

- Switch and VLAN Security -

Switch Port Security

Port Security adds an additional layer of security to the switching network.

The MAC address of a host generally does not change. If it is certain that a specific host will always remain plugged into a specific switch port, then the switch can filter all MAC addresses *except* for that host's address using Port Security. The host's MAC address can be statically mapped to the switch port, or the switch can dynamically learn it from traffic.

Port security *cannot* be enabled on trunk ports, dynamic access ports, Etherchannel ports, or a SPAN destination port.

To enable Port Security on an interface:

```
Switch(config)# interface fa0/5
Switch(config-if)# switchport port-security
```

By default, Port Security will allow **only one MAC** on an interface. The maximum number of allowed MACs can be adjusted, up to 1024:

```
Switch(config-if)# switchport port-security maximum 2
```

To statically specify the allowed MAC address(es) on a port:

```
Switch(config-if)# switchport port-security mac-address 0001.1111.2222
Switch(config-if)# switchport port-security mac-address 0001.3333.5555
```

Only hosts configured with the above two MAC addresses will be able to send traffic through this port. If the *maximum* number of MAC addresses for this port had instead been set to *10*, but only two were statically specified, the switch would dynamically learn the remaining eight MAC addresses.

MAC addresses that are dynamically learned with Port Security are referred to as **Sticky Addresses**. Dynamically learned addresses can be aged out after a period of inactivity (measured in minutes):

```
Switch(config-if)# switchport port-security aging time 10
```

Port Security aging is *disabled* by default.

(Reference: http://www.cisco.com/en/US/docs/switches/lan/catalyst6500/ios/12.1E/native/configuration/guide/port_sec.html)

* * *

All original material copyright © 2009 by Aaron Balchunas (aaron@routeralley.com), unless otherwise noted. All other material copyright © of their respective owners.

This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Updated material may be found at <http://www.routeralley.com>.

Switch Port Security (continued)

Port Security can instruct the switch on how to react if an unauthorized MAC address attempts to forward traffic through an interface (this is considered a **violation**). There are three violation actions a switch can take:

- **Shutdown** – If a violation occurs, the interface is placed in an errdisable state. The interface will stop forwarding all traffic, including non-violation traffic, until taken out of the errdisable state. This is the **default** action for Port Security.
- **Restrict** – If a violation occurs, the interface will stay online, forwarding legitimate traffic and dropping the unauthorized traffic. Violations are **logged**, either to a SYSLOG server or via an SNMP trap.
- **Protect** – If a violation occurs, the interface will stay online, forwarding legitimate traffic and dropping the unauthorized traffic. *No logging* of violations will occur.

To configure the desired Port Security violation action:

```
Switch(config-if)# switchport port-security violation shutdown
Switch(config-if)# switchport port-security violation restrict
Switch(config-if)# switchport port-security violation protect
```

To view Port Security configuration and status for a specific interface:

```
Switch# show port-security interface fastethernet 0/5
```

```
Port Security: Enabled
Port status: SecureUp
Violation mode: Shutdown
Maximum MAC Addresses: 10
Total MAC Addresses: 10
Configured MAC Addresses: 2
Aging time: 10 mins
Aging type: Inactivity
SecureStatic address aging: Enabled
Security Violation count: 0
```

Note that the *Maximum MAC Addresses* is set to *10*, and that the *Total MAC Addresses* is currently at *10* as well. If another MAC address attempts to forward data through this interface, it will be placed in an errdisable state, as the violation action is set to *Shutdown*.

All original material copyright © 2009 by Aaron Balchunas (aaron@routeralley.com), unless otherwise noted. All other material copyright © of their respective owners.

This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Updated material may be found at <http://www.routeralley.com>.

802.1x Port Authentication

802.1x Port Authentication forces a host device to *authenticate* with the switch, before the switch will forward traffic on behalf of that host. This is accomplished using the **Extensible Authentication Protocol over LANs (EAPOL)**. 802.1x only supports **RADIUS** servers to provide authentication.

Both the switch *and* the host must support 802.1x to use port authentication:

- If the *host* supports 802.1x, but the switch does not – the host will not utilize 802.1x and will communicate normally with the switch.
- If the *switch* supports 802.1x, but the host does not – the interface will stay in an **unauthorized** state, and will not forward traffic.

A switch interface configured for 802.1x authentication stays in an **unauthorized** state until a client successfully authenticates. The only traffic permitted through an interface in an unauthorized state is as follows:

- EAPOL (for client authentication)
- Spanning Tree Protocol (STP)
- Cisco Discovery Protocol (CDP)

To globally enable 802.1x authentication on the switch:

```
Switch(config)# dot1x system-auth-control
```

To specify the authenticating RADIUS servers, and configure 802.1x to employ those RADIUS servers:

```
Switch(config)# aaa new-model
Switch(config)# radius-server host 192.168.1.42 key CISCO
Switch(config)# aaa authentication dot1x default group radius
```

Finally, 802.1x authentication must be configured on the desired interfaces. An interface can be configured in one of three 802.1x states:

- **force-authorized** – The interface will *always* authorize any client, essentially disabling authentication. This is the **default** state.
- **force-unauthorized** – The interface will *never* authorize any client, essentially preventing traffic from being forwarded.
- **auto** – The interface will actively attempt to authenticate the client.

```
Switch(config)# interface fa0/5
Switch(config-if)# dot1x port-control auto
```

(Reference: http://www.cisco.com/en/US/docs/switches/lan/catalyst2950/software/release/12.1_9_ea1/configuration/guide/Sw8021x.html)

* * *

All original material copyright © 2009 by Aaron Balchunas (aaron@routeralley.com), unless otherwise noted. All other material copyright © of their respective owners.

This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Updated material may be found at <http://www.routeralley.com>.

VLAN Access-Lists

Normally, access-lists are used to filter traffic *between* networks or VLANs. **VLAN Access-Lists (VACLs)** filter traffic *within* a VLAN, with granular precision. VACLs can filter IP, IPX, or MAC address traffic.

Assume that host 10.1.5.10 should be filtered from communicating to any other device on the 10.1.x.x/16 network, in VLAN 102. First, an access-list must be created to identify the traffic to be filtered *within* the VLAN:

```
Switch(config)# ip access-list extended BLOCKTHIS
Switch(config-ext-nacl)# permit ip host 10.1.5.10 10.1.0.0 0.0.255.255
```

The first line creates an *extended* named *access-list* called *BLOCKTHIS*. This contains a single entry, *permitting* host *10.1.5.10* to reach any other device on the *10.1.0.0* network.

Confused as to why the *10.1.5.10* host was *permitted*, and not *denied*? In this instance, the access-list is **not being used to deny traffic**, but merely to **identify the traffic**. The *permit* functions as a *true* statement, and a *deny* would function as a *false* statement.

The next step is to create the actual VACL:

```
Switch(config)# vlan access-map MYVACL 5
Switch(config-access-map)# match ip address BLOCKTHIS
Switch(config-access-map)# action drop

Switch(config-access-map)# vlan access-map MYVACL 10
Switch(config-access-map)# action forward

Switch(config)# vlan filter MYVACL vlan-list 102
```

The first line creates a *vlan access-map* named *MYVACL*. Traffic that *matches* entries in the *BLOCKTHIS* access-list will be *dropped*.

The final *vlan access-map* entry contains only an *action to forward*. This will apply to **all other traffic**, as no IP or access-list was specified. The above configuration would block all traffic from the 10.1.5.10 host to any other host on *VLAN 102*, while passing all other traffic.

Notice that every *access-map* statement contains a sequence number (in the above example, 5 and 10). This dictates the order in which these rules should be followed.

* * *

All original material copyright © 2009 by Aaron Balchunas (aaron@routeralley.com), unless otherwise noted. All other material copyright © of their respective owners.

This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Updated material may be found at <http://www.routeralley.com>.

Private VLANs

Private VLANs (PVLANS) allow for further segmentation of a subnet *within* a VLAN. Essentially, multiple sub-VLANs (considered **secondary** VLANs) are created beneath a **primary** VLAN.

The *secondary* VLAN **can only communicate** with the *primary* VLAN, and not any other secondary VLANs. There are two types of secondary VLANs:

- **Community** – interfaces *within* the secondary VLAN *can* communicate with each other.
- **Isolated** – interfaces *within* the secondary VLAN *cannot* communicate with each other.

Private VLANs are only locally-significant to the switch - VTP will not pass this information to other switches.

Each switch interface in a private VLAN assumes a specific role:

- **Promiscuous** - communicates with the primary VLAN and all secondary VLANs. Gateway devices such as routers and switches should connect to promiscuous ports.
- **Host** – communicates only with promiscuous ports, or ports within the local community VLAN. Host devices connect to host ports.

PVLANS thus allow groups of host devices to be segmented within a VLAN, while still allowing those devices to reach external networks via a promiscuous gateway.

(Reference: http://www.cisco.com/en/US/docs/switches/lan/catalyst3750/software/release/12.2_25_see/configuration/guide/swpvlan.html)

* * *

All original material copyright © 2009 by Aaron Balchunas (aaron@routeralley.com), unless otherwise noted. All other material copyright © of their respective owners.

This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Updated material may be found at <http://www.routeralley.com>.

Private VLAN Configuration

The first step to configuring PVLANS is to specify the secondary VLANs:

```
Switch(config)# vlan 100
Switch(config-vlan)# private-vlan community
```

```
Switch(config)# vlan 101
Switch(config-vlan)# private-vlan isolated
```

Next, the primary VLAN must be specified, and the secondary VLANs associated with it:

```
Switch(config)# vlan 50
Switch(config-vlan)# private-vlan primary
Switch(config-vlan)# private-vlan association 100,101
```

Secondary VLANs *100* and *101* have been associated with the *primary* VLAN *50*.

Next, Host ports must be identified, and associated with a primary and secondary VLAN:

```
Switch(config)# interface range fa0/5 – 6
Switch(config-if)# switchport private-vlan host
Switch(config-if)# switchport private-vlan host-association 50 101
```

Interfaces *fa0/5* and *fa0/6* have been identified as *host* ports, and associated with primary VLAN *50*, and secondary VLAN *101*.

Finally, promiscuous ports must be identified, and associated with the primary VLAN and *all* secondary VLANs.

```
Switch(config)# interface range fa0/20
Switch(config-if)# switchport private-vlan promiscuous
Switch(config-if)# switchport private-vlan mapping 50 100.101
```

Interface *fa0/20* has been identified as a *promiscuous* port, and associated with primary VLAN *50*, and secondary VLANs *100* and *101*.

(Reference: http://www.cisco.com/en/US/docs/switches/lan/catalyst3750/software/release/12.2_25_see/configuration/guide/swpvlan.html)

* * *

All original material copyright © 2009 by Aaron Balchunas (aaron@routeralley.com), unless otherwise noted. All other material copyright © of their respective owners.

This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Updated material may be found at <http://www.routeralley.com>.

DHCP Snooping

Dynamic Host Control Protocol (DHCP) provides administrators with a mechanism to *dynamically* allocate IP addresses, rather than manually setting the address on each device.

DHCP servers **lease** out IP addresses to DHCP clients, for a specific period of time. There are four steps to this DHCP process:

- When a DHCP client first boots up, it broadcasts a **DHCPDiscover** message, searching for a DHCP server.
- If a DHCP server exists on the local segment, it will respond with a **DHCPOffer**, containing the “offered” IP address, subnet mask, etc.
- Once the client receives the offer, it will respond with a **DHCPRequest**, indicating that it will accept the offered protocol information.
- Finally, the server responds with a **DHCPACK**, acknowledging the clients acceptance of offered protocol information.

Malicious attackers can place a rogue DHCP server on the trusted network, intercepting DHCP packets while masquerading as a legitimate DHCP server. This is one form of a **Spoofing** attack, or an attack aimed at gaining unauthorized access or stealing information by sourcing packets from a *trusted* source. This is also referred to as a *man-in-the-middle* attack.

DHCP attacks of this sort can be mitigated by using **DHCP Snooping**. Only specified interfaces will accept DHCPOffer packets – unauthorized interfaces will discard these packets, and then place the interface in an errdisable state.

DHCP Snooping must first be globally enabled on the switch:

```
Switch(config)# ip dhcp snooping
```

Then, DHCP snooping must be enabled for a specific VLAN(s):

```
Switch(config)# ip dhcp snooping vlan 5
```

By default, all interfaces are considered *untrusted* by DHCP Snooping. Interfaces connecting to legitimate DHCP servers must be *trusted*:

```
Switch(config)# interface fa0/15
Switch(config)# ip dhcp snooping trust
```

(Reference: <http://www.cisco.com/en/US/docs/switches/lan/catalyst4500/12.1/12ew/configuration/guide/dhcp.pdf>)

* * *

All original material copyright © 2009 by Aaron Balchunas (aaron@routeralley.com), unless otherwise noted. All other material copyright © of their respective owners.

This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Updated material may be found at <http://www.routeralley.com>.

Dynamic ARP Inspection

Another common man-in-the-middle attack is **ARP Spoofing** (sometimes referred to as **ARP Poisoning**). A malicious host can masquerade as another host, by intercepting ARP requests and responding with its *own* MAC address.

Dynamic ARP Inspection (DAI) mitigates the risk of ARP Spoofing, but inspecting all ARP traffic on *untrusted* ports. DAI will confirm that a legitimate MAC-to-IP translation has occurred, by comparing it against a trusted database. This MAC-to-IP database can be statically configured, or DAI can utilize the DHCP Snooping table (assuming DHCP Snooping has been enabled).

DAI can be globally enabled for a specific VLAN(s):

```
Switch(config)# ip arp inspection vlan 100
```

By default, all interfaces in VLAN 100 will be considered **untrusted**, and subject to inspection by DAI. Interfaces to other switches should be configured as **trusted** (no inspection will occur), as each switch should handle DAI locally:

```
Switch(config)# interface fa0/24
Switch(config-if)# ip arp inspection trust
```

To create a manual MAC-to-IP database for DAI to reference:

```
Switch(config)# arp access-list DAI_LIST
Switch(config-acl)# permit ip host 10.1.1.5 mac host 000a.1111.2222
Switch(config-acl)# permit ip host 10.1.1.6 mac host 000b.3333.4444
```

```
Switch(config)# ip arp inspection filter DAI_LIST vlan 100
```

If an ARP response does not match the MAC-to-IP entry for a particular IP address, then DAI drops the ARP response and generates a log message.

(Reference: http://www.cisco.com/en/US/docs/switches/lan/catalyst3560/software/release/12.2_20_se/configuration/guide/swdynarp.html)

* * *

All original material copyright © 2009 by Aaron Balchunas (aaron@routeralley.com), unless otherwise noted. All other material copyright © of their respective owners.

This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Updated material may be found at <http://www.routeralley.com>.