.31 eq 443
access-list 112 permit tcp any host 123.123.10.32 eq 443
access-list 112 permit tcp any host 123.123.10.33 eq 443
access-list 112 permit tcp any host 123.123.10.34 eq 443
access-list 112 permit tcp any host 123.123.10.35 eq 443
access-list 112 permit tcp any host 123.123.10.36 eq 443
access-list 112 permit tcp any host 123.123.10.37 eq 443
access-list 112 deny tcp any any eq 443
```

apply the same restriction for ssl

```
access-list 112 deny tcp any any eq 8000
access-list 112 deny tcp any any eq 8080
access-list 112 deny tcp any any eq 8888
```

Block known HTTP proxy ports

9. "Small services"

The so-called "small services" reside on the ports below 20/tcp and 20/udp. They include services like 'echo', 'chargen' or 'daytime'. These services can be very helpful for trouble shooting network problems. In a time where everybody on the Internet played according to the rules, these services would be used for their intended purposes and nobody had to worry about them.

Unfortunately times have changed and many attackers use these services to either probe or flood our networks. A well-known DoS attack uses the 'chargen' service of one machine and the 'echo' service of another machine to generate never-ending traffic. Since this phenomenon is well known, there are two simple commands to disable these services altogether:

```
no service udp-small-servers
no service tcp-small-servers
```

We also have to manually disable one more 'small' service ('time' 37/tcp and 37/udp):

```
access-list 112 deny udp any any eq 37
access-list 112 deny tcp any any eq 37
```

10. Miscellaneous

The security policy list a number of further services that should not be accessible from the external network:

- **TFTP (69/udp)**

  The 'trivial file transfer protocol' allows transferring files between computers. In contrast to 'ftp' it uses the udp protocol, which is stateless but can be faster under certain circumstances. While there might be a justification to use 'tftp' within a LAN, it is definitely obsolete for transfers over the public network.

- **finger (79/tcp)**

  This service allows a remote user to query information about the local users. This information can include their real names, phone and room numbers and even projects they are working on. In short it provides a lot of information that you probably don't want to share.

- **NNTP (119/tcp)**

  The 'Net News Transfer Protocol' allows users to post and retrieve messages from central message boards. If you decide to offer this service at all, it should be run from a dedicated (and properly configured) server, and access to it should probably be restricted to internal users.

- **NTP (123/tcp)**

  'Network Time Protocol' provides the accurate time. It is essential to synchronize your hosts,

e.g. if they are accessing the same data or if you rely on the exact time for a central authorization service. In a large organization it might necessary to deploy a number of different ntp servers to guarantee the availability of the service. But unless you are a very large organization that offers this service to a part of the public network, you should restrict access to this service too to internal users only.

- **LPD (515/tcp)**

  The 'line printer daemon' listens to print requests from remote hosts. It could run on a 'fully networked' printer or on a host that offers this service for locally connected printer. Either way, if we don't allow our own users to login to their accounts from the external network, we definitely don't want anybody to remotely print on our printers.

- **syslogd (514/udp)**

  Syslogd is a very useful tool that allows logging many different events on a large number of hosts on a central logging facility. This is very convenient if you have to administer a large number of hosts. The only(?) problem with syslogd is the fact that it accepts connections from any host. If you allow access to external hosts they could easily launch a Denial of Service attack by flood your logging facility.

- **SNMP (161and 162, tcp and udp) and BGP (170/tcp)**

  The 'Simple Net Management Protocol' and the 'Border Gateway Protocol' are used to communicate with and configure network devices such as routers or switches. Nobody from the outside should even know the IP addresses of these devices; much less have access to them.

- **SOCKS (1080/tcp)**

  A Socks proxy server allows multiple hosts to share a common network connection. It is typically used in home environments to share the access cost for the network. The problem with SOCKS is that is a symmetrical proxy: just as it allows internal machines access to the public network, it will possibly allow any host on the public net to 'share' your access.

  This will allow external users to use hosts on your local area network as the 'source' for their traffic.

The following entries to our growing EXTENDED ACL block these services:

```
access-list 112 deny udp any any eq 69
access-list 112 deny tcp any any eq 79
access-list 112 deny tcp any any eq 119
access-list 112 deny udp any any eq 123
access-list 112 deny tcp any any eq 123
access-list 112 deny udp any any eq 161
access-list 112 deny tcp any any eq 161
access-list 112 deny udp any any eq 162
access-list 112 deny tcp any any eq 162
access-list 112 deny udp any any eq 179
access-list 112 deny tcp any any eq 179
```

```
access-list 112 deny udp any any eq 514
access-list 112 deny tcp any any eq 515
access-list 112 deny tcp any any eq 1080
access-list 112 deny udp any any eq 1080
```

11. ICMP

The Internet Control Message Protocol is (supposed to be) used to troubleshoot network connections. A remote user can establish if a local host is running, which protocols and ports are available, if source routing is allowed, if there was a fragmentation error, and many more rather detailed information about your local host. While all these messages have very legitimate purposes it is no longer a good idea to volunteer this information to just everybody who asks for it.

As a first step, we have to make sure that no icmp requests can reach our hosts from the public network. This is done by blocking the icmp 'echo' command:

```
access-list 112 deny icmp any any eq echo
```

But some of these messages are sent as a result of a failed connection attempt. These 'error messages' can be very specific and we want to disable that service to external hosts.

The security policy therefore also requires blocking some outgoing icmp packets.

Since these blocks have to be applied to outbound traffic, we also have to define an EGRESS access list on the border routers. And because we want to filter based on protocol (and even more specific on service) we have to use an EXTENDED ACL.

So far I have been using the actual port numbers to describe the filters in the access list. This could be also done for ICMP services, but it seems to be more common to use the service type instead. ICMP messages are identified by a 'type' and a 'code' field (RFC792 and others). By blocking the type of service we automatically block all codes within that type.

The required blocks are realized by the following (short) EGRESS ACL. We first explicitly deny three services and than permit all (others). Note that we have to use a different access-list number for this ACL:

```
access-list 122 deny icmp any any eq echo-reply
access-list 122 deny icmp any any eq time-exceeded
access-list 122 deny icmp any any eq unreachable
access-list 122 permit ip any any
```

Putting it all together

The security policy contains a long list of required blocks. It is by no means comprehensive but constitutes a minimal requirement. In addition to the filters required by the security policy I would recommend to

- extend the EGRESS ACL to block spoofed addresses from leaving your network
- disable all direct broadcasts (`no ip direct-broadcast`)
- protect the router itself (`no ip bootp server, no ip http server, no service finger, banner incoming`) and limit access to it (define `access-class`, secure snmp with a community name other than 'public' or 'private')

- enable logging to syslog. I include logging of the most important blocks in the listing below. This is not part of the requirements, but I think it is indispensable. We log all denied packets explicitly; the packets to hosts other than the announced external hosts (web server, mail server, DNS server) are logged with the line 'permit ip any any log' at the end of the ACL. (The address of the syslog server is assumed to be 123.123.100.1)

These recommendations cover only the most important additions. Two excellent prototype ACL's can be found at:

http://geek-speak.net/papers/access_lists.html

and: http://www.pasadena.net/cisco/secure.html.

In designing the final version of our ACL we have to keep three things in mind:

1. Readability
   Our ACL will be quite long. If cannot remember why we put one or the other line in there, it should at least be possible to find out again later.

2. Performance
   ACL's are examined from top to bottom. Each logical operation costs CPU time, which will slow down the router and consume resources. Therefore the filer rules for the most common packets should be as high in the ACL as possible.

3. Order dependence
   The order of some filter rules is critical. We used the combination of 'specific permit'/'general deny' to regulate access to some of our servers (dns, mail web). It is essential that these rules be not preceded by any rule that would allow actually permitted traffic.

I tried for the best possible compromise of these conflicting requirements for the INGRESS ACL:

First we have to deny access to the 'invalid' source addressed (spoofing). Otherwise they might be allowed by a subsequent 'permit any' rule.

Then we apply the rules for the most frequent services: www and mail. (Within these blocks the order is exactly as described above.)

We continue with the third service that has to be restricted to dedicated servers, dns.

At this point, the order of the remaining blocks doesn't matter anymore. We could probably do some fine-tuning according to the expected use of one or the other server. Or we could try to group the filters according to our security policy. I decided to go with readability: the remaining filters are listed in increasing port number.

The following list could be used to 'cut and paste' program the order routers. For each of the routers we have to login and enter the privileged mode ('enable'). In additions to the filter rules described in the preceding sections I added the commands necessary to actually activate the ACL's. The list contains some comments, indicated by the exclamation mark (!), the standard "comment" command for Cisco routers.

All 'local' IP addresses ('123.123.X.Y') are fictitious.

167.216.133.33 is **not** the address of our secondary name server (it is the name server of SANS)

Please substitute these addresses with the correct addresses of your local network.

```
! set some options
   logging 123.123.100.1
   no ip source-route
   no service udp-small-servers
   no service tcp-small-servers

! reset ACL 112 (INGRESS)
   no access-list 112

! block invalid addresses
   access-list 112 deny ip 10.0.0.0 0.255.255.255 any log
   access-list 112 deny ip 172.16.0.0 0.15.255.255 any log
   access-list 112 deny ip 192.168.0.0 0.0.255.255 any log
   access-list 112 deny ip 127.0.0.0 0.255.255.255 any log
   access-list 112 deny ip 123.123.0.0 0.0.255.255 any log

! 'www' (allow access to our external web servers 123.123.10.30-37)
   access-list 112 permit tcp any host 123.123.10.30 eq 80
   access-list 112 permit tcp any host 123.123.10.31 eq 80
   access-list 112 permit tcp any host 123.123.10.32 eq 80
   access-list 112 permit tcp any host 123.123.10.33 eq 80
   access-list 112 permit tcp any host 123.123.10.34 eq 80
   access-list 112 permit tcp any host 123.123.10.35 eq 80
   access-list 112 permit tcp any host 123.123.10.36 eq 80
   access-list 112 permit tcp any host 123.123.10.37 eq 80
   access-list 112 deny tcp any any eq 80 log
   access-list 112 permit tcp any host 123.123.10.30 eq 443
   access-list 112 permit tcp any host 123.123.10.31 eq 443
   access-list 112 permit tcp any host 123.123.10.32 eq 443
   access-list 112 permit tcp any host 123.123.10.33 eq 443
   access-list 112 permit tcp any host 123.123.10.34 eq 443
   access-list 112 permit tcp any host 123.123.10.35 eq 443
   access-list 112 permit tcp any host 123.123.10.36 eq 443
   access-list 112 permit tcp any host 123.123.10.37 eq 443
   access-list 112 deny tcp any any eq 443 log

! SMTP (allow access to our external mail servers 123.123.10.20-22)
   access-list 112 permit tcp any host 123.123.10.20 eq 25
   access-list 112 permit tcp any host 123.123.10.21 eq 25
   access-list 112 permit tcp any host 123.123.10.22 eq 25
   access-list 112 deny tcp any any eq 25 log

! DNS (allow access to our external DNS and zone transfer to 167.216.133.33)
   access-list 112 permit udp any host 123.123.10.10 eq 53
   access-list 112 permit udp any host 123.123.10.11 eq 53
   access-list 112 deny udp any any eq 53 log
   access-list 112 permit tcp host 167.216.133.33 host 123.123.10.10 eq 53
   access-list 112 permit tcp host 167.216.133.33 host 123.123.10.11 eq 53
   access-list 112 deny tcp any any eq 53 log

! block all tcp/udp to these ports
   access-list 112 deny tcp any any eq 21 log
   access-list 112 deny tcp any any eq 22 log
   access-list 112 deny tcp any any eq 23 log
   access-list 112 deny udp any any eq 37 log
```

```
   access-list 112 deny tcp any any eq 37 log
   access-list 112 deny udp any any eq 69 log
   access-list 112 deny tcp any any eq 79 log
   access-list 112 deny tcp any any eq 109 log
   access-list 112 deny tcp any any eq 110 log
   access-list 112 deny tcp any any eq 111 log
   access-list 112 deny udp any any eq 111 log
   access-list 112 deny tcp any any eq 119 log
   access-list 112 deny udp any any eq 123 log
   access-list 112 deny tcp any any eq 123 log
   access-list 112 deny tcp any any eq 135 log
   access-list 112 deny udp any any eq 135 log
   access-list 112 deny udp any any eq 137 log
   access-list 112 deny udp any any eq 138 log
   access-list 112 deny tcp any any eq 139 log
   access-list 112 deny tcp any any eq 143 log
   access-list 112 deny udp any any eq 161 log
   access-list 112 deny tcp any any eq 161 log
   access-list 112 deny udp any any eq 162 log
   access-list 112 deny tcp any any eq 162 log
   access-list 112 deny udp any any eq 179 log
   access-list 112 deny tcp any any eq 179 log
   access-list 112 deny tcp any any eq 389 log
   access-list 112 deny udp any any eq 389 log
   access-list 112 deny tcp any any eq 445 log
   access-list 112 deny udp any any eq 445 log
   access-list 112 deny tcp any any eq 512 log
   access-list 112 deny tcp any any eq 513 log
   access-list 112 deny tcp any any eq 514 log
   access-list 112 deny udp any any eq 514 log
   access-list 112 deny tcp any any eq 515 log
   access-list 112 deny tcp any any eq 1080 log
   access-list 112 deny udp any any eq 1080 log
   access-list 112 deny tcp any any eq 2049 log
   access-list 112 deny udp any any eq 2049 log
   access-list 112 deny tcp any any eq 4045 log
   access-list 112 deny udp any any eq 4045 log
   access-list 112 deny tcp any any range 6000 6255 log
   access-list 112 deny tcp any any eq 8000 log
   access-list 112 deny tcp any any eq 8080 log
   access-list 112 deny tcp any any eq 8888 log

! icmp
   access-list 112 deny icmp any any eq echo log

! allow everything else
   access-list 112 permit ip any any log

! reset ACL 122 (EGRESS)
   no access-list 112

! icmp
   access-list 122 deny icmp any any eq echo-reply log
   access-list 122 deny icmp any any eq time-exceeded log
   access-list 122 deny icmp any any eq unreachable log

! allow everything else
   access-list 122 permit ip any any log

! 'S0' is our external interface
   int s0
```

```
ip access-group 112 in
ip access-group 122 out
exit
```