1. Portsecurity

Portsecurity is a feature that can limit the MAC adresses that are allowed on a physical port. A violation of the port security occurs if

— the number if MAC adresses on a port is reached an the MAC id is different from the allowed list

— an allowed MAC id is connected to a differnt port in the same VLAN

Port security is seen as a deterrent for people that want to connect rogue devices to your network.

You can allow a single MAC to connect, or allow multiple MAC IDs. The maximum number of MAC IDs depends on the switch model and IOS version.

There are 3 modes for determining the allowed MAC IDs:

— Dynamic: The switch automatically learns the MAC addresses of devices that connect to the port. The switch automatically learns the MAC addresses of devices that connect to the port. When the switch restarts or the port goes down, these dynamically learned addresses are lost.

— Static or fixed: You manually configure the allowed MAC addresses on the switchport using commands. These statically configured addresses are stored in the switch's configuration file. They remain even after a switch restart.

— The switch dynamically learns MAC addresses, like in the dynamic method. Then, it "sticks" those learned addresses, adding them to the running configuration. If you save the running configuration to the startup configuration, these sticky addresses will persist across switch restarts.

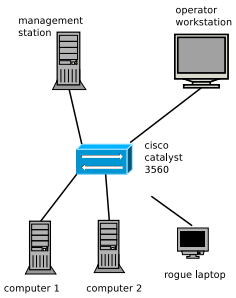
Port security depends on MAC-IDs. It is therefore vulnerable for MAC ID spoofing, where you copy the MAC ID from an allowed device to your own adapter. On Linux, this is trivial to do.

GNS3, which I use for most simulations, does not provide a switch that supports portsecurity. Therefore, this is not a simulated lab, but a real switch was used.

2. The network

2.1 The switch

Because GNS3 does not support port security, the test needs to be done with physical hardware.



The switch is a Cisco Catalyst 3560 with IOS 12.2, because that is what I got. most cisco switches will support port security, so it doesn't really matter. The switch has a DHCP server and all ports are assigned to VLAN 10. Syslog is redirected to the management server

The most relevant parts of the general config are:

```
ip dhcp excluded-address 192.168.10.1 192.168.10.10
ip dhcp excluded-address 192.168.10.51 192.168.10.255
!
ip dhcp pool CLIENT
   network 192.168.10.0 255.255.255.0
!
spanning-tree mode pvst
spanning-tree portfast default
spanning-tree extend system-id
vlan 10
!
interface Vlan10
 ip address 192.168.10.1 255.255.255.0
!
logging 192.168.10.2
```

2.2 The management server

The management server is a raspberry pi. Besides the reception of syslog, it is used to launch scripts that automate the tedious cisco command line use.

The pi uses rsyslogd. Standard, it does not receive syslog from other servers. In `/etc/rsyslog.conf` you need to add:

```
$ModLoad imudp
$UDPServerRun 514

# provides TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 514
```

In the catch-all section, I added `local7` to `/var/log/messages` because cisco logs on local7:

```
*.=info;*.=notice;*.=warn;\
        auth,authpriv.none;\
        cron,daemon.none;\
        local7.*;\
        mail,news.none          -/var/log/messages
```

2.3 The rest

For computer1 and computer2, you can use anything that has an ethernet NIC. The admin workstation needs to be able to ssh into the pi.

3. Port security

3.1 Enabling portsecurity

In our switch, we set all ports on VLAN 10. For most interfaces, we set:

```
interface FastEthernet0/1
 switchport access vlan 10
 switchport mode access
 switchport port-security
 switchport port-security mac-address sticky
```

For most interfaces; not for all:

— I noticed some strange behaviour when the Raspberry Pi boots. It triggers a port security violation, but only sometimes.

— I do not have a console cable, so I left port 23 and port 24 without port security to prevent locking myself out.

Let's examine the configuration.

```
switchport access vlan 10
```

Port security in this older versions can onl be applied on access ports.

```
 switchport port-security
 switchport port-security mac-address sticky
```

This defines the port security. Here, we have chosen to use sticky. You have the folowing options:

type	command	explanation
fixed	switchport port-security mac-address xxxx.yyyy.zzzz	Puts a fixed MAC ID on the port
sticky	switchport port-security mac-address sticky	Allows MAC ID to be dynamically learned and stores the MAC ID in the config
dynamic	(no additional command)	Allows MAC ID to be dynamically learned but does not store the MAC ID in the config

In general, sticky will be the preferred option. If you have a very good administration of devices, MAC IDs and cabling, fixed could be an option. Dyanamic is the default.

There are some additional options, for which I have chosen the default.

```
switchport port-security maximum 1
```

Defines the maximum number of MAC IDs allowed on the port. The default is 1.

```
switchport port-security violation {protect | restrict | shutdown}
```

Default is shutdown; this means that if a port has a vioaltion, it will be closed. Options are:

protect	Drops packets with unknown source addresses until you remove a sufficient number of secure MAC addresses to drop below the maximum value.
restrict	Drops packets with unknown source addresses until you remove a sufficient number of secure MAC addresses to drop below the maximum value and causes the SecurityViolation counter to increment.
shutdown	Puts the interface into the error-disabled state immediately and sends an SNMP trap notification.

You can also put some aging parameters on the port security. When the aging type is configured with the absolute keyword, all the dynamically learned secure addresses age out when the aging time expires. When the aging type is configured with the inactivity keyword, the aging time defines the period of inactivity after which all the dynamically learned secure addresses age out. I have not tried this.

3.2 And testing it...

I have tested two scenario's with mac address sticky.

3.2.1 Abusing the port and restoring the original situation

In this scenario, we removed one of the fixed computers and attached the rogue laptop. If the laptop is blocked, we put back the original computer. This is our original situation (an extract from `sh run` for the two interfaces):

```
interface FastEthernet0/1
 switchport access vlan 10
 switchport mode access
 switchport port-security
 switchport port-security mac-address sticky 0080.6494.adba
interface FastEthernet0/2
 switchport access vlan 10
 switchport mode access
 switchport port-security
 switchport port-security mac-address sticky 0080.6494.b208
```

The two MAC adresses are dynamically learned.

So now we put our rogue laptop on the switch. The port will be blocked and in our syslog we see:

```
Oct  3 16:10:09 192.168.10.1 311: 08:51:11: %PM-4-ERR_DISABLE: psecure-violation error detected on Fa0/1, putting Fa0/1 in err-disable state
Oct  3 16:10:10 192.168.10.1 312: 08:51:11: %PORT_SECURITY-2-PSECURE_VIOLATION: Security violation occurred, caused by MAC address 847b.eb43.c06c
```

Puting back the cable in the original computer will still block any trafic; the port is still shut. To get things working again, we have to issue:

```
config t
int fa0/1
shut
no shut
end
```

Which will result in the following syslog:

```
Oct  3 13:11:41 192.168.10.1 182: 05:52:42: %SYS-5-CONFIG_I: Configured from console by cisco on vty0 (192.168.10.13)
Oct  3 13:12:31 192.168.10.1 183: 05:53:31: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to down
Oct  3 13:12:32 192.168.10.1 184: 05:53:32: %LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to down
Oct  3 13:12:36 192.168.10.1 185: 05:53:37: %LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to up
Oct  3 13:12:36 192.168.10.1 186: 05:53:38: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to up
```

Now the port is back up after some manual intervention.

3.2.2 Put a new device in

This time, we do not put in a rogue laptop, but a replacement device. Attaching the device will ofcourse lead to a port security violation:

```
Oct  3 23:49:53 192.168.10.1 678: 3d01h: %PM-4-ERR_DISABLE: psecure-violation error detected on Fa0/2, putting Fa0/2 in err-disable state
Oct  3 23:49:54 192.168.10.1 679: 3d01h: %PORT_SECURITY-2-PSECURE_VIOLATION: Security violation occurred, caused by MAC address 847b.eb43.c06d or
```

This time, we not only need to reset the port, but will also need to remove the sticky MAC ID to allow a new one to be learned:

```
config t
int fa0/2
shut
no switchport port-security mac-address sticky 0080.6494.b208
no shut
end
```

And, as expected, the new MAC ID is learned and the port comes back up.

3.3 Some problems I encountered

If the MAC ID of computer_1 is allowed on Fa0/1, connecting that computer to a new port (for example Fa0/5) will cause a port security violation. In other words: if your computer connects to a specific port, you cannot just connect it to another port. This means that you need to keep an administration of which port has which MAC ID.

For some inexplicable reason, the portsecurity sticky did not learn the MAC ID correctly. A reload of the switch did nothing to improve the situation. I had to remove all cables, remove all MAC IDs and disable/re-enable port security to get everything working again.

4. Scripts

4.1 Intro

When I have to do things twice, I get bored. When I have to do it three times, I create a script to automate it.

My switch still uses telnet (yes I know) and I used expect to interface with my switch.

The following template will be used.

```
#!/usr/bin/expect

set timeout 60
set ip 192.168.10.1
set user cisco
set password secret

spawn telnet $ip
expect "ername:"
send "$user\r"
expect "assword: "
send "$password\r"
expect "DEMO"
send "enable\r"
expect "assword: "
send "$password\r"



expect "DEMO"
send -- "exit\r"
```

Because I have a test of short duration and becaise I am on a stand-alone network without connection to the rest of the world, I can put the password in my script. In a real life situation that would be unthinkable.

the DEMO is a part of the prompt for the switch. If you are very precise, you could put in the exact prompt that will be presented (`DEMO>` or `DEMO#` in enable mode, and so on for the different config modes. That will make the scripts better, but this will do for our test.

4.2 Resetting scripts with expect

If a port needs to be reset, you can log in to the switch, do `enable` and `conf t` and type all the IOS commands by hand. Or you could use an expect script.

```
#!/usr/bin/expect
set port [lindex $argv 0]
set mac [lindex $argv 1]

set timeout 60
set ip 192.168.10.1
set user cisco
set password secret

spawn telnet $ip
expect "ername:"
send "$user\r"
expect "assword: "
send "$password\r"
expect "DEMO>"
send "enable\r"
expect "assword: "
send "$password\r"

expect "DEMO"
send "conf t\r"
expect "DEMO"
send "int fa0/$port\r"
expect "DEMO"
send "shut\r"
expect "DEMO"
send "end\r"


expect "DEMO"
send "conf t\r"
expect "DEMO"
send "int fa0/$port\r"
expect "DEMO"
send "no shut\r"
expect "DEMO"
send "end\r"


expect "DEMO"
send -- "exit\r"
```

It should not be dificult to create scripts for the main transactions. I created expect scripts for the following.

script	purpose	commands
down_port.xpct	make a port admin down	conf t int fa0/port shut end
remove_mac_on_port_sticky.exp	remove a sticky mac from a port	conf t int fa0/port shut no switchport port-security mac-address sticky mac no shut
reset_port.xpct	reset a port	conf t int fa0/port shut no shut end
show_port.xpct	show settings	sh run interface fa0/port sh port-security interface fa0/port

4.3 A demo menu

This test was part of a demonstration of how port security works. I used the following script to hide the nitty gritty details from the audience.

```
#!/bin/bash
TMP=/tmp/rgh.$$
sudo tail -f /var/log/messages | grep PSECURE_VIOLATION &
tokill=$!
```

The syslog messages about port security violations are written over the menu. This shows that we actually see an alert when something is happening.

```
disp(){
          printf "%4.4s  %16.16s  %16.16s  %12.12s\n" port allowed last_seen status
          for port in 1 2 3 4 5 6 7 8 9 10 11 12 ; do
                    expect show_port.xpct $port | sed 's/\r//' > $TMP
                    mac=`awk '/switchport port-security mac-address.*[0-9a-f]+\.[0-9a-f]/{print $NF}' $TMP`
                    lastseen=`sed -n 's/Last Source Address:Vlan *: *//p' $TMP | sed 's/:1*0//'`
                    status=`sed -n 's/Port Status *: *//p' $TMP`
                    nstat='unknown'
                    case a$status in
                    (aSecure-up)        nstat='UP' ;;
                    (aSecure-down)      nstat='No device' ;;
                    (aSecure-shutdown)
                                        if [ "$mac" = "" ] ; then
                                                  nstat='No device'
                                        else
                                                  nstat="Violation"
                                        fi
                                        ;;
                    esac
                    if grep '^ *shutdown' $TMP > /dev/null ; then
                              nstat='adm down'
                    fi
                    printf "%4.4s  %16.16s  %16.16s  %12.12s\n" "$port" "$mac" "$lastseen" "$nstat"
          done
          rm -f $TMP
}
```

The goal of this function is to provide a display of what is actually happening. It shows a table that is easier to interpret than the standard IOS output.

```
rmmac(){
          port=$1
          expect show_port.xpct $port | sed 's/\r//' > $TMP
          mac=`awk '/switchport port-security mac-address.*[0-9a-f]+\.[0-9a-f]/{print $NF}' $TMP`
          if [ "$mac" != "" ] ; then
                    expect remove_mac_on_port_sticky.exp $port $mac
          fi
}
```

This function removes a MAC ID from a port. The function first looks which MAC ID is available on the port.

```
reset(){
          port=$1
          expect reset_port.xpct $port
}

down(){
          port=$1
          expect down_port.xpct $port
}
```

These functions reset a port (shut, no shut) or just shut the port.

```
quit=no

while [ $quit = no ] ; do
          clear
          disp
          echo "d for shut down a port"
          echo "n for new MAC id"
          echo "r for reset port"
          echo "q for quit"
          echo "enter for refresh"
          read line
          case a$line in
          (ar)      echo -n "Port nummer : "
                    read line
                    reset $line
                    ;;
          (ad)      echo -n "Port nummer : "
                    read line
                    down $line
                    ;;
          (an)      echo -n "Port nummer : "
                    read line
                    rmmac $line
                    ;;
          (aq)      quit=yes
                    ;;
          esac
done

sudo pkill tail
exit
```

The main loop presents a simple menu.

CONTENTS