ICT2203/ICT2203X

Network Security

SINGAPORE INSTITUTE OF TECHNOLOGY

Lab 3 Notes:

More Attacks and Defense of LAN with Switches

2021-2022 Trimester 3

Lab Exercise 1

SINGAPORE INSTITUTE OF TECHNOLOGY

1.1–1.4

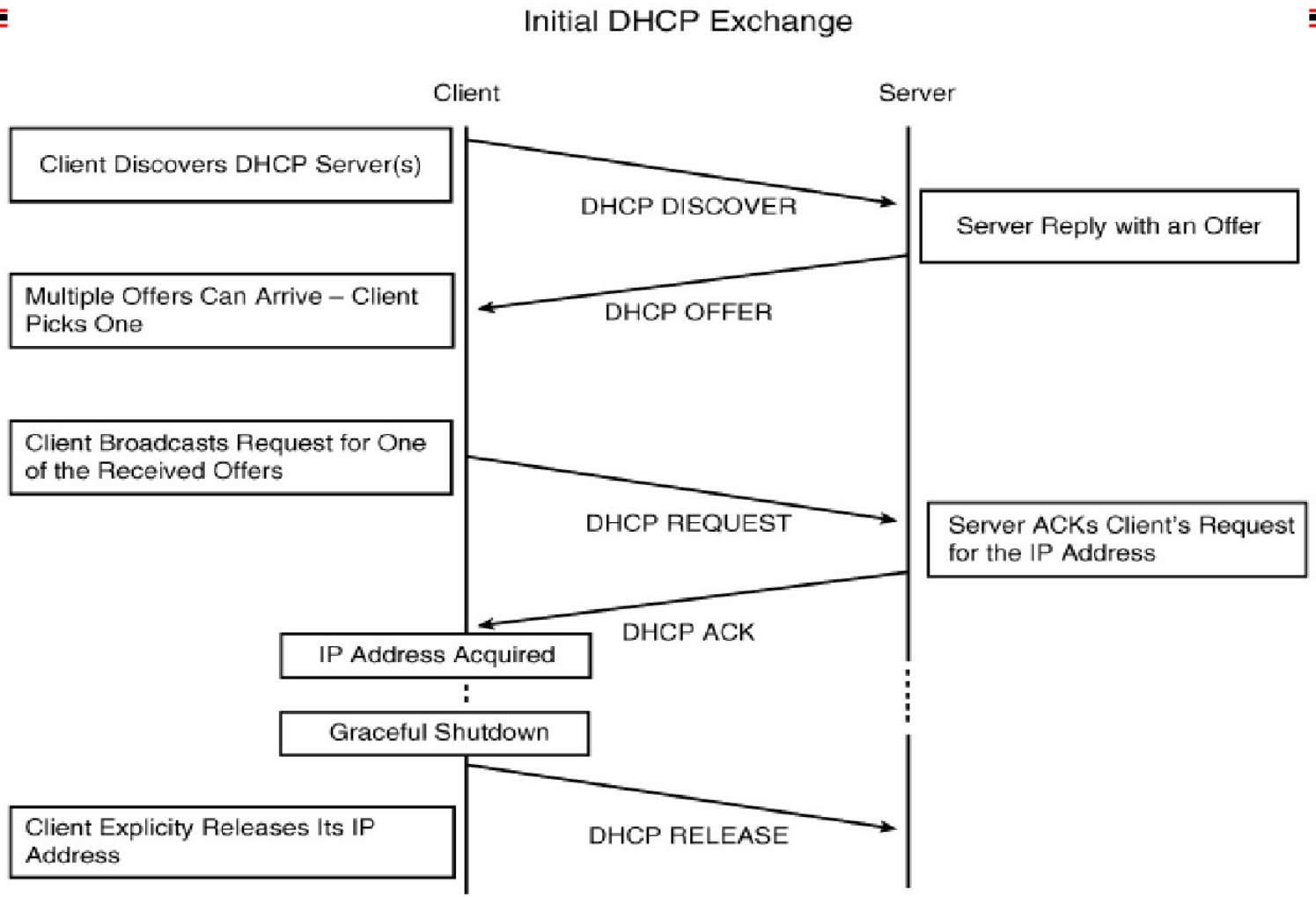
DHCP starvation and DHCP server spoofing attacks, which can lead to DNS server spoofing and website spoofing attacks

1.5

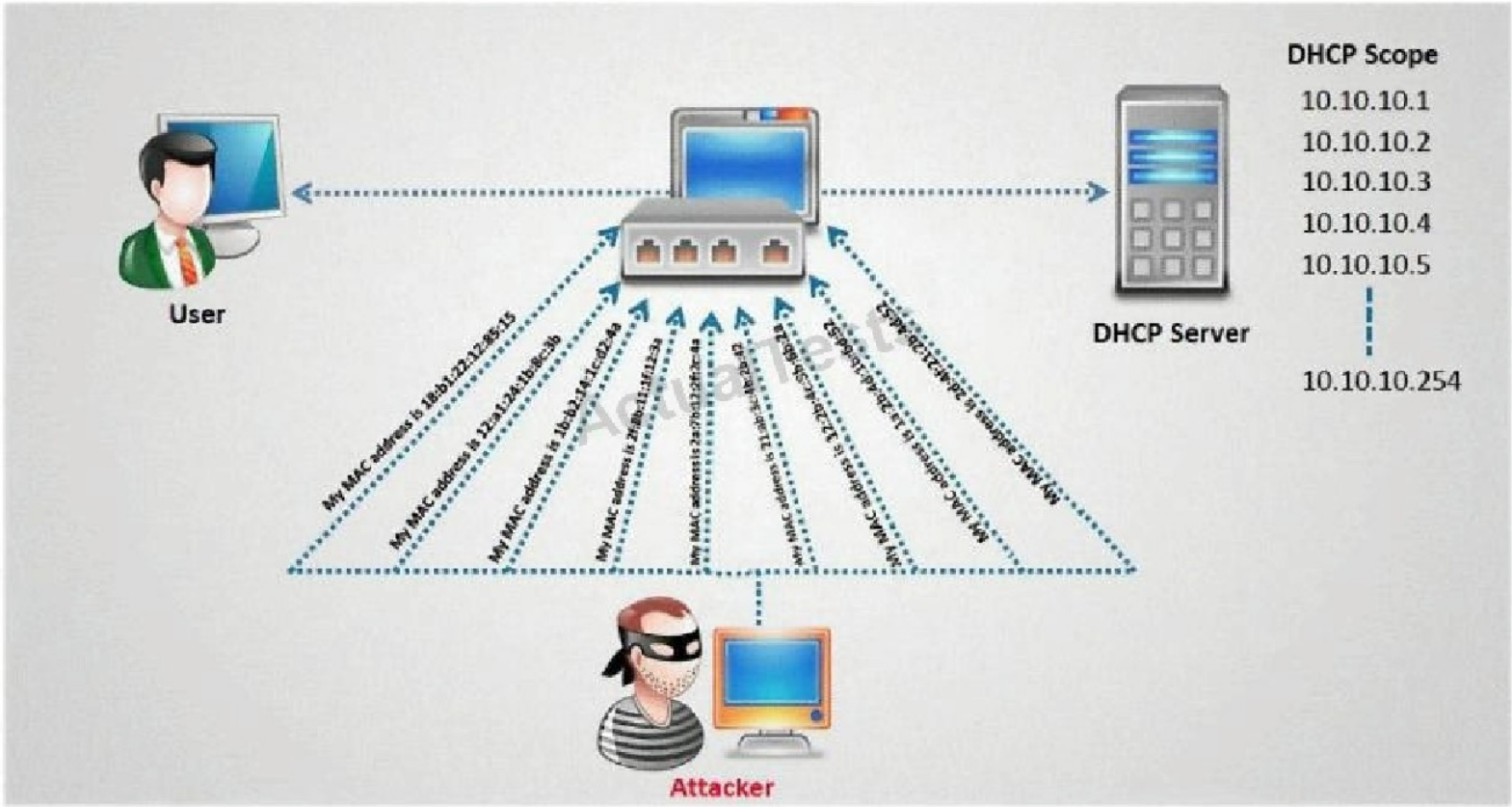
Defense:

- Configure DHCP snooping in switches

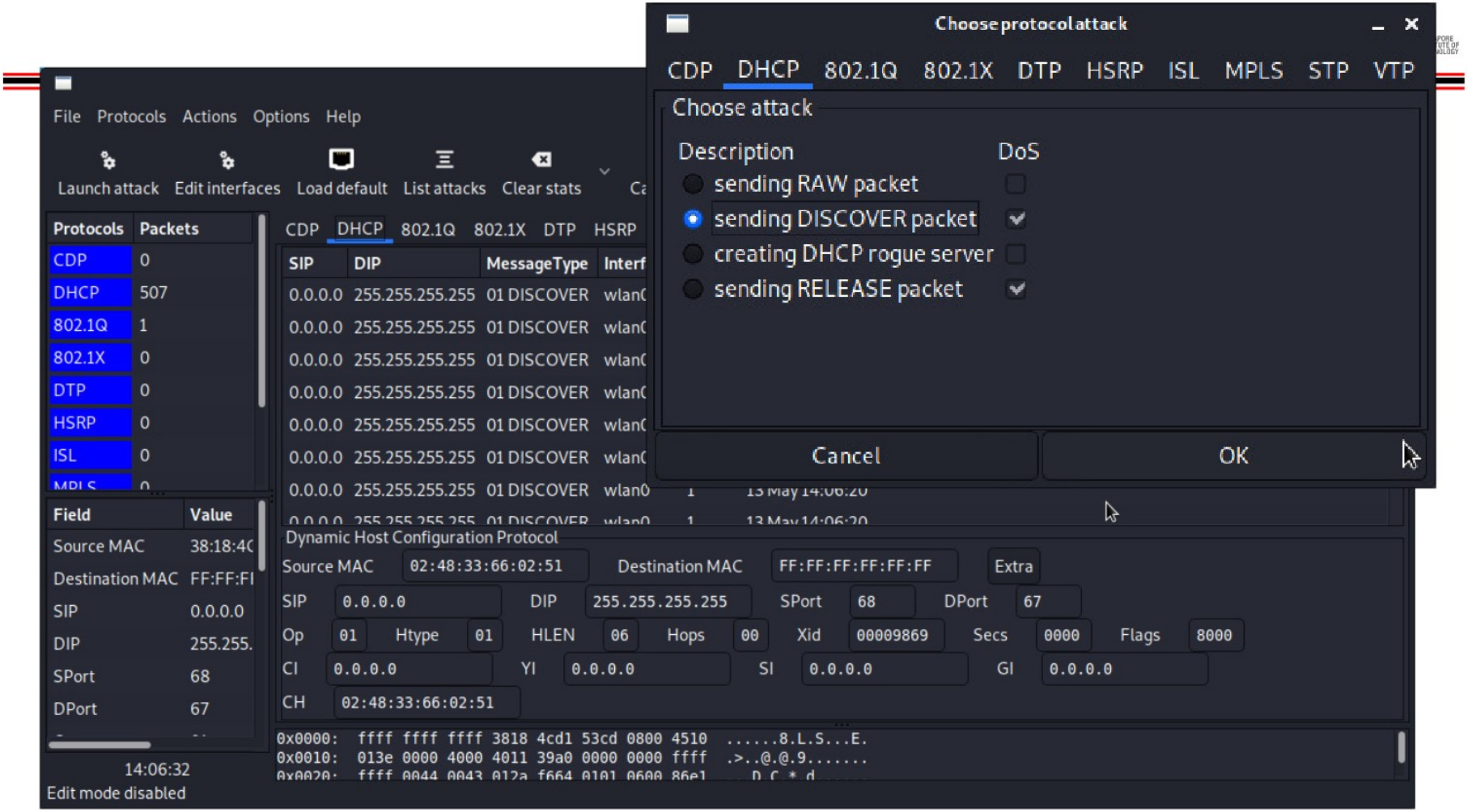
In ICT1010, you have learned DHCP (Dynamic Host Control Protocol) for hosts to automatically obtain IP address, default gateway, DNS, etc. in order to function.



Unfortunately, DHCP has no security! An obvious attack is starvation/exhaustion/DoS attack where an attacker generates many DHCP Discover messages to seize all IP addresses.

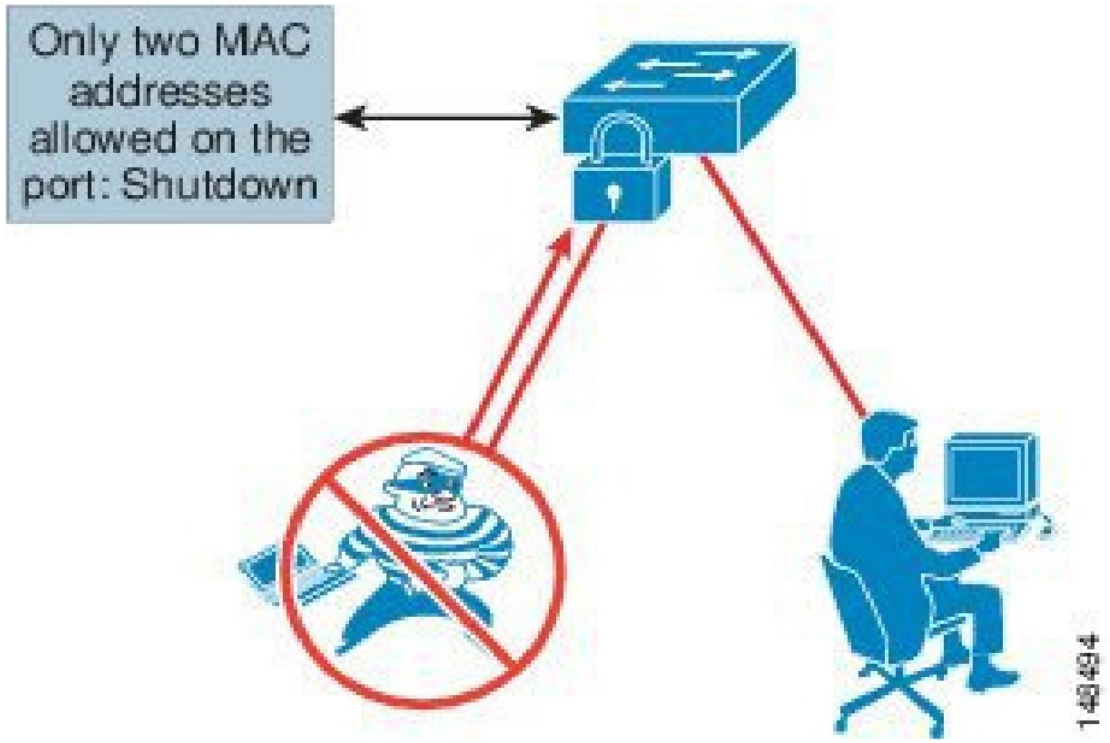


Again, Yersinia can be used to conduct DHCP starvation attack.



Do you think Port Security is an effective defense to counter DHCP starvation attack?

Port Security limits the MAC addresses allowed on a port, which can prevent an attacker from sending many DHCP Discover messages with different MAC addresses.



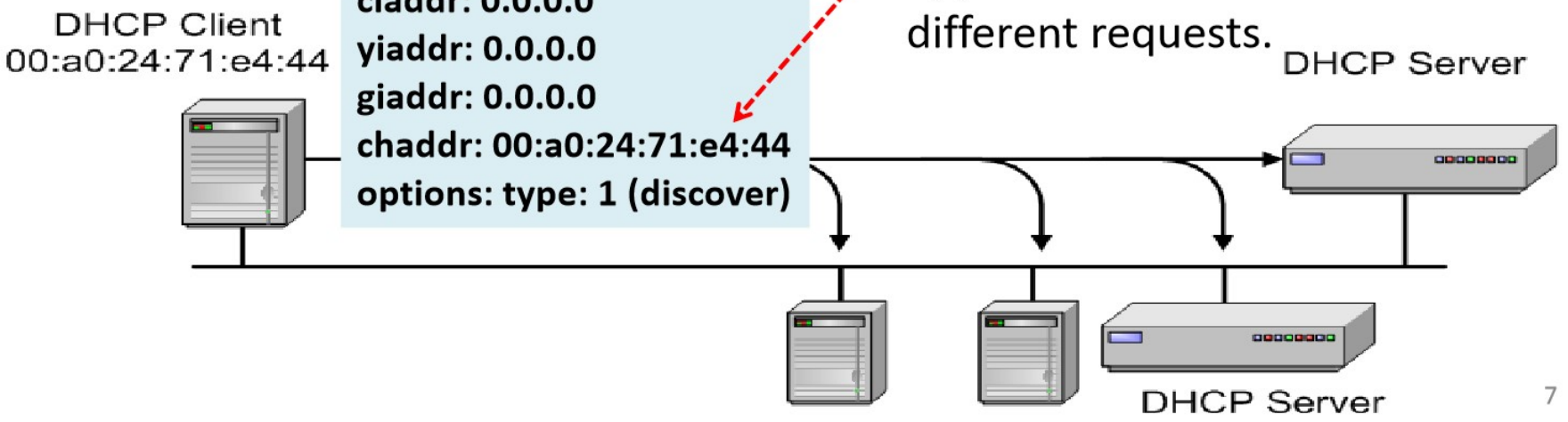
Unfortunately, an advanced attacker can still generate many DHCPDiscover messages without violating Port Security as follow: (This shows why you need to know details!)

Remember DHCP is at application layer, i.e. the lower layers are removed by the OS before DHCP server receives the final DHCP message.

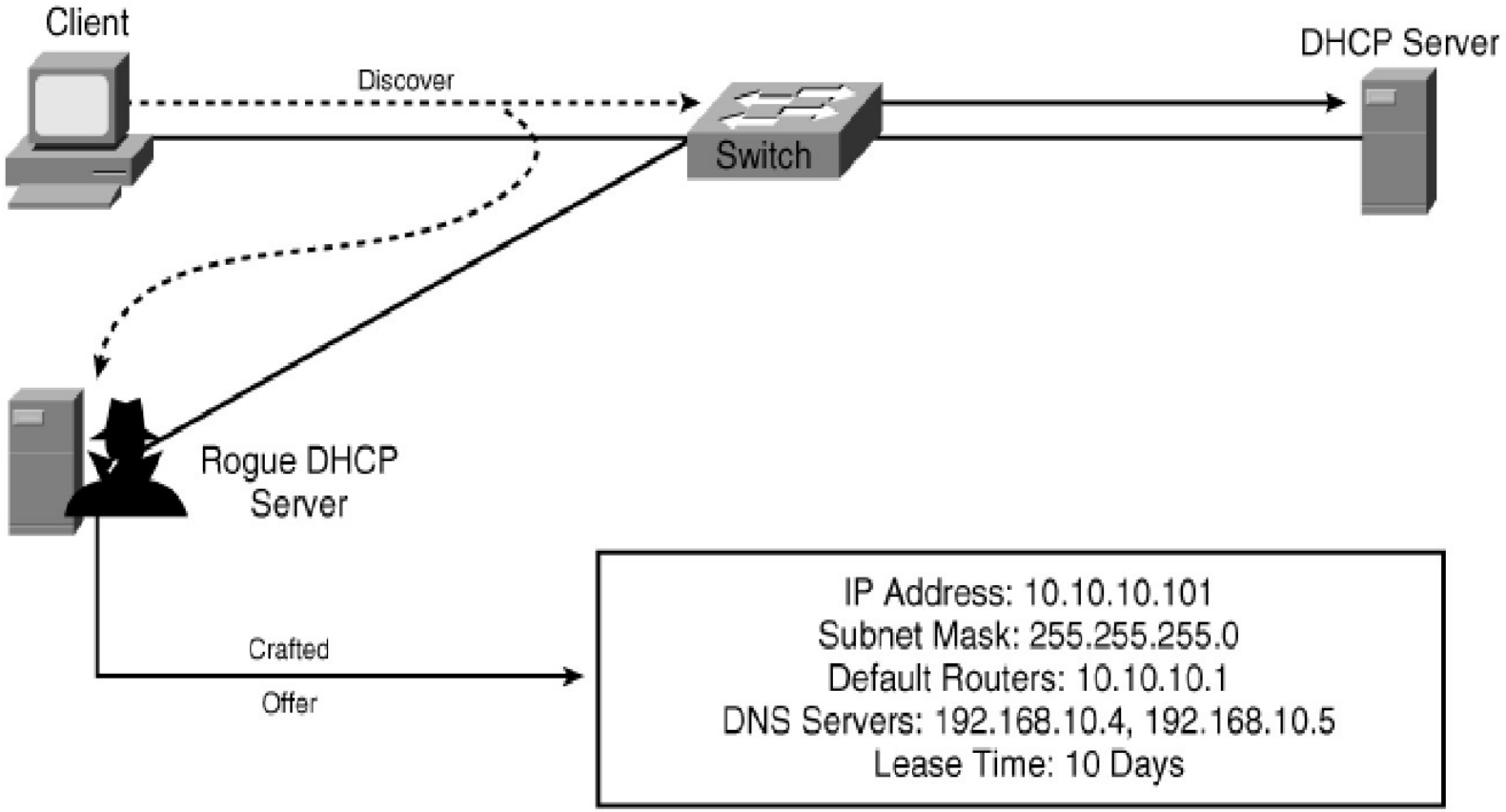
MAC	**src: 00:a0:24:71:e4:44** **dest: FF:FF:FF:FF:FF:FF**
IP/UDP	**src: 0.0.0.0:68** **dest: 255.255.255.255:67**
DHCP	**code: 1 (request)** **trans ID: 654** **ciaddr: 0.0.0.0** **yiaddr: 0.0.0.0** **giaddr: 0.0.0.0** **chaddr: 00:a0:24:71:e4:44** **options: type: 1 (discover)**

Fixed source MAC, no violation of Port Security.

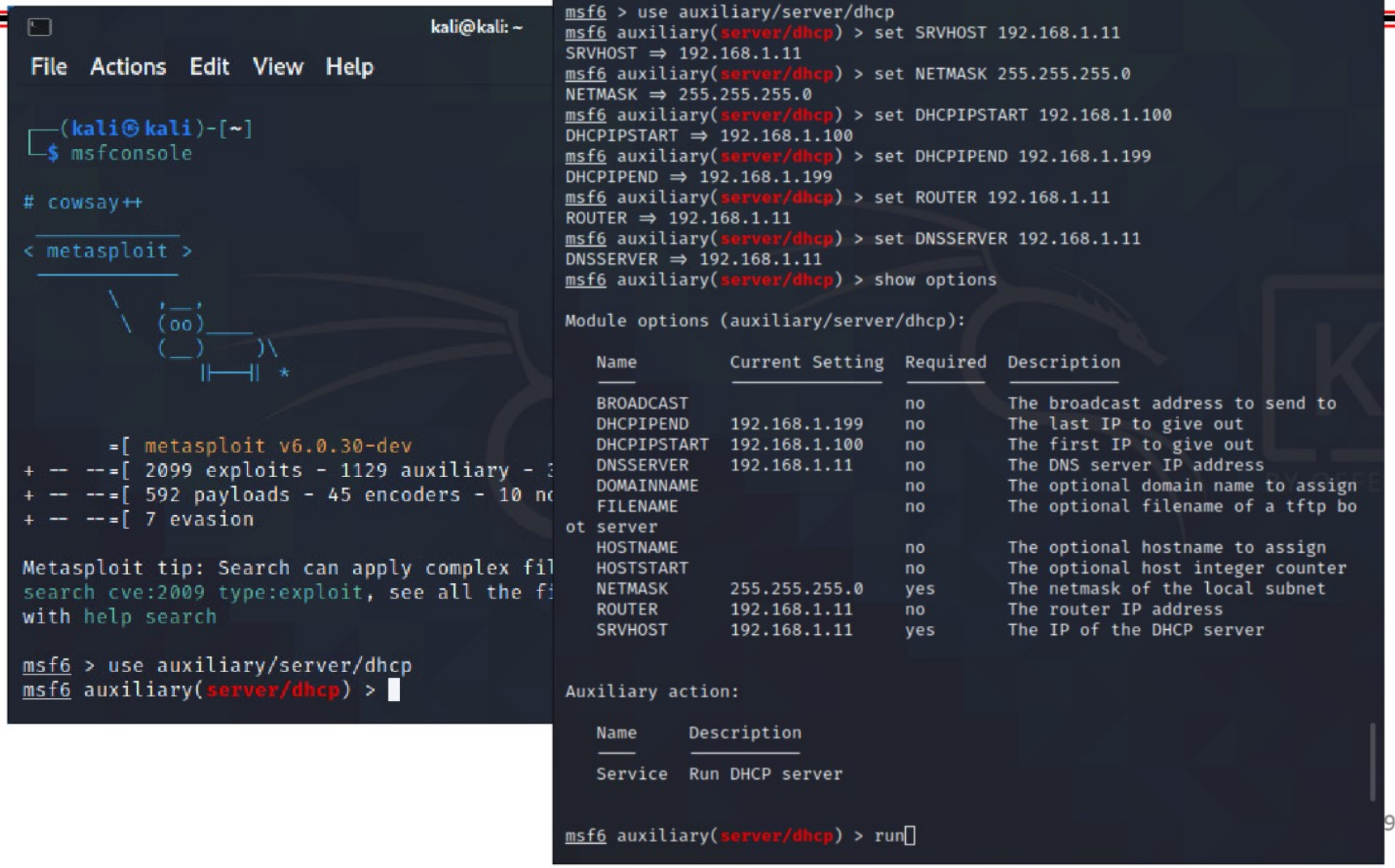
Random source MAC, which appears to DHCP server as different requests.



After DHCP starvation attack, an attacker may follow up with DHCP server spoofing attack which is to set up a rogue DHCP server to respond to users DHCP messages.

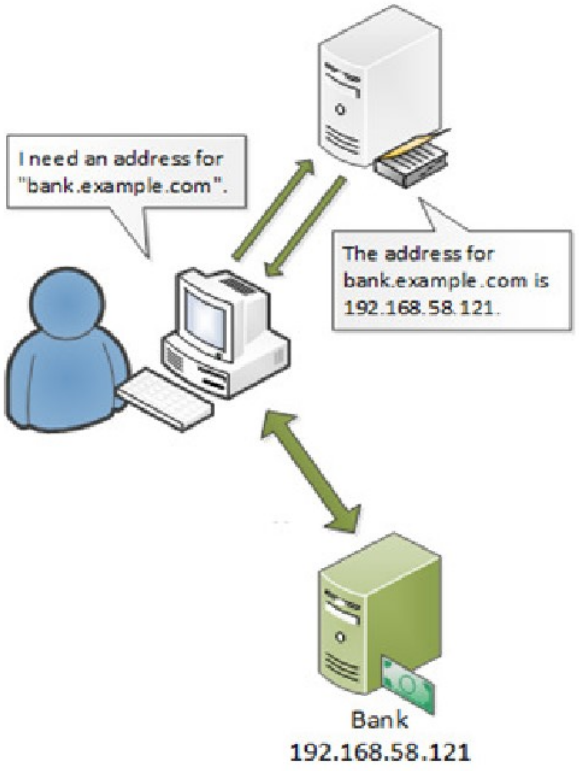


The Metasploit tool in Kali Linux has a DHCP module which can be used to run a rogue DHCP server.

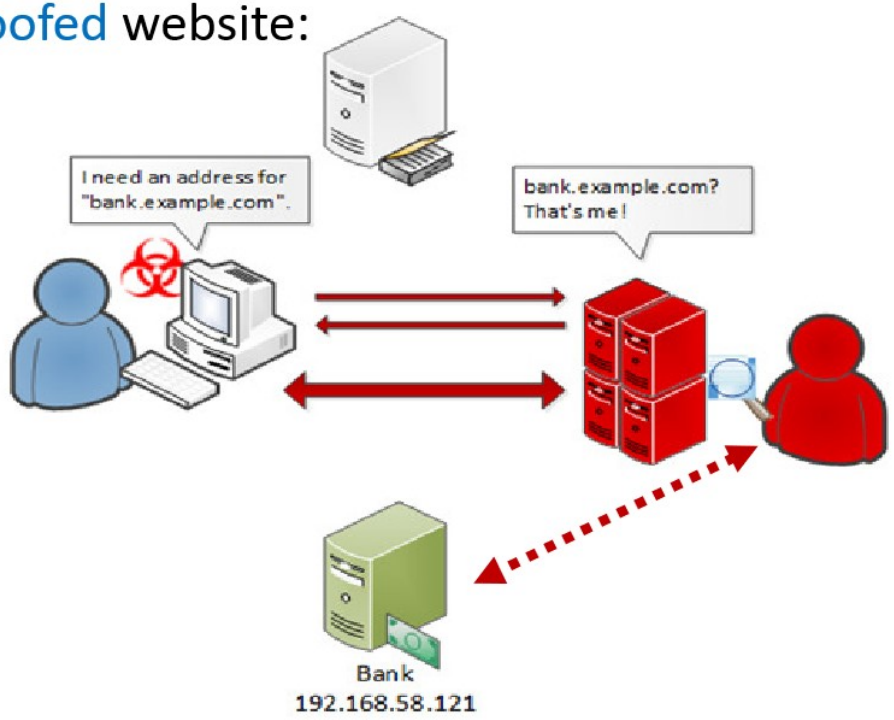


With DHCP server spoofing attack, an attacker can now easily direct users to rogue DNS server to mis-translate domain name to IP address to lead them to the spoofed website!

Before attack:



After DHCP server spoofing attack directing user to rogue DNS server and then spoofed website:

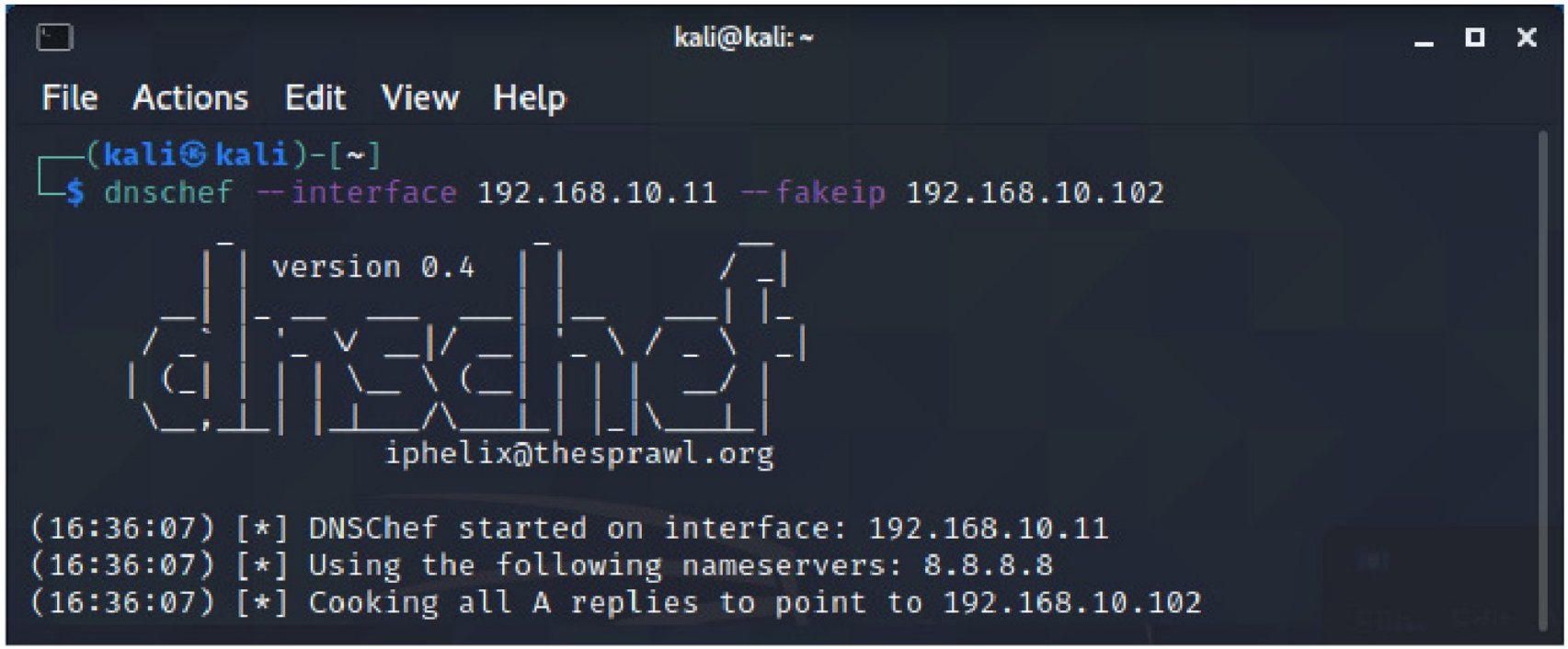


An attacker may then use the information obtained to masquerade as the authorized user to login to the real website!

The DNSChef tool in Kali Linux can be used to run the rogue DNS server.

Refer to *https://www.kali.org/tools/dnschef* on dnschef usage

e.g.: To start DNS server to listen at interface 192.168.10.11 and mis-translating all DNS replies to attacker's fake IP address 192.168.10.102



To prevent DHCP starvation and spoofing attacks, a defence called DHCP snooping is developed which enables a switch to perform deep-packet inspections of higher layers.

Enable DHCP snooping in the switch and then specify the required VLANs as follows:

```
Switch(config)# ip dhcp snooping
Switch(config)# ip dhcp snooping vlan vlan-id [vlan-id]
```

Configure DHCP snooping MAC address verification to verify layer 2 source MAC matches application layer DHCP client hardware address (chaddr), and drop if different.

```
Switch(config)# ip dhcp snooping verify mac-address
```

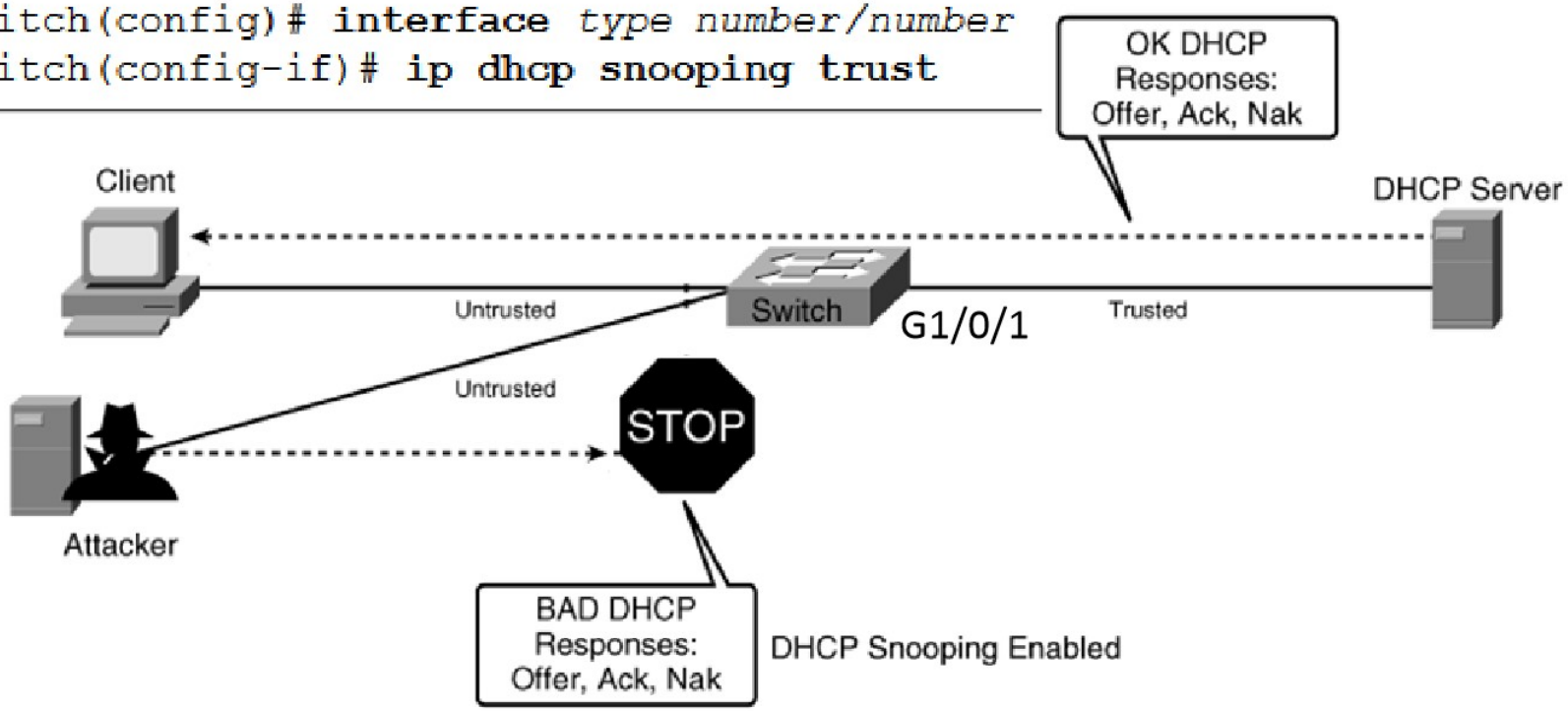
For each port, DHCP snooping rate-limiting (from 1 – 2048 DHCP pps) can also be configured. If exceeded, the port is put into err-disabled state.

```
Switch(config)# interface type number/number
Switch(config-if)# ip dhcp snooping limit rate rate
```

When DHCP snooping is enabled, all ports are by default untrusted which will not allow DHCP Offer and DHCP Ack messages, thus making it impossible to run rogue DHCP server.

To support DHCP servers, the ports connected to the authorized DHCP servers must be manually configured to be trusted as follows:

```
Switch(config)# interface type number/number
Switch(config-if)# ip dhcp snooping trust
```



An example of configuring DHCP snooping on a switch and verifying the status:

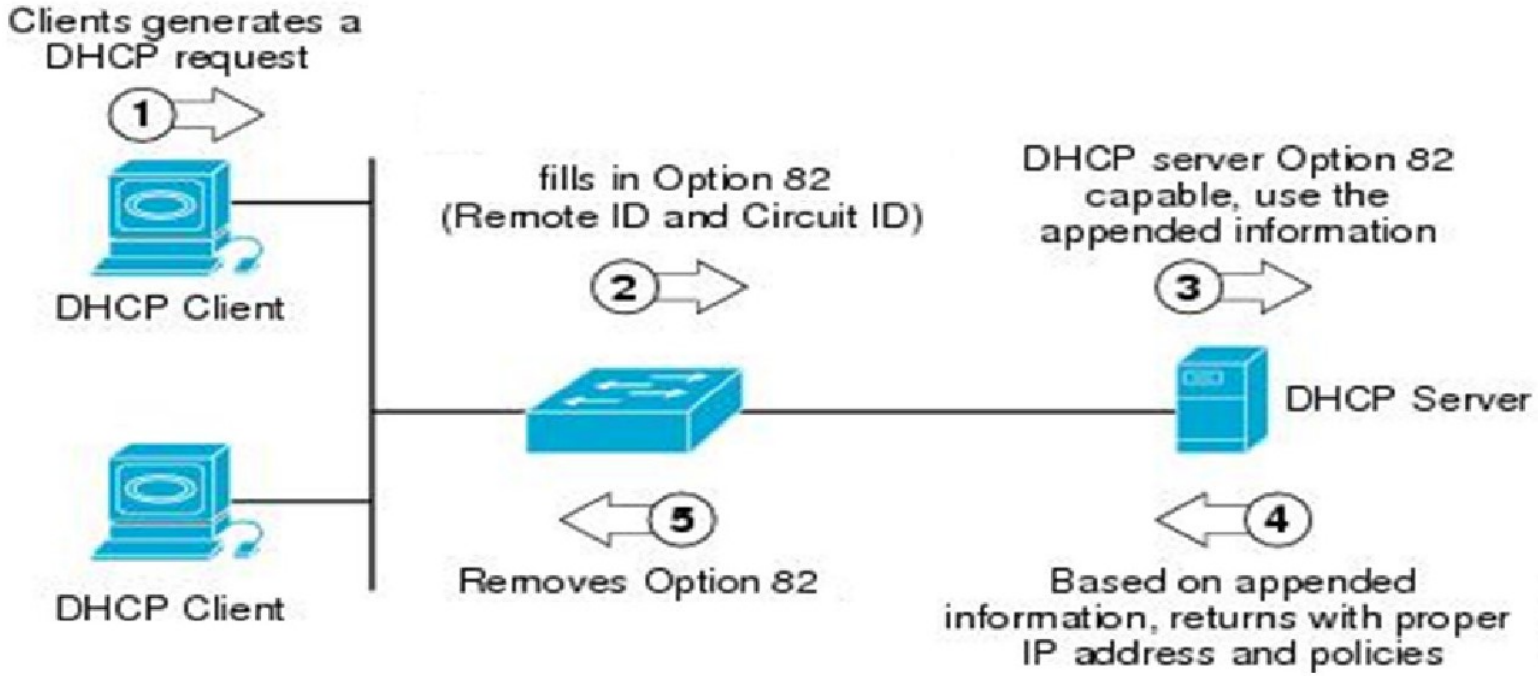
```
Switch(config)# ip dhcp snooping
Switch(config)# ip dhcp snooping vlan 10
Switch(config)# interface range gigabitethernet 1/0/35 - 36
Switch(config-if)# ip dhcp snooping limit rate 3
Switch(config-if)# interface gigabitethernet 1/0/1
Switch(config-if)# ip dhcp snooping trust
```

authentic DHCP server at interface G1/0/1

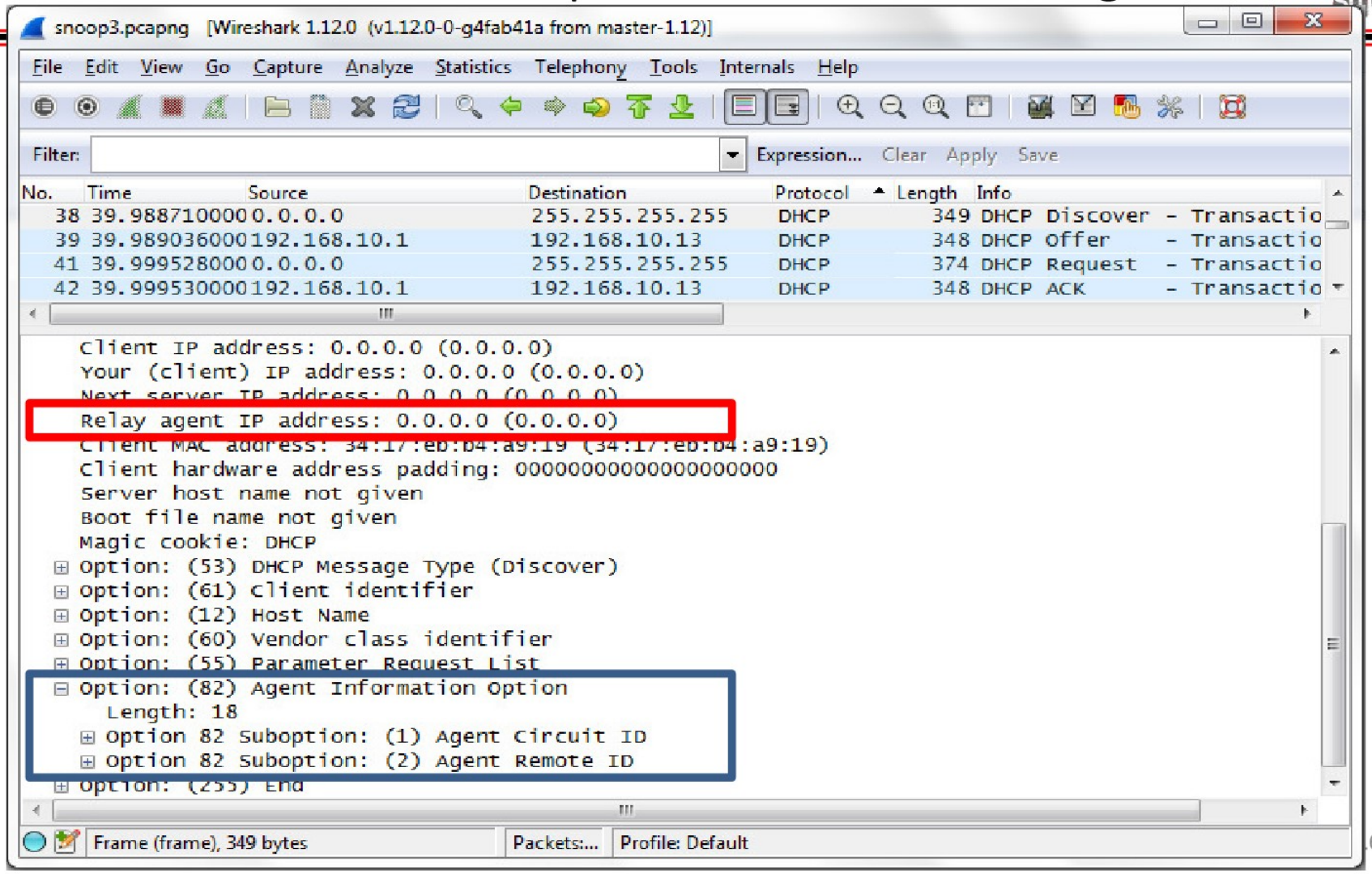
```
Switch# show ip dhcp snooping
Switch DHCP snooping is enabled
DHCP snooping is configured on following VLANs:
10
Insertion of option 82 is enabled
Interface                    Trusted     Rate limit (pps)
------------------------     -------     ----------------
GigabitEthernet1/0/35          no          3
GigabitEthernet1/0/36          no          3
GigabitEthernet1/0/1           yes         unlimited
Switch#
```

Note that by default, a switch enabled with DHCP snooping will automatically insert an option-82 (relay agent information option) field into the original DHCP messages sent by the client.

The option-82 field contains circuit ID (containing VLAN and port number) and remote ID (containing base MAC address) of the switch which provides information about location of client that sends the DHCPDiscover message.

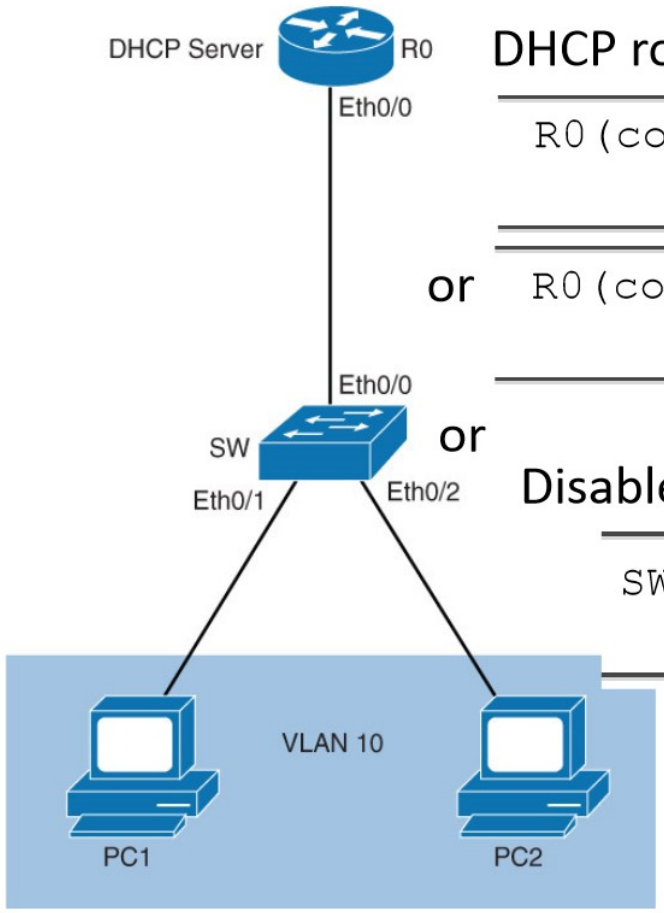


However, the switch is not relay agent, so the relay agent IP address (giaddr) is zeros while option-82 agent field is being inserted as shown in example DHCPDiscover message below:



Unfortunately, a DHCP packet with option-82 but relay agent IP address (giaddr) zeros will by default be discarded by the next receiving DHCP relay agent/server.

To resolve this problem, configure either:



DHCP router to trust DHCP messages:

```
R0(config)# ip dhcp relay information
            trust-all
```

or

```
R0(config-if)# ip dhcp relay information
               trusted
```

or

Disable option-82 on a DHCP snooping switch:

```
SW(config)# no ip dhcp snooping
            information option
```

To support DHCP snooping, the switch also maintains a binding table containing DHCP assigned addresses for untrusted ports which can be used to defend other attacks (next exercises).

The DHCP snooping binding table is updated automatically:

- Upon seeing DHCPACK, new entry is added into binding table.
- Upon seeing DHCPNack, DHCPRelease or DHCPDecline, corresponding entry is removed.

```
SW# show ip dhcp snooping binding
MacAddress          IpAddress        Lease(sec)  Type           VLAN  Interface
------------------  ---------------  ----------  -------------  ----  ----------
00:24:13:47:AF:C2   192.168.1.4      85858       dhcp-snooping   10   GigabitEthernet1/0/35
00:24:13:47:7D:B1   192.168.1.5      85859       dhcp-snooping   10   GigabitEthernet1/0/36
Total number of bindings: 2
```

Lab Exercise 2

SINGAPORE INSTITUTE OF TECHNOLOGY

2.1–2.3

ARP poisoning attack which can lead to MitM (Man-in-the-Middle) attack

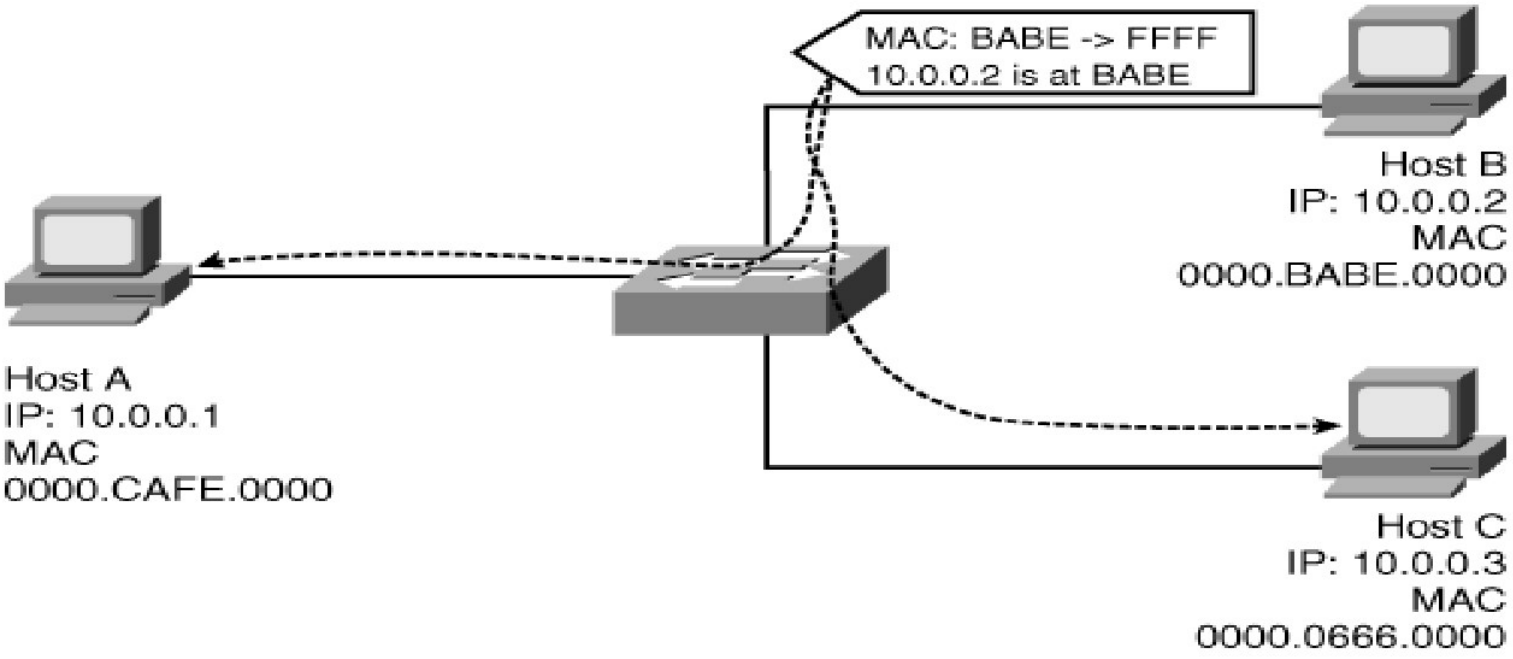
2.4

Defense:

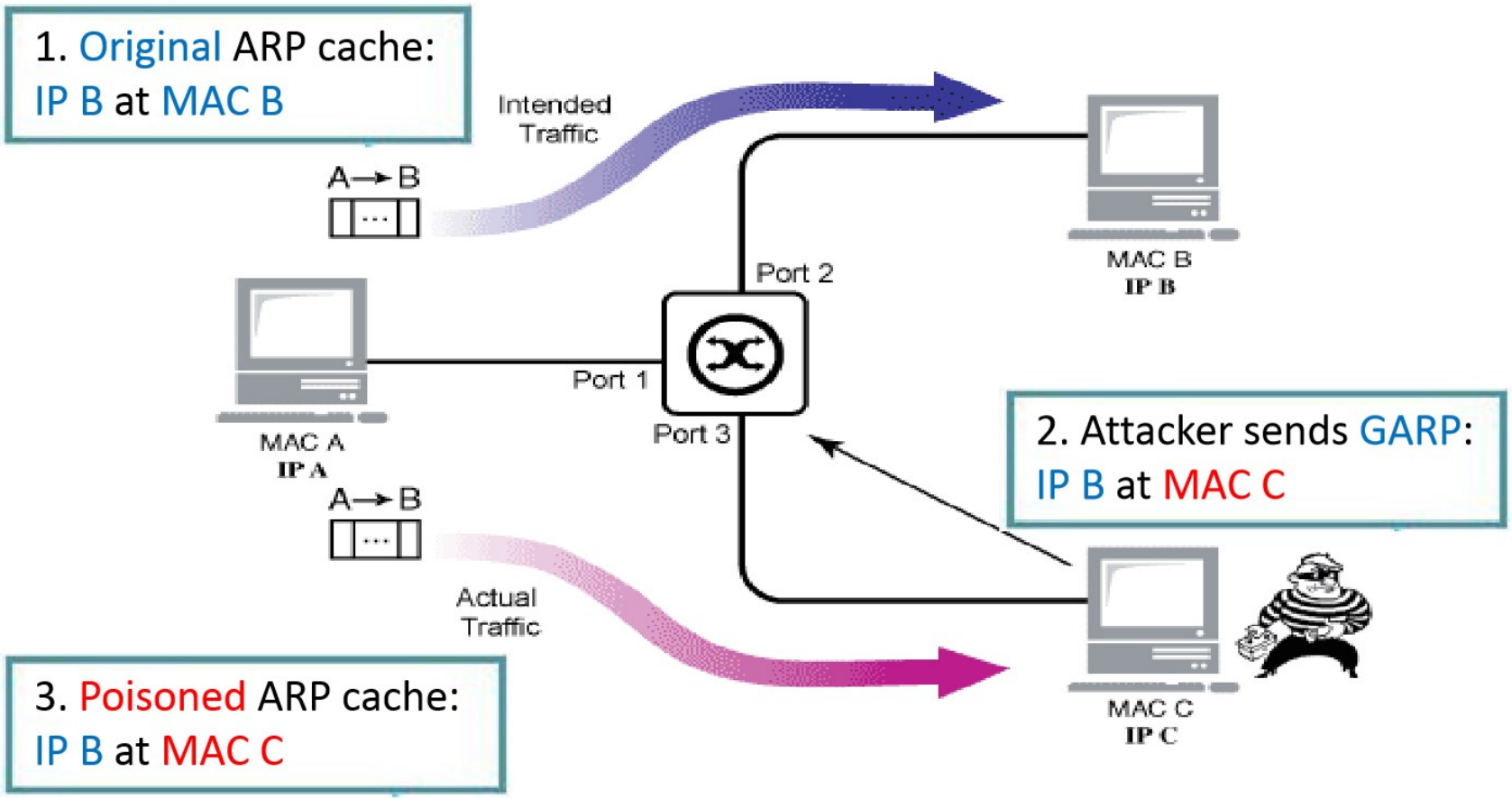
- Configure DAI (Dynamic ARP Inspection) in switches

In ICT1010, you learned about ARP over layer 2 Ethernet. In fact, there is also gratuitous ARP (GARP), which is ARP announcement or unsolicited ARP reply sent without a request.

Based on RFC826, a compliant host will update its ARP cache with the sender's IP/MAC upon receiving ARP request or ARP reply.

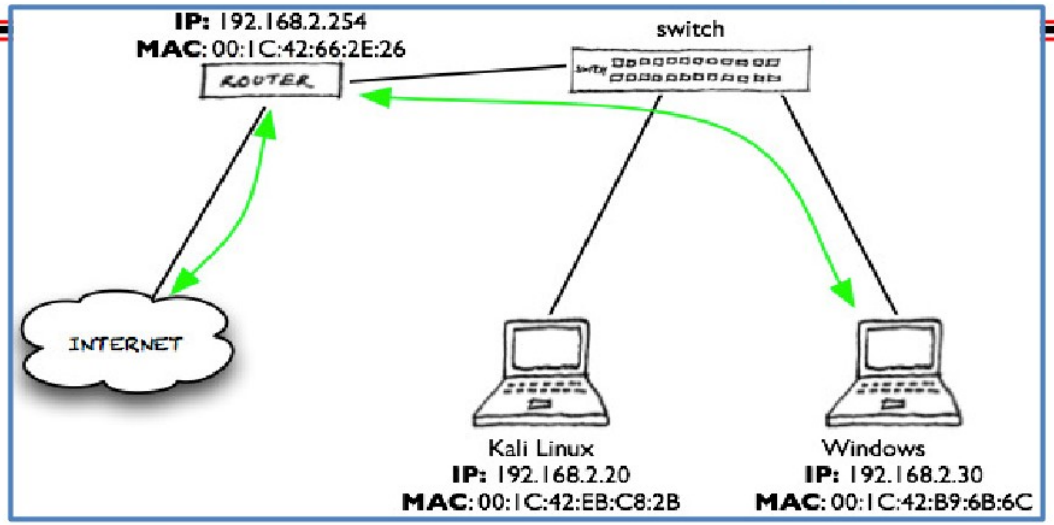


Unfortunately, gratuitous ARP can be easily misused to perform ARP poisoning attack to poison the ARP cache of victim host.

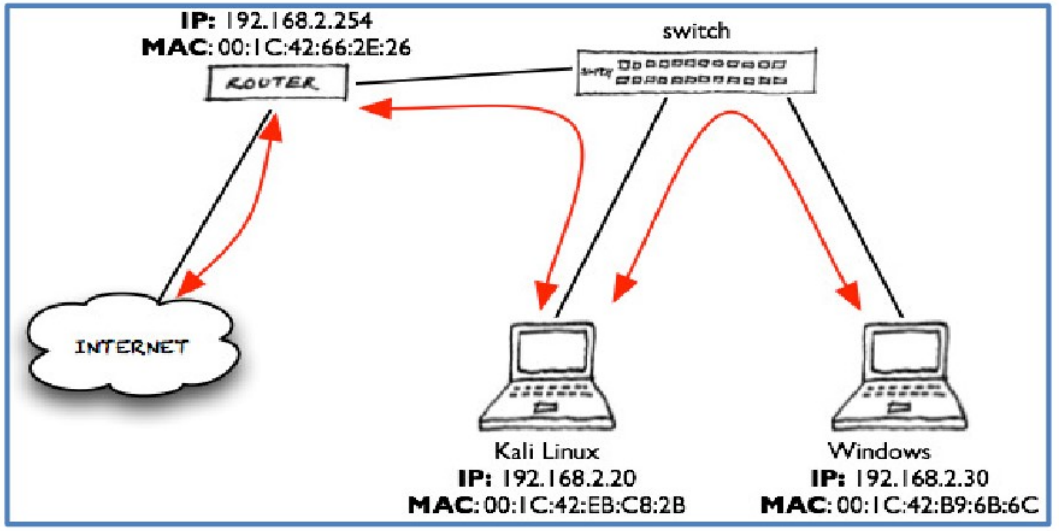


To make it worse, ARP poisoning attack can be extended to poison the ARP caches of victim host and default gateway simultaneously to conduct Man-in-the-Middle (MitM) attack!

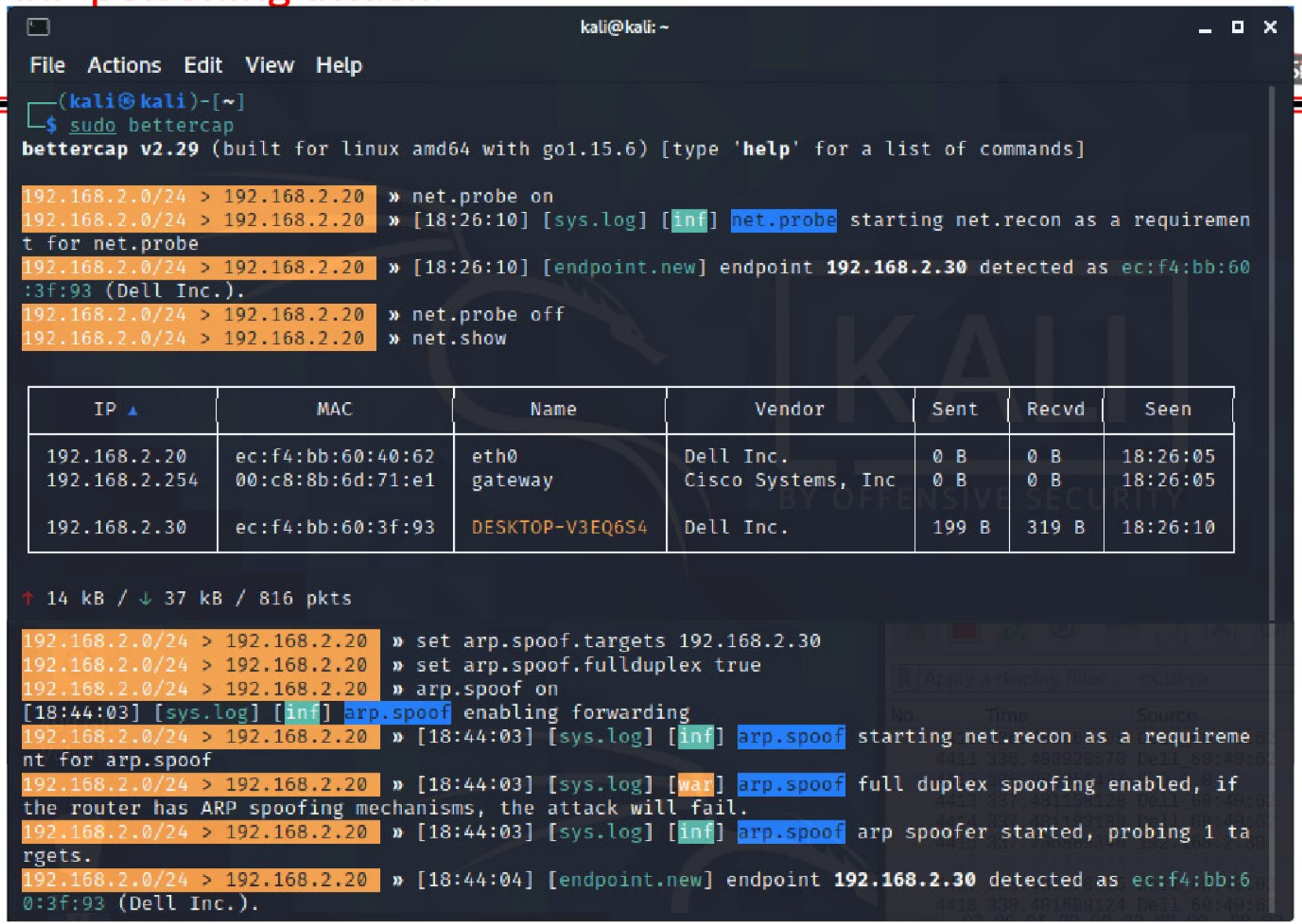
Before attack:



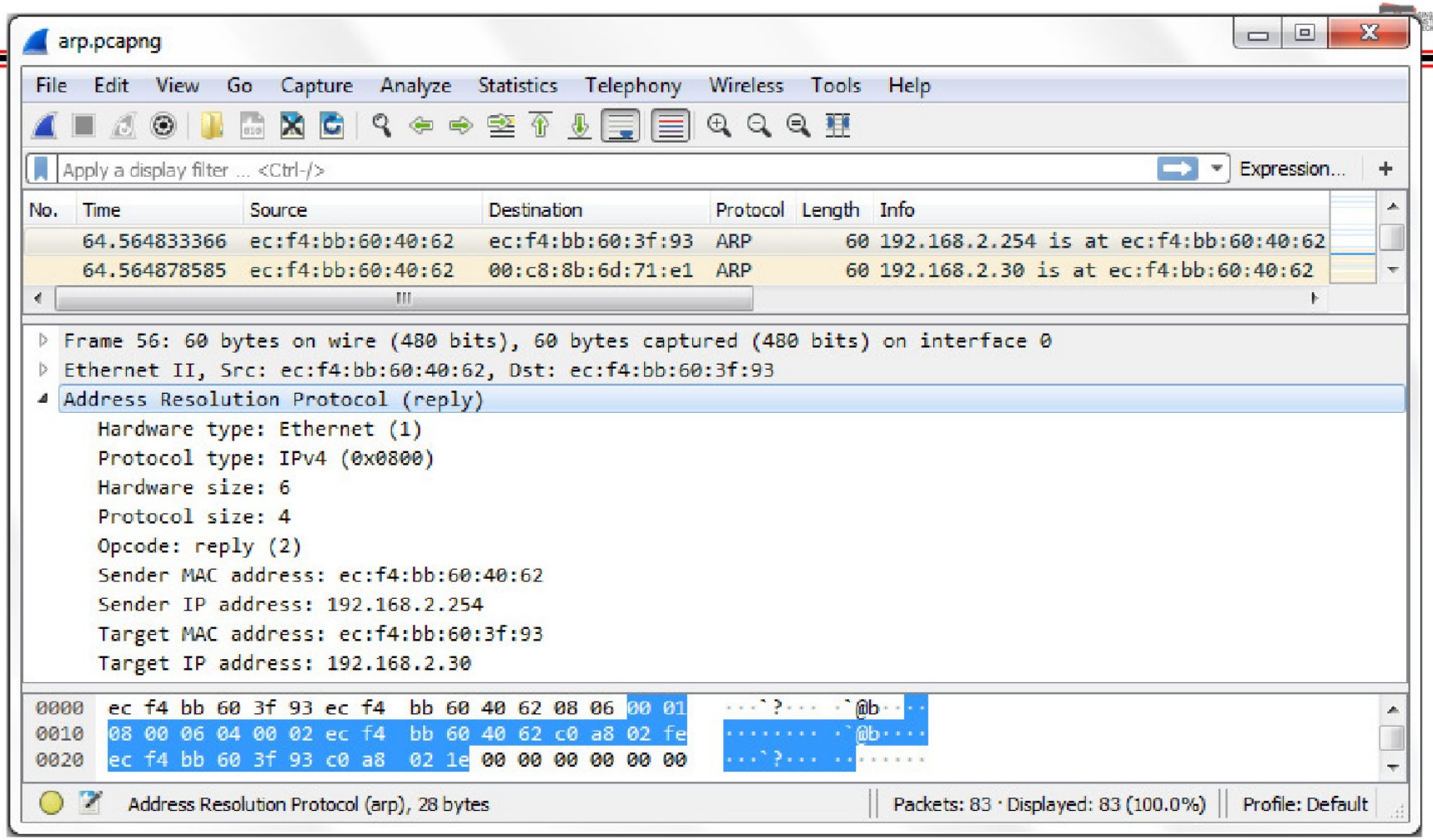
After ARP poisoning, all traffic between Windows host and default gateway are now going through the attacker leading to MitM attack!



In Kali Linux, a tool called Bettercap could be used to conduct ARP poisoning attack!



Notice that gratuitous ARP replies were sent out by Bettercap upon initiating ARP poisoning attack.



After ARP poisoning attack, all communications of victim are redirected through attacker and could be captured.

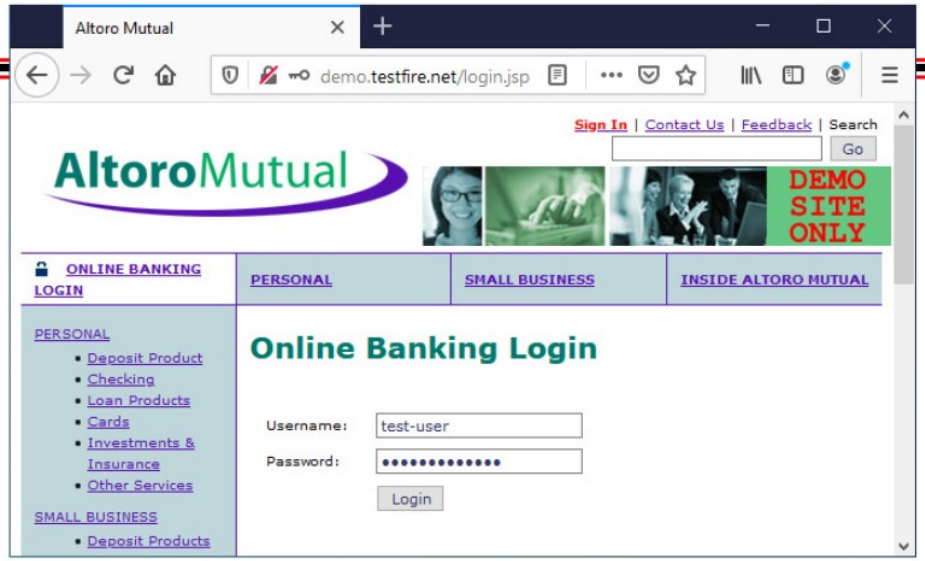
Enabling sniffing in Bettercap:

```
192.168.2.0/24 > 192.168.2.20 » set net.sniff.local true
192.168.2.0/24 > 192.168.2.20 » net.sniff on
[18:49:57] [net.sniff.http.request] http DESKTOP-V3EQ6S4 GET demo.testfire.net/login.jsp
192.168.2.0/24 > 192.168.2.20 » [18:49:57] [net.sniff.http.request] http DESKTOP-V3EQ6S4 GET demo.test
fire.net/login.jsp
[18:49:57] [net.sniff.http.response] http 65.61.137.117:80 200 OK → DESKTOP-V3EQ6S4 (5.4 kB text/html;
charset=ISO-8859-1)
192.168.2.0/24 > 192.168.2.20 » [18:49:57] [net.sniff.http.response] http 65.61.137.117:80 200 OK → D
ESKTOP-V3EQ6S4 (1.2 kB text/html;charset=ISO-8859-1)
[18:49:57] [net.sniff.http.request] http DESKTOP-V3EQ6S4 GET demo.testfire.net/style.css
192.168.2.0/24 > 192.168.2.20 » [18:49:57] [net.sniff.http.request] http DESKTOP-V3EQ6S4 GET demo.test
fire.net/style.css
```

Username and password could be captured if not encrypted!

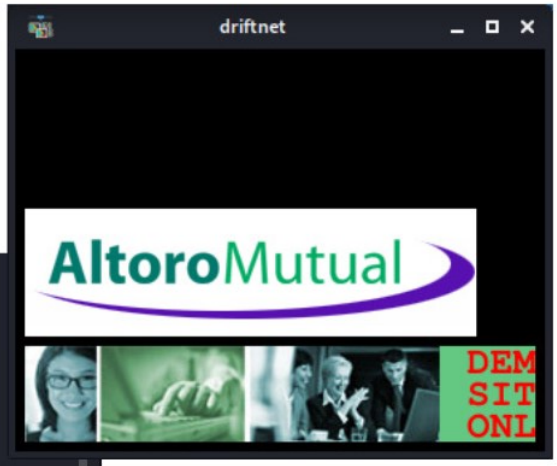
```
192.168.2.0/24 > 192.168.2.20 »
POST /doLogin HTTP/1.1
Host: demo.testfire.net
Accept-Encoding: gzip, deflate
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 49
Origin: http://demo.testfire.net
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0) Gecko/20100101 Firefox/72.0
Upgrade-Insecure-Requests: 1
Referer: http://demo.testfire.net/login.jsp
Cookie: JSESSIONID=B7161875E87F1744E6A8FED9A5E5B74E

uid=test-user&passw=test-password&btnSubmit=Login
```

You may also try driftnet in Kali to sniff and display the images that the victim is seeing!



Victim logging in at his/her machine.

```
┌──(kali㉿kali)-[~]
└─$ sudo driftnet -i eth0
```

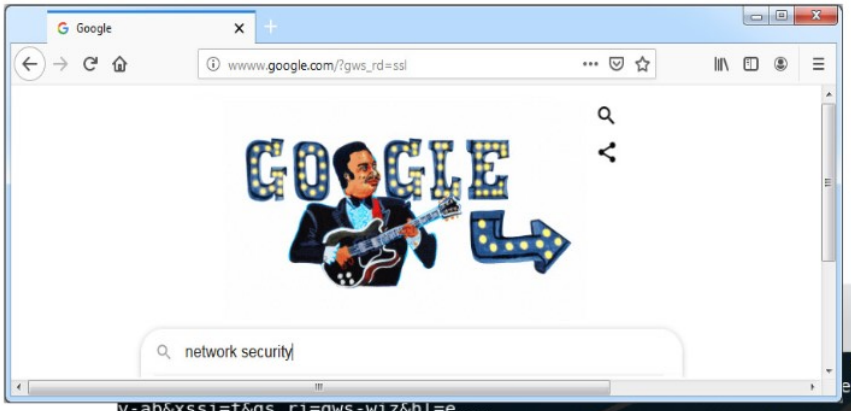


Displaying and storing of images captured by driftnet at attacker's machine:

```
┌──(kali㉿kali)-[~]
└─$ sudo su
┌──(root💀kali)-[/home/kali]
└─# cd /tmp

┌──(root💀kali)-[/tmp]
└─# ls driftnet*
driftnet-60a411cb327b23c6.gif   driftnet-60a411cb6b8b4567.gif
driftnet-60a411cb643c9869.jpeg  driftnet-60a411cc66334873.jpeg
```

Even if visiting https website, an attacker may try SSL stripping or HSTS bypass attacks!

- HSTS (HTTP Strict Transport Security) (RFC 6797) is a security mechanism for web sites to declare to browsers to access via secure connections only.



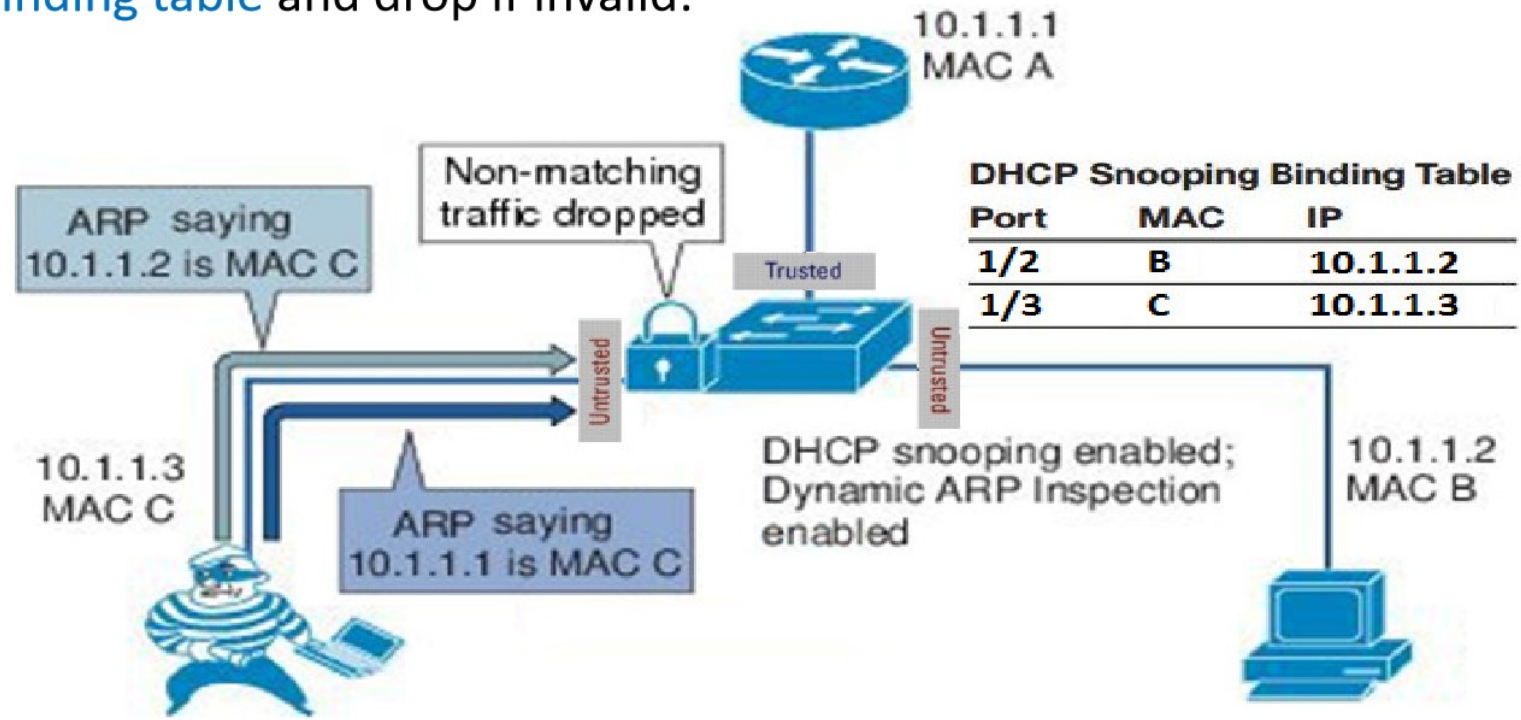
Instead of https://www.google.com notice that victim is redirected to wwww.google.com.

```
network secu
ecur&cp=13&clie
y-ab&xssi=t&gs_ri=gws-wiz&hl=e...
[192.168.2.30:1342 > 172.217.24.100:80] Searched wwww.google.com: network secur
[192.168.2.20] GET www.google.com/complete/search?q=network%20secur&cp=13&client
-ab&xssi=t&gs_ri=gws-wiz&hl=en...
[192.168.2.20:35836 > 74.125.68.147:80] Searched www.google.com: network secur
[192.168.2.30] GET wwww.google.com/complete/search?q=network%20securi&cp=14&cli
sy-ab&xssi=t&gs_ri=gws-wiz&hl=...
[192.168.2.30:1342 > 172.217.24.100:80] Searched wwww.google.com: network securi
[192.168.2.20] GET www.google.com/complete/search?q=network%20securi&cp=14&clie
y-ab&xssi=t&gs_ri=gws-wiz&hl=e...
[192.168.2.20:35838 > 74.125.68.147:80] Searched www.google.com: network securi
[192.168.2.30] GET wwww.google.com/complete/search?q=network%20securit&cp=15&client=
psy-ab&xssi=t&gs_ri=gws-wiz&hl...
[192.168.2.30:1342 > 172.217.24.100:80] Searched wwww.google.com: network securit
[192.168.2.20] GET www.google.com/complete/search?q=network%20securit&cp=15&client=p
sy-ab&xssi=t&gs_ri=gws-wiz&hl=...
[192.168.2.20:35840 > 74.125.68.147:80] Searched www.google.com: network securit
[192.168.2.30] GET wwww.google.com/complete/search?q=network%20security&cp=16&client
=psy-ab&xssi=t&gs_ri=gws-wiz&h...
[192.168.2.30:1342 > 172.217.24.100:80] Searched wwww.google.com: network security
[192.168.2.20] GET www.google.com/complete/search?q=network%20security&cp=16&client=
psy-ab&xssi=t&gs_ri=gws-wiz&hl...
```

Thus, attacker can still eavesdrop victim's communications.

To prevent ARP poisoning attack, a switch enabled with DHCP snooping can be further configured with DAI (Dynamic ARP Inspection) to drop invalid ARP traffic on untrusted ports.

After enabling DAI, all ports are untrusted by default similar as DHCP snooping. The MAC and IP addresses in the ARP packets received on these untrusted ports will then be validated against the DHCP snooping binding table and drop if invalid:



An example of configuring DAI together with DHCP snooping on a switch and verifying the status:

```
SW(config)# ip dhcp snooping
SW(config)# ip dhcp snooping vlan 10
SW(config)# ip arp inspection vlan 10
SW(config)# interface fa1/1
SW(config-if)# ip dhcp snooping trust
SW(config-if)# ip arp inspection trust
```

Configuring trusted DAI port similar as DHCP snooping

Each port can also be configured with ARP rate-limit (default is 15 ARP pps). If exceeded, the port will transit to err-disabled state.

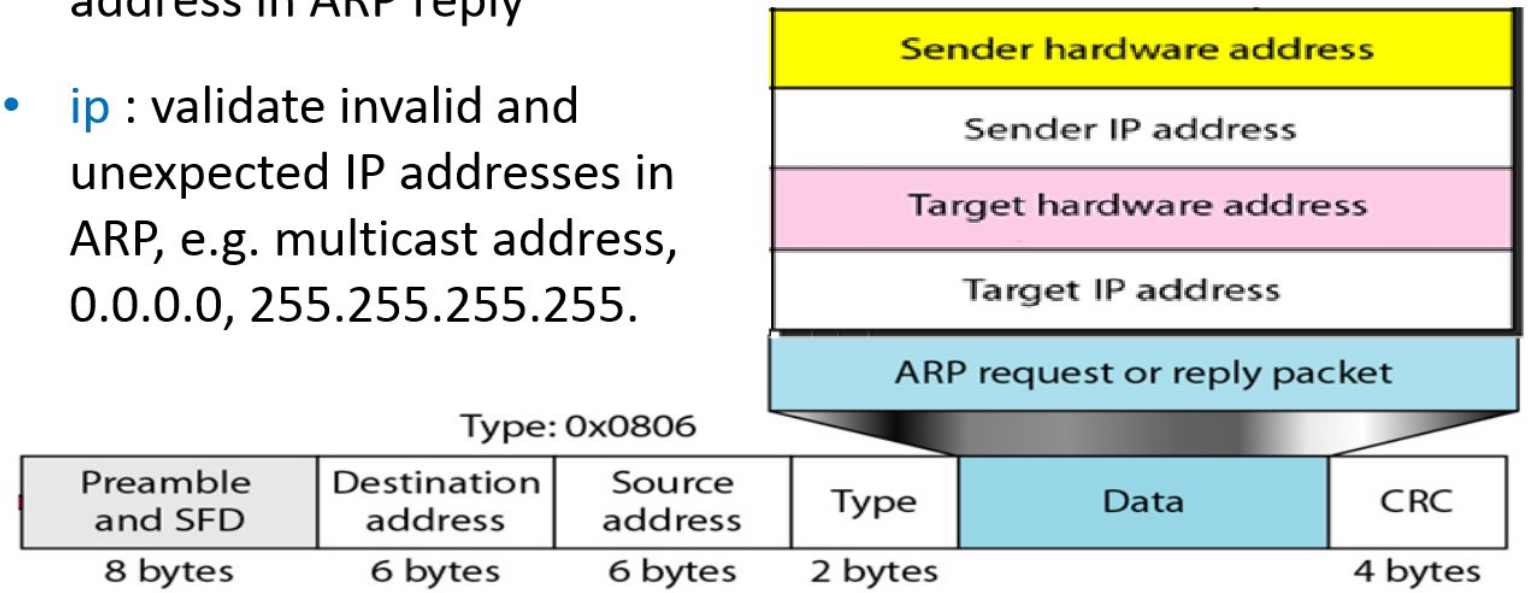
```
Switch(config-if)# ip arp inspection limit {rate
pps [burst interval seconds] | none}
```

```
sw2# show ip arp inspection interfaces
 Interface          Trust State      Rate (pps)     Burst Interval
 ---------------    -----------      ----------     --------------
 Fa1/1              Trusted                None                N/A
 Fa1/2              Untrusted                15                  1
 Fa1/3              Untrusted                15                  1
! output removed for brevity
```

For added security, DAI can also be configured to validate contents of ARP reply matching layer 2 Ethernet addresses as follows:

```
Switch(config)# ip arp inspection validate {[src-mac] [dst-mac] [ip]}
```

- src-mac : validate layer 2 source MAC against sender hardware address in ARP reply
- dst-mac : validate layer 2 destination MAC against target hardware address in ARP reply
- ip : validate invalid and unexpected IP addresses in ARP, e.g. multicast address, 0.0.0.0, 255.255.255.255.



For non-DHCP environment with no entry in DHCP snooping table, DAI can also be configured by using ARP ACL (access control list) as follow:

Configure IP to MAC mapping manually in ARP ACL, but can be tedious:

```
Switch(config)# arp access-list arp-acl-name
Switch(config-arp-nacl)# permit ip host sender-ip mac host sender-mac
[Repeat the previous command as needed]
Switch(config-arp-nacl)# exit
```

Next, apply ARP ACL to DAI:

```
Switch(config)# ip arp inspection filter arp-acl-name
                vlan vlan-range [static]
```

The optional 'static' keyword is to indicate checking ARP ACL only because by default, intercepted ARP on untrusted ports are first matched with ARP ACL, and if no match, then the DHCP snooping binding table.

An example of configuring DAI for non-DHCP environment:


```
Switch(config)# arp access-list StaticARP
Switch(config-arp-nacl)# permit ip host 192.168.1.10 mac host 0006.5b02.a841
Switch(config-arp-nacl)# exit
Switch(config)# ip arp inspection vlan 10
Switch(config)# ip arp inspection filter StaticARP vlan 10
Switch(config)# interface gigabitethernet 1/0/49
Switch(config-if)# ip arp inspection trust
```

```
sw2# show ip arp inspection vlan 10

Source Mac Validation      : Disabled
Destination Mac Validation : Disabled
IP Address Validation      : Disabled

 Vlan     Configuration    Operation   ACL Match          Static ACL
 ----     -------------    ---------   ---------          ----------
   10     Enabled          Active      StaticARP          No

 Vlan     ACL Logging      DHCP Logging      Probe Logging
 ----     -----------      ------------      -------------
   10     Deny             Deny              Off
```

Lab Exercise 3

SINGAPORE INSTITUTE OF TECHNOLOGY

3.1–3.3 IP address spoofing and MAC address spoofing attacks

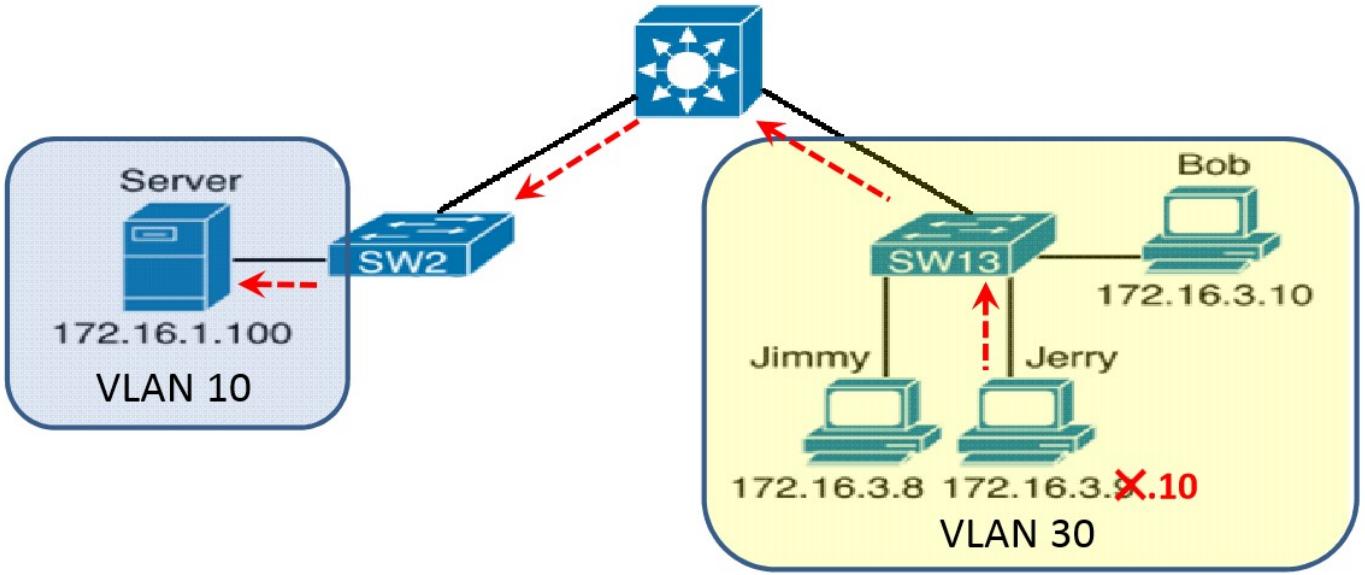
3.4 Defense:

- Configure IPSG (IP Source Guard) in switches

In a LAN, access control list (ACL) may be applied at switches to filter traffic in inter-VLAN routing, but an attacker may perform IP address spoofing attack to bypass it!

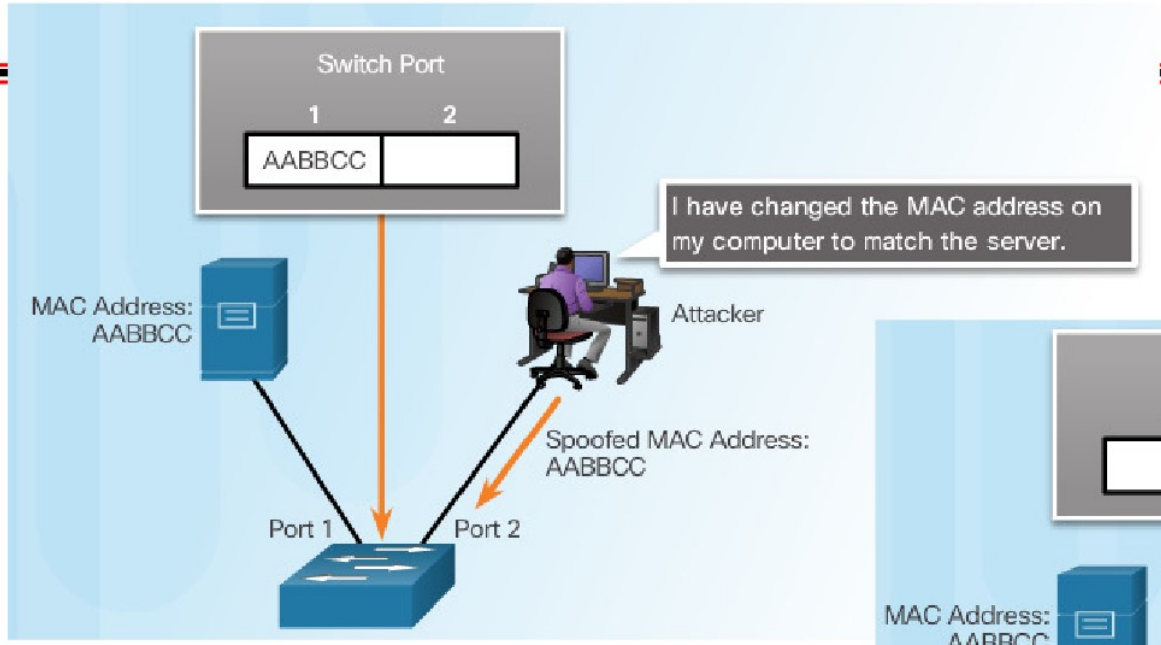
For example, apply ACL to only allow Bob to access server as follows:

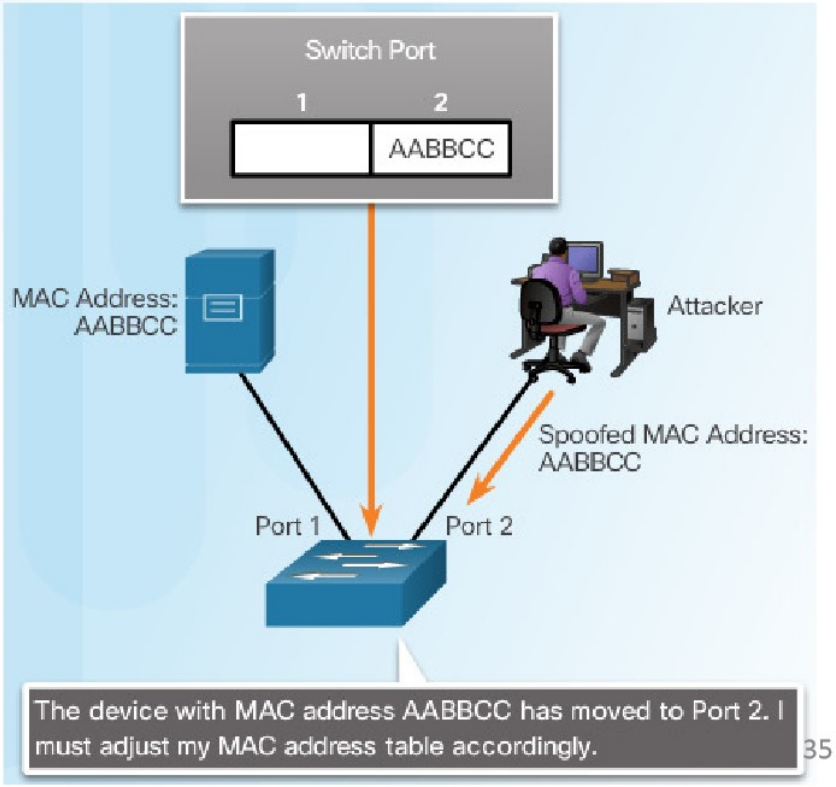
```
switch(config)# access-list 1 permit 172.16.3.10
switch(config)# int vlan 10
switch(config-if)# ip access-group 1 out
```



Unfortunately, an attacker Jerry performing IP address spoofing attack and is still able to access the server!

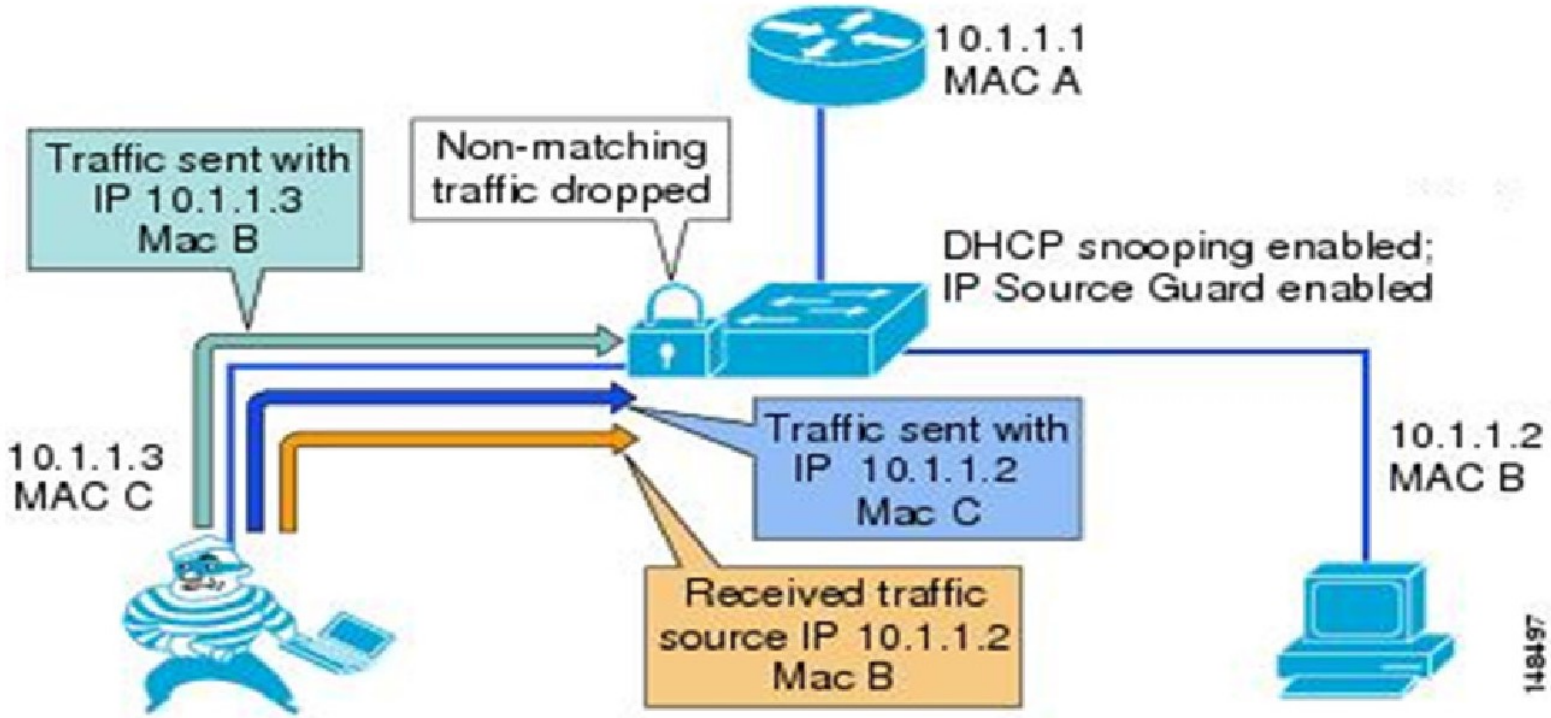
An attacker may also perform MAC address spoofing attack in an attempt to steal network traffic.





To prevent IP spoofing attack, a switch configured with DHCP snooping can be further configured with IP source guard (IPSG).

When IPSG is configured on untrusted ports, all traffic are blocked except DHCP. The untrusted port will only allow traffic after IP address is assigned by DHCP, or static IP has been configured.



An example of configuring IP source guard on the untrusted interface using IP filter to prevent IP spoofing attack.

Configure IPSG using ip filter to defend against IP spoofing attack as follows:

```
Switch(config)# interface gigabitethernet0/1
Switch(config-if)# ip verify source
Switch# show ip verify source
Interface  Filter-type  Filter-mode  IP-address       Mac-address        Vlan
---------  -----------  -----------  ---------------  -----------------  ----
Gi0/1      ip           active       40.1.1.24                           1
```

With ip filter, any received packet with a different source IP address will be dropped.

IP source guard may also be configured using IP-MAC filter to prevent IP and MAC address spoofing attacks.

Configure IPSG using ip-mac filter to prevent IP and MAC spoofing attacks:

```
Switch(config)# interface gigabitethernet0/1
Switch(config-if)# ip verify source port-security
Switch# show ip verify source
Interface  Filter-type  Filter-mode  IP-address       Mac-address        Vlan
---------  -----------  -----------  ---------------  -----------------  ----
Gi0/1      ip-mac       active       40.1.1.24        00:00:00:00:03:04  1
```

Note: Port Security must be configured for mac filter to work. In addition, DHCP option 82 must be enabled; otherwise the switch will drop DHCP server reply and client will not be able to obtain IP address.

To implement IPSG, the switch maintains an IP source binding table which is automatically learned from DHCP snooping or statically configured manually.

To manually configure static IP source binding:

```
Switch(config)# ip source binding mac-address vlan vlan-id ip-address
                interface type number/number
```

To verify IP source binding table:

```
Switch# show ip source binding
MacAddress          IpAddress       Lease(sec)  Type           VLAN  Interface
------------------  --------------  ----------  -------------  ----  ------------------
00:02:B3:3F:3B:99   55.5.5.2        6522        dhcp-snooping  10    FastEthernet6/10
00:00:00:0A:00:0B   11.0.0.1        infinite    static         10    FastEthernet6/11
```