

## Security

### 目录

Log和log-input作用 .....	1
remark .....	4
Autocommand.....	6
Privilege Levels .....	9
基于时间的ACL.....	16
Reflexive ACL.....	19
Context-Based Access Control (CBAC) .....	24
Port to Application Mapping (PAM) .....	30
Lock-and-Key Security (Dynamic ACL) .....	34
TCP Intercept.....	39
Unicast Reverse Path Forwarding (uRPF) .....	42
AAA.....	50
IP Source Tracker .....	56
Secure Shell (SSH) .....	57
Intrusion Prevention System (IPS) .....	61
Zone-Based Policy Firewall .....	66
Control Plane Policing (CoPP) .....	80

## Log 和 log-input 作用

### 概述

当路由器为用户转发了数据之后，如果管理员想查看路由器曾经为哪些用户转发过数据，在正常情况下，这是无法查证的。但是，可以通过接口配置 **ACL**，并且强制 **ACL** 记录下曾经转发过的用户记录，这样，就能从路由器得知哪些用户是发起

过数据的，并且发送了多少数据，但是用户发出的数据内容，是无法记录的。

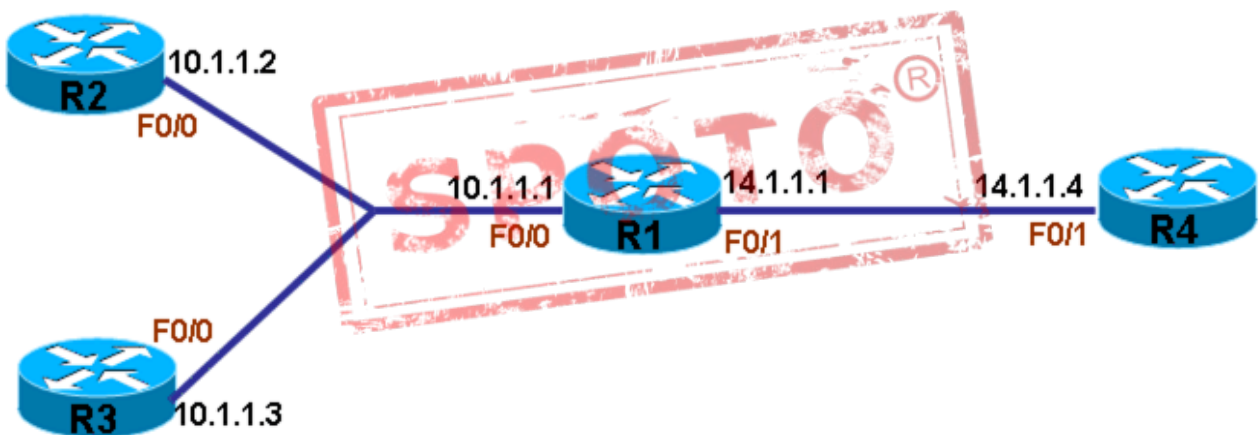
要达到以上目的，那在配置 ACL 时，使用 Log 和 log-input 的功能，并且将配置好的 ACL 用于接口上。

Log 和 log-input 的区别是：

Log 只能记录数据包通过时的源 IP 和目的 IP，

而 log-input 除了记录源 IP 和目的 IP 之外，还会记录源的 MAC 地址。

配置



## 1.配置 ACL 中的 Log

**说明：**配置路由器 R1，让其允许 R2 发来的数据包通过，但拒绝 R3 的数据包通过，并且记录下它们数据量。

### (1) 配置 ACL

**说明：**配置 ACL，允许 R2，拒绝 R3，分别使用 log 关键字

```
r1(config)#access-list 100 permit ip host 10.1.1.2 any log
```

```
r1(config)#access-list 100 deny ip host 10.1.1.3 any log
```

## (2) 应用 ACL

```
r1(config)#int f0/0
```

```
r1(config-if)#ip access-group 100 in
```

## (3) 测试结果

**说明：**从 R2 和 R3 分别 ping R4，查看 R1 上的 log

```
Oct  1 14:15:26: %SEC-6-IPACCESSLOGDP: list 100 permitted icmp 10.1.1.2 -> 14.1.1.4 (0/0), 5 packets
```

```
Oct  1 14:16:46: %SEC-6-IPACCESSLOGDP: list 100 denied icmp 10.1.1.3 -> 14.1.1.4 (0/0), 5 packet
```

**说明：**从 R1 上弹出的日志可以看出，R2 到 R4 的数据包是被放行了的，而 R3 到 R4 的数据包被丢弃了。

## (4) 查看 ACL 记录

```
r1#sh ip access-lists
```

```
Extended IP access list 100
```

```
10 permit ip host 10.1.1.2 any log (25 matches)
```

```
20 deny ip host 10.1.1.3 any log (5 matches)
```

**说明：**从 ACL 中也可以看出，R2 的流量被放行，R3 的流量被拒绝了。

## 2.配置 ACL 中 log-input

**说明：**配置路由器 R1，让其允许所有数据包通过，不仅记录下它们数据量，还将记录下源 MAC。

### (1) 配置 ACL:

说明：配置 ACL，允许所有数据包通过，并且使用 log-input 关键字

```
r1(config)#access-list 130 permit ip any any log-input
```

## (2) 应用 ACL

```
r1(config)#int f0/0
```

```
r1(config-if)#ip access-group 130 in
```

## (3) 查看 R2 的源 MAC

```
r2#show interfaces f0/0
```

FastEthernet0/0 is up, line protocol is up

Hardware is AmdFE, address is 0013.1a2f.1200 (bia 0013.1a2f.1200)

Internet address is 10.1.1.2/24

## (4) 从 R2 ping R4, 查看 R1 上的 log

```
Oct  1 14:23:21: %SEC-6-IPACCESSLOGDP: list 130 permitted icmp 10.1.1.2  
(FastEthernet0/0 0013.1a2f.1200) -> 14.1.1.4 (0/0), 1 packet
```

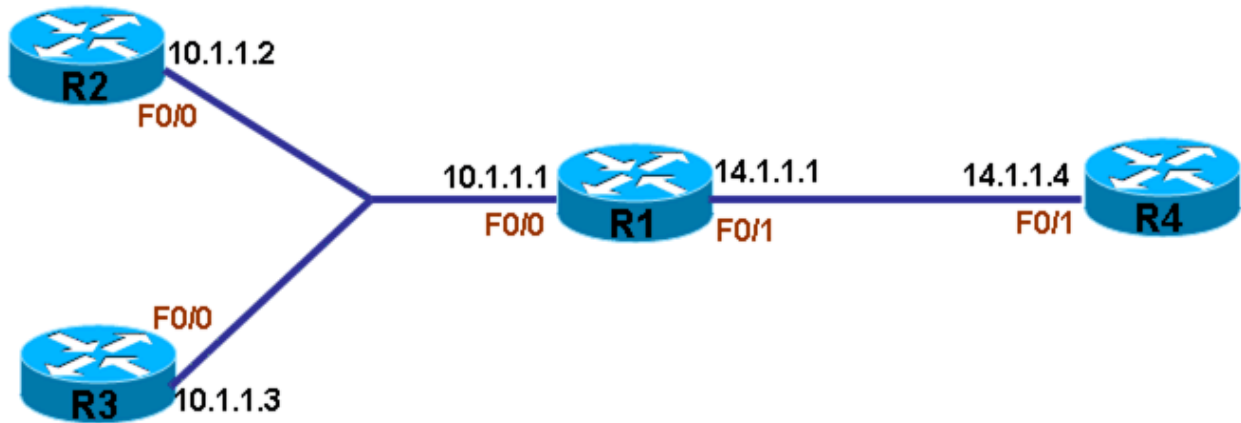
**说明：**从 R1 上弹出的日志可以看出，R2 到 R4 的数据包是被放行了的，并且还看到 R2 的源 MAC。

## remark

### 概述

在配置 ACL 时，有时因为条目太多，ACL 结构复杂，事后可能很难辨别出每条 ACL 的作用分别是什么，在这种情况下，就可以在配置 ACL 时，给 ACL 中的条目写上标记，类似于说明文字，这通过 remark 来实现，remark 可以在条目的前一行，也可以在后一行，由自己决定，但 remark 不能和条目同一行。

配置



1.配置 ACL，并使用 remark

**说明：**如上图，在 R1 上配置 ACL 时，拒绝 R2，但允许 R3，还分别为每个条目写上注释。

r1(config)#access-list 100 remark Deny\_R2                      写上拒绝 R2 的注释

r1(config)#access-list 100 deny ip host 10.1.1.2 any

r1(config)#access-list 100 remark Permit\_R3                      写上拒绝 R3 的注释

r1(config)#access-list 100 permit ip host 10.1.1.3 any

2.查看结果

**说明：**在 ACL 中是无法查看注释的，但在 running-configuration 中可以看出。

access-list 100 remark Deny\_R2

access-list 100 deny ip host 10.1.1.2 any

access-list 100 remark Permit\_R3

access-list 100 permit ip host 10.1.1.3 any

## Autocommand

### 概述

有时，当客户的网络出现故障时，需要远程工程师协助或指导客户解决故障，这时就需要远程工程师 telnet 到客户的网络设备上，但是却并不希望远程工程师去直接更改用户设备的配置，在这种情况下，就可以在用户的设备上为远程工程师配置一个用户，通过这样的用户登陆设备之后，可以自动执行远程工程师想要执行的命令，从而达到了远程工程师查看设备配置的目的，又不违反修改配置的规矩。

要实现这样的功能，就可以在设备上配置 Autocommand 的功能，这样，当相应的用户 telnet 到设备时，就可以自动执行其想要的命令。

这样的 Autocommand 可以配置为所有 VTY 连上来的人执行，即配置在 VTY 接口下，也可以单独为某个用户执行，即配置在用户名之后。但这样的命令都只能执行一条。

### 配置



#### 1.在 VTY 下为所有用户配置自动执行命令

(1) 在 R2 上配置用户名和密码

```
r2(config)#username ccie password cisco
```

(2) 配置为所有 VTY 用户自动执行命令

**说明：**要为所有 VTY 用户执行命令，就配置在 VTY 接口下，这里配置自动执行

# CCIE LAB认证经验分享千人群：539730342

命令为 show ip interface brief

```
r2(config)#line vty 0 935
```

```
r2(config-line)#login local
```

```
r2(config-line)#autocommand show ip interface brief
```

## (3) 测试结果

**说明：**从 R1 telnet 到 R2，输入正确用户名和密码，即可看到命令执行后的输出


```
r1#telnet 12.1.1.2
```

```
Trying 12.1.1.2 ... Open
```

```
User Access Verification
```

```
Username: ccie
```

```
Password:
```



Interface	IP-Address	OK?	Method	Status	Protocol
FastEthernet0/0	12.1.1.2	YES	manual	up	up
FastEthernet0/1	unassigned	YES	unset	administratively down	down
Serial1/0	unassigned	YES	unset	administratively down	down
Serial1/1	unassigned	YES	unset	administratively down	down
Serial1/2	unassigned	YES	unset	administratively down	down
Serial1/3	unassigned	YES	unset	administratively down	down

```
[Connection to 12.1.1.2 closed by foreign host]
```

```
r1#
```

**说明：**从结果中可以看出，R1 telnet 到 R2 后，并不需要输入命令，就弹出之前命令的自动执行命令的输出结果，这样即达到了查看配置的目的，又不违反更改配

置的规矩。

## 2.为单个用户配置自动执行命令

### (1) 配置用户名

```
r2(config)#username test password test
```

### (2) 为单个用户配置自动执行命令

**说明：**这里配置自动执行命令：show ip route

```
r2(config)#username test autocommand show ip route
```

### (3) 从 R1 上 telnet 到 R2 做测试

```
r1#telnet 12.1.1.2
```

```
Trying 12.1.1.2 ... Open
```

```
User Access Verification
```

```
Username: test
```

```
Password:
```

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
```

```
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
```

```
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
```

```
E1 - OSPF external type 1, E2 - OSPF external type 2
```

```
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
```

```
ia - IS-IS inter area, * - candidate default, U - per-user static route
```

```
o - ODR, P - periodic downloaded static route
```



Gateway of last resort is not set

12.0.0.0/24 is subnetted, 1 subnets

C 12.1.1.0 is directly connected, FastEthernet0/0

[Connection to 12.1.1.2 closed by foreign host]

r1#

**说明：**可以看到 R1 从 VTY 连上 R2 时，输入相应的 用户名和密码后，便自动执行了相应的命令，并且证明用户的命令优先于接口的命令。



## Privilege Levels

### 概述

在 Cisco 设备中，将所有用户的操作权限分为 0-15 共 16 个等级，0 为最低等级，15 为最高等级。等级越高，能执行的命令就越多，权限就越大。要给用户赋予等级，可以在配置用户名或者密码时赋予。

在 Cisco 设备中，有一种最初级的模式，称为用户模式，即 User EXEC mode，默认表示为 Router>，在这个模式下，默认等级是 1，能执行的命令命令是相当少的，而需要注意的是，在这个模式下，永远只能执行等级为 1 的命令，如果要将等级提高到更高，就需要手动进入更高等级的模式，这个模式被称为特权模式，即 Privileged EXEC mode，表示为 Router#，通常此模式被称为 enable 模式。在没有指定等级而进入 enable 模式后，默认等级为 15，也就表示可以完全控制设备。如果要进入比 1 级更高的模式而又不是 15 级，可以在进入 enable 模式时手工指定要进入的等级。除此之外，在创建本地用户数据库时，也可以为相应用户指定相应等级，如果不指定，默认用户等级为 1 级。

# CCIE LAB认证经验分享千人群：539730342

通过以上方法为相应用户或密码分配等级之后，他们所能执行的命令也只是相应等级范围内的，比如 5 级的用户是不能执行 15 级的命令的，但是可以手工赋予每个等级可以执行哪些命令，如让 5 级的用户能执行某 15 级的命令，如果 5 级用户能执行某 15 级的命令，那么 6-15 也都是可以执行的。

注：如果 15 级 Privileged EXEC mode 没有密码，默认只有本地终端直连可以登陆，VTY 是不能登陆的。

## 配置

### 1.查看 User EXEC mode 的默认等级

```
r1>show privilege
```

```
Current privilege level is 1
```

```
r1>
```

说明：从结果中看出，在 User EXEC mode 下，默认的等级为 1 级，也永远只能执行 1 级的命令。



### 2.查看 Privileged EXEC mode 的默认等级

#### （1）不指定等级登陆 Privileged EXEC mode

```
r1>enable
```

```
r1#
```

#### （2）查看查看 Privileged EXEC mode 的默认等级

```
r1#show privilege
```

```
Current privilege level is 15
```

```
r1#
```

**说明：**从结果中看出，在 Privileged EXEC mode 下，默认的等级为 15 级

### 3.创建不同等级的密码

(1) 创建一个 5 级的密码，为 cisco5

注：只有在创建 secret 密码时，才能指定等级，password 是不可以的。

```
r1(config)#enable secret level 5 cisco5
```

(2) 创建一个 6 级的密码，为 cisco6

```
r1(config)#enable secret level 6 cisco6
```

### 4.测试 5 级的密码

(1) 登陆 5 级的密码

```
r1>enable 5
```

Password:            输入 cisco5

```
r1#
```

(2) 查看当前等级

```
r1#show privilege
```

```
Current privilege level is 5
```

```
r1#
```

**说明：**从结果中看出，当前等级为 5 级。

(3) 查看 show run 配置

```
r1#show run
```



^

% Invalid input detected at '^' marker.

r1#

**说明：**从结果中看出，5级用户是不能执行 show run 命令的。

## 5.测试 6 级的密码

### （1）登陆 6 级的密码

r1>enable 6

Password: 输入 cisco6

r1#

### （2）查看当前等级

r1#show privilege

Current privilege level is 6

r1#

**说明：**从结果中看出，当前等级为 6 级。

### （4）查看 show run 配置

r1#show run

^

% Invalid input detected at '^' marker.

r1#

**说明：**从结果中看出，6级用户也是不能执行 show run 命令的。

## 6.赋予 5 级用户可以执行 show run 命令

**说明：**要赋予某个等级用户可以执行某个命令时，必须使用 15 级的等级来指定，还必须说明该等级是在哪个模式下执行命令的。

```
r1(config)#privilege exec level 5 show run
```

**说明：**赋予 5 级用户可以在 exec 执行 show run 命令

## 7.测试 5 级用户的命令

```
r1>enable 5
```

Password:

```
r1#show run
```

Building configuration...

Current configuration : 55 bytes

!

boot-start-marker

boot-end-marker

!

end

```
r1#
```

**说明：**可以看到,5 级用户已经可以执行 show run 命令，但要注意的是，该等级只能查看权限范围内能够配置的参数情况。

## 8.查看 6 级是否可以执行 show run

```
r1>enable 6
```

Password:

r1#

r1#

r1#sh run

Building configuration...

Current configuration : 55 bytes

!

boot-start-marker

boot-end-marker

!

!

end

r1#



**说明：**当一条命令赋予某个等级之后，比此等级更高的等级同样获得该命令的执行权。

## 9.创建默认等级的本地用户数据库

### （1）配置用户名和密码

r1(config)#username aaa password bbb

### （2）登陆已配置的用户名和密码

r1>login

Username: aaa

Password:           输入密码 bbb

r1>

### (3)查看默认用户名的等级

R1>show privilege

Current privilege level is 1

R1>

**说明：**可以看到创建的用户默认是 1 级

## 10.创建等级为 15 的用户

### (1) 创建用户

R1(config)#username ccc privilege 15 password ddd

### (2) 登陆用户

r2>login

Username: ccc

Password:

r2#

### (3) 查看该用户等级

R1#show privilege

Current privilege level is 15

R1#

**说明：**可以看到创建的用户是 15 级



## 基于时间的 ACL

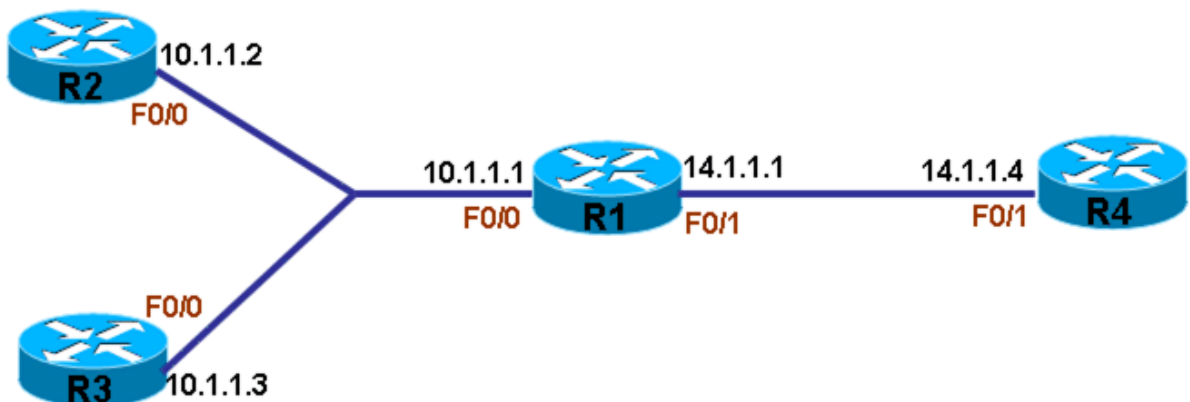
### 概述

一个很通常的需求，就是在某个公司里，有时希望限制员工在某个时间范围内才可以访问网页，即 HTTP 服务，或其它服务，在时间范围之外，就不能访问，那么这样的需求，就可以通过配置基于时间的 ACL 来实现。

要通过 ACL 来限制用户在规定的时间内访问特定的服务，首先设备上必须配置好正确的时间。在相应的时间要允许相应的服务，这样的命令，在配置 ACL 时，是正常配置的，但是，如果就将命令正常配置之后，默认是在所有时间内允许的，要做到在相应时间内允许，还必须为该命令加上一个时间限制，这样就使得这条 ACL 命令只在此时间范围内才能生效。而要配置这样的时间范围，是通过配置 time-range 来实现的，在 time-range 中定义好时间，再将此 time-range 跟在某 ACL 的条目之后，那么此条目就在该时间范围内起作用，其它时间是不起作用的。

在定义 time-range 时，常用的时间简单分为两种，第一种叫做绝对时间（absolute），即这个时间只生效一次，比如 2010 年 1 月 1 日 15:00；另一种时间叫做周期时间（periodic），即这个时间是会多次重复的，比如每周一，或者每周一直到周五。

### 配置



前提：在 R1 路由器上需要提前配置好正确的时间，此步骤省略。



# CCIE LAB认证经验分享千人群：539730342

## 1.配置 time-range

```
r1(config)#time-range TELNET
```

```
r1(config-time-range)#periodic weekdays 9:00 to 15:00
```

说明：定义的时间范围为每周一到周五的 9:00 to 15:00

## 2.配置 ACL

**说明：**配置 R1 在上面的时间范围内拒绝 R2 到 R4 的 telnet，其它流量全部通过。

```
r1(config)#access-list 150 deny tcp host 10.1.1.2 any eq 23 time-range TELNET
```

```
r1(config)#access-list 150 permit ip any any
```

## 3.应用 ACL

```
r1(config)#int f0/0
```

```
r1(config-if)#ip access-group 150 in
```

## 4.测试时间范围内的流量情况

### （1）查看当前 R1 的时间

```
r1#sh clock
```

```
14:34:33.002 GMT Thu Oct 1 2009
```

```
r1#
```

**说明：**当前时间为周四 14:34，即在所配置的时间范围内。

### （2）测试 R2 向 R4 发起 telnet 会话

```
r2#telnet 14.1.1.4
```

```
Trying 14.1.1.4 ...
```

```
% Destination unreachable; gateway or host down
```

r2#

**说明：**可以看到，在规定的时间内，R2 向 R4 发起 telnet 会话是被拒绝的。

### （3）测试除 telnet 外的其它流量

r2#ping 14.1.1.4

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 14.1.1.4, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

r2#

**说明：**可以看到，在规定的时间内，除了 telnet 之外，其它流量不受限制。

### （4）测试除 R2 之外的设备 telnet 情况

r3#telnet 14.1.1.4

Trying 14.1.1.4 ... Open

r4>

**说明：**可以看到，除 R2 之外，其它设备 telnet 并不受限制。

## 5.测试时间范围外的流量情况

### （1）查看当前 R1 的时间

r1#sh clock

15:01:15.206 GMT Thu Oct 1 2009

r1#

**说明：**当前时间为周四 15:01，即在所配置的时间范围之外。

## (2) 测试 R2 向 R4 发起 telnet 会话

```
r2#telnet 14.1.1.4
```

```
Trying 14.1.1.4 ... Open
```

```
r4>
```

**说明：**在时间范围之外，所限制的流量被放开。

## Reflexive ACL

### 概述

在某些网络中，为了考虑安全性，不希望外网的用户主动向内网发起连接，因为怀疑这样的动作可能是攻击行为。但是内网用户主动向外部发起的连接，外网的回包可以进入内网。这样的需求，如果使用普通的 ACL 在外网进来的接口上拒绝所有数据包，这肯定是不行的，因为这样虽然保证外网不能访问内网了，安全目的达到了，但是内网主动向外网发起的连接，外网回包时也进不来了，所以这种普通 ACL 不可行。更好的方法就是，先拒绝所有外网主动向内网发起的连接，但是在内网主动向外网发起的连接中，作好记录，打好标记，等到外网回包时，能够让其顺利进入内网，这样即保证了外网不能主动访问内网，实现了安全，又保证了内网发起的连接，外网可以回应，也不妨碍通信。要实现这样的功能，就可以通过特殊的 ACL，即 Reflexive ACL 来实现。

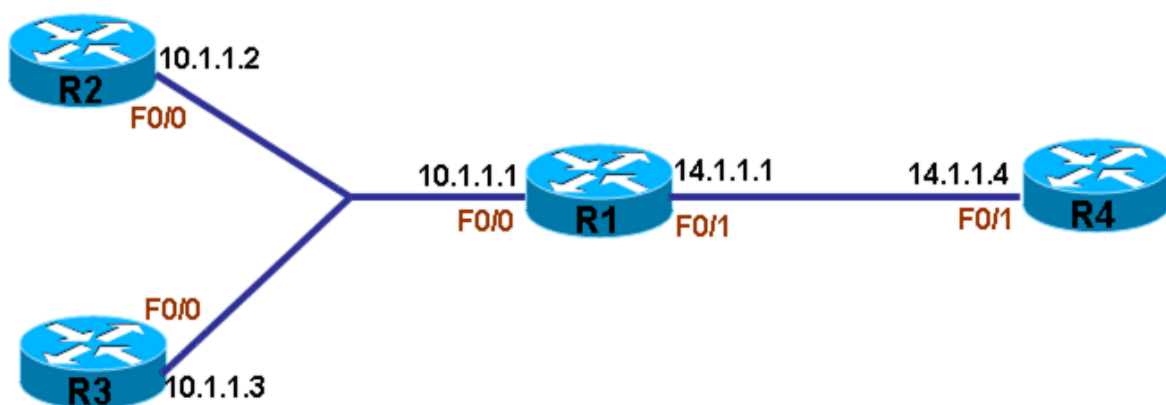
Reflexive ACL 就是根据以上所述，先拒绝外网向内网发送数据，然后允许内网向外网发送数据，但是在内网的数据发向外网时，这些数据的会话会被记录，被标记，等外网发回的数据和这些有记录的会话属于同一会话时，便可临时在进来的方向上打开缺口，让其返回，其它外网发来的不在记录中的数据，统统不能进入内网。所以根据这些原理，Reflexive ACL 需要有两个 ACL 来配合使用，一个 ACL 是用于外网到内网的方向，以拒绝外网的主动连接，另一个 ACL 是用于内网到外网的方向，用来检测内网有数据发向外网时，做上记录，等外网回包时，就在之前那个 ACL 中打开一个临时缺口，让外网的回包进入，这样就实现了之前所说的安全功能。

Reflexive ACL 有许多功能限制，只支持命名的扩展 ACL，并且思科官方文档会解释此 ACL 在最后没有像通常那样隐含拒绝所有，所以请注意你使用的 IOS 是否隐

含拒绝了所有数据通过。此 ACL 正因为外网数据在主动进入内网时被拒绝的，所以当内网数据出去时，这个会话会做好记录，会在进的方向给打开缺口，让其返回，这个被打开的缺口，在会话结束后，缺口被关闭。其实大家都知道，只有 TCP 的数据才存在会话，所以当 TCP 数据传完之后，会立马关闭缺口，但是对于没有会话的 UDP，就不能使用上面的方法了，就软件根据 timeout 来判断数据是否传完，如果没有给 ACL 指定 timeout，默认使用全局 timeout，默认全局是 timeout 是 300 秒，在 timeout 结束后，缺口被关闭。正因为这些从内网发到外网的数据被记录了，只有返回的数据和记录中相符，才能进入内网，所以如果返回的数据不符，就进不来，因此，会话在中途端口号是不能更换的，一旦更换，就无法匹配记录了。而像 FTP 这样的会话，在中途要改变端口号，所以 FTP 在有 Reflexive ACL 时，不能很好的工作。

在 Reflexive ACL 拒绝外网数据进入内网时，外网是不能先向内网发起连接的，但是并不需要将所有数据都拒绝，在配置时，某些数据就可以放开，让它和正常数据一样没有限制，比如路由协议的数据。而在定义什么样的数据出去之后被记录，可以返回，也可以只选择特定的数据。当在定义这个需要被记录的数据时，使用 ACL 来匹配，只能写一条，如果写多条，除了第一条，其它统统无效。

## 配置



**说明：** R4 为外网，R2 和 R3 为内网。

# CCIE LAB认证经验分享千人群：539730342

## 1.配置拒绝外网主动访问内网

**说明：**拒绝外网主动访问内网，但是 ICMP 可以不受限制

(1) 配置允许 ICMP 可以不用标记就进入内网，其它的必须被标记才返回

```
r1(config)#ip access-list extended come
```

```
r1(config-ext-nacl)#permit icmp any any
```

被允许的 ICMP 是不用标记即可进入内网的

```
r1(config-ext-nacl)#evaluate abc
```

其它要进入内网的，必须是标记为 abc 的

(2) 应用 ACL

```
r1(config)#int f0/1
```

```
r1(config-if)#ip access-group come in
```

## 2.测试结果

(1) 测试外网 R4 的 ICMP 访问内网

```
r4#ping 10.1.1.2
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

```
r4#
```

**说明：**可以看到，ICMP 是可以任意访问的

(2) 测试外网 R4 telnet 内网

```
r4#telnet
```

```
r4#telnet 10.1.1.2
```

Trying 10.1.1.2 ...

% Destination unreachable; gateway or host down

r4#

**说明：**可以看到，除 ICMP 之外，其它流量是不能进入内网的。

## （1）测试内网 R2 的 ICMP 访问外网

r2#ping 14.1.1.4

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 14.1.1.4, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/4 ms

r2#

**说明：**可以看到，内网发 ICMP 到外网，也正常返回了

## （2）测试内网 R2 发起 telnet 到外网

r2#telnet 14.1.1.4

Trying 14.1.1.4 ...

% Connection timed out; remote host not responding

r2#

**说明：**可以看到，除 ICMP 之外，其它流量是不能通过的。

## 3.配置内网向外网发起的 telnet 被返回

**说明：**外网和内网之间的 ICMP 可以不受限制，外网不能 telnet 内网，但内网 telnet 外网时，需要配置记录，让其返回，根据上面的 ACL 配置，可以返回的，必须是标

# CCIE LAB认证经验分享千人群：539730342

为 abc 的，所以在此为内网发向外网的 telnet 标为 abc，返回时，就会有缺口，因此内网能正常 telnet 外网，但外网不可主动 telnet 内网。

(1) 配置内网出去时，telnet 被记录为 abc,将会被允许返回

```
r1(config)#ip access-list extended goto
```

```
r1(config-ext-nacl)#permit tcp any any eq telnet reflect abc timeout 60 telnet 已记为 abc
```

```
r1(config-ext-nacl)#permit ip any any
```

(2) 应用 ACL

```
r1(config)#int f0/1
```

```
r1(config-if)#ip access-group goto out
```

## 4.测试结果

(1) 查看 R2 到外网的 ICMP

```
r2#ping 14.1.1.4
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 14.1.1.4, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

```
r2#
```

**说明：**ICMP 属正常

(3) 查看内网向外网发起 telnet

```
r2#telnet 14.1.1.4
```

Trying 14.1.1.4 ... Open



r4>

**说明：**可以看出，此时内网发向外网的 telnet 因为被标记为 abc，所以在回来时，开了缺口，也就可以允许返回了。

## （4）查看 ACL

r1#sh ip access-lists

Reflexive IP access list abc

permit tcp host 14.1.1.4 eq telnet host 10.1.1.2 eq 23395 (16 matches) (time left 33)

Extended IP access list come

10 permit icmp any any (86 matches)

20 evaluate abc

Extended IP access list goto

10 permit tcp any any eq telnet reflect abc

20 permit ip any any (20 matches)

r1#

**说明：**可以看到，有一条为 abc 的 ACL 为允许外网到内网的 telnet，正是由于内网发到外网的 telnet 被标记了，所以也自动产生了允许其返回的 ACL，并且后面跟有剩余时间。

## Context-Based Access Control (CBAC)

### 概述

CBAC 要实现的功能，就是 Reflexive ACL 所要实现的功能，但是由于 Reflexive ACL 有许多不足之处，所以需要 CBAC 来更好的支持。



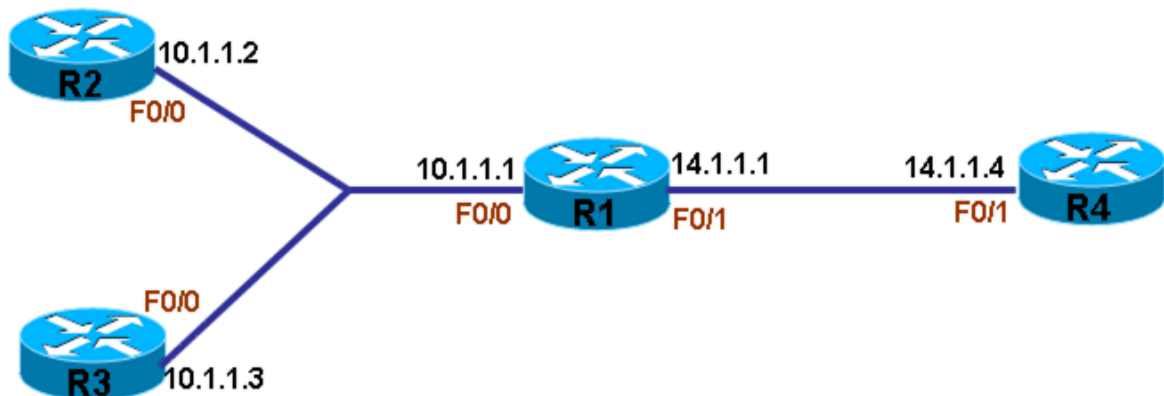
# CCIE LAB认证经验分享千人群：539730342

Reflexive ACL 不支持中途变换端口的协议，如 FTP，这在 CBAC 中，是可以支持的，Reflexive ACL

对于什么样的数据包需要允许返回，是需要写 ACL 来精确匹配的，并且只能写一条，而 CBAC 可以直接写数据的协议名称，不用写 ACL 去匹配，CBAC 所写的协议，就是 OSI 第 7 层应用层的协议，所以很方便用户匹配数据，并且可以写多个协议。CBAC 在思科官方文档中也会说不支持 ICMP 这个协议，所以请注意你的 IOS，在实际中，支持 CBAC 的，都是支持 ICMP 的。

用户不希望某些数据能主动从外网发向内网，只有当数据是从内网发向外网时，被允许返回，对于这样的数据，应该一开始就拒绝从外网进入内网，然后从内网发向外网时，让 CBAC 记住这个会话，并且在从外网进入内网的接口上打开缺口，方便其返回时顺利进入内网。所以在使用 CBAC 时，就需要先写好一个 ACL，应用在外网进内网的接口上，用以拒绝某些数据的进入，当这些数据从内网发起外网时，CBAC 就在进入的接口上临时创建缺口，让其返回。这个用在拒绝外网进入内网的 ACL，必须是扩展 ACL。以路由器自身为源或目的的数据，不被 CBAC 所记录。CBAC 同样有超时，由自己定义，并且可以打开审记功能，产生的日志就可以被记录。

配置



说明：R4 为外网，R2 和 R3 为内网。

# CCIE LAB认证经验分享千人群：539730342

## 1.配置拒绝所有的数据包从外网进入内网

(1) 在 R1 上配置 ACL 防止所有数据包进来:

```
r1(config)#access-list 100 deny ip an any
```

(2) 应用 ACL

```
r1(config)#int f0/1
```

```
r1(config-if)#ip access-group 100 in
```

## 2.测试结果

(1) 使用 ICMP 和 telnet 测试外网访问内网

```
r4#ping 10.1.1.2
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

U.U.U

Success rate is 0 percent (0/5)

```
r4#telnet 10.1.1.2
```

Trying 10.1.1.2 ...

% Destination unreachable; gateway or host down

```
r4#
```

**说明：**从结果中看出，外网向内网发起的 ICMP 和 telnet 均不能通过。

(2) 使用 ICMP 和 telnet 测试内网访问外网

```
r2#ping 14.1.1.4
```

Type escape sequence to abort.

# CCIE LAB认证经验分享千人群：539730342

Sending 5, 100-byte ICMP Echos to 14.1.1.4, timeout is 2 seconds:

.....

Success rate is 0 percent (0/5)

r2#telnet 14.1.1.4

Trying 14.1.1.4 ...

% Connection timed out; remote host not responding

r2#

**说明：**从结果中看出，内网向外网发起的 ICMP 和 telnet 也不能通过。

## 3.配置 CBAC 允许相应协议被返回

(1) 在 R1 上配置 CBAC 记录 telnet 会话，因此可以返回

r1(config)# ip inspect name ccie tcp audit-trail on timeout 60

r1(config)#int f0/1

r1(config-if)#ip inspect ccie out

**说明：**测试 IOS 没有单独的 telnet 协议，只能选整个 TCP 协议。

## 4.测试 CBAC 效果

(1) 测试外网向内网发起 ICMP

r4#ping 10.1.1.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

U.U.U

Success rate is 0 percent (0/5)

r4#

**说明：**外网向内网发起的 ICMP 是不能进入的。

## (2) 测试外网向内网发起 telnet

r4#telnet 10.1.1.2

Trying 10.1.1.2 ...

% Destination unreachable; gateway or host down

r4#

**说明：**外网向内网发起的 telnet 是不能进入的。

## (3) 测试内网向外网发起 ICMP

r2#ping 14.1.1.4

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 14.1.1.4, timeout is 2 seconds:

.....

Success rate is 0 percent (0/5)

r2#

**说明：**因为 CBAC 只记录 TCP，所以 ICMP 没有被允许返回。

## (4) 测试内网向外网发起 telnet

r2#telnet 14.1.1.4

Trying 14.1.1.4 ... Open

r4>

**说明：**因为 CBAC 记录 TCP，所以 telnet 被允许返回。

## （1）查看 CBAC 的会话记录

```
r1#sh ip inspect sessions
```

Established Sessions

Session 835AC510 (10.1.1.2:63276)=>(14.1.1.4:23) tcp SIS\_OPEN

```
r1#
```

**说明：**内网到外网的 telnet 被 CBAC 成功记录

## （2）查看 ACL 中 CBAC 打开的缺口

```
r1#sh ip access-lists
```

Extended IP access list 100

permit tcp host 14.1.1.4 eq telnet host 10.1.1.2 eq 20029 (13 matches)

10 deny ip any any (48 matches)

```
r1#
```

**说明：**可以看到，原本拒绝所有的 ACL，被 CBAC 临时创建了允许 telnet 返回的条目。

## 5.看日志

Oct 1 18:52:22: %FW-6-SESS\_AUDIT\_TRAIL: tcp session initiator (10.1.1.2:62155) sent 30 bytes -- responder (14.1.1.4:23) sent 32 bytes

```
r1#
```

**说明：**在 CBAC 检测时，审计打开，并且会产生日志。

## Port to Application Mapping (PAM)

### 概述

常用的协议 HTTP 对应 TCP 端口号 80，常用的协议 telnet 对应 TCP 端口号 23，这些端口号，Cisco 设备也是遵守默认的端口号规则，而这些协议对应的端口号，是任何时候都是可以随意改动的。在正常情况下，看到 TCP 80，就会把它当成 HTTP 来处理，看到 TCP 23，就会当成 telnet 来处理。

当设备上开启了 CBAC 后，CBAC 是根据相应的协议来做好会话记录，从而在 ACL 为其返回的流量打开缺口的。比如已经配置 CBAC 检测 HTTP 协议，也就是说当设备检测到有 TCP 80 的数据通过时，就会记录下会话，并且为其打开缺口，而检测 TCP 其它端口号，是不会这么做的。但是，在某些特殊时候，如果你发起的 HTTP 会话，端口号已经被改变，如已经改成 1000，那么这个时候你发起的 HTTP 会话就是 TCP 1000，对于 TCP 1000 这样的数据，在已配置好的 CBAC 中，是不会为其记录并打开缺口的，如果要让 CBAC 也知道你现在所使用的 HTTP 已经不再是标准的端口号了，但还希望 CBAC 为你服务，那么就必须手动告诉设备你已经将 HTTP 改为了 TCP 1000。这个功能就是靠 PAM 来实现的。

原本的 CBAC 只支持协议的标准端口，但是要让 CBAC 支持非常规端口协议，就需要 PAM 来完成这个工作。

要让设备知道相应协议是用哪些端口号，可以配置协议和对应的端口号，如果没有人为配置，系统中也同样会存在这样的对应表。表里面提供三种信息，分别为：

系统定义的映射

用户定义的映射

主机映射

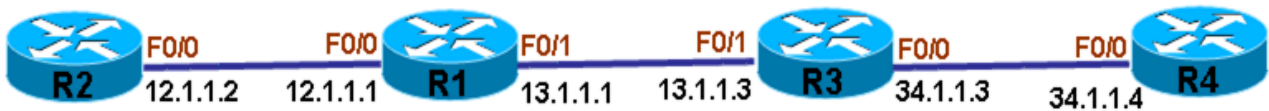
其中系统定义的就是标准的协议端口号，是不能更改的；

而用户定义的可以随意更改和删除，如果同时存在系统定义的和用户定义的，那么会优先使用用户定义的。

主机映射就是可以使用重复端口，即是基于每台主机的，比如定义某台主机 HTTP 使用 TCP 1000，而定义另外一台主机 FTP 使用 TCP 1000。

也可以为某个协议定义一个范围的端口号，通过多次输入命令实现。

## 配置



**说明：**在 R3 上配置 NAT，让 telnet 到 13.1.1.100 的结果被转到 telnet 34.1.1.4

### 1.在 R3 上配置 NAT

**说明：**配置让 telnet 到 13.1.1.100，目标端口为 1000 的，结果被转到 telnet 34.1.1.4

#### (1) 定义 NAT 方向

```
r3(config)#int f0/0
```

```
r3(config-if)#ip nat inside
```

```
r3(config)#int f0/1
```

```
r3(config-if)#ip nat outside
```

#### (2) 配置映射

```
r3(config)#ip nat inside source static tcp 34.1.1.4 23 13.1.1.100 1000
```

**说明：**当 telnet 13.1.1.100 1000 时，结果为 telnet 到 34.1.1.4

### 2.测试 telnet 结果

#### (1) 测试从 R2 telnet 13.1.1.100，目标端口为 1000

# CCIE LAB认证经验分享千人群：539730342

```
r2#telnet 13.1.1.100 1000
```

```
Trying 13.1.1.100, 1000 ... Open
```

```
r4>
```

**说明：**从结果中看出，当 R2 telnet 13.1.1.100，目标端口为 1000 时，结果为结果为 telnet 到 34.1.1.4。

## 3.配置 CBAC

**说明：**在 R1 上配置 CBAC，拒绝从 F0/1 进来的所有数据，但是记录 R2 发起的 telnet 数据，并允许其返回

### （1）配置 ACL 拒绝所有数据进入

```
r1(config)#access-list 100 deny ip any any
```

```
r1(config-if)#ip access-group 100 in
```

### （2）配置 CBAC 允许 telnet 返回

```
r1(config)#ip inspect name ccie telnet timeout 100
```

```
r1(config)#int f0/1
```

```
r1(config-if)#ip inspect ccie out
```

## 4.测试 CBAC 结果

（1）测试 R2 telnet 13.1.1.100，目标端口为 1000 时，CBAC 是否能为其记录会话并打开缺口

```
r2#telnet 13.1.1.100 1000
```

```
Trying 13.1.1.100, 1000 ...
```

```
% Connection timed out; remote host not responding
```

```
r2#
```



**说明：**从结果中看出，CBAC 并不会为端口号为 1000 的数据创建返回缺口，因为已配置的 CBAC 只记录 telnet，也就是标准 TCP 23 端口，而现在是 TCP 1000 端口，所以并不被关心。

## 5.配置 PAM

**说明：**已配置的 CBAC 只记录 telnet，也就是 TCP 为 23 端口的数据包，而现在的 telnet 为 TCP 端口号 1000，所以并没有被记录，因此配置 PAM，改变设备的默认 telnet 端口，应改为 1000，从而让 CBAC 根据此端口映射表作记录。

### （1）配置 telnet 端口号为 1000

```
r1(config)#ip port-map telnet port tcp 1000
```

## 6.测试 CBAC 支持 PAM 非常规端口

### （1）再次测试 R2 telnet 13.1.1.100，目标端口为 1000 时，CBAC 是否打开缺口

```
r2#telnet 13.1.1.100 1000
```

```
Trying 13.1.1.100, 1000 ... Open
```

```
r4>
```

**说明：**可以看出，CBAC 已经认为 telnet 为 TCP 端口号 1000，并成功为其打开缺口。

### （2）查看 CBAC 中的会话

```
r1#sh ip inspect sessions
```

```
Established Sessions
```

```
Session 66C93DE4 (12.1.1.2:17502)=>(13.1.1.100:1000) telnet SIS_OPEN
```

```
r1#
```

**说明：**看到 CBAC 中成功理解 telnet 为端口号 1000。

## 7.定义范围端口给协议

(1) 定义端口号 **8001** 到 **8004** 都给 HTTP

```
r1(config)#ip port-map http port 8001
```

```
r1(config)#ip port-map http port 8002
```

```
r1(config)#ip port-map http port 8003
```

```
r1(config)#ip port-map http port 8004
```

## 8.定义到主机映射

**说明：**可以让同一个端口被不同主机使用

(1) 定义主机 **10.1.1.1** 的 HTTP 使用端口号 **8000**

```
r1(config)#access-list 10 permit 10.1.1.1
```

```
r1(config)#ip port-map http port 8000 list 10
```

(2) 定义主机 **20.1.1.1** 的 FTP 使用端口号 **8000**

```
r1(config)#access-list 20 permit 20.1.1.1
```

```
r1(config)#ip port-map ftp port 8000 list 20
```

## Lock-and-Key Security (Dynamic ACL)

### 概述

有一种特殊的需求，比如一台连接了内网和外网的路由器，某些时候想限制内网的用户访问外网，如果用户想要获得访问外网的权限，就必须通过认证。

路由器在最初阻挡用户的数据通过，在用户通过认证后，再放行。要实现这样的功能，可以使用 **Dynamic ACL**。**Dynamic ACL** 在一开始拒绝用户相应的数据包通过，当用户认证成功后，就临时放行该数据，但是在会话结束后，再将 **ACL** 恢复最初的配置。要定义 **Dynamic ACL** 什么时候恢复最初的配置，可以定义会话超时，即会话多久没有传数据，就断开，也可以定义绝对时间，即无论会话有没有结束，到了规定时间，也要断开。

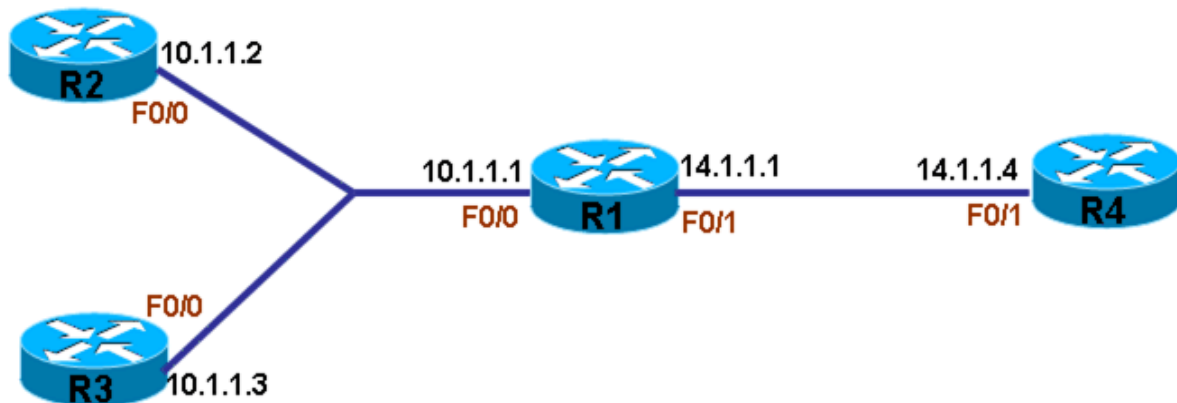
**Dynamic ACL** 给用户提供的认证方法有多种，最常用的可以使用 **AAA**，本地用户数据库。只要用户提供了正确的用户名和密码，就可以获得网络访问权。用户要向路由器提供用户名和密码以获得认证，就必须 **telnet** 配置了 **Dynamic ACL** 的路由器，当 **telnet** 到路由器时，输入了正确的用户名和密码之后，认证就算通过。而要注意的是，用户输入的用户名，并不是普通的用户名，必须是具有访问功能的用户名，如果用户输入的用户名是普通的，那么等于普通的 **telnet**，并不能成为认证，只有输入的用户名是具有访问功能的，才能通过认证并获得网络访问权。要赋予一个用户名网络访问权的功能，就需要配置 **autocommand** 来实现。

当使用 **AAA** 认证时，用户名要有 **autocommand**，也可以在配置本地用户数据库时为用户名添加，并且还可以加在 **VTY** 接口下。

会话结束的时间以分钟为单位，有空闲时间和绝对时间，如果两个时间同时配，空闲时间必须小于绝对时间，如果两个时间都不配，**Dynamic ACL** 打开的访问缺口必须手工清除。

因为 **Dynamic ACL** 在认证时是依靠用户 **telnet** 自己，所以一定要为用户打开 **telnet** 访问权限，某些数据也可以让其默认通过，比如路由协议的数据。认证通过之后，就可以访问相应的服务，在认证通过之后，具体可以访问哪些，需要配置 **ACL** 来定义，也就是配置了 **dynamic** 的条目，这样的动态条目只能配置一条，如果配多条动态 **ACL**，**IOS** 只认第一条，要注意 **Dynamic ACL** 只支持扩展的 **ACL**。

## 配置



**说明：**

R2 和 R3 为内网，R4 为外网，配置 R1，默认允许所有 telnet 通过，因为要使用 telnet 做认证，然后只有当认证通过之后，ICMP 才可以通过。

**1.配置 Dynamic ACL**

(1) 配置默认不需要认证就可以通过的数据，如 telnet

```
r1(config)#access-list 100 permit tcp any any eq telnet
```

(2)配置认证之后才能通过的数据，如 ICMP，绝对时间为 2 分钟。

```
r1(config)#access-list 100 dynamic ccie time out 2 permit icmp any any
```

(3) 应用 ACL

```
r1(config)#int f0/0
```

```
r1(config-if)#ip access-group 100 in
```

**2.测试访问**

(1) 测试内网 R2 telnet 外网 R4

```
r2#telnet 14.1.1.4
```

```
Trying 14.1.1.4 ... Open
```

r4>

**说明：**从结果中看出，telnet 不受限制。

## （2）测试测试内网 R2 ping 外网 R4

r2#ping 14.1.1.4

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 14.1.1.4, timeout is 2 seconds:

U.U.U

Success rate is 0 percent (0/5)

r2#

**说明：**内网在没有认证之前，ICMP 是无法通过的。

## 3.配置本地用户数据库

r1(config)#username ccie password cisco

## 4.配置所有人的用户名具有访问功能

r1(config)#line vty 0 181

r1(config-line)#login local

r1(config-line)#autocommand access-enable 这条必加

## 5.内网 R2 做认证

r2#telnet 10.1.1.1

Trying 10.1.1.1 ... Open

User Access Verification

# CCIE LAB认证经验分享千人群：539730342

Username: ccie

Password:

[Connection to 10.1.1.1 closed by foreign host]

r2#

**说明：**当 telnet 路由器认证成功后，是会被关闭会话的。

## 6.测试内网到外网的 ICMP 通信功能

r2#ping 14.1.1.4

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 14.1.1.4, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

r2#

**说明：**认证通过之后，ICMP 被放行。

## 7.查看 ACL 状态

r1#sh ip access-lists

Extended IP access list 100

10 permit tcp any any eq telnet (105 matches)

20 Dynamic ccie permit icmp any any

permit icmp any any (5 matches)

r1#

**说明：**可以看到动态允许的流量已放行。

## 8.host 功能

有配置访问功能时，命令有

```
r1(config-line)#autocommand access-enable
```

```
r1(config-line)# autocommand access-enable host
```

两种，并且后面可以跟时间，这里的时间为空闲时间，必须比之前的绝对时间要小，在配置访问功能时，如果没有加 host，那么内网一台主机通过认证之后，所有主机都能访问外网，加了 host，就变成谁通过了认证，谁才能访问外网。

## TCP Intercept

### 概述

通常所说的 TCP 要比 UDP 可靠，是因为 TCP 是有会话的，是面向连接的，任何两点之间在通信时，都必须有一条会话，所有被丢失的数据包，都会重传。而 TCP 在建立这条会话之前，必须完成的任务是三次握手。

TCP 只有在完成了三次握手之后，才能建立会话。比如 R1 是客户端，要与服务器 R2 建立 TCP 会话，这三次握手的过程是：

第一次握手：R1 向 R2 发送一个数据包，并且携带一个序列号，如 100。

第二次握手：R2 向 R1 回一个数据包，先回答 R1 的序列号，为  $100+1$ ，结果是 101，然后也会向 R1 发出一个序列号，比如是 200，并等待对方的回答，直到 TCP 超时为止。

第三次握手：R1 根据 R2 提出的序列号，作出回答， $200+1$ ，结果为 201。

当这样的过程完成之后，即表示三次握手成功完成，即可建立 TCP 会话。

从上面的过程可以看到，如果 R1 在给服务器 R2 发起握手之后，服务器 R2 回应之后，如果 R1 并不作第三次握手的回应，那么服务器将启动一个等待计时器，直到

# CCIE LAB认证经验分享千人群：539730342

超时为止。这样的握手被称为半开连接。因此，如果客户端向服务器发起大量的半开连接，就会导致服务器由于启动过多的计时器而耗尽系统资源，从而停止工作。

当一台网络上的正常服务器向用户提供服务时，如果用户对其进行上述的 TCP 攻击，将导致该服务器停止工作，所以就试图寻找一种方法来避免服务器遭受这样的 TCP 攻击。很显然，要避免这样的攻击，可以让服务器尽早的清除半开连接，从而避免握手数量的累加。也可以让服务器限制最大握手数量，即累加到一定数量的半开连接后，便停止客户端的握手请求。

这样的工作，可以在服务器上实现，而我们现在要做的，是在路由器上实现，因为当客户访问服务器时，会话是通过路由器的，所以中间的路由器可以开启监测功能，来监测这些握手会话，当某些握手长时间不完成时，便可认为是攻击，就向服务器发送重置该会话的请求，路由器也可以限制客户到达服务器的半开连接数量。

在这里，路由器要为服务器提供 TCP 保护，有两种工作模式。第一种是客户向服务器发起的握手请求被正常转发到服务器，但是路由器会监视这条会话，当超过规定时间，三次握手还没完成，那么路由器就向服务器发送重置该会话的请求，从而保护了服务器。这种模式被称为 **watch** 模式。第二种是当客户向服务器发起握手请求时，路由器收到这样的包之后，将其拦下，只有当三次握手完成之后，再将该会话交给服务器，让服务器和客户机之间正常连接，否则，路由器就清除该会话。

在配置 TCP Intercept 时，需要配置 ACL 要告诉路由器监测哪些 TCP 会话，如果 ACL 不配，路由器是不会采取任何动作的。

## 配置：

### 1.配置需要监视的 TCP

**说明：**通过配置 ACL 来实现

```
r1(config)#access-list 100 permit tcp any any
```

### 2.应用 ACL 到 TCP Intercept

```
r1(config)#ip tcp intercept list 100
```

### 3.配置 TCP Intercept 的模式



## (1) 配置 watch 模式

```
r1(config)#ip tcp intercept mode watch
```

## (2) 配置 intercept 模式

```
r1(config)#ip tcp intercept mode intercept
```

## 4.配置半开连接最长等待时间（默认 30 秒）

```
r1(config)#ip tcp intercept watch-timeout 60
```

**说明：**半开连接在 60 秒未完成三次握手，连接将被清除。

## 5.配置连接超时

**说明：**既然三次握手成功完成，TCP 会话也建立，而路由器也会默认跟踪该会话 24 小时，可以缩短该时间，比如 1 小时，可以理解为 TCP 会话建立后，多长时间不传数据，会话将断开。

```
r1(config)#ip tcp intercept connection-timeout 3600
```

## 6.定义总的未完成数

**说明：**定义半开连接数到多少就开始清除会话，有高低两个，是阈值，默认是 900 和 1100

```
r1(config)#ip tcp intercept max-incomplete low 800
```

```
r1(config)#ip tcp intercept max-incomplete high 1000
```

## 7.定义每分钟的未完成数，有高低两个，是阈值，默认是 900 和 1100

```
r1(config)#ip tcp intercept one-minute low 800
```

```
r1(config)#ip tcp intercept one-minute high 1000
```

## 8.配置丢弃模式

**说明：**当会话到达最大数量后，默认是清除最老（时间最长）的会话，也可以配置随机清除。

```
r1(config)#ip tcp intercept drop-mode oldest/random
```

## Unicast Reverse Path Forwarding (uRPF)

### 概述

uRPF 被称为单播的反向路径转发，功能是让路由器具备防 IP 欺骗或 IP 伪造的能力。

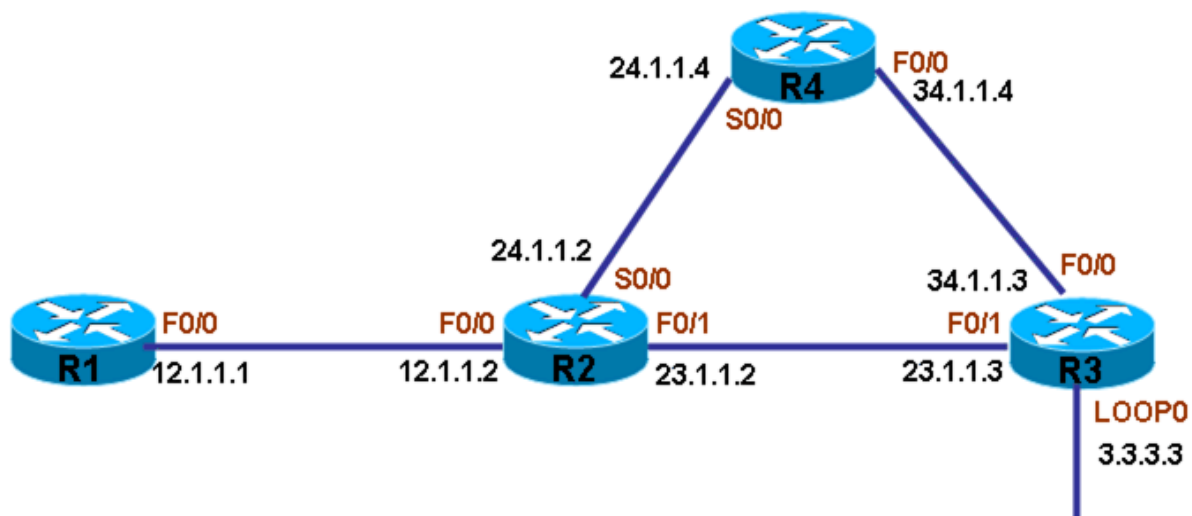
uRPF 所认为的 IP 伪造，是指某个 IP 的数据包的并不应该从某个接口进来，却从某个接口进来了，那么这样的数据包便认为是具有 IP 欺骗性质的，默认是被丢弃的。

当路由器从某个开启了 uRPF 的接口上收到数据包之后，都会检测该数据包的源 IP 地址，同时与路由表中的路由条目作对比，经过判断后，如果到达这个源 IP 的出口确实是这个开了 uRPF 的接口，则数据包被转发，否则被丢弃。比如路由器从接口 F0/0 收到一个源 IP 为 10.1.1.1 的数据包，那么就将 IP 地址 10.1.1.1 和路由表作对比，只要去往 10.1.1.1 的出口不是 F0/0（比如是 F0/1），那么该数据包被丢弃。

因为 uRPF 开启后，所有从此接口进入的数据包都要被检测，速度将会变慢，所以必须开启 CEF 后，才能开启 uRPF。uRPF 只能在 in 方向上开启，在做检查时，所有到源 IP 的最优路径都认为是可行的，EIGRP 非等价出口也算正常，并且即使是默认路由，也可通过检查。

在正常情况下，如果一个数据包无法通过 uRPF 检查，那么该数据包默认是丢弃的，但是有时因为特殊原因，可以让某些即使检查失败的数据包也能通过，要做到这一点，就可以在开启 uRPF 除加 ACL，其中检查失败的数据包，是丢弃还是放行，全由 ACL 来决定，ACL 允许，就放行，ACL 拒绝，就丢弃。这里的 ACL 和常用 ACL 一样配置，可以带 log 和 log-input 参数。

### 配置



说明：

R1 到任何网段的数据包都发向 12.1.1.2(即 R2)

R3 到任何网段的数据包都发向 34.1.1.4 (即 R4)

R2 到 R3 的 loopback0 (3.3.3.3)都发向 23.1.1.3(即 R3)

### 1.确认网络路径

说明：先测试网络的路径走向

#### (1) 查看 R3 的路由表

```
r3#sh ip route
```

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP

D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

# CCIE LAB认证经验分享千人群：539730342

E1 - OSPF external type 1, E2 - OSPF external type 2

i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2

ia - IS-IS inter area, \* - candidate default, U - per-user static route

o - ODR, P - periodic downloaded static route

Gateway of last resort is 34.1.1.4 to network 0.0.0.0

34.0.0.0/24 is subnetted, 1 subnets

C 34.1.1.0 is directly connected, FastEthernet0/0

3.0.0.0/32 is subnetted, 1 subnets

C 3.3.3.3 is directly connected, Loopback0

23.0.0.0/24 is subnetted, 1 subnets

C 23.1.1.0 is directly connected, FastEthernet0/1

S\* 0.0.0.0/0 [1/0] via 34.1.1.4

r3#

**说明：**从路由表中可以看出，R3 到任何网络都是发往 34.1.1.4(即 R4)的。

**(2)从 R3 跟踪到 R1 的路径走向**

r3#traceroute 12.1.1.1

Type escape sequence to abort.

Tracing the route to 12.1.1.1

1 34.1.1.4 0 msec 4 msec 0 msec

2 24.1.1.2 12 msec 12 msec 12 msec

3 12.1.1.1 12 msec \* 8 msec

r3#

**说明：**从结果中看出，R3 到 R1，是先发往 R4，然后 R4 从 R2 的 s0/0 发过来，最后到 R1 的。

### (3)查看 R2 到 R3 的 loopback0 (3.3.3.3)

r2#show ip route

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP

D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

E1 - OSPF external type 1, E2 - OSPF external type 2

i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2

ia - IS-IS inter area, \* - candidate default, U - per-user static route

o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

34.0.0.0/24 is subnetted, 1 subnets

S 34.1.1.0 [1/0] via 23.1.1.3

3.0.0.0/32 is subnetted, 1 subnets

S 3.3.3.3 [1/0] via 23.1.1.3

23.0.0.0/24 is subnetted, 1 subnets

C 23.1.1.0 is directly connected, FastEthernet0/1

24.0.0.0/24 is subnetted, 1 subnets

C 24.1.1.0 is directly connected, Serial0/0

12.0.0.0/24 is subnetted, 1 subnets

C 12.1.1.0 is directly connected, FastEthernet0/0

r2#

**说明：**从路由表中可以看出，R2 到 R3 的 loopback0 (3.3.3.3)是发往 23.1.1.3(即 R3)的。

(4) 从 R2 跟踪到 loopback0 (3.3.3.3)的路径走向

r2#traceroute 3.3.3.3

Type escape sequence to abort.

Tracing the route to 3.3.3.3

1 23.1.1.3 0 msec \* 0 msec

r2#

**说明：**可以看到，R2 到 R3 的 loopback0 (3.3.3.3)是直接从自己的 F0/1 发出去就到了。

(5) 再看 R3 以 loopback0 (3.3.3.3)为源到 R1 的连通性

r3#ping 12.1.1.1 source loopback 0

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 12.1.1.1, timeout is 2 seconds:

Packet sent with a source address of 3.3.3.3

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms

r3#

**说明：**可以看到通信正常

(6) 再看 R3 以 loopback0 (3.3.3.3)为源到 R1 的路径

r3#traceroute 12.1.1.1 source loopback 0

Type escape sequence to abort.

Tracing the route to 12.1.1.1

1 34.1.1.4 0 msec 4 msec 0 msec

2 24.1.1.2 12 msec 12 msec 12 msec

3 12.1.1.1 12 msec \* 8 msec

r3#

**说明：**在任何情况下到达 R1 都是同样的路径。

## 2.在 R2 上开启 uRPF

(1) 在 R2 的 S0/0 上开启 uRPF

r2(config)#int s0/0

r2(config-if)#ip verify unicast reverse-path

## 3.测试开启 uRPF 后的通信情况

(1) 以 R3 的 loopback0 (3.3.3.3)为源，向 R1 发送数据

r3#ping 12.1.1.1 source loopback 0

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 12.1.1.1, timeout is 2 seconds:

Packet sent with a source address of 3.3.3.3

.....

Success rate is 0 percent (0/5)

r3#

**说明：**可以看到，在 R2 的 S0/0 上开启 uRPF 之后，网络不通了。

#### 4.查看 R2 的 uRPF 情况

r2#sh ip interface s0/0

IP verify source reachable-via RX, allow default

5 verification drops

0 suppressed verification drops

r2#

**说明：**可以看到，有 5 个数据包因为 uRPF 被丢弃，正是因为 R2 到 R3 的 loopback0 (3.3.3.3)是从 F0/1 出去的，所以以 R3 的 loopback0 (3.3.3.3)为源的数据包必须也从 F0/1 进来，所以从 S0/0 进来被 uRPF 检查失败，所以默认丢弃，如果不想丢弃，必须配置 ACL 允许。

#### 5.以 R3 的 F0/0 为源，向 R1 发送数据

r3#ping 12.1.1.1

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 12.1.1.1, timeout is 2 seconds:

.....



Success rate is 0 percent (0/5)

r3#

**说明：**默认不指定源，即以 **F0/0** 发出数据，数据照样被丢弃，原因同上。

## 6.查看 R2 的 uRPF 情况

r2#sh ip interface s0/0

IP verify source reachable-via RX, allow default

10 verification drops

0 suppressed verification drops

r2#

**说明：**可以看到丢弃的数据包。

**7.在 R2 上配置 ACL 允许以 R3 的 loopback0 (3.3.3.3)为源的数据包即使 uRPF 检查失败也放行。**

r2(config)#access-list 100 permit ip host 3.3.3.3 any

r2(config)#int s0/0

r2(config-if)#ip verify unicast reverse-path 100

## 8.测试以 R3 的 loopback0 (3.3.3.3)为源的数据包通信情况

r3#ping 12.1.1.1 source loopback 0

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 12.1.1.1, timeout is 2 seconds:

Packet sent with a source address of 3.3.3.3

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/32 ms

r3#

**说明：**可以看到由于 uRPF 中的 ACL 允许此失败的数据包通过，所以网络正常。

9.再看以 R3 的 F0/0 为源，向 R1 发送数据的通信情况

r3#ping 12.1.1.1

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 12.1.1.1, timeout is 2 seconds:

.....

Success rate is 0 percent (0/5)

r3#

**说明：**因为 uRPF 中的 ACL 并没有允许以 R3 的 F0/0 为源数据检查失败也通过，所以数据包被丢弃。



## AAA

### 概述

在 Cisco 设备中，许多接口，许多地方，为了安全，都需要开启认证服务，比如我们通常配置的 console 下的密码，进入 Privileged EXEC 模式的 enable 密码，VTY 线路下的密码，以及其它二层接口的认证密码等等。这些密码配置在哪里，那里就开启了相应的认证服务。并且这些认证服务方式是单一的，也没有备份认证方式，更重要的是，这些密码只能读取本地，却不可以通过读取远程服务器的认证方式来给自己提供认证服务。要想将一个接口的认证方式调用到另外一个接口，或者让某接口使用远程服务器的认证方式，那就需要 AAA 中的认证来实现。

AAA 中的认证可以给设备提供更多的认证方式，AAA 认证就相当于一个装有认证方式的容器一样，里面可以有多个认证方式，当这个容器被安装在某个接口，那么这个接口也就拥有了容器中所有的认证方式。而几乎所有的认证方式都可以被放入这个容器中，包含远程服务器的认证方式。

# CCIE LAB认证经验分享千人群：539730342

AAA 认证中，这个装有多认证方式的容器，被称为列表，即 **list**，这个 **list** 加载到某个接口，那么这个接口就拥有 **list** 中所有的认证方式。**List** 中的多个认证方式是排列有序的，当前一个认证方式不可用时，才能使用第二个，以此类推。如果第一个可以使用，绝不会开启第二个，所以这样就提供了认证备份。那么何时 **AAA** 才认为第一个认证方式不可用呢？何时才使用第二个呢，这是当 **AAA** 询问一个认证方式时，如果认证数据库没有返回信息，即认证数据丢失或失败了，那么才使用下一个认证方式，如果某个认证方式是可用的，但用户提供了错误的用户名或密码，这样的情况是不会使用下一个认证方式的。

**AAA** 中的认证方式除了调用设备本地的认证方式之外，还可以调用远程服务器的认证方式，比如 **radius**、**tacacs+**。而一个 **AAA list** 中最多可以有四个认证方式，当第一个无法响应了，便使用第二个，第二个如果同样不响应，才使用第三个。但是如果前面的认证方式是在设备本地的，如本地用户数据库，当数据库中没有用户名时，**IOS** 可能并不认为是认证不响应，所以很难使用下一个认证方式，只有当前一个是远程服务器时，远程服务器不响应，才会使用下一个，这是 **IOS** 的不足之处。如果 **list** 中所有认证方式全部没有响应，那么 **IOS** 是不会让用户登陆的，除非最后有 **none** 指示放弃认证。

前面提到 **AAA** 的所有认证方式都应该放入 **list** 中，**list** 手动应用到某个接口上，这个接口就拥有了 **list** 中的认证方式，如果配置好的 **list** 没有应用到接口，那么是毫无意义的。一个 **list** 是有名字的，但是如果 **list** 的名字为 **default**，那么这个 **list** 不用手动应用到接口，因为默认情况下，名为 **default** 的 **list** 自动应用于所有相关接口，所以请谨慎创建名为 **default** 的 **list**。创建的 **list**，并不是所有接口都是可以用的，要看创建的是为什么样的接口使用的，比如是给登陆认证的，就要指定为 **login**，如果是给 **dot1x** 认证的，就要指定为 **dot1x**。所有支持 **login** 认证的方法有：

**enable**

**krb5**

**krb5-telnet**

**line**

**local**

**local-case**

**none**

# CCIE LAB认证经验分享千人群：539730342

```
group radius
```

```
group tacacs+
```

```
group group-name
```

并且 login 中是不执行 autocommands 的。

AAA 的 list 可以调用多个认证方式，同样也可以调用远程服务器，比如 radius、tacacs+，不仅如此，还可以允许远程服务器有多个备份组，以便一台服务器坏了，另外一台可以继续顶替，要实现这样的服务器备份组，就必须配置 AAA 服务器组，在里面添加多台服务器的 IP 地址即可，并且配置了 AAA 的 Cisco 设备和远程服务器之间也可以应用密码认证。

配置：



## 1.开启 AAA

### (1) 开启 AAA

```
r1(config)#aaa new-model
```

### (2) 测试认证

```
r2#telnet 12.1.1.1
```

Trying 12.1.1.1 ... Open

User Access Verification

# CCIE LAB认证经验分享千人群：539730342

Username:

**说明：**当一台设备上开启 AAA 之后，所有 VTY 接口默认被加入 local 本地用户数据库认证。

## 2.为 VTY 使用 AAA 认证

(1) 创建 list，并指定认证方法顺序为 local 和 enable

```
r1(config)#aaa authentication login list1 local enable
```

**说明：**因为为 VTY 认证，所以认证关键字为 login

(2) 在 VTY 使用 AAA 认证

```
r1(config)#line vty 0 935
```

```
r1(config-line)#login authentication list1
```

## 3.创建认证密码

(1) 创建 enable 密码

```
r1(config)#enable secret cisco
```

## 4.测试认证

(1) 测试使用的认证方法

```
r2#telnet 12.1.1.1
```

Trying 12.1.1.1 ... Open

User Access Verification

Username:

**说明：**在没有 local 认证的情况下，AAA 并没有跳到下一个 enable 认证，所以只有远程服务器不响应，才认为认证为响应，才会使用下一个认证方式。

## 5.采用远程服务器认证

# CCIE LAB认证经验分享千人群：539730342

**说明：**将远程服务器认证做为第一个，当远程服务器不响应时，直接跳到下一个认证方式

(1) 创建 AAA list，第一个为 tacacs+，第二个为 enable

```
r1(config)#aaa authentication login list2 group tacacs+ enable
```

```
r1(config)#line vty 0 935
```

```
r1(config-line)#login authentication list2
```

```
r1(config-line)#
```

(2) 测试认证

```
r2#telnet 12.1.1.1
```

```
Trying 12.1.1.1 ... Open
```

```
User Access Verification
```

```
Password:
```



```
r1>
```

**说明：**在没有配置远程服务器时，AAA 认证无响应，所以切换到第二个认证方式 enable 认证。

## 6.测试更多认证

(1) 配置第一个为 tacacs+，第二个为 enable，第三个为空，并且删除 enable 密码

```
r1(config)#no enable secret
```

```
r1(config)#aaa authentication login list3 group tacacs+ enable none
```

```
r1(config)#line vty 0 935
```

```
r1(config-line)#login authentication list3
```

```
r1(config-line)#
```

## (2) 测试认证

```
r2#telnet 12.1.1.1
```

Trying 12.1.1.1 ... Open

User Access Verification

Password:

**说明：**当不是远程服务器无响应的认证方式失败时，都不会跳到下一个认证方式。

## 7.配置远程服务器组：

### (1) 配置多个远程服务器

```
r1(config)#aaa group server tacacs+ ggg
```

```
r1(config-sg-tacacs+)#server 10.1.1.1
```

```
r1(config-sg-tacacs+)#server 20.1.1.1
```

```
r1(config-sg-tacacs+)#exit
```

### (2) 定义远程服务器密码

```
r1(config)#radius-server host 10.1.1.1 key cisco
```

## 8.更多提示

**说明：**AAA list 中，当第一项为服务器时，检测不可用，才会往后退，如果服务器后的认证方式为 local，并且下一项为 none 时，随便输入什么认证都无项，直接让用户登陆，所以请小心此类配置。

### (1)配置第一项为服务器，第二项为 local，且紧跟 none

```
r1(config)#aaa authentication login list4 group tacacs+ local none
```

```
r1(config)#line vty 0 935
```

```
r1(config-line)#login authentication list4
```

## (2)测试认证

```
r2#telnet 12.1.1.1
```

```
Trying 12.1.1.1 ... Open
```

```
User Access Verification
```

```
Username: abc
```

```
r1>
```

**说明：**可以看到，此类配置，随便输入任何认证，都为通过。



## IP Source Tracker

### 概述

此功能是让路由器能够收集网络中可能正在遭受 DOS 攻击的主机，并且记录攻击源，创建必要的描述 DOS 攻击易用的信息，可以跟踪多个 IP。

记录日志的时间间隔是可以随意定义的，定义的最大主机数量也是可以定义的，并且这些信息全部可以输出到远程服务器，如 GRP 和 RSP，也只有高端系列 75，12000 才支持。

### 配置

**1.配置跟踪的主机，可以配置多个主机。**



```
r1(config)#ip source-track 100.10.0.1
```

2.配置产生日志的间隔，单位为分

```
r1(config)#ip source-track syslog-interval 2
```

3.阶段配置输出的时间间隔

```
r1(config)#ip source-track export-interval 60
```

4.配置最多记录的地址数量

```
r1(config)#ip source-track address-limit 3
```

## Secure Shell (SSH)

### 概述

在对设备进行远程连接的方法中，最常用的是 telnet，而所有通过 telnet 会话传递的数据都是以明文（非加密）方式传递的，当这些数据被截取之后，很轻松就能读取原文意思，为了安全性，有一种在 telnet 会话之上的连接方法，将数据进行加密后传输，这就是 Secure Shell (SSH)。

SSH 共有两个版本，ver 1 和 ver 2，Cisco 设备在没有指定版本的情况下，默认就是指 ver 1。要支持 SSH，设备必须拥有支持 IPsec (DES or 3DES) 加密的 IOS，从 12.1(1)T 或之后都是可以的。除此之外，必须为设备配置主机名和域名，否则会报错。最关键的是必须配置 RSA key，配置之后，SSH 就自动打开了，否则不能开启。删除 RSA 使用命令 `crypto key zeroize rsa`，如果密码被删除，则表示 SSH 被禁用。

在 Cisco 设备上配置 SSH，SSH 分为 server 和 client 两种，server 就是提供 SSH 登陆的设备，而 client 就是发起 SSH 会话的设备，而 Cisco 设备无法单独开启 client，client 在配置 server 功能后自动开启，并且自身是不需要任何命令打开的，也没有特定命令能够打开 client。当 SSH 会话没有数据传递时，默认超时是 120 秒，即使是手工配置也不能超过这个值。并且 SSH 的最大连接数量就是 VTY 所允许的数量。

SSH 版本 2 的 RSA 至少是 768 位。

## 配置



### 1.配置双方主机名和域名

**注：** server 和 client 之间的域名是可以不一样的。

#### (1) 配置 R1 的主机名和域名

```
router (config)#hostname r1
```

```
r1(config)#ip domain-name cisco.com
```

#### (2) 配置 R2 的主机名和域名

```
router (config)#hostname r2
```

```
r2(config)#ip domain-name cisco.com
```

### 2.配置双方的 RSA key

**注：** 双方的 RSA key 位数必须一致。

#### (1)配置 R1 的 RSA key

```
r1(config)#crypto key generate rsa
```

The name for the keys will be: r1.cisco.com

Choose the size of the key modulus in the range of 360 to 2048 for your

General Purpose Keys. Choosing a key modulus greater than 512 may take

a few minutes.

# CCIE LAB认证经验分享千人群：539730342

How many bits in the modulus [512]: 1024

% Generating 1024 bit RSA keys ...[OK]

r1(config)#

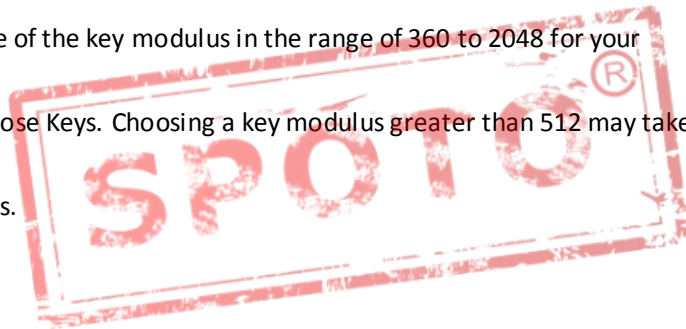
## (2)配置 R2 的 RSA key

r2(config)#crypto key generate rsa

The name for the keys will be: r2.cisco.com

Choose the size of the key modulus in the range of 360 to 2048 for your

General Purpose Keys. Choosing a key modulus greater than 512 may take  
a few minutes.



How many bits in the modulus [512]: 1024

% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]

\*Mar 1 05:24:34.940: %SSH-5-ENABLED: SSH 1.99 has been enabled

r2(config)#

## 3.创建用户名和密码， client 通过此用户名和密码登陆

r1(config)#username ccie password cisco

## 4.在 VTY 下开启认证，并指定 SSH 可以登陆

r1(config)#line vty 0 181

# CCIE LAB认证经验分享千人群：539730342

```
r1(config-line)#login local
```

```
r1(config-line)#transport input ssh telnet
```

## 5.测试 SSH 登陆

```
r2#ssh -l ccie 10.1.1.1
```

Password:

```
r1>
```

**说明：**可以成功登陆

## 6.查看 SSH 版本：

```
r1#sh ip ssh
```

SSH Enabled - version 1.5

Authentication timeout: 120 secs; Authentication retries: 3

```
r1#
```



```
r2#sh ip ssh
```

SSH Enabled - version 1.99

Authentication timeout: 120 secs; Authentication retries: 3

```
r2#
```

**说明：**所有在 2.0 以下，都算是版本 1。

## 7.将 SSH 启用版本 2

### (1) 修改 SSH 为版本 2

```
r2(config)#ip ssh version 2
```

(2) 查看版本：

```
r2#sh ip ssh
```

```
SSH Enabled - version 2.0
```

```
Authentication timeout: 120 secs; Authentication retries: 3
```

```
r2#
```

**说明：**已经改为版本 2。

注：版本 1 和版本 2 是可以互相登陆的。

## 8.指定源地址

**说明：**当使用 telnet 登陆远程时，可以指定源 IP 地址，在使用 SSH 时，需要在配置中修改源 IP 地址。

```
r2(config)#ip ssh source-interface Loopback0
```

**说明：**已将 SSH 的源 IP 地址改为 Loopback0 的地址。

## 9 同时开启两个 SSH 版本

```
R1 (config)# no ip ssh version
```

**说明：**加了此命令，表示 SSH 两个版本同时开启。

# Intrusion Prevention System (IPS)

## 概述

在路由器连接了公司内网和外网的时候，通常希望路由器能够保护内网服务器

# CCIE LAB认证经验分享千人群：539730342

抵御外网的攻击，这就需要配置路由器拒绝某些外网发向内网的可疑流量。但是，一个普通的网络操作人员，可能无法了解太多的可疑流量，也很难辨别什么样的流量算是可疑流量。基于以上原因，厂商就尽量将所有可疑流量的特征码写成一个文件，而管理员将此特征码文件导入路由器，让路由器根据特征码文件中的描述来拒绝相应的流量，从而抵御外网的攻击和入侵。这就是 IPS（入侵防御系统）的工作，而厂商写的特征码文件，被称为 **signatures**（签名文件）。

**SDF**（签名定义文件）

**signatures**（签名文件）被称为 **SDF**，并且里面有描述攻击流量的条目数量。Cisco 设备上可用的 **SDF** 有两种：

- 1 默认的，也就是系统内置的签名（共 100 条）
- 2 厂商最新的，也就是使用路由器或 **SDM** 下载最新的签名文件

需要注意的是，如果要从厂商下载最新的 **SDF**，强烈建议使用 **SDM** 下载，如果从路由器下载，会有意想不到的问题。并且从路由器的 **CLI** 模式不能调整 IPS 的动作的，所以配置 IPS，建议使用 **SDM**。

如果要做二层透明 IPS，只支持以太网，并且签名是三层的，没有二层，组播也不支持。

厂商的签名文件现有以下三种：

**attack-drop.sdf** (83 条)，适用于内存 128MB 以下

**128MB.sdf** (300 条)，适用于内存 128 MB 或更多

**256MB.sdf** (500)，适用于内存 256 MB 或更多。

这些文件可从 **flash** 直接加载进 IPS，但是 **flash** 被清除，**SDF** 也将清除，那么就使用内置的，

**SDF** 只能用于 12.4(9)Tx 或更早 IOS，主线 IOS 也支持。

系统内置的 **SDF** 是在 IOS 文件里面的，不需要下载。

IPS 检测所有经过自己的数据包，数据可以定义的，根据签名文件来判断，当检测到可疑流量时，可以先产生日志，或者发向 **SDEE**（也会有日志跳出），而产生的

动作包含如下：

- 1 发送警报给日志服务器，或者集中管理器
- 2 丢包
- 3 重置连接
- 4 在规定时间内拒绝该源数据包
- 5 在规定时间内拒绝该连接数据

IPS 提供两种警报：syslog 和 SDEE，而 SDEE 其实是在运行的，但是并没处理 IPS 事件，除非开启 notification，并且要开 SDEE，必须开 http server。

在 IPS 检测流量时，也可以使用 ACL 要定义哪些流量需要检测，哪些不需要检测。命名和数字 ACL 都支持，但是 12.3(8)T，只支持标准数字 ACL。

当 IOS 中加载了 SDF 之后，要告诉设备从哪些接口检测进来还是出去的流量，都需要手工定义，可以两个方向同时定义。如果使用厂商 SDF，当在系统中出现错误时，默认使用内置 SDF，但可以禁止在厂商 SDF 失败时使用内置 SDF。

## 配置

### 1.安装内置 SDF

**说明：**因为有内置和厂商两种 SDF，如果要安装厂商的 SDF，也要先安装内置的。

#### （1）导入内置 SDF

```
r1(config)#ip ips sdf builtin
```

### 2.开启 IPS

#### （1）创建策略名

```
r1(config)#ip ips name ipp
```

#### （2）在接口上开启 IPS

```
r1(config)#int f0/0
```

```
r1(config-if)#ip ips ips in
```

### 3.查看 IPS 结果

#### (1) 查看 IPS 情况

```
r1#sh ip ips all
```

Configured SDF Locations:

flash:

Builtin signatures are enabled and loaded

Last successful SDF load time: 01:43:33 UTC Mar 1 2002

IPS fail closed is disabled

IPS deny-action ips-interface is false

Fastpath ips is enabled

Quick run mode is enabled

Event notification through syslog is enabled

Event notification through SDEE is disabled

Total Active Signatures: 135

Total Inactive Signatures: 0

Signature 50000:0 disable

Signature 50000:1 disable

Signature 50000:2 disable

Signature 1107:0 disable





IPS Rule Configuration

IPS name ipp

Interface Configuration

Interface FastEthernet0/0

Inbound IPS rule is ipp

Outgoing IPS rule is not set

r1#

**说明：**可以看到，已经加载了内置的 SDF，并且接口 F0/0 上也应用了进方向的 IPS。

#### 4.将内置和厂商 SDF 结合使用

**说明：**可以同时使用两个 SDF，但内置必须先装载。

##### (1) 加载厂商 SDF

R1# copy disk2:attack-drop.sdf ips-sdf

##### (2) 保存为新的 SDF

R1# copy ips-sdf disk2:my-signatures.sdf

##### (3) 加载新的 SDF

R1(config)# ip ips sdf location disk2:my-signatures.sdf

##### (4) 在接口配置 IPS 的方法和之前一样

#### 5.开启 SDEE 报警功能

##### (1) 打开 SDEE

R1(config)# ip ips notify sdee

##### (2) 配置条目数：最多 1000



```
R1(config)# ip sdee events 500
```

## 6.不加载内置 SDF

**说明：**在使用厂商 SDF 的情况下，默认失败后直接使用内置的，但可以在失败时也不使用内置 SDF

### （1）关闭使用内置 SDF

```
R1(config)# no ip ips location in builtin
```

## 7.关闭丢包功能

**说明：**在 IPS 没有正常加载 SDF 时，可以选择丢弃所有数据包

```
R1(config)# ip ips fail closed
```



## Zone-Based Policy Firewall

### 概述

在 Cisco IOS 版的路由器中，可以实施一种防火墙功能，被称为 **Zone-Based Policy Firewall**，也就是说这种防火墙是基于 **zone** 的，是基于区域的。既然是基于区域的防火墙，那么配置的防火墙策略都是在数据从一个区域发到另外一个区域时才生效，在同一个区域内的数据是不会应用任何策略的。而要配置这些策略，方法像使用 MQC 来配置 QOS 一样配置防火墙策略，但是两个的配置方法并不完全一致，因为双方的格式会有一些不同。

要完全理解 **Zone-Based Policy Firewall** 的工作，必须理解以下一些参数：

### Zone

**Zone** 就是区域，因为区域化防火墙是基于区域的，策略也只能在区域间传递数

据时才生效，在区域内是不生效的，所以我们可以将需要使用策略的接口划入不同的区域，这样就可以应用我们想要的策略。但是，有时某些接口之间可能不需要彼此使用策略，那么这样的接口只要划入同一个区域，它们之间就可以任意互访了。Zone 是应用防火墙策略的最小单位，一个 zone 中可以包含一个接口，也可以包含多个接口。

## Security Zones

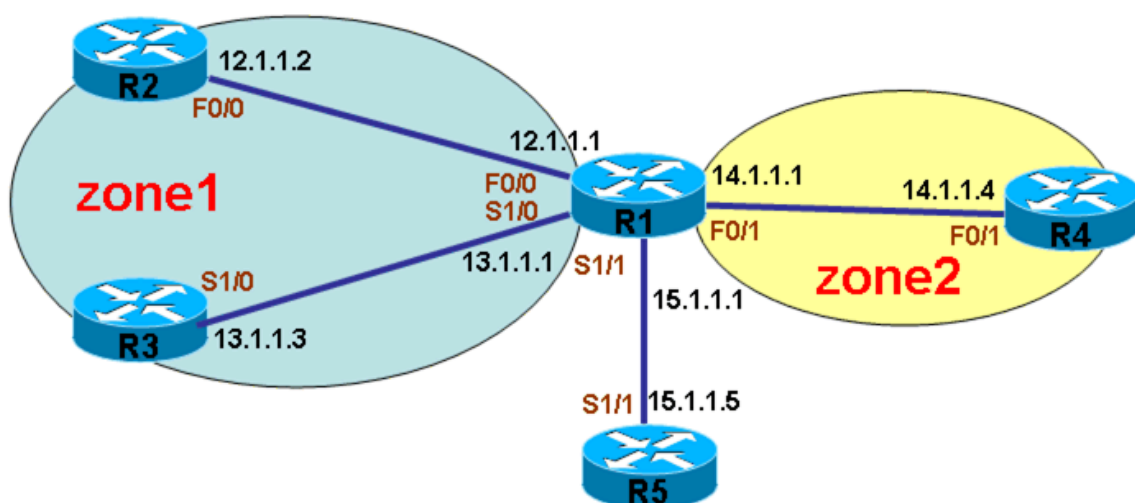
Security Zones 和上面的 zone 并没有区别，意思完全相同，只是因为 Security Zones 是指应用了策略的 zone，而且 Security Zones 应该是要包含接口的。

在配置好 Security Zones 之后，当一个接口属于 Security Zones 的成员时，所有发到此接口和从此接口发出去的（但发到路由器和路由器发给它的除外）数据是默认被丢弃的，也就是说区域间的接口默认是不能通信的，数据将全部被丢弃。

Virtual Interfaces 也可以作为 Security Zones 的成员，在将 Virtual Interfaces 划入 Security Zones 时，使用 `interface virtual-template`。

特别的是，如果一个接口不属于任何一个 zone，那么这个接口永远也不可以和任何 zone 的任何接口通信。

以下图解释通信过程：



R2 和 R3 属于区域 zone1

R4 属于区域 zone2

R5 不属于任何区域

默认的通信权限为：

R2 和 R3 可以自由通信，

Zone1 的 R2 和 R3 不能和 zone2 的 R4 互访，因为双方属于不同 zone，必须明确配置策略允许。

而 R5 永远不能和任何路由器通信，因为它不属于任何区域，即使策略也做不到。

## Zone-Pairs

因为在 Security Zones 之间的所有数据默认是全部被丢弃的，所以必须配置相应的策略来允许某些数据的通过。要注意，同区域的接口是不需要配置策略的，因为他们默认就是可以自由访问的，我们只需要在区域与区域之间配置策略，而配置这样的区域与区域之间的策略，必须定义从哪个区域到哪个区域，即必须配置方向，比如配置从 Zone1 到 Zone2 的数据全部被放行。可以看出，Zone1 是源区域，Zone2 是目的区域。配置一个包含源区域和目的区域的一组策略，这样的区域组，被称为 Zone-Pairs。因此可以看出，一个 Zone-Pairs，就表示了从一个区域到另一个区域的策略，而配置一个区域到另一个区域的策略，就必须配置一个 Zone-Pairs，并加入策略。

当配置了一个区域到另一个区域的策略后，如果策略动作是 inspect，则并不需要再为返回的数据配置策略，因为返回的数据是默认被允许的。

如果有两个 zone，并且希望在两个方向上都应用策略，比如 zone1 到 zone2 或 zone2 到 zone1，就必须配置两个 zone-pairs，就是每个方向一个 zone-pairs。

有时可以将 self zone 即作源又作目的。self zone 是 system-defined zone，即系统定义的 zone，它是没有接口的。当 zone-pair 包含 self zone 时，被应用的策略只对发到路由器和路由器发出的数据生效，通过路由器中转的数据不生效。

路由器上最好有两个 zone 来做策略，其中不包含 self zone。

## 策略

当接口被划入不同的 zone 之后，想要相互通信，就必须配置策略，再将配置好的策略应用于 Zone-Pairs，因为一个 Zone-Pairs 就表示了一个区域到另一个区域的策略情况。

在为 zone 之间配置策略，使用的方法类似于用 MQC 配置 QOS，但格式会有略微的差异。配置策略的方法为先使用 Class Map 匹配出指定的数据，然后再利用 Policy Map 调用 Class Map 匹配到的数据，做出相应的策略动作，最终将 Policy Map 应用于 Zone-Pairs。下面分别针对 Zone-Based Policy Firewall 中的 Class Map 和 Policy Map 来做详细介绍。

### 3/4 层 Class Maps 和 Policy Maps

因为 Class Map 可以匹配 OSI 中第三层数据和第四层数据，也可以匹配第七层应用层数据，但是匹配三四层数据和匹配第七层数据是完全不一样的，我们把匹配三四层数据的 Class Maps 和 Policy Maps 称为顶级 Class Maps 和顶级 Policy Maps。他们与普通 MQC 中的 Class Map 和 Policy Map 的区别在于，顶级 Class Maps 和顶级 Policy Maps 分别表示为 inspect class maps 和 inspect policy maps，而除了顶级 Class Maps 和顶级 Policy Maps 可以用在 zone-pair 中，其它统统不可以应用于 zone-pair 中。

顶级 Policy Maps 只能调用顶级 Class Maps，并且能执行的动作只有：drop, inspect, police, pass, service-policy, and urlfilter。而顶级 Class Maps 只能匹配 OSI 第三层数据和第四层数据，无法匹配第七层数据。

MQC 的 Police Maps 是在接口的，而 inspect policy Maps 是对 zone-pair 的，如果两个都配，zone-pair policer 是在接口进方向策略之后的，但在出策略之前。但两不冲突。

如果在 inspect policy Maps 中，默认没有被匹配到的数据是被丢弃。

### 7 层 Class Maps 和 Policy Maps

7 层 class maps 只能应用于 7 层 Policy Maps，而 7 层 Policy Maps 也只能调用 7 层 class maps，并且 7 层 Policy Maps 不能直接应用于 zone-pair 中，只能嵌套于 3/4 层 Policy Maps 中。如果 7 层 Policy Maps 嵌套在 3/4 层 Policy Maps 中，那么 3/4 层 Policy Maps 称为 parent policy，而 7 层 Policy Maps 称为 child policy。

# CCIE LAB认证经验分享千人群：539730342

7 层 Class Maps 和 Policy Maps 在配置时，必须指定协议名，比如 HTTP 协议，就配置 Class Maps 为 class-map type inspect http，Policy Maps 为 policy-map type inspect http。

当 7 层 Policy Maps 没有匹配到的数据，默认是要返回让顶层 Policy Maps 来处理的。

## Parameter Maps

是用来定义动作和标准的，分别用在 policy map and 和 class map 中，有三种：

Inspect parameter map

URL Filter parameter map

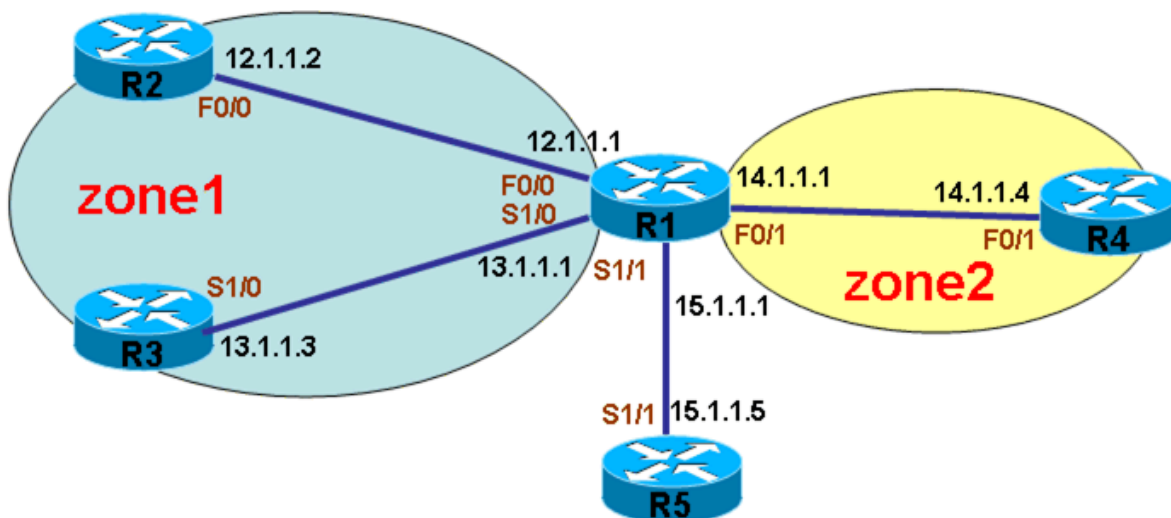
Protocol-specific parameter map

Inspect parameter map 是可选的，如果两级都有，低等级的有效。

URL Filter parameter map 在 URL 过滤时需要，在 34 层 policy MAP 中

Protocol-specific parameter map 只有 7 层 policy map 需要。

## 配置



#### 1.测试默认通信:

**说明:** 在没有配置防火墙的情况下, 测试通信情况

(1) 测试 R2 到 R3、R4、R5 的 ICMP 通信情况:

```
r2#ping 13.1.1.3
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 13.1.1.3, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 72/144/244 ms

```
r2#ping 15.1.1.5
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 15.1.1.5, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 72/147/368 ms

```
r2#ping 14.1.1.4
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 14.1.1.4, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 40/157/352 ms

```
r2#
```

**说明：**可以看到，默认没有配置防火墙的情况下，ICMP 畅通无阻。

(2) 测试 R2 到 R4 的 telnet 情况：

```
r2#telnet 14.1.1.4
```

Trying 14.1.1.4 ... Open

```
r4>
```

**说明：**可以看到，默认没有配置防火墙的情况下，telnet 畅通无阻。

## 2.创建 security zone

(1) 创建 zone1

```
r1(config)#zone security zone1
```

```
r1(config-sec-zone)#
```

(2) 创建 zone2

```
r1(config)#zone security zone2
```

```
r1(config-sec-zone)#exi
```



## 3.将接口划入 zone

### (1)将连 R2 和 R3 的接口划入 zone1

```
r1(config)#interface f0/0
```

```
r1(config-if)#zone-member security zone1
```

```
r1(config-if)#exit
```

```
r1(config)#int s1/0
```

```
r1(config-if)#zone-member security zone1
```

```
r1(config-if)#exit
```

### (2)将连 R4 的接口划入 zone2

```
r1(config)#int f0/1
```

```
r1(config-if)#zone-member security zone2
```

```
r1(config-if)#exit
```

### (3) 查看结果

```
r1#sh zone security
```

```
zone self
```

```
Description: System defined zone
```

```
zone zone1
```

```
Member Interfaces:
```

```
FastEthernet0/0
```

```
Serial1/0
```

```
zone zone2
```



Member Interfaces:

FastEthernet0/1

r1#

**说明：**结果与配置一致。

## 4.测试通信

### （1）测试 zone1 同区域的通信情况

r2#ping 13.1.1.3

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 13.1.1.3, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 56/188/408 ms

r2#

**说明：**在没有配置策略的情况下，同区域的通信不受限制

### （2）测试不通区域的通信情况

r2#ping 15.1.1.5

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 15.1.1.5, timeout is 2 seconds:

.....

Success rate is 0 percent (0/5)

r2#

r2#ping 14.1.1.4

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 14.1.1.4, timeout is 2 seconds:

.....

Success rate is 0 percent (0/5)

r2#

**说明：**从结果中看出，从 zone1 到 zone2，再到没有区域的网段，都是不通的。

## 5.创建 zone-pair

**说明：**创建 zone1 为源， zone2 为目的的 zone-pair

r1(config)#zone-pair security ccie source zone1 destination zone2

r1(config-sec-zone-pair)#exit

r1(config)#

## 6.配置策略

(1) 配置 class-map 匹配 zone1 到 zone2 的流量

r1(config)#access-list 100 permit ip any host 14.1.1.4

r1(config)#class-map type inspect c1

r1(config-cmap)#match access-group 100

**说明：**7 层 class-map 能匹配的协议很少，所以用 3/4 层

(2) 配置 policy-map 允许 zone1 到 zone2 的流量

r1(config)#policy-map type inspect p11

r1(config-pmap)#class type inspect c1

r1(config-pmap-c)#pass



```
r1(config-pmap-c)#exit
```

**说明：**动作 `pass` 是不会创建返回流量的。

## 7.应用策略到 `zone-pair`

```
r1(config)#zone-pair security ccie source zone1 destination zone2
```

```
r1(config-sec-zone-pair)#service-policy type inspect ppp
```

```
r1(config-sec-zone-pair)#
```

## 8.测试通信情况

```
r2#ping 14.1.1.4
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 14.1.1.4, timeout is 2 seconds:

.....

Success rate is 0 percent (0/5)

```
r2#
```

**说明：**已经应用了策略，`zone1` 到 `zone2` 还是不通，因为动作 `pass` 没有创建返回的流量。

## 9.创建 `zone2` 到 `zone1` 返回流量

### （1）配置 `class-map` 匹配流量

```
r1(config)#access-list 101 permit ip any any
```

```
r1(config)#class-map type inspect c2
```

```
r1(config-cmap)#match access-group 101
```

```
r1(config-cmap)#exit
```

### （2）配置 `policy-map` 允许流量

```
r1(config)#policy-map type inspect p22
```

```
r1(config-pmap)#class type inspect c2
```

```
r1(config-pmap-c)#pass
```

```
r1(config-pmap-c)#exit
```

### (3) 应用策略到返回流量

```
r1(config)#zone-pair security ccsp source zone2 destination zone1
```

```
r1(config-sec-zone-pair)#service-policy type inspect p22
```

## 10.测试 zone1 到 zone2 通信情况

### (1) 测试 R2 到 R4 的 ICMP 通信情况

```
r2#ping 14.1.1.4
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 14.1.1.4, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 36/112/360 ms

```
r2#
```

**说明：**ICMP 通信正常。

### (2) 测试 R2 到 R4 的 telnet 通信情况

```
r2#telnet 14.1.1.4
```

Trying 14.1.1.4 ... Open

```
r4>
```

### (3)zone1 到非 zone 的通信情况

```
r2#ping 15.1.1.5
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 15.1.1.5, timeout is 2 seconds:
```

```
.....
```

```
Success rate is 0 percent (0/5)
```

```
r2#
```

**说明：**区域中和非区域是永远也不能通信的。

### 11.配置 3/4 层 class-map 直接匹配协议

**说明：**使用 3/4 层 class-map 直接匹配 telnet 协议，其它协议不管

#### (1) 配置 class-map 匹配 telnet 协议

```
r1(config)#class-map type inspect c3
```

```
r1(config-cmap)#match protocol telnet
```

```
r1(config-cmap)#exit
```

#### (2) 配置 policy-map 允许 telnet 协议

```
r1(config)#policy-map type inspect p33
```

```
r1(config-pmap)#class c3
```

```
r1(config-pmap-c)#inspect
```

```
r1(config-pmap-c)#
```

**说明：**这里的动作是 inspect，所以不用创建返回流量的允许策略。

## (3) 应用策略到 zone-pair

```
r1(config)#zone-pair security ccie source zone1 destination zone2
```

```
r1(config-sec-zone-pair)#service-policy type inspect p33
```

```
r1(config-sec-zone-pair)#exit
```

```
r1(config)#
```

## 12.测试通信情况

### (1) 测试 zone1 到 zone2 的 ICMP 通信情况

```
r2#ping 14.1.1.4
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 14.1.1.4, timeout is 2 seconds:

.....

Success rate is 0 percent (0/5)

```
r2#
```

**说明：**因为没有允许除 telnet 外的协议，所以 ICMP 不能通过。

### (2) 测试 telnet 通信情况

```
r2#telnet 14.1.1.4
```

Trying 14.1.1.4 ... Open

```
r4>
```

**说明：**因为策略明确允许 telnet 通过，所以 telnet 通信正常。

### (3)测试到同区域的 ICMP

```
r2#ping 13.1.1.3
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 13.1.1.3, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 156/224/352 ms

r2#

**说明：**同区域中，不需要策略允许，所有通信正常。

## （4）测试到非区域的通信情况

r4#telnet 12.1.1.2

Trying 12.1.1.2 ...

% Connection timed out; remote host not responding

r4#

**说明：**区域到不同区域的通信永远不能通过。



## Control Plane Policing (CoPP)

### 概述

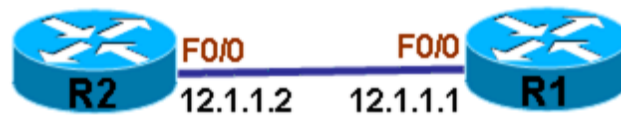
Control Plane Policing (CoPP)被称为控制面板策略，控制面板策略这个特性让用户通过配置 QOS 过滤来管理控制面板中的数据包，从而保护路由器和交换机免受 DOS 的攻击，控制面板可以无论在流量多大的情况下都能管理数据包交换和协议的状态情况。

在控制面板中，只能通过 MQC 配置常规的 QOS，并且 out 方向的 QOS 并不是所有 IOS 都支持，请自行检查，其中配置的 QOS 策略中，只有 drop 和 policy 两个动



作可以使用。而且 NBAR 功能也不能很好的支持。当在控制面板中配置 QOS 后，不用在接口下应用该策略，因为控制面板下的策略对所有接口生效。

## 配置



### 1.测试 R2 到 R1 的数据流量

```
r2#ping 12.1.1.1
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 12.1.1.1, timeout is 2 seconds:
```

```
!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

```
r2#
```

**说明：** R1 在没有使用任何 QOS 的情况下，通信正常

### 2.配置 QOS

**说明：** 这里配置 QOS 丢弃所有的包，以作测试用。

#### （1）配置匹配源自 R2 的数据

```
r1(config)#access-list 2 permit 12.1.1.2
```

```
r1(config)#class-map r2
```

```
r1(config-cmap)#match access-group 2
```

```
r1(config-cmap)#exit
```

## （2）配置丢弃源自 R2 的数据

```
r1(config)#policy-map copp
```

```
r1(config-pmap)#class r2
```

```
r1(config-pmap-c)#drop
```

## 3.将 QOS 应用于 COPP

```
r1(config)#control-plane
```

```
r1(config-cp)#service-policy input copp
```

## 4.测试到 R1 的通信

```
r2#ping 12.1.1.1
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 12.1.1.1, timeout is 2 seconds:

.....

Success rate is 0 percent (0/5)

```
r2#
```

**说明：**可以看到，并没有将所配置的 QOS 用于接口，只用在了控制面板下，所有接口都执行控制面板下的 QOS 策略，从而数据包被丢弃了，网络不通。