Extended Access Control Lists and Domains

How it works

Extended access control lists can filter by more criteria than just source IP address: source and destination IP, protocol, port. In other words: filtering by various information in the L3 and L4 headers.

Unlike standard ACLs, they are placed as close to the source as possible so that the forbidden traffic puts as little load on the network as possible.

Creating an ACE of a numbered extended ACL:

```
access-list 105 permit protokol source-ip dest-ip eq dest-port
access-list 105 permit protokol source-ip eq source-port dest-ip eq dest-port
access-list 105 permit tcp source-ip dest-ip dest-port established
```

Instead of the address, we can specify `any` if this parameter is not to be distinguished. The options for the protocol are identified by a question mark (tcp, udp, icmp, smtp, ftp, ftp-data,...). Ports can be specified by number or by name (e.g. http 80, https 443 – options are again indicated by a question mark).

The ACE `deny ip any any` is assumed at the end, however it may be practical to state explicitly.

Creating a named extended ACL:

```
ip access-list extended SOME_NAME
      permit tcp……
      exit
```

If we add the keyword `established` to the end of the ACL, the match is evaluated only if the corresponding TCP segment has the ACK or RST bit set, which are not present in the first TCP segment in the handshake (for the UDP protocol there is no sense, there is no handshake). So if we want to make it harder for hackers to try to establish a connection to our part of the network where we have no servers, we can set:

```
permit tcp any eq 80 protected.network.address established
```

or for the opposite direction in the port:

```
permit tcp protected.network.address any eq 80 established
```

and we disable the rest of the traffic, we set this ACL on the input interface from the "unsafe" network or the output interface to the "protected" network. The consequence is that here specifically, connections to web servers can be established from the protected network to the outside, but not to the inside.

The following entry is very often the first ACE into a serverless network:

```
permit tcp any any established
```

Which means that traffic that has been established from the internal network out will pass quickly and without problems, and only the next will be filtered in more detail. Beware of UDP traffic, if we didn't allow it with some other rule, a lot of multimedia traffic wouldn't work.

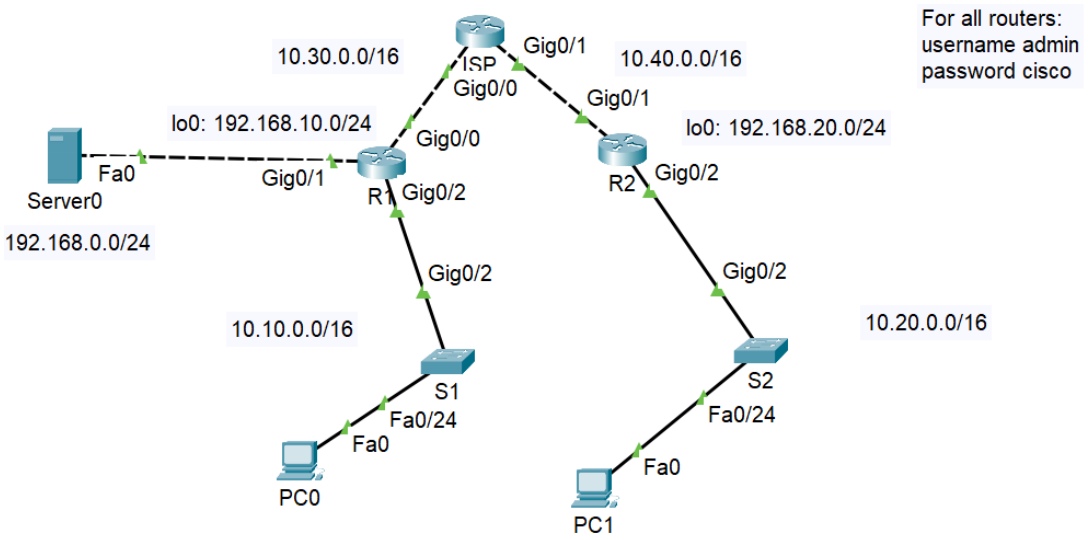
The default "prohibition of others" can be explicitly stated as follows:

```
deny ip any any
```

This means that all IP traffic (that is, what all TCP and UDP segments are encapsulated in) will be disabled. Only what remains at the L2 layer, such as ARP, CDP, and others, will pass.

Task:

On the web, there is a pre-prepared PKT file for standard ACLs. There are three routers, two switches, one server and two computers. Go through their settings, see where SSH access is enabled. If SSH is not enabled on R1 or R2, fix it.



On a computer with the address 10.10.0.11, try to ping the other computer (10.20.0.11). Will it work?

Set up ACLs on the network as follows:

- It will be a named extended ACL called WEB_ACCESS.
- A computer with the address 10.10.0.11 can access the R2 router via SSH.
- Web traffic from the 10.10.0.0/16 network can go anywhere.

On a computer with address 10.10.0.11, try to ping the other computer (10.20.0.11). Does it work?

Next, display the website 192.168.0.11 in a web browser on the same computer. Does it work?

Try SSH access to routers R1 and R2 on the same computer. Which one works? They are the following:

```
ssh -l admin 10.40.0.2
ssh -l admin 10.10.0.1
```

ICMP Filter

There are several "ports" for ICMP (although of course it is not about ports, but rather about ICMP types):

- unreachable
- time-exceeded
- echo
- echo-reply
- parameter-problem
- source-quench

The latter two types are important, for example, when determining the MTU on the path.

So if we want to enable all but echo, we can either list them in individual ACEs, or it can be like this:

```
deny icmp any any echo
permit icmp any any
… others
```

Create an ACL (numbered with 102) on router R2 to filter traffic coming from the ISP as follows:

- allow all TCP traffic that has been established from internal networks,
- allow ICMP echo reply and miscellaneous ICMP error detecting traffic (list in individual ACEs),
- disallow all other IP traffic.

Now you will probably need to remove the ACL on router R1.

Test the ping to and from the protected network, then write down the statistics.

A few Recommendations on ACLs

First, a general comment on ACLs: various devices usually support both named and numbered lists, but this need not be the rule. Exceptionally, we can find a device (not so much a router as a hardware firewall) where some of this "doesn't work".

In the examples (not only here) we usually see IP addresses, but in reality, name addresses can also be used. For example:

```
access-list 104 permit tcp any host web.domena.cz eq www
```

However, the device on which we use something like this must be able to translate the name.

There are some recommendations in the ACL about the ordering of ACE entries:

- we take into account that the ACEs in the list are processed from top to bottom, i.e. more specific items should come first and then more general ones,
- usually permit type entries first, but it depends on subsets/supersets of networks: for example, we may want some settings to apply to some devices on a given network, but other settings to apply to the rest of the network, so we list the subset first),
- if it doesn't "fight" with the previous item, we place first what will be used more often.

What about the second point, how do we determine the typical frequency of use of specific ACEs? Simply by first deploying the ACE entries by guessing, but after a certain amount of time of use, we can view the statistics using show access-list. For each ACE, the number of "hits" will be listed, and we'll rearrange accordingly. This improves the throughput of filtering, and also saves the computational resources of the device.

There is one most important rule: clarity. If the application of the above rules reduces the clarity, it doesn't matter for a list with a few items, but for a list with dozens or hundreds of items it can matter very significantly.

DNS Resolution Locally on the Router

We can enable local name translation directly on the router. For example:

```
ip host webserver 192.168.25.4
...
ping webserver
```

Then we can use this name address instead of the IP address in various commands (not only in ping). However, this binding will really only be visible locally on this device.

We can also set the name server address:

```
ip name-server some.ip.adress
```

If we have been playing with ACLs, we need to enable UDP traffic on port 53 (i.e. "domain"):

```
access-list 105 permit udp any any eq domain
access-list 105 permit udp any eq domain any
```

For security reasons, on the other hand, DNS resolution is often disabled if we make a mistake when entering a command:

```
no ip domain-lookup
```

Task:

Try this: if we create a name for a server and then disable domain lookup, will the ping to the created name work?