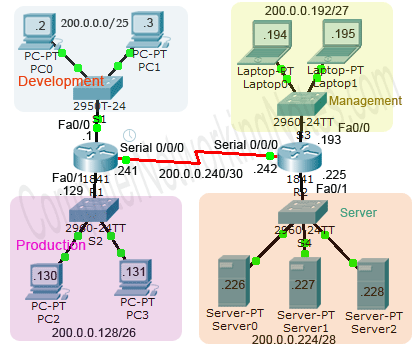
**Standard Access Control List**

A standard ACL can be used for several purpose. In this tutorial we will see how it can be used in controlling the unwanted network traffic. With standard ACL, we can define certain conditions for the network traffic passing through the router. Once defined, Standard ACL works like a gate keeper that will allow only the authorized people (packets). All unwanted people (packets) are kicked out from the gate.

For demonstration purpose I will use packet tracer network simulator software. You can use it or can use any other network simulator software such as Boson, NetSim, GNS etc.

Create a topology as illustrate in following figure.



This network is built with single class C IP address 200.0.0.0/24. Through VLSM network is divided in following sections:-

* Development (200.0.0.0/25)
* Production (200.0.0.128/26)
* Management (200.0.0.192/27)
* Server (200.0.0.224/28)

These sections are connected via two routers. Routers are running RIVv2 routing protocol.

**VLSM Chart for Subnetted networks**

| Block size | Slash notation | Interface | Network address | Subnet mask | Wildcard mask |
| --- | --- | --- | --- | --- | --- |
| 128 | /25 | Fa0/0 (R1) | 200.0.0.0 | 255.255.255.128 | 0.0.0.127 |
| 64 | /26 | Fa0/1 (R1) | 200.0.0.128 | 255.255.255.192 | 0.0.0.63 |
| 32 | /27 | Fa0/0 (R2) | 200.0.0.192 | 255.255.255.224 | 0.0.0.31 |
| 16 | /28 | Fa0/1 (R2) | 200.0.0.224 | 255.255.255.240 | 0.0.0.15 |
| 4 | /30 | Serial 0/0/0 (R1-R2) | 200.0.0.240 | 255.255.255.252 | 0.0.0.3 |

In this network, at this moment all sections are connected with each other’s. Users are able to access all resources from other sections as well as their own. You are hired to secure this network.

This network has following security requirements.

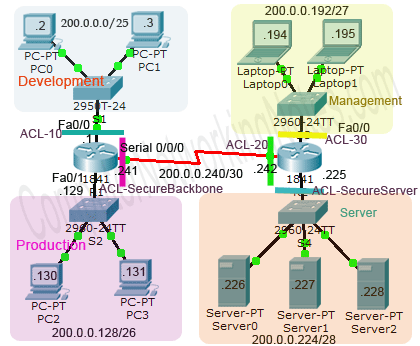
**Section level requirement**

* Development section should be able to access only production section. It should not be able to access management section and server section.
* Production section should be able to access only development section. It should not be able to access management section and server section.

**User level requirement**

* One user (PC0) from development section should not be able to access anything except its own section.
* One user (PC2) from production section should also be able to access management section but not server section.
* One user (PC3) from production section should be able to access server section but not management section.
* One user (laptop0) from management section should be able to access only Server section not the development section and production section.

For above requirements we need to secure five locations. For each location we need a separate ACL.



**As you know we can create a standard ACL in three ways:-**

1. Classic Numbered
2. Modern Numbered
3. Modern Named

To give you a better overview of these methods I will include all of them in this example.

| ACL Number / Name | ACL Type | ACL Direction | Applied Interface |
| --- | --- | --- | --- |
| 10 | Classic Numbered | Inbound | R1’s Fa0/0 |
| 20 | Modern Numbered | Outbound | R2’s Serial 0/0/0 |
| 30 | Classic Numbered | Outbound | R2’s Fa0/0 |
| SecureBackbone | Modern Named | Outbound | R1’s Serial 0/0/0 |
| SecureServer | Modern Named | Outbound | R2’s Fa0/1 |

**Understanding ACL requirements**

ACL is just like a double edge sword. We need to be extra careful while working with ACLs. A little mistake can mesh entire network data flow. Instead of creating ACL conditions directly in router, it’s always a better idea to create them in paper first. This way we can update / reorder or remove conditions without recreating entire ACL.

For example our first requirement from section level requirements says “block production department from gaining access in management section”. For this requirement we have to create a deny statement at section level. Suppose we created necessary condition for this requirement directly in router without reading remaining requirement. And later we came to know that one user from production section needs permission to access management section.

In this situation if we have created ACL directly in router using classical number method then the only way to allow this user is to delete the existing ACL and recreate it with allow statement prior to deny statement. But if we have created these conditions in paper then we could easily reorder / update /change them without recreating entire ACL. Once we are satisfy with conditions in paper, we can easily create them in router.

Okay let’s create ACL conditions from section level requirements. Our requirements are

*Development section should be able to access only production section. It should not be able to access management section and server section.*

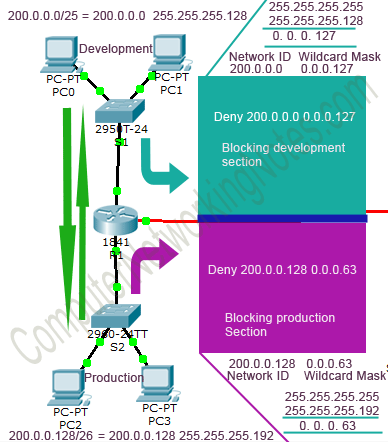
*Production section should be able to access only development section. It should not be able to access management section and server section.*

By default router does not filter any traffic unless we manually put an ACL. This behavior fulfills our half requirement. Production section and development section are able to access each other. We only need to control them from accessing management section and sever section.

In order to access Management section and Server section, both (Development and Production) section need to go through the Serial 0/0/0 interface. If we put deny condition in *SecureBackbone* ACL for development and production section, above requirements will be fulfilled.

ACL-SecureBackbone

* Deny 200.0.0.0 0.0.0.127 (*Blocking development section traffic from going outside*)
* Deny 200.0.0.128 0.0.0.63 (*Blocking production section traffic from going outside*)



Okay now let’s see our user level requirement one by one from ACLs point of view.

Our first requirement is

*One user (PC0) from development section should not be able to access anything except its own section*.

This requirement needs Inbound ACL. As user only needs to access its own section which he can access through the LAN (switch) network. This user has nothing to access from other sections. We should drop the traffic from this user as soon as it enters in the interface (Fa0/0 of R1).

**ACL-10**

* deny 200.0.0.2 0.0.0.0 (*Blocking single user from development section*)
* permit any (*allowing all remaining traffic.*)

If we do not create ***permit any*** statement then router will block all traffic coming in this interface. As we know, as soon as we create our first statement, an Implicit Deny Statement would be added automatically in the end of ACL.

Our next requirement is

*One user (PC2) from production section should also be able to access management section but not server section.*

Let’s see this requirement from ACL’s point of view:-

User belongs to Production section. Being a member of production section:-

He should be able to access Development section (*Already doing, no action is required*).

He should not be able to access Management section and Server section. (*Here group level permission is restricting user from gaining access on management section and server section. But his individual permission is allowing him to access management section.*)

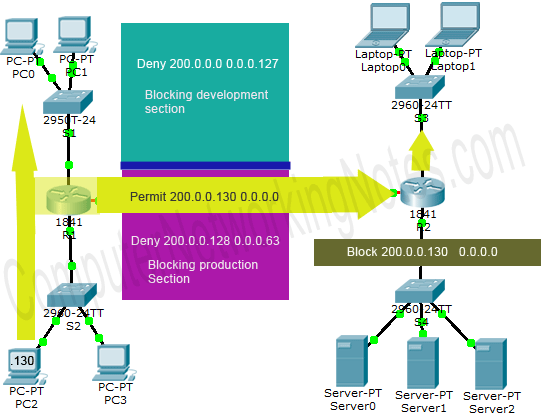
Whenever there is a conflict between User level permission and Group level permission, User level permission always override the Group level permission.

But wait…. we have already blocked group in SecureBackbone ACL at R1’s Serial 0/0/0. So how could we allow single user from group while blocking the rest?

If you are reading this article from first, then answer should have already clicked in your mind. If you are guessing about ordering of ACL then you are absolutely right. With proper ordering, we can easily achieve this goal. As we know ACL conditions are processed from top to down without skipping. Once a match found, no further conditions are processed for that packet. So if we put permit condition for this host before the deny condition for the group then *SecureBackbone* ACL will do exactly what we want.

With permit condition, we will create a window for PC2 in SecureBackbone wall. Through this window, PC2 will be able to access the sections attached with R2.

R2 has two sections; Management and Server. PC2 will be able to access both sections. But as per requirement it should be allowed to access only Management section. We need to block it from accessing server section. For this goal we need to put a deny condition in *SecureServer* ACL.



**Oaky lets update ACLs**

ACL-SecureBackbone

* Deny 200.0.0.0 0.0.0.127 (*Blocking development section traffic from going outside*)
* Permit 200.0.0.130 0.0.0.0 (*Allowing single host traffic from production section*)
* Deny 200.0.0.128 0.0.0.63 (*Blocking production section traffic from going outside*)

ACL -SecureServer

* Deny 200.0.0.130 0.0.0.0 (*Blocking single host from accessing server section*)

Our next requirement is identically same as previous requirement

*One user (PC3) from production section should be able to access server section but not management section.*

For this requirement we need a permit condition is SecureBockbone ACL and one deny condition in ACL 30 for this PC3.

ACL-SecureBackbone

* Deny 200.0.0.0 0.0.0.127 (*Blocking development section traffic from going outside*)
* Permit 200.0.0.130 0.0.0.0 (*Allowing single host traffic from production section*)
* Permit 200.0.0.131 0.0.0.0 (*Allowing single host traffic from production section*)
* Deny 200.0.0.128 0.0.0.63 (*Blocking production section traffic from going outside*)

ACL -30

* Deny 200.0.0.131 0.0.0.0 (Blocking single host from accessing management section)

Our last requirement is fairly simple.

*One user (laptop0) from management section should be able to access only Server section not the development section and production section.*

Simply creating a block condition in ACL 20 (R2’s Serial 0/0/0) will do this job.

* deny 200.0.0.194 0.0.0.0 (*Blocking single host from management section*)

We have gone through all the requirements. Let’s have quick look on ACL conditions

**ACL-10 (*Filtering incoming traffic on R1’s Fa0/0*)**

* deny 200.0.0.2 0.0.0.0 (*Blocking incoming traffic from single host*)
* permit any (*Allowing remaining all hosts.*)

**ACL-SecureBackbone (*Filtering outgoing traffic on R1’s Serial 0/0/0*)**

* deny 200.0.0.0 0.0.0.127 (*Blocking development section* )
* permit 200.0.0.130 0.0.0.0 (*Allowing single host from production section* )
* permit 200.0.0.131 0.0.0.0 (*Allowing single host from production section*)
* deny 200.0.0.128 0.0.0.63 (*Blocking production section*)

**ACL-20 (*Filtering outgoing traffic on R2’s Serial 0/0/0*)**

* deny 200.0.0.194 0.0.0.0 (*Blocking single host from management section*)
* permit any (*Allowing remaining traffic*)

**ACL-30 (*Filtering traffic going from R2’s Fa0/0*)**

* deny 200.0.0.131 0.0.0.0 (*Blocking single user from production section from gaining unauthorized on management section.*)
* permit any (*Allowing remaining traffic*)

**ACL-SecureServer (*Filtering traffic going from R2’s Fa0/1*)**

* deny 200.0.0.130 0.0.0.0 (*Blocking single user from production section from gaining unauthorized on server section.*)
* permit any (*Allowing remaining traffic*)

That’s all paper work we need to do before creating real ACLs.

Well… you may be a little bit annoyed with all above preparation. But believe me friends; it will save a lot of time and effort in Cisco exams and as well as in job life.

**Create Standard ACL**

A standard ACL can be created in two ways:-

1. Classic numbered method
2. Modern numbered or named method

Classic numbered method uses following global configuration mode command

Router(config)# access-list ACL\_Identifier\_number permit/deny matching-parameters

Modern numbered or named method uses following global configuration mode commands

Router(config)#ip access-list standard ACL\_Number / ACL\_Name

Router(config-std-nacl)#permit / deny Source Address

Router(config-std-nacl)#exit

Router(config)#

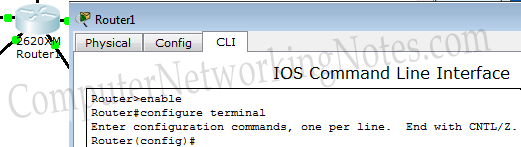
I have already explained above commands and parameters in detail with examples in previous part of this article. For this part I assume that you are familiar with above commands.

In our example we will create two ACLs (10 and SecureBackbone) in Router1 and three ACLs (20, 30 and SecureServer) in Router2.

Okay let’s create them one by one

ACL-10 (Configuration style - *Classical Numbered*)

Access CLI prompt of Router1 and enter in global configuration mode



Enter following commands

Router(config)#access-list 10 deny 200.0.0.2 0.0.0.0

Router(config)#access-list 10 permit any

Router(config)#

Great job, we have just created our first ACL with classic numbered method. Now let’s create our second ACL, but this time use modern named method.

**ACL-SecureBackbone (Configuration style – *Modern Named*)**

Router(config)#ip access-list standard SecureBackbone

Router(config-std-nacl)#deny 200.0.0.0 0.0.0.127

Router(config-std-nacl)#permit 200.0.0.130 0.0.0.0

Router(config-std-nacl)#permit 200.0.0.131 0.0.0.0

Router(config-std-nacl)#deny 200.0.0.128 0.0.0.63

Router(config-std-nacl)#exit

Router(config)#

Good going, we have finished our ACL creation task or router R1. Now access the global configuration mode of router R2 and enter following commands to create ACL20

**ACL-20 (Configuration style – *Classical Numbered*)**

Router(config)#ip access-list standard 20

Router(config-std-nacl)#deny 200.0.0.194 0.0.0.0

Router(config-std-nacl)#permit any

Router(config)#

Following commands will create ACL-30

**ACL-30 (Configuration style – *Modern Numbered*)**

Router(config)#access-list 30 deny 200.0.0.131 0.0.0.0

Router(config)#access-list 30 permit any

Router(config)#

Finally use following commands to create our last ACL-SecureServer

**ACL-SecureServer (Configuration style – *Modern Named*)**

Router(config)#ip access-list standard SecureServer

Router(config-std-nacl)#deny 200.0.0.130 0.0.0.0

Router(config-std-nacl)#permit any

Router(config-std-nacl)#exit

Router(config)#

Now our security guards (ACLs) have an authorized persons (conditions) list. Right now they are just sitting in office (router). From here they will do nothing. We need to send them on their job place (interface) where they will perform their jobs (filtrations).

**Assign Standard ACLs in interfaces**

Regardless what method we used in creating the ACLs, assigning them in interfaces are the same steps process:-

Router(config)#interface type [slot\_#] port\_#

Router(config-if)#ip access-group ACL\_# in|out

Commands and parameters are explained in previous part of this article. In this part we will use these commands in assigning the ACLs.

Let’s assign our ACLs in their respective interfaces

**ACL-10 (R1’s Fa0/0 interface, *Inbound direction*)**

Router(config)#interface fastethernet 0/0

Router(config-if)#ip access-group 10 in

Router(config-if)#exit

Router(config)#

**ACL-SecureBackbone ( R1’s Serial 0/0/0, *Outbound direction*)**

Router(config)#interface serial 0/0/0

Router(config-if)#ip access-group SecureBackbone out

Router(config-if)#exit

Router(config)#

**ACL-20 (R2’s Serial 0/0/0 interface, *Outbound direction*)**

Router(config)#interface serial 0/0/0

Router(config-if)#ip access-group 20 out

Router(config-if)#exit

Router(config)#

**ACL-30 (R2’s Fa0/0 interface – *Outbound direction*)**

Router(config)#interface fastethernet 0/0

Router(config-if)#ip access-group 30 out

Router(config-if)#exit

Router(config)#

**ACL-SecureServer (R2’s Fa0/1 interface – *Outbound direction*)**

Router(config)#interface fastethernet 0/1

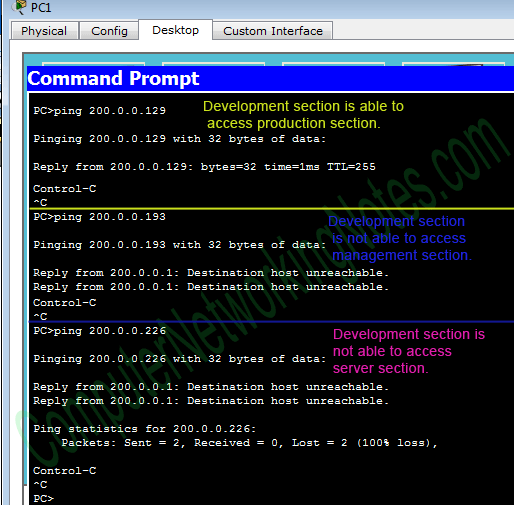
Router(config-if)#ip access-group SecureServer out

Router(config-if)#exit

Router(config)#

**Testing Standard ACLs**

To verify the implementation, we can use **ping** command. **ping** command is used to test the connectivity between source and destination. For example in following figure I tested our first requirement from PC1 (belongs to development section).

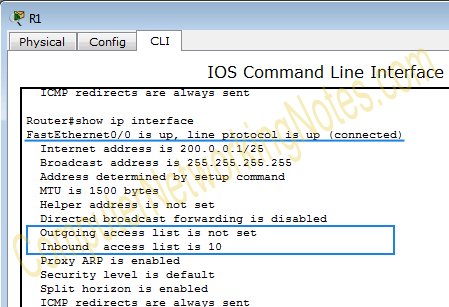


Now it’s your turn to test remaining conditions. If you have followed all above steps then requirements should be fulfilled. If you are missing any requirement or not getting result as expected, use my practice topology for cross check. You can download my practice topology from here.

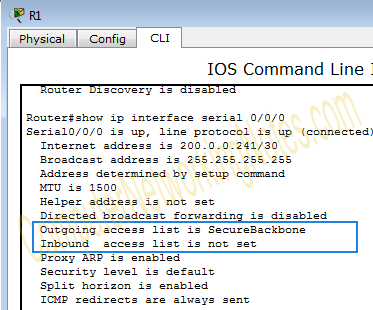
**Verifying Standard Access List configuration**

Once created and activated ACLs, we can verify them with following privilege exec mode commands.

To show which ACLs are activated on which interfaces in which direction, we can use ***show ip interface*** command

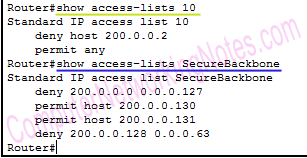


From output we can see that ACL-10 is applied in inbound direction on FastEthernet0/0. By default above command will list all interfaces. To view a single interface, we need to specify it in above command as command line option. For example, to view only serial interface use **show ip interface *serial 0/0/0*** command.

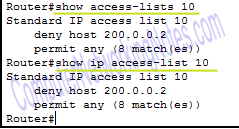


To view the conditions in ACL, we have two commands

Router# show access-lists ACL\_Number\_or\_Name (Optional, used to see the specific ACL)

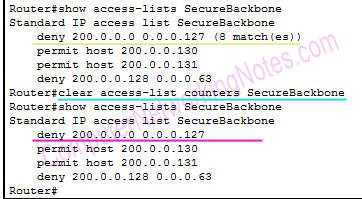


Router# show ip access-list ACL\_Number\_or\_Name (Optional, used to see the specific ACL)

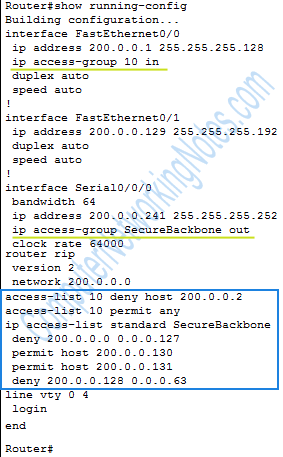


Have you notice any difference between outputs? Second command provides more detailed information about modern style ACLs. It lists the sequence number of each condition in ACL. Sequence numbers are used to edit or delete any condition from ACL. Sequence numbers are available only when you create ACL from modern style.

Router keeps track of every match on every condition. To reset this counter, use ***clear*** command.



We can also view all running configuration including ACLs from ***show running-config*** command.

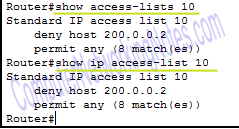


Editing / Updating Standard ACLs

We can edit or update a standard ACL only if it is created from modern configuration style. If it is created from classical configuration style then we cannot edit or update it, we can only append it.

How will I know which ACL is created from which style?

ACLs created from modern way have sequence numbers. We can use ***show ip access-list*** command to know whether a specific ACL is created from classic style or modern style. If output of this command shows sequence numbers in front of conditions then that ACL is created from modern style. For example following figure illustrates the output of show ip access-list command from router R1.



As we can see in output, ACL-10 has no sequence number while ACL-SecureBackbone has it. So ACL-10 is created from classical numbered approach while ACL-SecureBackbone is created from modern named style.

Okay now we know how to find out the configuration style of ACLs. Let’s edit them. Suppose we have two tasks, one for each ACL:-

* For ACL-10 :- Deny host 200.0.0.3
* For ACL-SecureBackbone Deny host 200.0.0.130

**For ACL-10**

As we know that this ACL is created from classical numbered method, so it cannot be edited. We have only one option, delete existing ACL and create new one with requirement.

**For ACL-SecureBackbone**

This ACL is created from modern named method. We can edit it directly. We are asked to deny the **host 200.0.0.130**, which is currently allowed (*20 permit host 200.0.0.130*).

Okay let’s update this ACL step by step.

Verify current status

Router#show ip access-list SecureBackbone

Standard IP access list SecureBackbone

10 deny 200.0.0.0 0.0.0.127

20 permit host 200.0.0.130

30 permit host 200.0.0.131

40 deny 200.0.0.128 0.0.0.63

Remove old permission

Router#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#ip access-list standard SecureBackbone

Router(config-std-nacl)#no 20

Router(config-std-nacl)#exit

Router(config)#exit

Confirm removal

Router#show ip access-list SecureBackbone

Standard IP access list SecureBackbone

10 deny 200.0.0.0 0.0.0.127

30 permit host 200.0.0.131

40 deny 200.0.0.128 0.0.0.63

Insert new condition in the place of old condition

Router#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#ip access-list standard SecureBackbone

Router(config-std-nacl)#20 deny 200.0.0.130 0.0.0.0

Router(config-std-nacl)#exit

Router(config)#exit

Verify update

Router#show ip access-list SecureBackbone

Standard IP access list SecureBackbone

10 deny 200.0.0.0 0.0.0.127

20 deny host 200.0.0.130

30 permit host 200.0.0.131

40 deny 200.0.0.128 0.0.0.63

Router#

How to delete a Standard ACL

We have two commands to delete a standard ACL.

Router(config)#no access-list [ACL\_Number] *and* Router(config)#no ip access-list standard [ACL\_Number\_or\_Name]

First command is used to delete numbered ACL while second command is used to delete both numbered and named ACLs. Let’s have an example of both commands.

Delete both ACLs from router R1.

Router(config)#no access-list 10

Router(config)#no ip access-list standard SecureBackbone

That’s all for this part. In next part of this article I will explain Extended Access List configuration commands in detail with examples.