Introduction

- **Port security** is a security feature on Cisco switches.
- It allows you to control:
  - **What source MAC addresses** are allowed on a switch port.
  - **How many MAC addresses** are allowed on a switch port.
- It is covered under **exam topic 5.7**, which requires the ability to configure Layer 2 security features, including:
  - DHCP snooping
  - ARP inspection
  - Port security
- **DHCP snooping** and **ARP inspection** will be covered in upcoming videos.
- This video focuses on **port security**.

Topics Covered in This Video

- What port security is.
- Why port security is used and what security benefits it offers.
- Various **port security configuration commands**.
- A **bonus question** from Boson Software's ExSim for CCNA at the end of the video.

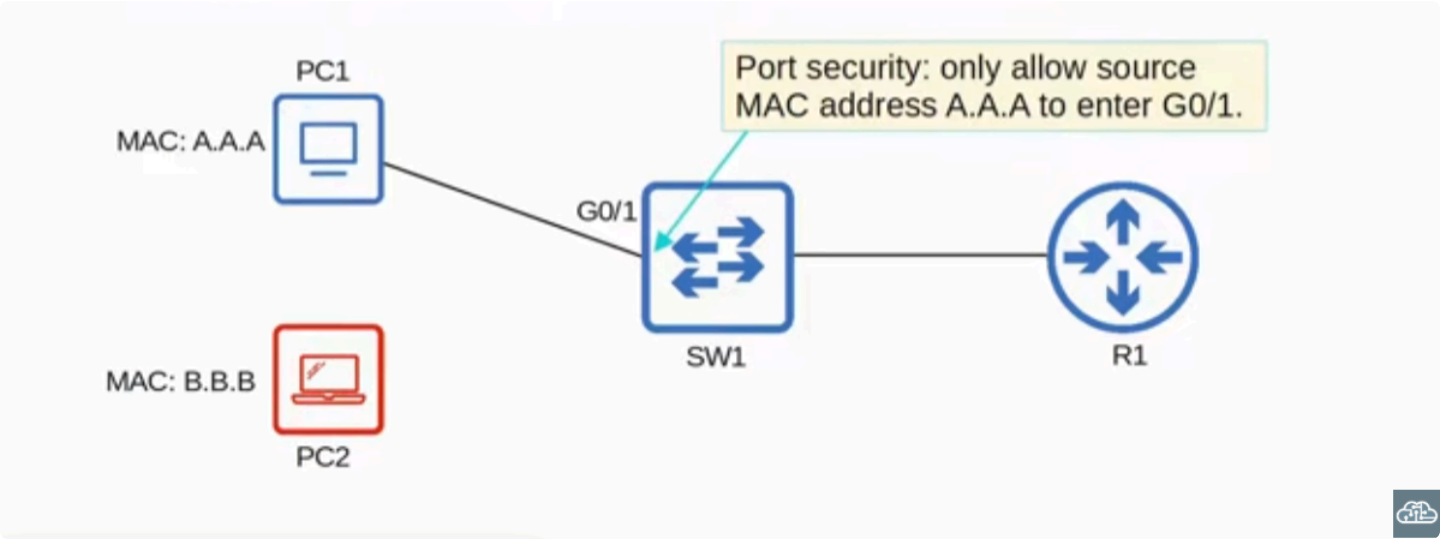
What is Port Security?

- **Port security** is a security feature of Cisco switches.

- It allows you to control **which source MAC address, or addresses, are allowed to enter a switchport**.
- It is configured on a **per-interface basis**.
- **Note:** Throughout the video, the terms **port** and **interface** are used interchangeably — they mean the same thing.
- When a frame with an **unauthorized source MAC address** enters the port, an **action** will be taken.
- There are a few possible actions that you can configure.
- The **default action** is to place the interface in an **err-disabled state**.
  - In effect, this is like **shutting down the interface**.
  - Traffic will no longer be sent or received by that interface.

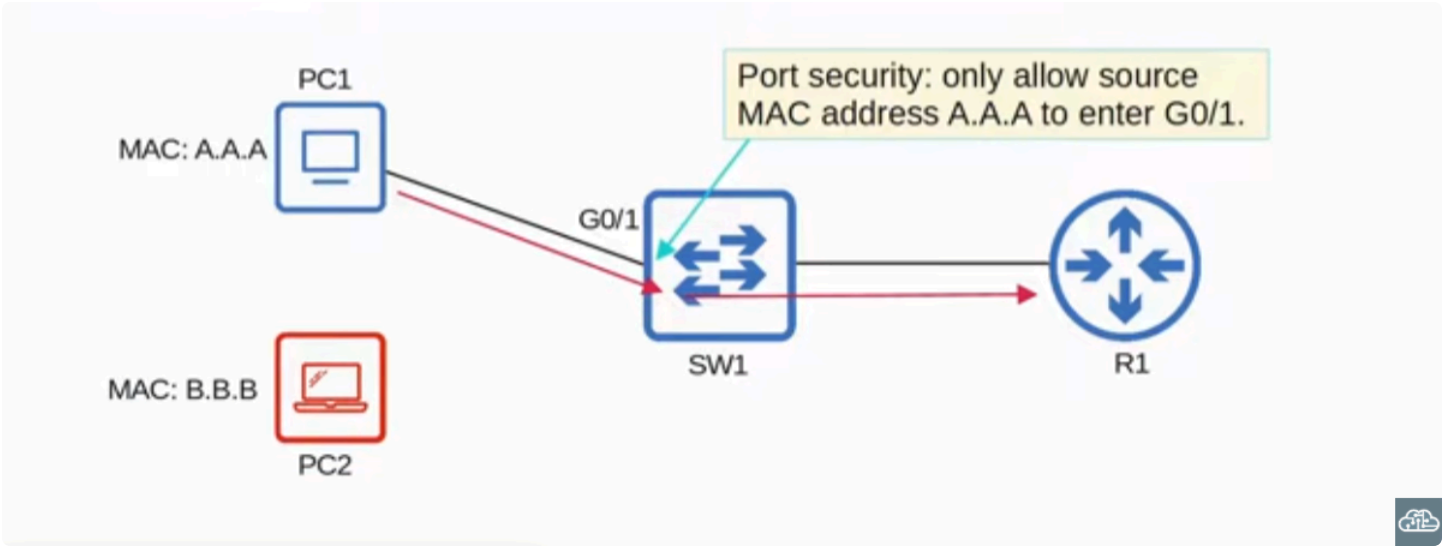
Example: Unauthorized Device Triggers Err-Disabled State

Scenario Setup

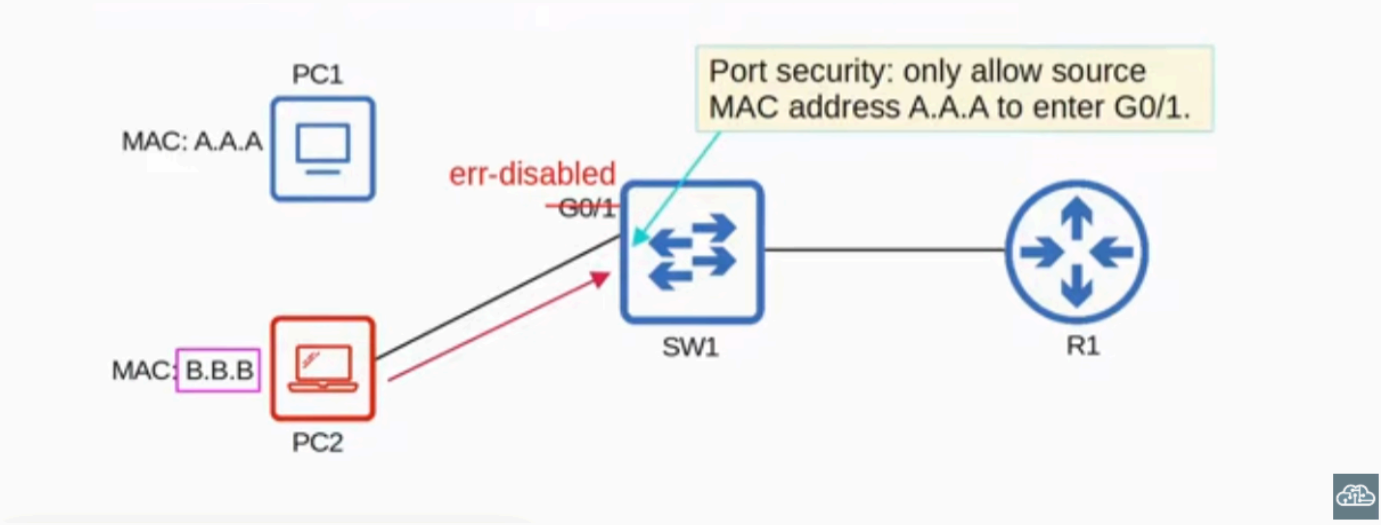


- **PC1** is connected to SW1's **G0/1** interface.
- PC1's MAC address is **A.A.A** (shortened for readability — actual MAC addresses are 12 hexadecimal characters).
- The user of PC1 brought in his personal laptop, **PC2**, which has a MAC address of **B.B.B**.
- The network admin configured port security on SW1's **G0/1** interface so it will only allow frames with a source MAC address of **A.A.A** to enter G0/1.

Sequence of Events

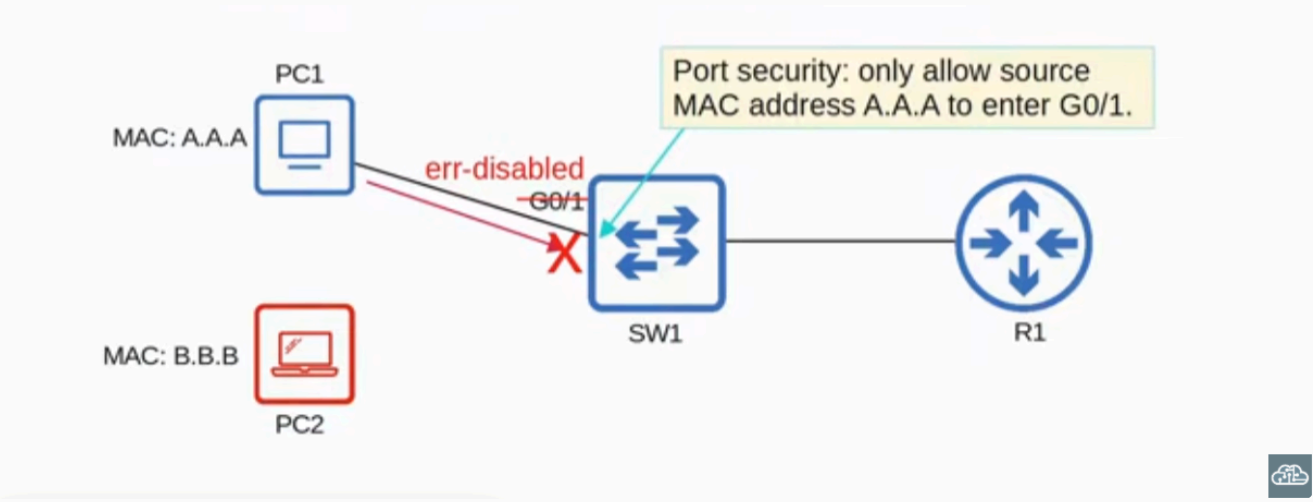


1. **PC1 sends a frame** — SW1 checks the source MAC address and sees that it is A.A.A, so it forwards it to the destination as normal.



2. **The user unplugs the cable from PC1** and connects it to the laptop, PC2.

3. **PC2 sends a frame** — SW1 checks the source MAC address and notices that it is B.B.B, but only A.A.A should be allowed. SW1 places G0/1 in an **err-disabled state**.
    - The interface will not send or receive data until it is enabled again.
    - For this example, the default action of **shutdown** is assumed; other possible actions are explained later in the video.



4. **The user notices his laptop is unable to communicate** — he unplugs the cable from the laptop and connects it back to PC1.
5. **PC1 sends a frame** — the interface is still err-disabled, so **PC1 is also unable to communicate** over the network.

Note: There are **two ways to enable an interface** that has been disabled by port security; they are covered later in the video.

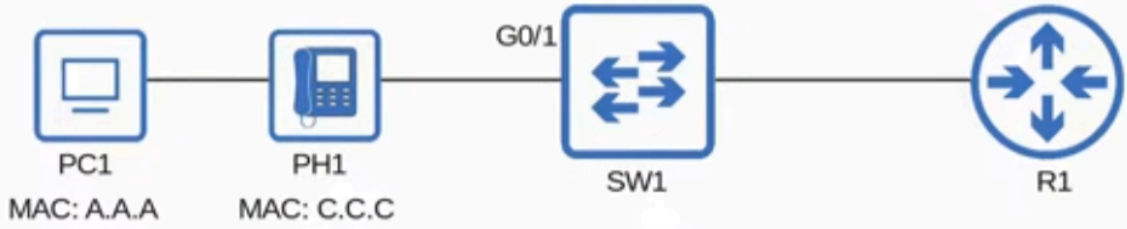
More Points About Port Security

- When you enable port security on an interface with the **default settings**, **one MAC address** is allowed.
- You can configure the allowed MAC address **manually** if you want.

- If you don't configure it manually, the switch will allow the **first source MAC address** that enters the interface.
  - That MAC address will be allowed on the interface.
  - **All others will be unauthorized.**
- However, you can change the **maximum number of MAC addresses** allowed.

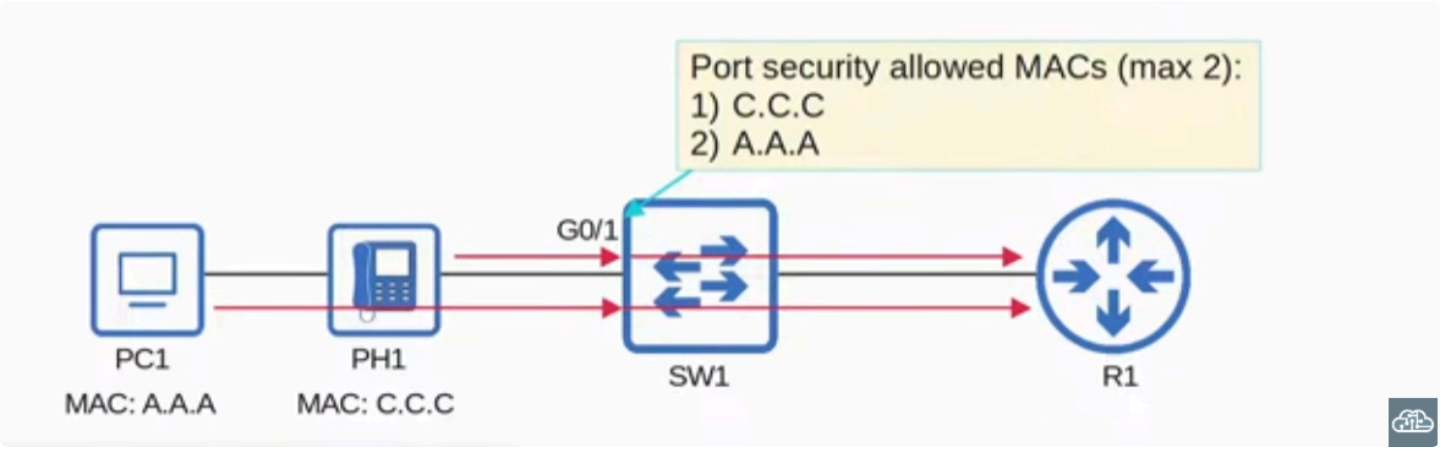
Example: Increasing the Maximum Number of MAC Addresses

Scenario Setup



- **Phone1** is directly connected to SW1.
- **PC1** is connected to Phone1.
- The default port security setting, which allows **one MAC address**, won't work in this situation because both PC1 and Phone1 will send traffic using their own MAC address as the source — that's **two MAC addresses**.
- SW1's **G0/1** interface is configured to allow **two MAC addresses**.
- The MAC addresses are **not configured manually** — SW1 will **dynamically learn** which two MAC addresses to allow.

Sequence of Events

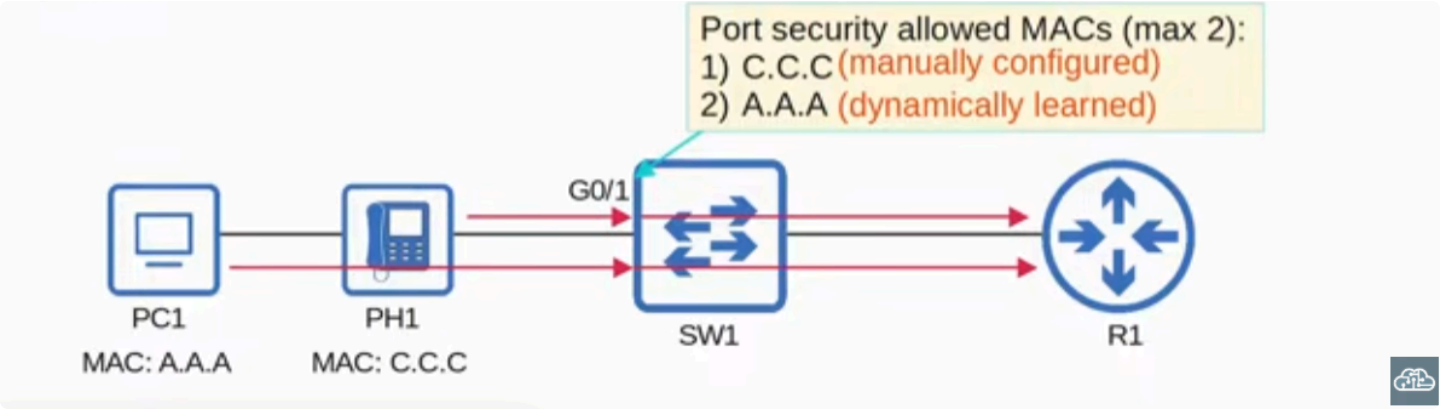


1. **Phone1 sends a frame** — SW1 adds it to the list of allowed MAC addresses and forwards it as normal.
2. **PC1 sends a frame** — SW1 adds it to the list of allowed MAC addresses and forwards it as normal.
3. **The interface has reached its maximum** number of allowed MAC addresses (configured as 2).
4. **Another device sends a frame** — SW1 shuts down the interface because the source MAC address is **not authorized**.

Two Main Points Introduced

- **Point 1:** The default number of allowed MAC addresses is **one** when you enable port security, but you can configure it to allow more.
- **Point 2:** The allowed MAC addresses can be **manually configured** or **dynamically learned**.

Example: Combination of Manual and Dynamic Configuration



- In the previous example, both MAC addresses were **dynamically learned**.

- However, we could have **manually configured** SW1 to allow **C.C.C** on G0/1.
- Then allowed it to **dynamically learn A.A.A** when PC1 sends a frame.
- If more than one MAC address is allowed, they don't all have to be manually configured or all dynamically learned — a **combination is possible**.

Why Use Port Security?

- **Port security** is useful because it allows network admins to **control which devices are allowed to access the network**.
- Someone can't just plug an **unauthorized device** into a switchport and gain access to the network.
- **However: MAC address spoofing** is a simple task.
  - It's easy to configure a device to send frames with a **different source MAC address**.
  - Be aware that port security **isn't a perfect solution** in this regard.
- Rather than manually specifying the MAC addresses allowed on each port, port security's ability to **limit the number of MAC addresses** allowed on an interface is **often more useful**.

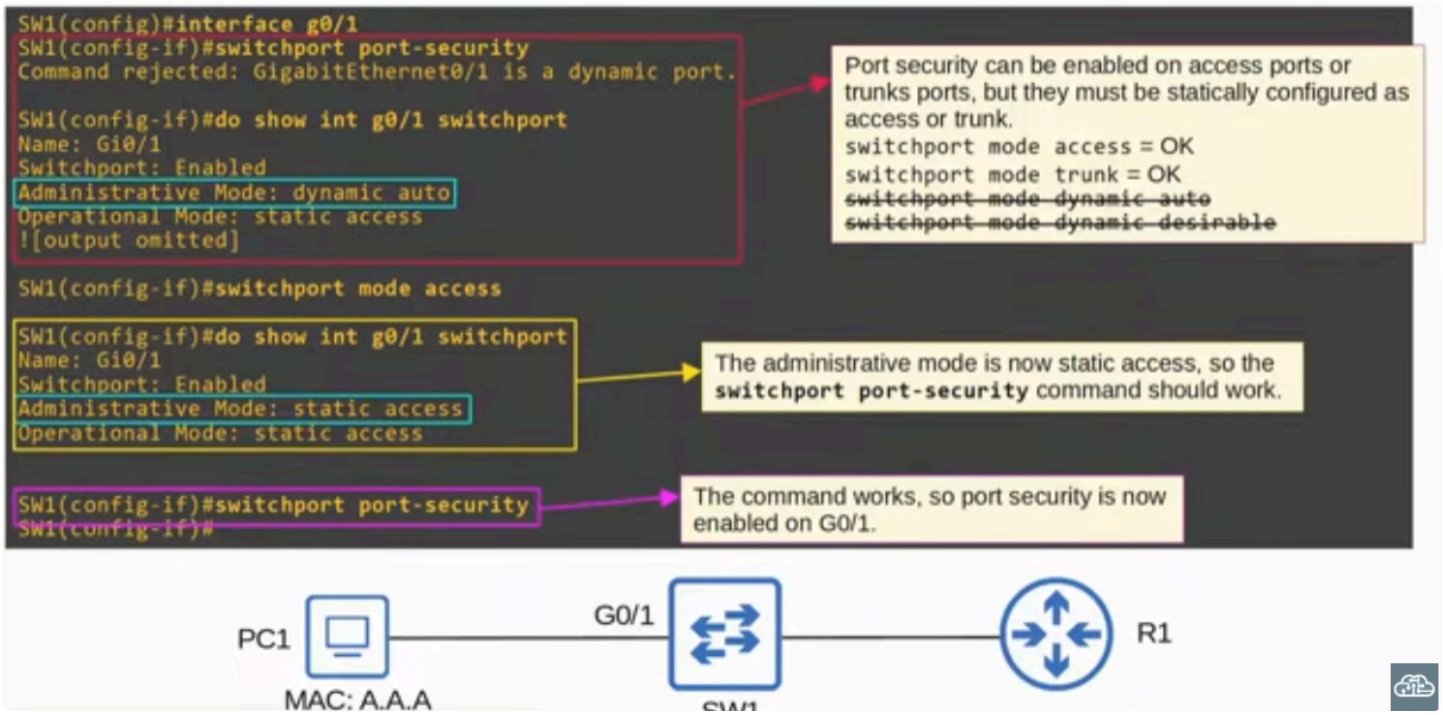
Example: DHCP Starvation Attack

- Think back to the **DHCP starvation attack** carried out in the Day 48 lab video.
- The attacker **spoofed thousands of fake MAC addresses**.
- The DHCP server assigned IP addresses to those fake MAC addresses, **exhausting the DHCP pool**.
- Not just that — switches **can't learn an infinite number of MAC addresses**, so the switch's **MAC address table can also become full** due to such an attack.
- Then the switch **can no longer learn new MAC addresses**, and it will **flood every packet it receives**.
- Using port security to **limit the number of MAC addresses** allowed on an interface can **protect against such attacks**.

Summary

- **Both aspects of port security are useful:**
  - Controlling **which MAC addresses** are allowed.
  - Controlling **how many MAC addresses** are allowed.
- The video will continue to look at **both aspects**.

Basic Port Security Configurations



- **Port security** is enabled directly on the interface.
- The professor enters interface configuration mode for **G0/1** and tries the command `switchport port-security`.

```
SW1(config)# interface gigabitethernet0/1
SW1(config-if)# switchport port-security
```

- **However, the command is rejected** with a message saying that GigabitEthernet0/1 is a **dynamic port**.
  - The professor then asks: "What does that mean?"
  - To check, he uses the `show interfaces g0/1 switchport` command.

Verification:

```
SW1# show interfaces g0/1 switchport
```

- The output shows that, by default, switchports have an **administrative mode of dynamic auto**.
  - This is the DTP command `switchport mode dynamic auto`, covered earlier in the course.
- **Port security can be enabled on access ports or trunk ports**, but they must be **statically configured** as access or trunk.
  - **Dynamic auto** and **dynamic desirable** are not allowed.
- So, the professor configures G0/1 as a **static access port** with `switchport mode access`.

```
SW1(config-if)# switchport mode access
```

- Then he checks the administrative mode again using `show interfaces g0/1 switchport`.

Verification:

```
SW1# show interfaces g0/1 switchport
```

- The administrative mode is now **static access**.
- The professor then issues the `switchport port-security` command again.

```
SW1(config-if)# switchport port-security
```

- **Indeed it works** — port security is now enabled on G0/1.
- When you use just this command, port security is enabled with the **default settings**.

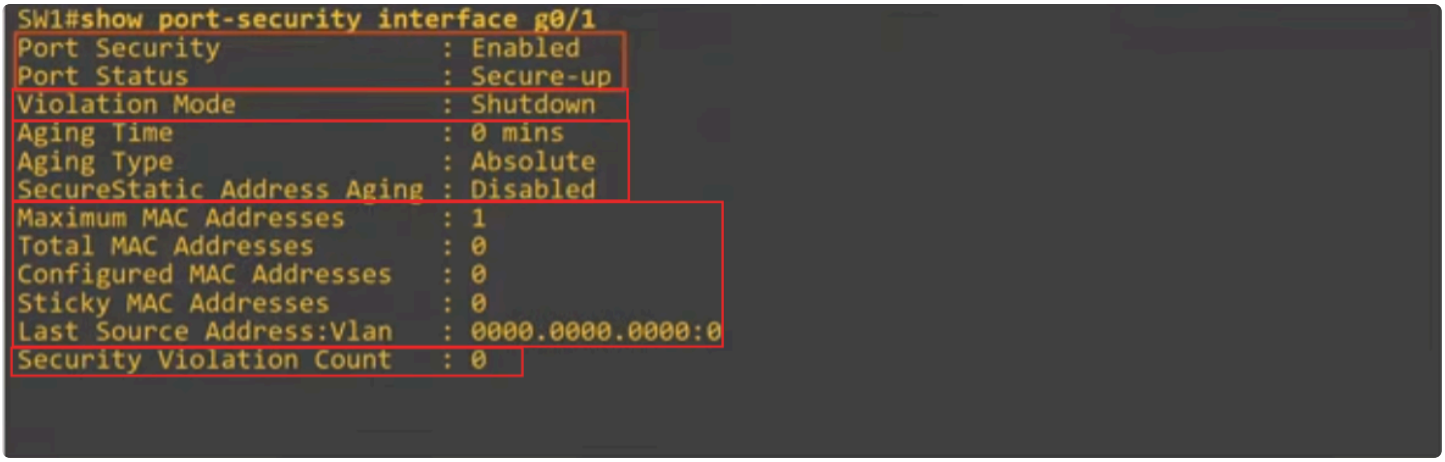
Note: The original command was rejected because the port was dynamically configured. A port must be statically set to access or trunk before port security can be enabled.

Viewing Port Security Settings

- The command **show port-security interface**, followed by the interface name, is very useful.

Verification:

```
SW1# show port-security interface g0/1
```



- **Port security is enabled**, and the **port status is secure-up**.
  - **Secure-up** just means port security is enabled, and the interface is up.
- The **default violation mode is shutdown**, as stated earlier.
  - If an unauthorized frame enters the interface, it will be **err-disabled**.
- There are some **default settings regarding the timers**:
  - The **aging time of 0 minutes** means that the addresses will **never age out** — there is no timer.
  - These will be explained later in the video.
- There is information about the **MAC addresses**:
  - The **maximum is 1**.
  - **Currently it knows 0**.
  - **0 have been manually configured**.
  - There are **0 sticky MAC addresses** — also mentioned later in the video.
- SW1 hasn't received any traffic on this interface yet, so the **last source address is all 0s**, with **VLAN number 0**.
- There have been **no violations**, so the **violation counter is at 0**.

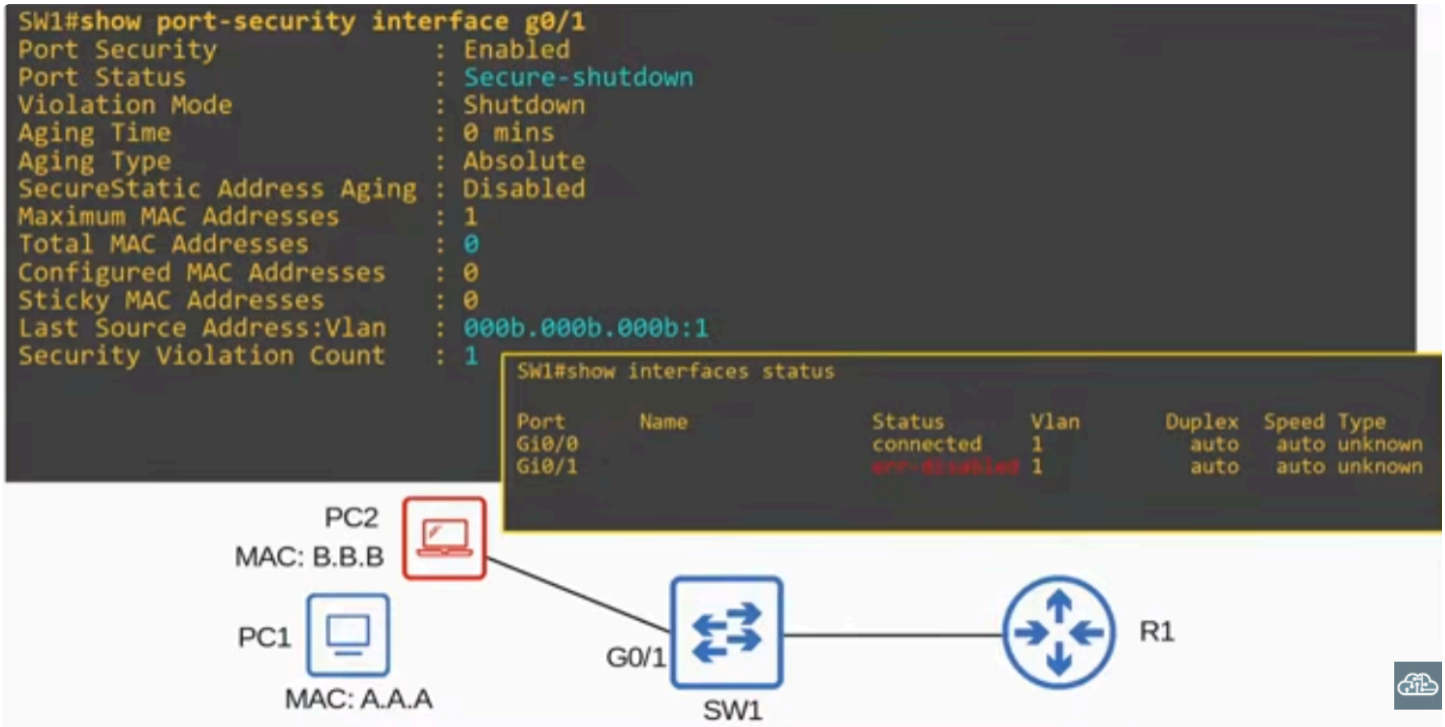
Example: PC1 Sends a Ping to R1

```
SW1#show port-security interface g0/1
Port Security              : Enabled
Port Status                : Secure-up
Violation Mode             : Shutdown
Aging Time                 : 0 mins
Aging Type                 : Absolute
SecureStatic Address Aging : Disabled
Maximum MAC Addresses      : 1
Total MAC Addresses        : 1
Configured MAC Addresses   : 0
Sticky MAC Addresses       : 0
Last Source Address:Vlan   : 000a.000a.000a:1
Security Violation Count   : 0
```

- The professor sends a ping from **PC1 to R1** and checks the output again.
- **Two places have changed:**

1. **Total MAC addresses** has changed from **0 to 1**, because SW1 learned PC1's MAC address.
    - **Note:** The maximum is also 1, so SW1 won't be able to learn any more MAC addresses on the interface.
2. **Last source address** has changed to **PC1's MAC address**, and the **VLAN is 1**, the default VLAN.

Example: PC2 is Connected Instead

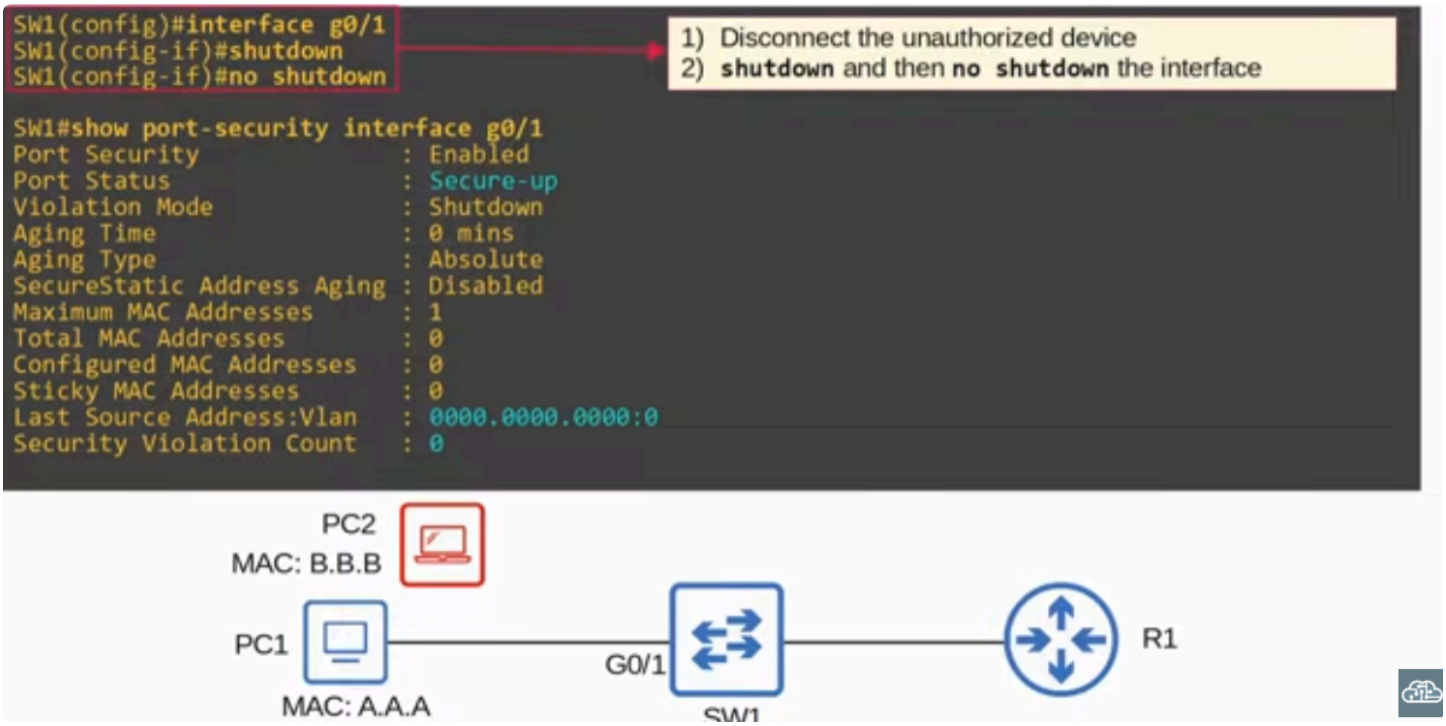


- The professor brings back PC2 and connects the cable to it instead.
- **What happens when PC2 tries to ping R1?**

1. From the top of the output, the **port status has changed from secure-up to secure-shutdown**.
    - **Note:** If you check `show interfaces status`, you will see the status **err-disabled**, as mentioned earlier. But in the `show port-security interface` command, it says **secure-shutdown**.
2. The **total MAC addresses count has reset to 0**.
    - So, it dynamically learned PC1's MAC address as a **secure MAC address**, but after the port was shutdown, it was **cleared**.

3. The **last source address** is **PC2's MAC address, B.B.B**.
4. The **security violation count is now 1**.

Re-Enabling an Interface Disabled by Port Security



- The professor demonstrates how to **manually re-enable** an interface that has been disabled by port security.

Step 1 — Disconnect the Unauthorized Device

- **Before entering any commands**, you should first **disconnect the unauthorized device**.

Step 2 — Manually Re-Enable the Interface

- After disconnecting the unauthorized device, enter the following commands on the interface:
  - `shutdown` — puts the interface in **administratively disabled mode**.
  - `no shutdown` — **re-enables** the interface.

```
SW1(config-if)# shutdown
SW1(config-if)# no shutdown
```

Verify the Interface Status

- Check the output of the `show port-security interface` command:

Verification:

```
SW1# show port-security interface g0/1
```

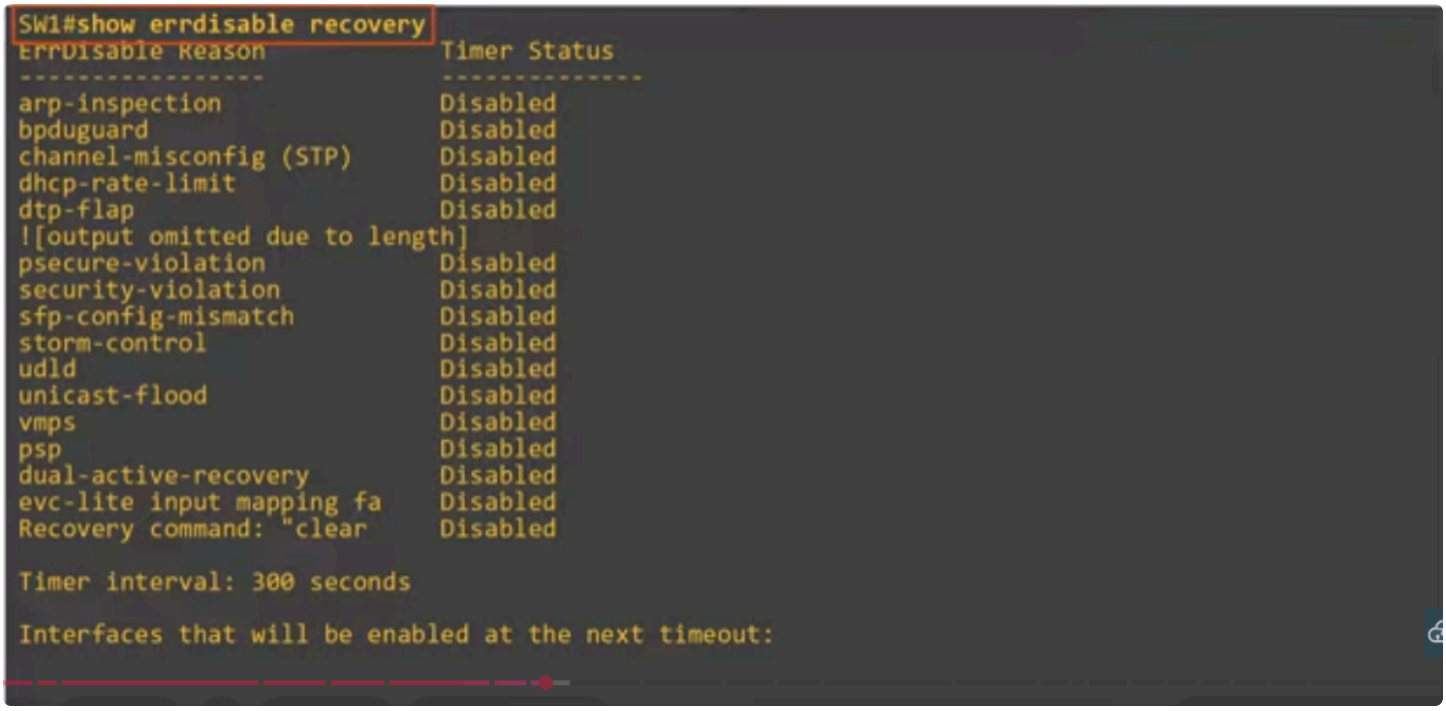
- The **port status is back to secure-up**.
- The **last source address**, which was PC2's MAC address before, has been **erased**.
- At the bottom, the **security violation count** was also **reset** when the interface came back up.

Note: With the default violation mode, **shutdown**, the security violation count **shouldn't go higher than 1**.

ErrDisable Recovery — Alternative Way to Re-Enable an Interface

- There is **another way** to re-enable an err-disabled interface, called **ErrDisable recovery**.
- It causes **err-disabled interfaces** to be **automatically re-enabled** after a certain period of time.
- There are actually **various reasons** an interface can enter an **err-disabled state**.

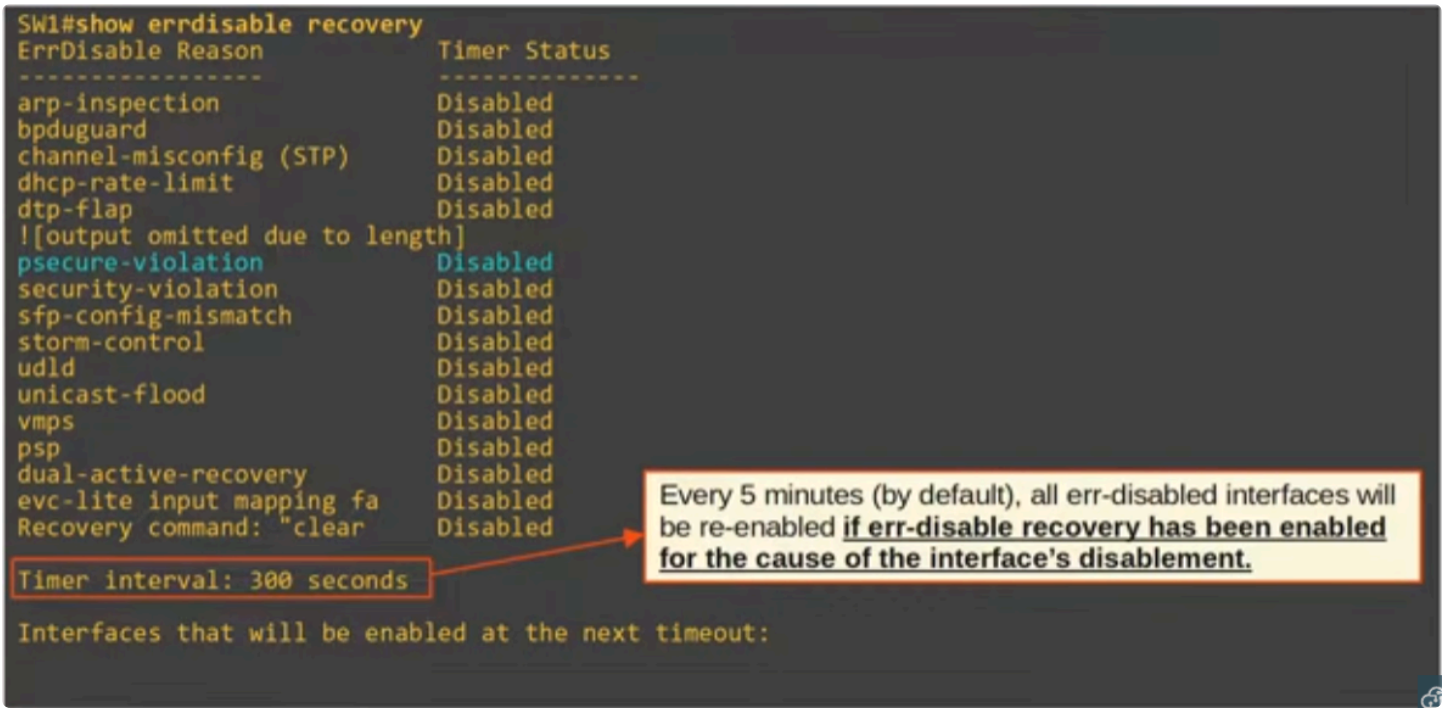
Viewing ErrDisable Recovery Information



- The professor uses the command `show errdisable recovery`, which lists all of the reasons.
- There are so many reasons that the professor had to omit a lot of them — the output doesn't fit on one screen.

Verification:

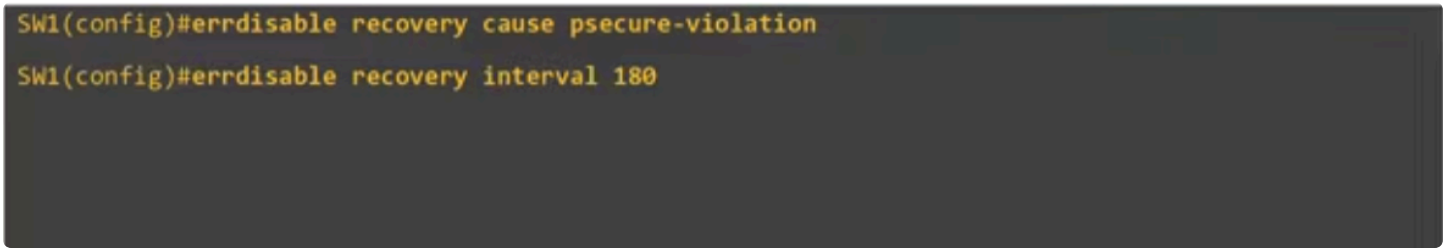
```
SW1# show errdisable recovery
```



- **On the left** is each **err-disable reason**.
- **On the right** it will show whether or not **err-disable recovery is enabled**.
- **By default, it is disabled for all reasons**, so err-disabled interfaces will **not** be automatically recovered.
- The reason we're looking for is **psecure-violation**, which means **port-security violation**.
- **Note:** The **default timer is 300 seconds**.

Key Point: By default, every **5 minutes**, all err-disabled interfaces will be re-enabled — but only if **err-disable recovery has been enabled** for the cause of the interface's disablement.

Enabling ErrDisable Recovery for Port Security Violations



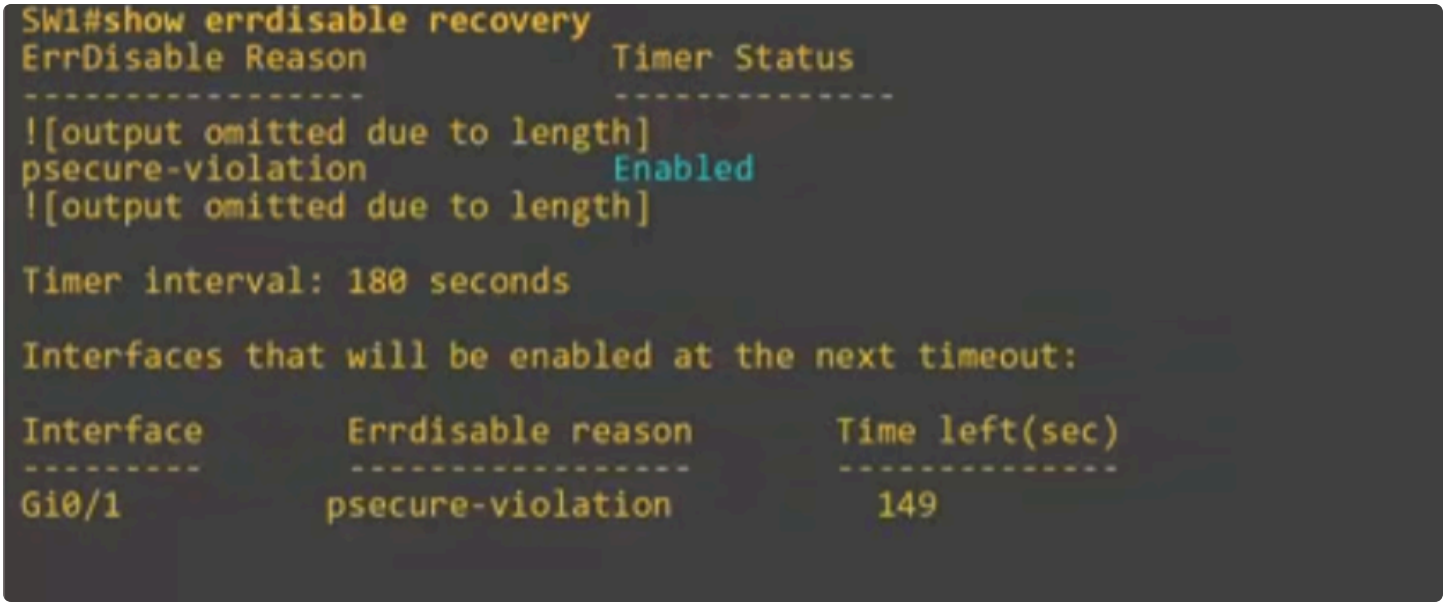
- The command to enable it is `errdisable recovery cause`, followed by the cause, which is **psecure-violation** in this case.

```
SW1(config)# errdisable recovery cause psecure-violation
```

- Just to demonstrate the command, the professor also **shortened the timer** with `errdisable recovery interval`, specifying **180 seconds**.

```
SW1(config)# errdisable recovery interval 180
```

Verifying the Configuration



- The professor checks `show errdisable recovery` again.

Verification:

```
SW1# show errdisable recovery
```

```
SW1(config)#errdisable recovery cause psecure-violation

SW1(config)#errdisable recovery interval 180


SW1#show errdisable recovery
ErrDisable Reason            Timer Status
-----------------            --------------
![output omitted due to length]
psecure-violation            Enabled
![output omitted due to length]

Timer interval: 180 seconds

Interfaces that will be enabled at the next timeout:

Interface       Errdisable reason       Time left(sec)
---------       -----------------       --------------
Gi0/1          psecure-violation          149
```

- The **psecure-violation recovery timer is now enabled**.
- The **timer interval is 180 seconds**, as configured.

```
SW1(config)#errdisable recovery cause psecure-violation

SW1(config)#errdisable recovery interval 180


SW1#show errdisable recovery
ErrDisable Reason            Timer Status
-----------------            --------------
![output omitted due to length]
psecure-violation            Enabled
![output omitted due to length]

Timer interval: 180 seconds

Interfaces that will be enabled at the next timeout:

Interface       Errdisable reason       Time left(sec)
---------       -----------------       --------------
Gi0/1          psecure-violation          149
```

- Just to demonstrate, the professor caused G0/1 to become err-disabled again.
  - You can see that it **will be enabled at the next timeout**.
  - There are **149 seconds left**.

Important Warning — Always Disconnect the Unauthorized Device

- This is a **useful feature**, but it is **useless** if you don't remove the device that caused the interface to enter the err-disabled state.
- **That will always be step 1**: disconnect the unauthorized device.
- Then, either:
  - **Manually re-enable** the interface, or
  - Let **errdisable recovery** do it for you automatically.

What Happens If You Don't Disconnect the Unauthorized Device?

- If you **manually configured the secure MAC address**: the interface will simply **become disabled again** when it receives another frame from the unauthorized device.
- If you let the switch **dynamically learn** the previous secure MAC address: it is **cleared when the interface is disabled**.
  - When the interface is re-enabled, the **unauthorized device's MAC address might become the new secure MAC address** on the interface.
  - This is **obviously not a good situation**.

Note: Remember to **disconnect the unauthorized device**.

Violation Modes

- The professor just showed the default mode, **shutdown**, and how to re-enable an interface shut down by port security.
- There are **three different violation modes** that determine what the switch will do if an unauthorized frame enters an interface configured with port security.

Shutdown Mode (Default)

- This is the **default** violation mode.
- It **effectively shuts down the port** by placing it in an **err-disabled state** if an unauthorized frame enters the port.
- It will also generate a **Syslog and/or SNMP message** to let you know that port security disabled the interface.

- **However**, after the interface is down, it **won't continue generating messages** even if the unauthorized device continues trying to send traffic.
    - Only **one message** is generated to say that the port was disabled.
- The **violation counter is set to 1** when the interface is disabled.
    - It will be **reset to 0** when the interface is re-enabled, as shown earlier.

Restrict Mode

- The switch will **discard traffic from unauthorized MAC addresses**.
- **However**, the interface is **not disabled**.
    - Devices with **authorized MAC addresses** will still be able to use the interface.
- The switch generates a **Syslog and/or SNMP message** each time an unauthorized MAC address is detected.
- The **violation counter is incremented by 1** for each unauthorized frame.

Protect Mode

- Like restrict mode, the switch **discards traffic from unauthorized MAC addresses**, and the interface is **not disabled**.
- **However**, it does **not generate Syslog or SNMP messages** for unauthorized traffic.
- It **doesn't increment the violation counter** either.
- It just **silently discards** unauthorized traffic.

Restrict Violation Mode

- The professor demonstrates the **restrict** violation mode with a configuration example.

Example: Configuring and Testing Restrict Mode

Step 1 — Start with a Fresh Port-Security Configuration

```
SW1(config-if)#switchport port-security
SW1(config-if)#switchport port-security mac-address 000a.000a.000a
```

- First, enable port security on the interface.

```
SW1(config)# interface gigabitethernet0/1
SW1(config-if)# switchport port-security
```

Step 2 — Manually Authorize PC1's MAC Address

```
SW1(config-if)#switchport port-security
SW1(config-if)#switchport port-security mac-address 000a.000a.000a
SW1(config-if)#switchport port-security violation restrict
```

- This time, the professor manually authorizes PC1's MAC address with the following command.

```
SW1(config-if)# switchport port-security mac-address aaaa.bbbb.cccc
```

Note: In this example, `aaaa.bbbb.cccc` represents PC1's MAC address.

Step 3 — Enable Restrict Mode

```
SW1(config-if)#switchport port-security
SW1(config-if)#switchport port-security mac-address 000a.000a.000a
SW1(config-if)#switchport port-security violation restrict
```

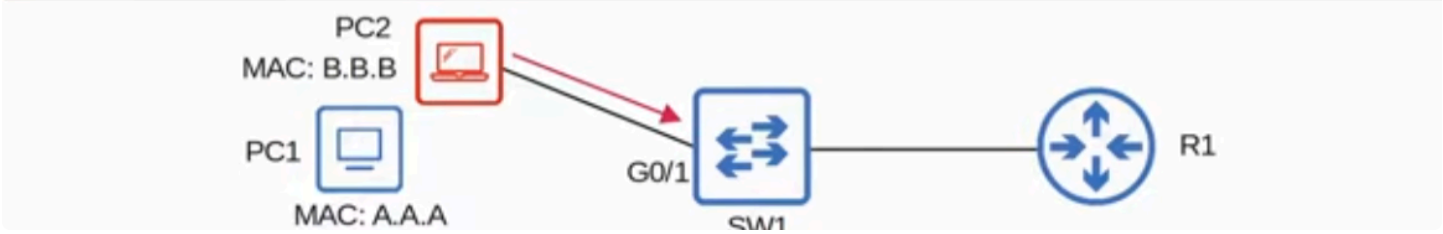
- Use the following command to set the violation mode to **restrict**.

```
SW1(config-if)# switchport port-security violation restrict
```

Step 4 — Send Traffic from PC2

```
SW1(config-if)#switchport port-security
SW1(config-if)#switchport port-security mac-address 000a.000a.000a
SW1(config-if)#switchport port-security violation restrict

*May 23 22:54:09.951: %PORT_SECURITY-2-PSECURE_VIOLATION: Security violation occurred, caused by MAC
address 000b.000b.000b on port GigabitEthernet0/1.
```



- The professor then tried to ping R1 from PC2.
- A bunch of **syslog messages** appeared, telling that a **security violation occurred**, caused by **PC2's MAC address**, on the **G0/1 interface**.

Step 5 — Verify the Output

Verification:

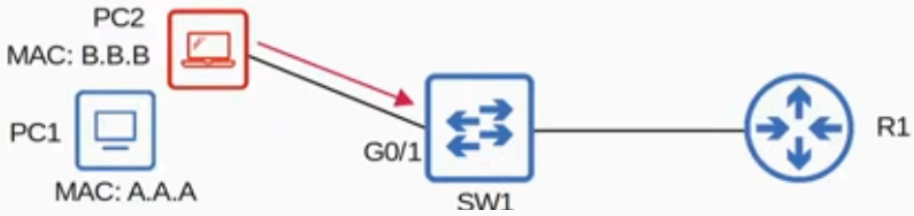
```
SW1# show port-security interface g0/1
```

- The output shows:

```
SW1(config-if)#switchport port-security
SW1(config-if)#switchport port-security mac-address 000a.000a.000a
SW1(config-if)#switchport port-security violation restrict

*May 23 22:54:09.951: %PORT_SECURITY-2-PSECURE_VIOLATION: Security violation occurred, caused by MAC
address 000b.000b.000b on port GigabitEthernet0/1.

SW1#show port-security interface g0/1
Port Security              : Enabled
Port Status                : Secure-up
Violation Mode             : Restrict
Aging Time                 : 0 mins
Aging Type                 : Absolute
SecureStatic Address Aging : Disabled
Maximum MAC Addresses      : 1
Total MAC Addresses        : 1
Configured MAC Addresses   : 1
Sticky MAC Addresses       : 0
Last Source Address:Vlan   : 000b.000b.000b:1
Security Violation Count   : 12
```



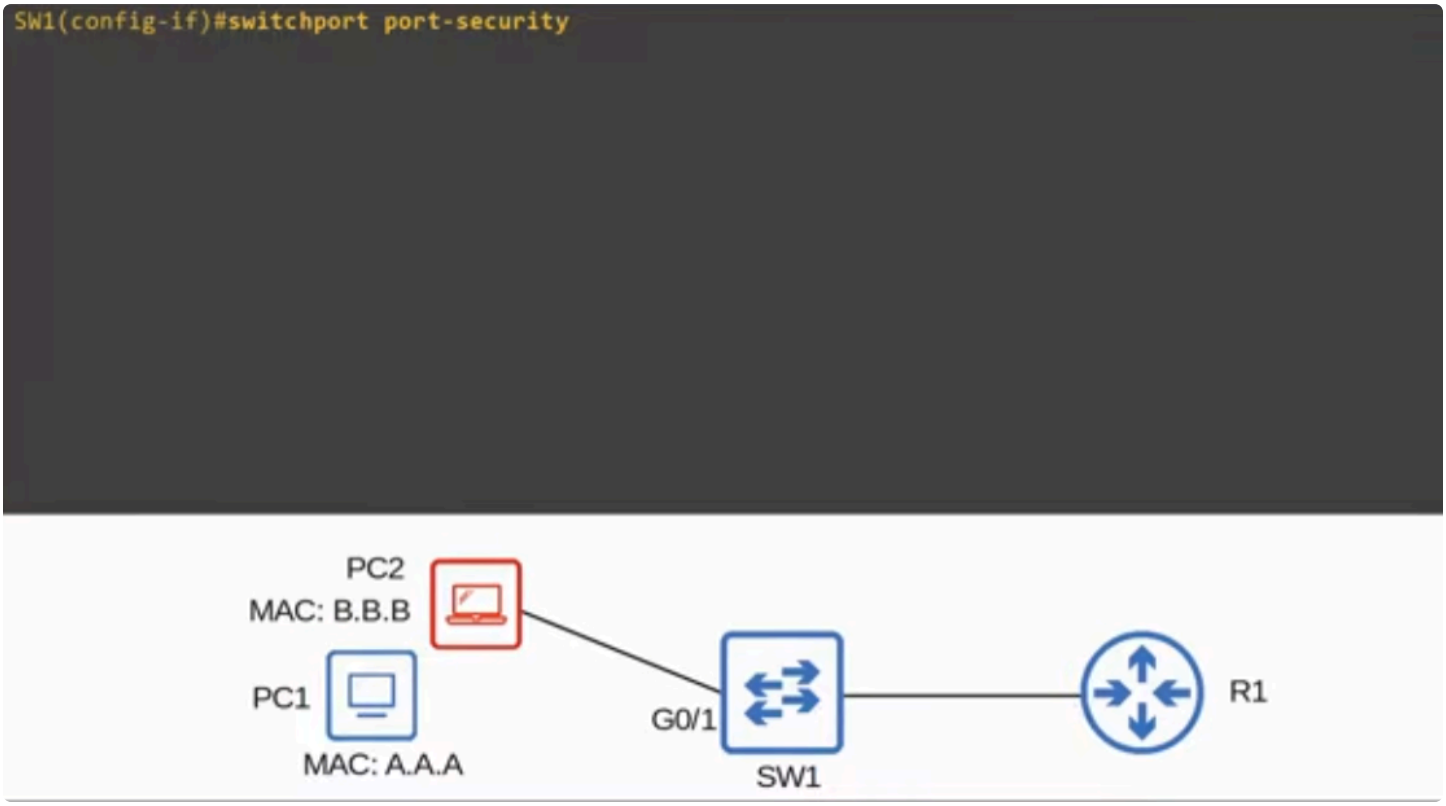
  - The **violation mode is restrict**.
  - The **violation count has gone to twelve** because the professor tried to send traffic from PC2.
  - **However**, the **port status is secure-up**, not secure-shutdown.
  - If the cable were connected back to PC1, it would **still be able to send traffic**, no problem, because the interface is still up and PC1's MAC address is **authorized**.

Protect Violation Mode

- The professor demonstrates the **protect** violation mode with a configuration example.

Example: Configuring and Testing Protect Mode

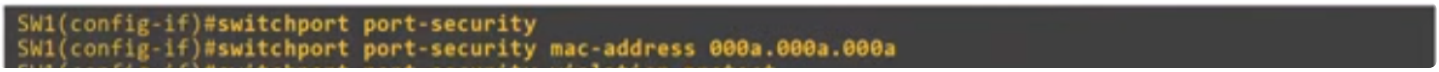
Step 1 — Start with a Fresh Port-Security Configuration



- Make sure to enable port security on the interface.

```
SW1(config)# interface gigabitethernet0/1
SW1(config-if)# switchport port-security
```

Step 2 — Manually Authorize PC1's MAC Address



- The professor once again manually authorizes PC1's MAC address.

```
SW1(config-if)# switchport port-security mac-address aaaa.bbbb.cccc
```

Note: In this example, `aaaa.bbbb.cccc` represents PC1's MAC address.

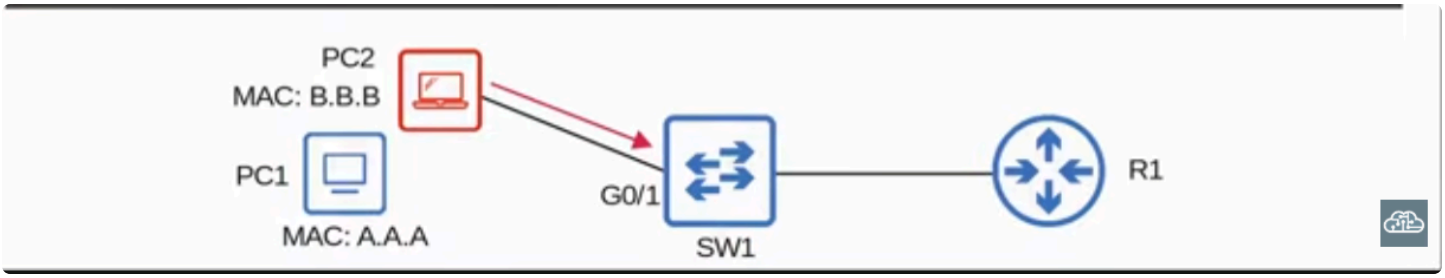
Step 3 — Enable Protect Mode

```
SW1(config-if)#switchport port-security
SW1(config-if)#switchport port-security mac-address 000a.000a.000a
SW1(config-if)#switchport port-security violation protect
```

- Configure the violation mode to **protect**.

```
SW1(config-if)# switchport port-security violation protect
```

Step 4 — Send Traffic from PC2



- The professor then sent some traffic from PC2.
- **PC2's pings failed**, but there were **no syslog messages** on SW1.

Step 5 — Verify the Output

Verification:

```
SW1# show port-security interface g0/1
```

- The output shows:

```
SW1(config-if)#switchport port-security
SW1(config-if)#switchport port-security mac-address 000a.000a.000a
SW1(config-if)#switchport port-security violation protect


SW1#show port-security interface g0/1
Port Security              : Enabled
Port Status                : Secure-up
Violation Mode             : Protect
Aging Time                 : 0 mins
Aging Type                 : Absolute
SecureStatic Address Aging : Disabled
Maximum MAC Addresses      : 1
Total MAC Addresses        : 1
Configured MAC Addresses   : 1
Sticky MAC Addresses       : 0
Last Source Address:Vlan   : 000b.000b.000b:1
Security Violation Count   : 0
```

  - The **port status is secure-up**.
  - The **violation mode is protect**.
  - The **violation counter is 0**.
- So, SW1 **discarded the traffic from PC2**, but **didn't display any syslog messages** or **increment the violation count**.

Summary of Violation Modes

There are three different violation modes that determine what the switch will do if an unauthorized frame enters an interface configured with port security.

- **Shutdown**
  - → Effectively shuts down the port by placing it in an err-disabled state.
  - → Generates a Syslog and/or SNMP message when the interface is disabled.
  - → The violation counter is set to 1 when the interface is disabled.
- **Restrict**
  - → The switch discards traffic from unauthorized MAC addresses.
  - → The interface is NOT disabled.
  - → Generates a Syslog and/or SNMP message each time an unauthorized MAC is detected.
  - → The violation counter is incremented by 1 for each unauthorized frame.
- **Protect**
  - → The switch discards traffic from unauthorized MAC addresses.
  - → The interface is NOT disabled.
  - → It does NOT generate Syslog/SNMP messages for unauthorized traffic.
  - → It does NOT increment the violation counter.

- These are how you control what the switch does when a **port-security violation** occurs.
- You should definitely learn the **actions taken by each violation mode**.
- **Remember: shutdown** is the **default mode**.

Secure MAC Address Aging

- The professor moves to the next part of the `show port-security interface` command — **secure MAC address aging**.
- **Definition:** MAC addresses **dynamically learned** or **statically configured** on a port-security enabled port are called **secure MAC addresses**.
- **By default**, secure MAC addresses will **not age out**.
  - There is **no timer** — they are **permanent** unless you:
    - **Manually delete** the learned MAC address, or
    - The port is **disabled and then re-enabled**.
  - That's what the **aging time of 0 minutes** means.

Configuring the Aging Time

- You can configure the timer with the following command, followed by the time in minutes:

```
SW1(config-if)# switchport port-security aging time <minutes>
```

Aging Type — Absolute

- If you do configure an aging time, the **default aging type is absolute**.
- **Absolute aging** means that:
  - After the secure MAC address is learned, the **aging timer starts**.
  - The MAC address is **removed after the timer expires**, even if the switch **continues receiving frames** from that source MAC address while it is counting down.
- **However**, after the MAC address ages out, it can then be **immediately re-learned** if another frame with that source MAC is received.

Aging Type — Inactivity

- The other aging type is **inactivity**.
- This is **like regular MAC address aging**:
  - After the MAC address is learned, the **aging timer starts**.
  - But it is **reset every time** a frame from that source MAC address is received.
  - So, if the switch **keeps receiving frames** from that MAC address, it will **never be aged out**.

Configuring the Aging Type

- You can configure the aging type with the following command, followed by `absolute` or `inactivity`:

```
SW1(config-if)# switchport port-security aging type absolute
```

or

```
SW1(config-if)# switchport port-security aging type inactivity
```

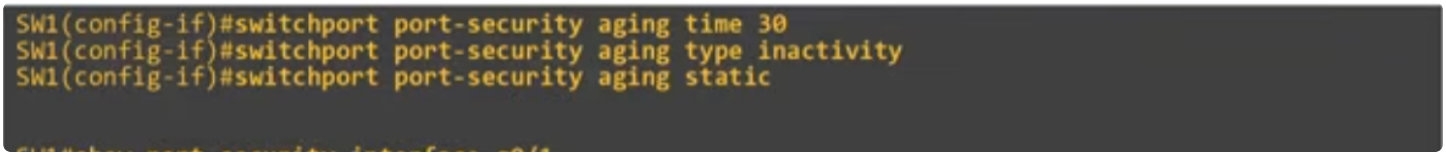
Aging of Static Secure MAC Addresses

- **By default**, only **dynamically-learned** secure MAC addresses will age out.
- If you configure a MAC with `switchport port-security mac-address` (shown earlier), it **won't age out**.
  - The command will **remain in the running-config**.
  - The MAC will **remain in the MAC address table**.
- **However**, with the following command, you can make the switch **age out static secure MAC addresses, too**:

```
SW1(config-if)# switchport port-security aging static
```

- If a static secure MAC address ages out:
  - The command will be **removed from the running config**.
  - The address will be **removed from the MAC address table**.

CLI Demonstration of Secure MAC Address Aging



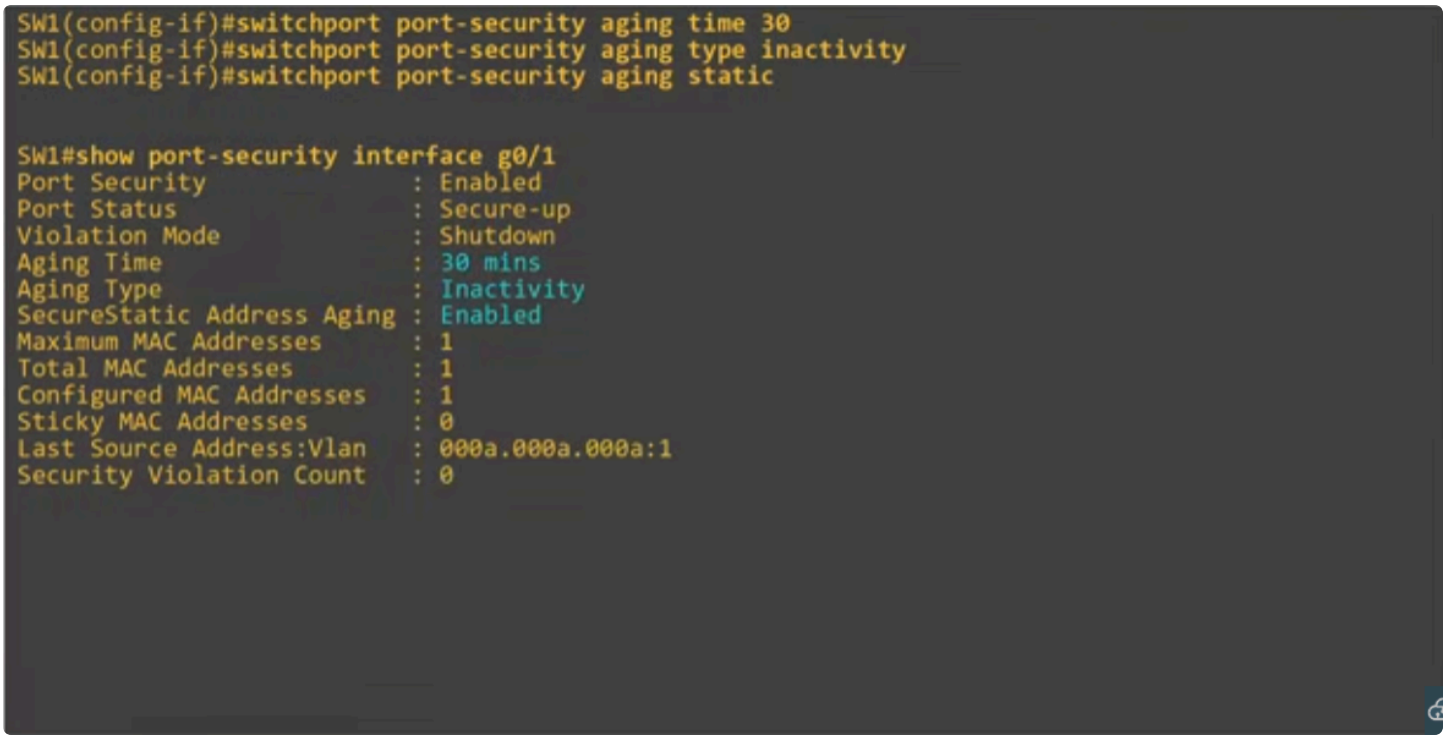
- The professor configures an **aging time of 30 minutes**, an **aging type of inactivity**, and **enables aging of static secure MAC addresses**.

```
SW1(config-if)# switchport port-security aging time 30
SW1(config-if)# switchport port-security aging type inactivity
SW1(config-if)# switchport port-security aging static
```

- Then he checks **`show port-security interface g0/1`** again.

Verification:

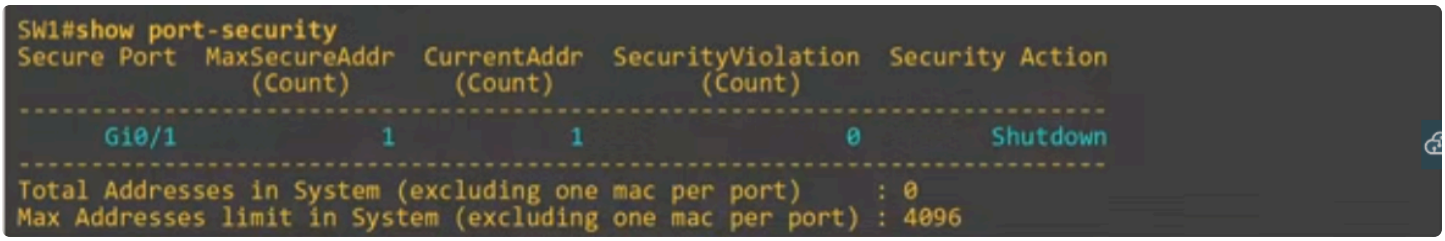
```
SW1# show port-security interface g0/1
```



- The output has changed:
  - **Aging time 30 minutes**.
  - **Aging type inactivity**.
  - **Secure static address aging enabled**.

Note: That's all you really need to know about the timers.

SHOW PORT-SECURITY — Overview Command



- Before moving on to the next topic, the professor shows one more useful command: **show port-security**.

Verification:

```
SW1# show port-security
```

- It displays:
  - **Which interfaces** have port security enabled.
  - The **max and current number of secure addresses** on those interfaces.
  - Their **security violation count**.
  - Their **security action**.
- In this case, port security is only enabled on **one port**, but if you have it enabled on many, this is a **useful command** to get an **overview** of your port security enabled interfaces.

Sticky Secure MAC Addresses

- **Sticky secure MAC address learning** is the last major topic of the video.
- It can be enabled with the following command:

```
SW1(config-if)# switchport port-security mac-address sticky
```

What Happens When Sticky Learning Is Enabled

- When enabled, **dynamically-learned secure MAC addresses** will be **added to the running config**.
- So, if you look in the running config, you'll see a command like this **automatically added**:

```
SW1(config-if)# switchport port-security mac-address sticky <learned MAC address>
```

- These **sticky secure MAC addresses** will **never age out**, even if you enable **static secure MAC address aging**.

Important Note — Saving the Configuration

- **However**, because they are added to the **running-config**, not the **startup-config**, you'll need to **save the running-config to the startup-config** to make them truly permanent.
  - For example, with the `write memory` command.

```
SW1# write memory
```

- If you don't do that, they will be **lost** if the switch **restarts**, or is **turned off and then on again**.

Conversion Behavior — Enabling Sticky Learning

- When you issue the `switchport port-security mac-address sticky` command:
  - All **current dynamically-learned secure MAC addresses** will be **converted to sticky secure MAC addresses**.
  - So, they will be **added to the running config**.

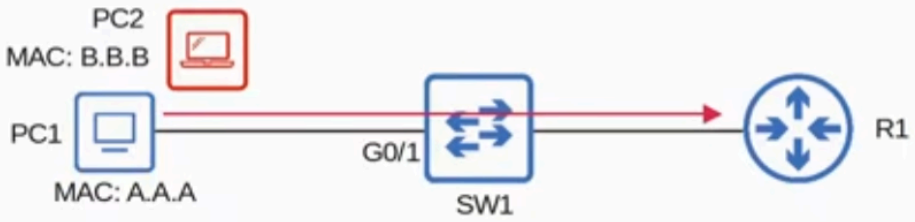
Conversion Behavior — Removing Sticky Learning

- The **opposite is true**, too.
- If you **remove sticky learning**, **sticky secure MAC addresses** will be **converted to regular dynamic secure MAC addresses**.

CLI Demonstration of Sticky Secure MAC Addresses

- The professor demonstrates sticky secure MAC address learning in the CLI.

```
SW1(config-if)#switchport port-security
SW1(config-if)#switchport port-security mac-address sticky
SW1(config-if)#do show running-config interface g0/1
!
interface GigabitEthernet0/1
 switchport mode access
 switchport port-security mac-address sticky
 switchport port-security mac-address sticky 000a.000a.000a
 switchport port-security
 negotiation auto
```



Example: Configuring and Verifying Sticky MAC Addresses

Step 1 — Enable Port Security

- As always, the professor enables port security first.

```
SW1(config)# interface gigabitethernet0/1
SW1(config-if)# switchport port-security
```

Step 2 — Enable Sticky MAC Address Learning

- Then he issues the following command:

```
SW1(config-if)# switchport port-security mac-address sticky
```

Step 3 — Send Traffic from PC1

- The professor sent a ping from **PC1 to R1**.

Step 4 — Check the Running-Config

- He then checked the **G0/1 interface** in the running-config.
- As you can see, an **extra command** was added:

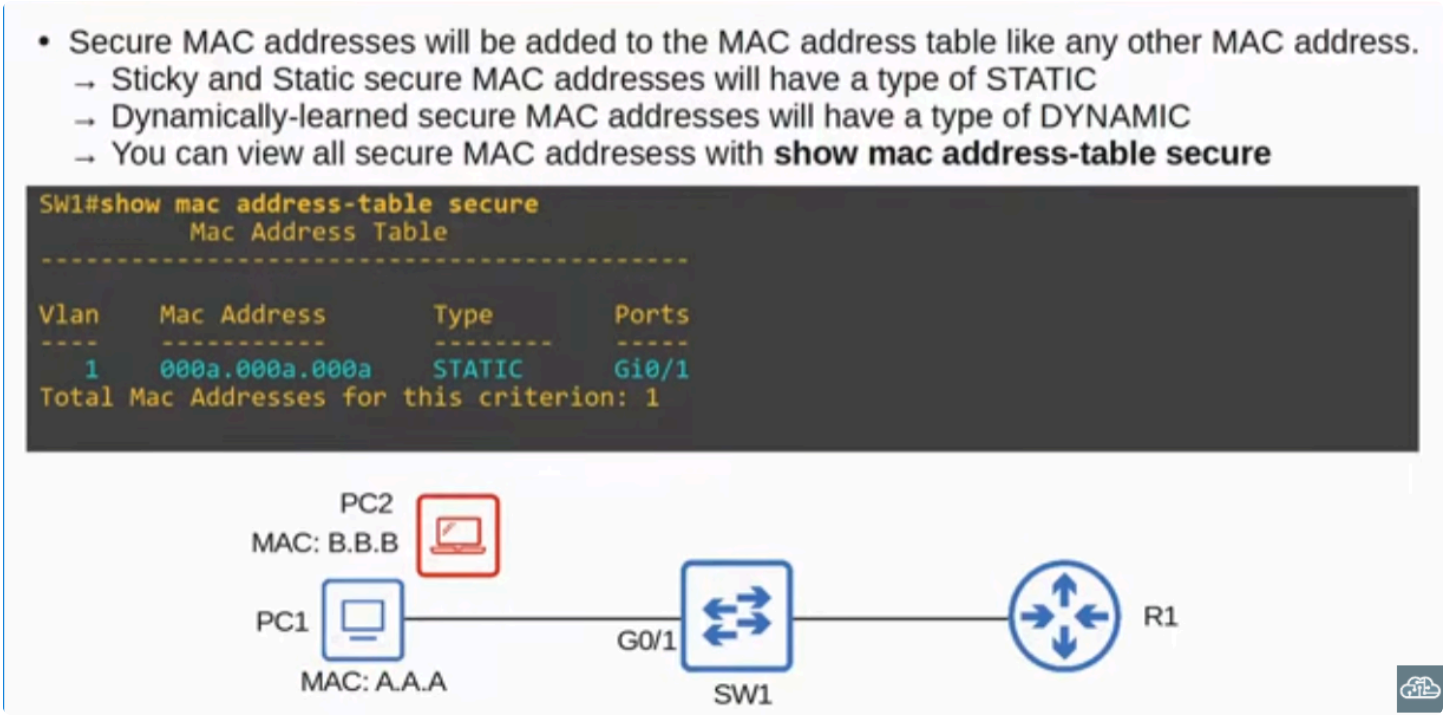
```
SW1(config-if)# switchport port-security mac-address sticky <PC1's MAC address>
```

Note: The professor did not configure that command — it was **added automatically** when PC1's frame entered the interface.

Summary

- **Sticky MAC addresses** are basically a way of **configuring static secure MAC addresses**, without actually having to **manually configure them**.

The MAC Address Table



- Before moving on to review and the quiz, the professor briefly mentions the **MAC address table**.
- **Secure MAC addresses** will be added to the MAC address table **like any other MAC address**.
- **Sticky and static secure MAC addresses** will have a type of **static**.
- **Regular dynamically-learned secure MAC addresses** that aren't sticky will have a type of **dynamic**.
- You can view all secure MAC addresses in the table with the following command:

Verification:

```
SW1# show mac address-table secure
```

Example: Viewing PC1's Sticky MAC Address

- The professor used the `show mac address-table secure` command.

- The output shows **PC1's MAC address** from the previous sticky MAC address example.
- **Notice:** The **type is static**, even though the professor didn't actually statically configure it himself.
    - It was **dynamically learned**.

Summary of Commands

- The professor provides a **summary of the commands** covered in the video.
- **Lots of new commands!**
- You'll definitely want to **experiment with these commands in a lab**.
    - Follow the packet tracer lab.
    - Also try making your own.
- If you don't remember any of these commands, **go back in the video to review**.

Review of What We Learned

- The professor reviews what was learned before moving on to the quiz.

Port Security's Basic Function

- First, the professor gave an **intro to port security** and explained its **basic function**.
- Basically, it allows you to control:
    - **What source MAC addresses** are allowed to enter a switch interface.
    - **How many source MAC addresses** are allowed to enter a switch interface.

Why We Should Use Port Security

- The professor briefly explained **why we should use port security**.
- First of all, it allows us to **prevent unauthorized devices** from accessing the network.

- Secondly, it helps **defend against attacks** such as the **DHCP exhaustion attack** shown in a previous video.
    - In a DHCP exhaustion attack, **thousands of spoofed MAC addresses** are used to send **DHCP discover messages**.

Configuration

- Then, while explaining various aspects of port security, the professor also showed **how to configure it**.
