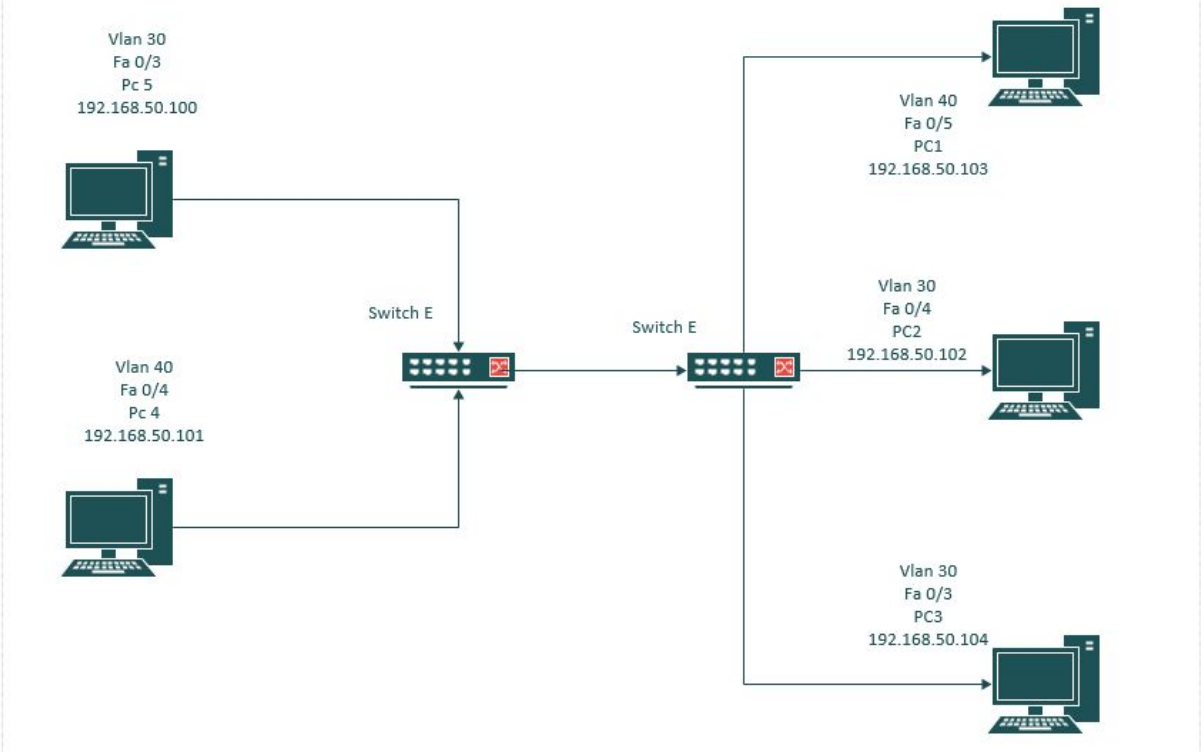
VLAN Hopping

Overview: Cisco's proprietary DTP [Dynamic Trunking Protocol] is a large security vulnerability. Trunk ports should be manually configured as best practice. Most modern cisco switches have DTP enabled by default. Because of this, DTP will always be on the lookout to automatically negotiate trunk links, which allows attackers to double encapsulate packets and capture a victims VLAN ARP traffic that they should not have access to. This attack is known as VLAN Hopping.

PART 1 : LAB Topology / Setup



Setting up & Pinging/Testing VLANS 30 & 40

VLAN 30 pings from PC5	**VLAN 40 Pings from PC 4**
Successful pings:	Successful Pings
192.168.50.102 [PC 2]	192.168.50.103 [PC 1]

192.168.50.104 [PC 3]	Unsuccessful Pings [Not In VLAN]
Unsuccessful Pings [Not In VLAN]:	192.168.50.102 [PC 2]
192.168.50.103 [PC 1]	192.168.50.104 [PC 3]

Switch 1 & Switch 2 VLAN Config

```
interface FastEthernet0/1
 switchport mode dynamic desirable
!
interface FastEthernet0/2
 switchport trunk encapsulation dot1q
 switchport mode trunk
!
interface FastEthernet0/3
 switchport access vlan 30
 switchport mode access
!
interface FastEthernet0/4
 switchport access vlan 30
 switchport mode access
!
interface FastEthernet0/5
 switchport access vlan 40
 switchport mode access
!
```

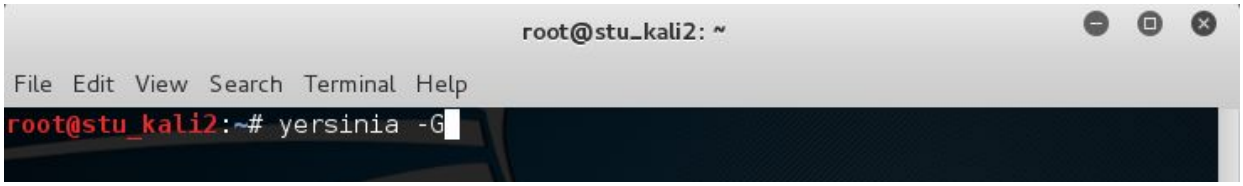
```
!
!
!
!
interface FastEthernet0/1
 switchport mode dynamic desirable
!
interface FastEthernet0/2
 switchport trunk encapsulation dot1q
 switchport mode trunk
!
interface FastEthernet0/3
 switchport access vlan 30
 switchport mode access
!
interface FastEthernet0/4
 switchport access vlan 40
 switchport mode access
!
```

Show Interfaces on Trunk

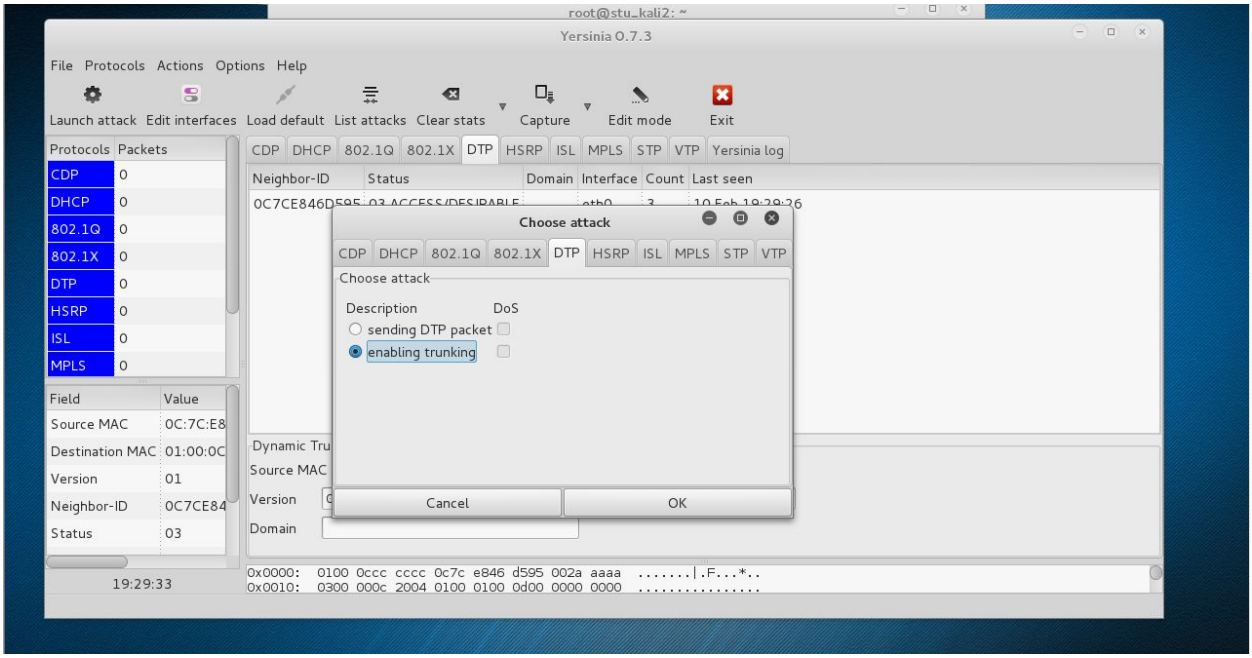
```
Switch#
Switch#
Switch#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/2       on               802.1q         trunking      1

Port        Vlans allowed on trunk
Fa0/2       1-4094
```

PART 2 : Launch Attack

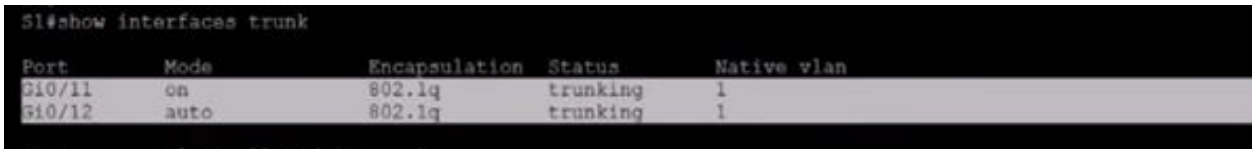
Step 1 : Open Kali **Yersinia** on Attack PC



Step 2: Inside the Yersinia GUI
Select Launch Attack → Choose **DTP** Tab → **Enable Trunking** → Select OK



Exploit Failure : What should have happened if the exploit was successful ?
Assuming that these were older switches in our labs, after running the exploit: the trunk interface would likely change to look like the [see image below], which shows double encapsulation. At this point, The attacker would be able to “hop” to the victim VLAN and capture ARP packets that should have stayed contained within only the victims VLAN.



Why Didn't the attack work ?
If the ports are set as trunks, they can be tagged before entering. However, If the ports are set as access ports, **they will always get dropped.** Cisco 3550 Switches have updated and past exploits like this VLAN hopping attack, no longer work with the strategy utilized in this lab.

Tagged vs. untagged Scapy packets
In the below example, I ping PC 5 with two scapy packet scripts. One packet includes a dot1q tag and the other does not.

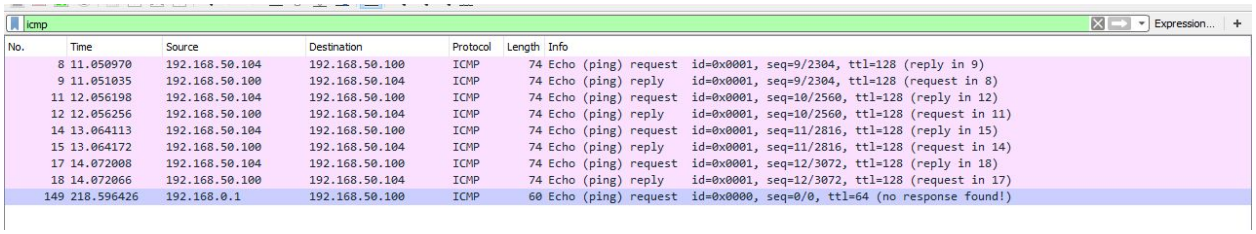
With dot1q tag

```
p1 = Ether(dst='ec:b1:d7:45:11:24', src='ec:b1:d7:40:d9:fb')/Dot1Q(vlan=30)/IP(dst='192.168.50.100', src='192.168.0.1')/ICMP()
```

Without dot1q tag

```
p1 = Ether(dst='ec:b1:d7:45:11:24', src='ec:b1:d7:40:d9:fb')/IP(dst='192.168.50.100', src='192.168.0.1')/ICMP()
```

Pings in Wireshark



icmp

No.	Time	Source	Destination	Protocol	Length	Info
8	11.050970	192.168.50.104	192.168.50.100	ICMP	74	Echo (ping) request id=0x0001, seq=9/2304, ttl=128 (reply in 9)
9	11.051035	192.168.50.100	192.168.50.104	ICMP	74	Echo (ping) reply id=0x0001, seq=9/2304, ttl=128 (request in 8)
11	12.056198	192.168.50.104	192.168.50.100	ICMP	74	Echo (ping) request id=0x0001, seq=10/2560, ttl=128 (reply in 12)
12	12.056256	192.168.50.100	192.168.50.104	ICMP	74	Echo (ping) reply id=0x0001, seq=10/2560, ttl=128 (request in 11)
14	13.064113	192.168.50.104	192.168.50.100	ICMP	74	Echo (ping) request id=0x0001, seq=11/2816, ttl=128 (reply in 15)
15	13.064172	192.168.50.100	192.168.50.104	ICMP	74	Echo (ping) reply id=0x0001, seq=11/2816, ttl=128 (request in 14)
17	14.072008	192.168.50.104	192.168.50.100	ICMP	74	Echo (ping) request id=0x0001, seq=12/3072, ttl=128 (reply in 18)
18	14.072066	192.168.50.100	192.168.50.104	ICMP	74	Echo (ping) reply id=0x0001, seq=12/3072, ttl=128 (request in 17)
149	218.596426	192.168.0.1	192.168.50.100	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)

Result: The packet script [without the tag] ICMP, is seen above in the wireshark capture. The other tagged example was never captured, because it was dropped by the switch.

PART 3 : Attack Mitigation

To secure the switch ports against the original attack that we tried to create:
The simplest mitigation technique is as follows..
Go into the ports that are not configured as trunk ports and make sure they are set as access ports. **[# switchport nonegotiate]**

switchport set to nonegotiate

```
!
interface FastEthernet0/3
 switchport access vlan 30
 switchport mode access
 switchport nonegotiate
!
```

What [# nonegotiate] command does ?

Conclusion: At this point, the nonegotiate command will make sure the port will not automatically negotiate trunk ports. The arp packets would not be available from the victim VLAN, which in turn successfully mitigates the VLAN hopping attack. For our specific lab

environment, this was not necessary, because the switch stopped the attack without the need for us to manually configure the ports to do so.