

Demystifying Layer 2 Attacks

BY ABHISHEK SINGH, CISSP

Prologue

The obvious question to ask is why the heck do I care for Layer 2?

The obvious answer to this is - no matter how secure you make your TCP/IP fortress if a hacker can punch in at any of the layers he has the keys of kingdom. Moreover most of the firewalls are not capable for detecting these kinds of attacks. Also to successfully conduct most of these attacks we have to be on the segment as of the victim.

So far VLANs was created as a means of secure LAN talks, but as we will see in the paper that with careful techniques these can also be subverted. Think of the scenario if the attacker can puncture the stack at Layer 2, he can now control all the above traffic.

I will try to provide a discussion on these attacks and their mitigation mechanism (indebted to the pioneer in the area CISCO)

So let's rock

What Layer 2 is/ does?

Basic purpose of Link Layer in TCP/IP protocol suite is to send and receive:

1. IP datagrams for the IP module,
2. ARP requests and replies for the ARP module, and
3. RARP requests and replies for the RARP module.

TCP/IP supports many different link layers, depending on the type of networking hardware being used: Ethernet, token ring, FDDI (Fiber Distributed Data Interface), RS-232 serial lines, but we will concentrate on Ethernet in this paper

Each device on Ethernet is identified by MAC (Media Access Control) address (hardware address or physical address) which is unique and comprises of 12-digit hexadecimal numbers (48 bits in length). By convention, MAC addresses are usually written in one of the following two formats:

MM:MM:MM:SS:SS:SS

MM-MM-MM-SS-SS-SS

The first half of a MAC address contains the ID number of the adapter manufacturer. These IDs are regulated by an Internet standards body (see sidebar). The second half of a MAC address represents the serial number assigned to the adapter by the manufacturer. In the example,

00-0D-60-C5-0C-C1

The prefix

00-0D-60

Indicates the manufacturer is IBM Corporation.

FF-FF-FF-FF-FF-FF is a broadcast MAC address.

Layer 2 Devices

Switches and Bridges are widely used L2 devices, replacing the shared medium with the switched one hence conserving the bandwidth because the data is sent only on that network segment which has the receiving device (in contrast to the broadcast used by hubs). Other benefits are:

1. Improved security (users are less able to tap-in into other user's data).
2. Better management (the ability to control who receives what information (i.e. Virtual LANs).
3. Limited impact of network problems.
4. Ability to operate some links in full duplex (rather than half duplex required for shared access).

To the core of switching lies a mechanism to identify where is which device located, so a sort of mapping (can be manual or automatic) is formed between ports and the MAC addresses of the devices (e.g. PC) connected to that port. In CISCO (I will be using CISCO as ref in my paper as it is most widely used vendor in networks today) this is referred to as **Content-Addressable Memory (CAM)** table. It is important to note that CAM tables are of fixed size.

How do switches learn the topology of segment?

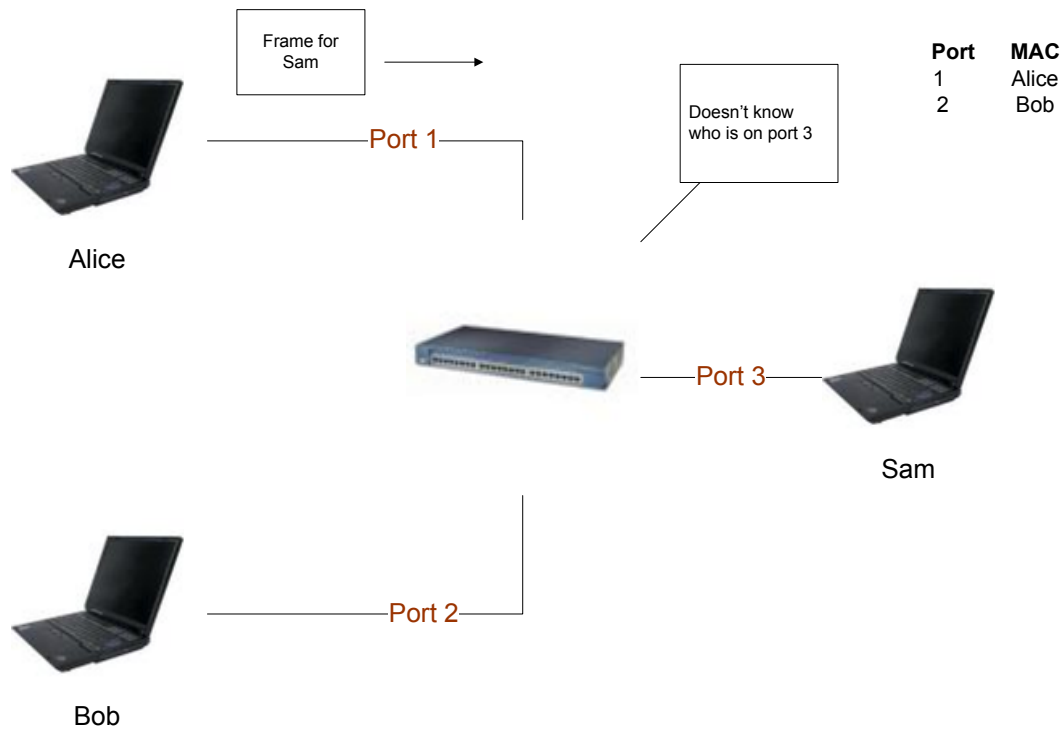


Fig 2. Frame recived for unknown MAC

Fig2 shows situation in which a frame is received for an unkown MAC, fig 3 shows the switch response of flooding the frame to all the ports except the one on which it was received.

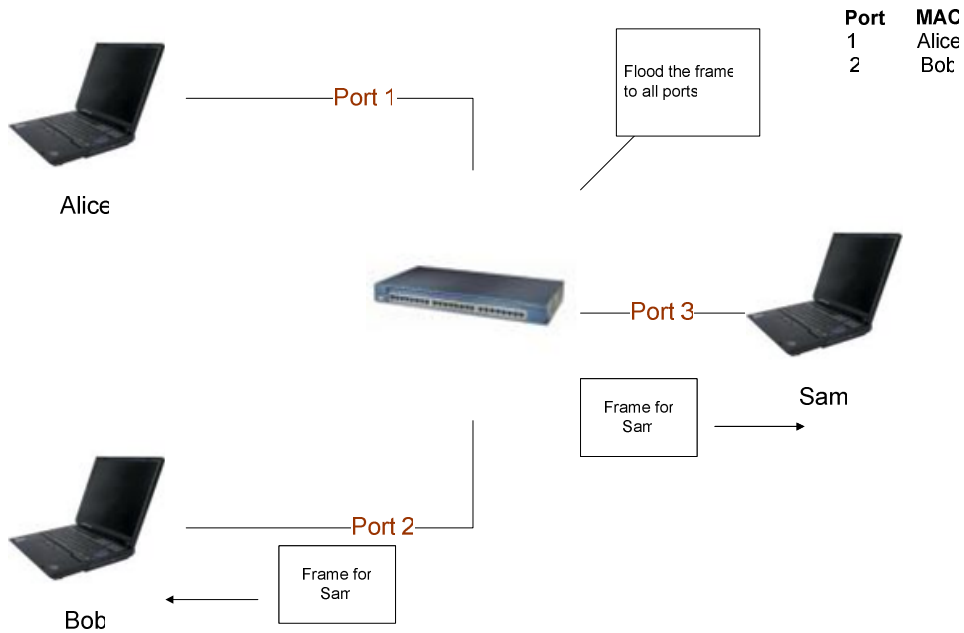


Fig 3. Flooding the frame

Now once Sam replies switch will learn that he is on Port 3 and enters this in its CAM table.

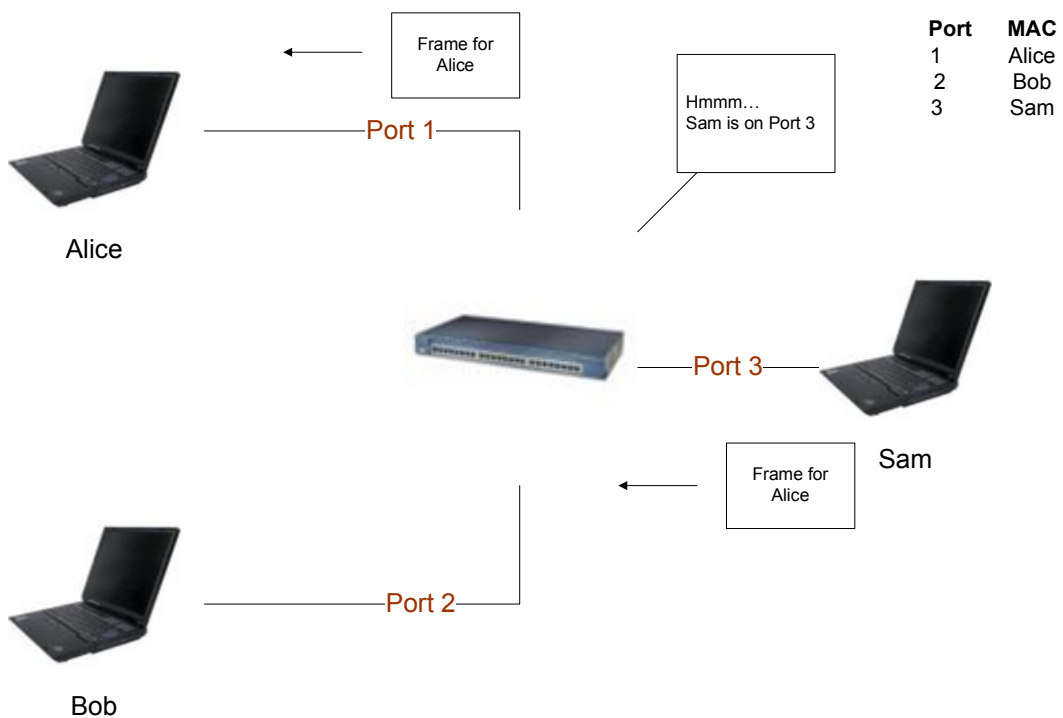


Fig 4. CAM entry made new MAC learnt

Layer 2 Attacks

Attack 1 - CAM table based attacks/ CAM table Overflow

This attack exploits the fact that CAM tables are of limited size and first appeared in *macof* tool in 1999, was written in PERL later incorporated into *dsniff*. Attacker sends a flood of invalid source MAC addresses to the extent that the CAM table is flooded (exploiting the way the switch learns and populate its CAM table with corresponding MAC addresses) and switch will start behaving like a hub. It has to be noted that in order for this attack to be successful flooding has to occur before the entries timeout in the switch CAM table.

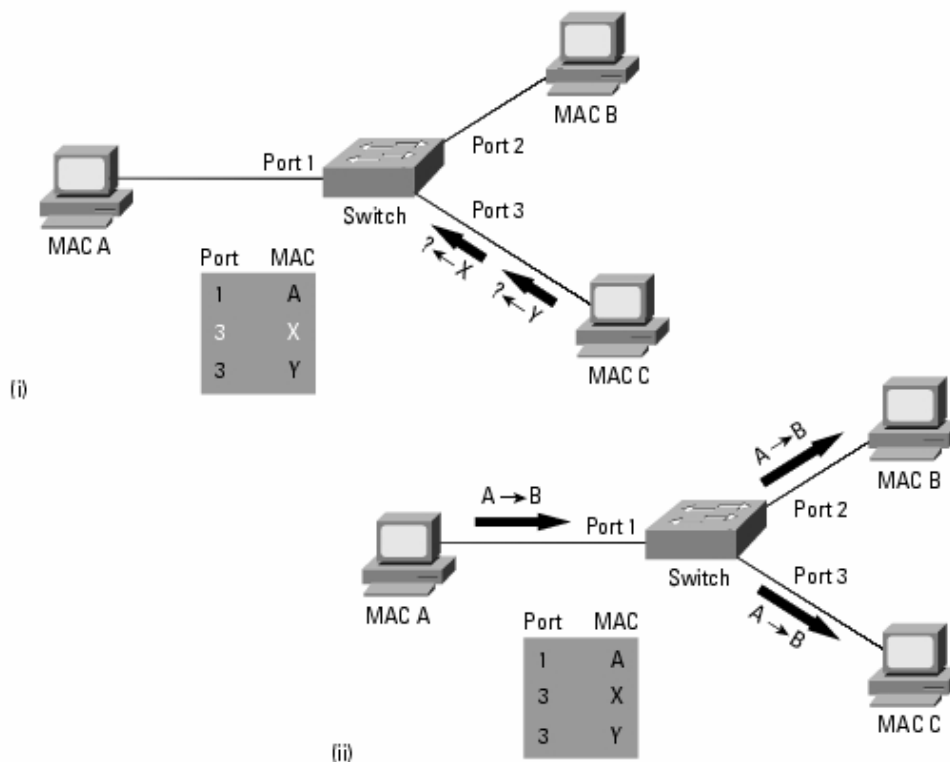


Fig 5. CAM table overflow attack

Consequence

Successful attack will render the switch “Fail - Open” in which case it will start behaving like a Hub and so all the traffic can be sniffed.

Mitigation of CAM table overflow attacks

Two methods

1. Configure Port Security – Manually configure CAM table to specify MAC address on each port, seems impossible on large networks.
2. Number of MAC’s on Port – Specify the number of MAC which can be learned on a switch port. An invalid entry will cause either blocking of the offending MAC or shutdown of the switch port.

Attack 2 – ARP Based attacks

Address Resolution Protocol (ARP) [RFC 826] is used for IP to MAC binding, when a host wants to send data to destination the information available to it is the IP Address, however in order to deliver the packet source needs to construct a frame now since it doesn’t have the MAC address of the destination it sends an ARP broadcast and the destination, if on the same LAN segment, replies back with its MAC address, in other cases the next hop gateway will respond.

- ARP entries are cached in an ARP table on the hosts for a specific period of time.
- ARP is a **Stateless** protocol.

Hardware type		Protocol type
HW addr lth	P addr lth	Opcode
Source hardware address		
Source protocol address		
Destination hardware address		
Destination protocol address		

ARP message

Fig 6. ARP Message Format

ARP Cache Poisoning

An attacker on the LAN can inject malicious IP – MAC bindings (ARP Packets) leading to a scenario in which traffic from/ to victim's machine are either delivered to another host or is not delivered at all.

This can be achieved by four methods:

1. **Malicious Response** – Attacker sends out a response which doesn't correspond to any request this response will be treated as legit by most of the implementations. Effect of this is a false binding in victims' cache, if this is a broadcast all the hosts on the LAN are affected.
2. **Malicious Request** – By nature of ARP hosts on LAN segment cache there table with data from ARP requests sent out by other host for example if host A sent a ARP Request for host B then host C might cache this host A binding in its table. Exploiting this, an attacker can send out legit looking requests and thereby poisoning the ARP cache of other hosts on the segment.
3. **Malicious Response to a legitimate Request** – Attacker might wait for the real host to respond to an ARP request, once this is done, attacker also responds to the same arp request with malicious MAC – IP binding, the receiver, by design in most of the cases, will tend to make an entry for the response which came last.
4. **Malicious Request and Malicious Response** – Attacker can send out a malicious request for some MAC on the LAN and then send out a corresponding ARP reply thereby poisoning the victim's ARP cache.
5. **Gratious ARP** – When a system boots up on LAN it send a gratuitous ARP (it's MAC – IP binding), now if a malicious GARP is sent the systems on LAN will force the systems on LAN to make a faulty entry in there cache table.

Consequences

DOS Attacks – A malicious entry with a non-existent MAC address can lead to a DOS attack, the situation becomes more dangerous if the malicious entry is made for the gateway.

Man-in-the-middle Attack

With Cache Poisoning an attacker inserts himself in the path between source and destination and cause MiM. Traffic is still forwarded to the destination by enabling IP forwarding and disabling ICMP redirect on the malicious host. Attacker now spoofs the client IP address and opens up the connection to the server any access control mechanisms based on IP address will then render ineffective (see dsniiff).

Advantages of MiM

Defeating the Firewalls - If there are systems on LAN which can have direct access to internet through the firewalls (like proxy, mail servers etc) attacker can abuse these to have direct connection to WAN.

Exploiting the Secure Sessions (SSL, SSH etc) – Attacker can hijack these sessions and provide fake certificates to the client, though these certificates can be easily proven to be fake but to most of the users these can cause the session establishment to a fake server (attacker).

SSL sniff can be used to accomplish this

"First, arpspoof convinces a host that our MAC address is the router's MAC address, and the target begins to send us all its network traffic. The kernel forwards everything along except for traffic destined to port 443, which it redirects to \$listenPort (10000, for example).

At this point, sslsniff receives the client connection, makes a connection to the real SSL site, and looks at the information in the server's certificate. sslsniff then generates a new certificate with an identical Distinguished Name and signs it with the end-entity certificate in \$certificateFile. sslsniff uses the generated certificate chain to do a SSL handshake with the client and proxy data between both hosts (while logging it, of course).

If the client is Internet Explorer, the browser will not notice anything unusual about the generated certificate due to the Certificate Chain vulnerability." – SSL Sniff homepage.

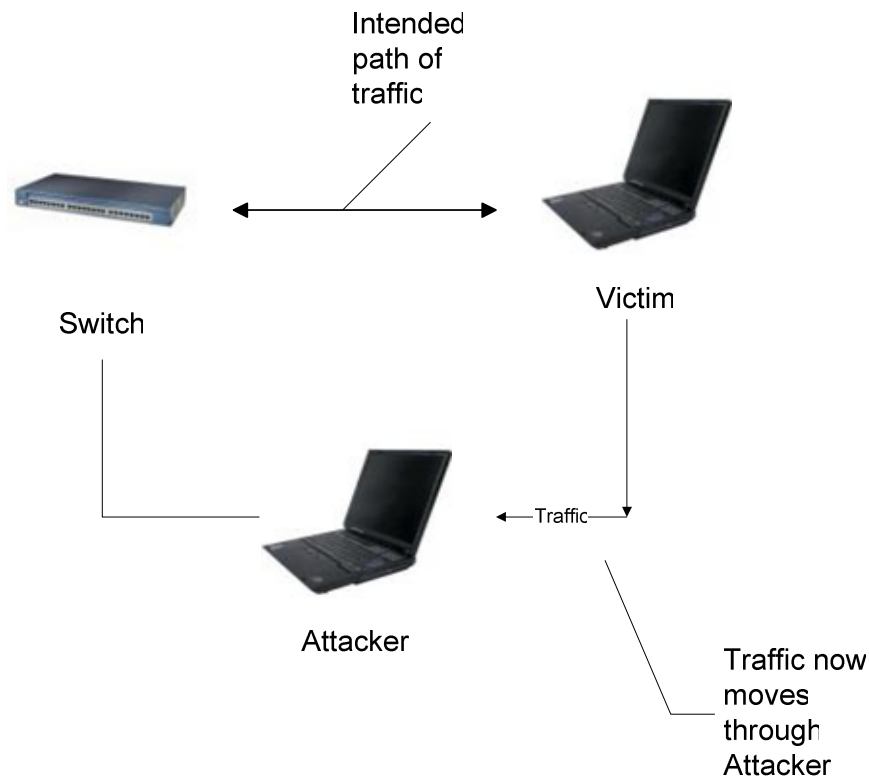


Fig 7. MiM Attack

Solaris Case - Study

As with any operating system defending against these ARP based attacks is very difficult so there are two options:

Use only static ARP entries – Manually add the entries to ARP table, seems impossible in realtime scenarios since LAN segments are big and evolving so making an entry for each machine on the LAN is near to impossible. (in some systems like Windows dynamic ARP entries take precedence)

Decrease the Entry Lifetime – Flushing the IP – MAC bindings often can be a partial solution in cases where 1 is not feasible

ndd can be used to change the ARP cache flush interval, as well as the IP routing table flush interval, as follows:

```
# ndd -set /dev/arp arp_cleanup_interval <time>
```

```
# ndd -set /dev/ip ip_ire_cleanup_interval <time>
```

In this example, **<time>** is the interval specified in milliseconds. Although this will not eliminate these types of attacks, it may slow the attacker down.

More stable Solution for Linux, BSD and Solaris Systems

Considering the fact that ARP is a Stateless protocol if we can implement a mechanism where we can make it Stateful, might solve the problem this is done by adding the kernel modules "Streams" depicted below:

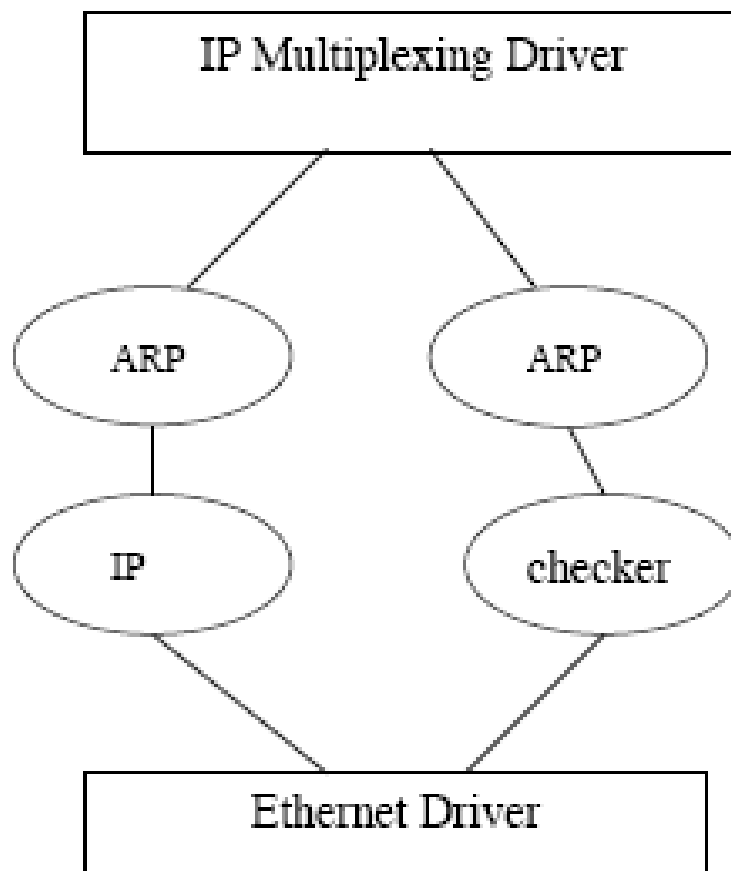


Fig 8. Stream based Solution for Cache Poisoning

In this case a Checker module:

Intercepts an ARP request (moving in downward direction) makes an entry for it.

Receives an ARP reply (moving in upward direction), checks to see if there is a corresponding ARP request, in which case this reply is considered legitimate.

Traffic is recorded in Lists, two separate lists are maintained one for each direction.

Algorithm can be briefed as:

When an ARP frame is received:

If this ARP frame is a response:

If a corresponding entry exists in the requested list:

Move the entry to the responded list and let the packet flow up to be processed by the host's ARP implementation.

Else, a corresponding entry is not found in the requested list, and:

Check if there exists a corresponding entry in the responded list,

If yes then multiple responses have been received, so:

Check whether the entry in the responded list corresponding to the IP address is same as that in the response.

If yes:

Refresh the entry in the ARP cache.

Else, the ARP cache entry is not consistent with the received packet:

Log the incident. Drop the frame. Mark the entry as Invalid.

Else, this is an unsolicited response.

Drop the frame and log the incident.

Else, this is a request, and:

Let the frame flow up and let the host's ARP implementation process it.

Else, a frame is being sent, and:

If the ARP frame is a response:

Let the frame flow down.

Else, this is a request, and:

Add a corresponding entry in the requested list.

Let the frame flow down.

Fig 9. Streams Module Algorithm for Cache Poisoning prevention

ARP Spoofing

This attack derives roots from above theory, in this, attacker spoofs the path between the server and client and inserts himself in between the communication path, resulting in MiM as described above.

Cisco suggests using Private VLANs for mitigation (see details http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/sw_7_1/conf_gd/vlans.htm#xtocid854519)

ARP Flooding

Similar to CAM overflow, this attack has its roots in flooding the switch with mass number of spurious ARP replies. Switch will then start acting as Hub and the traffic can be sniffed by attacker, also bandwidth loss will occur.

In order to mitigate ARP based attacks tools like **arpwatch** should be used, which will alert if any abnormal ARP activity is found on LAN.

Attack 3 – VLAN based Attacks

VLAN are used to logically divide a physical network into a group of systems which can communicate with each other, this is accomplished by configuring ports on switch which can receive the traffic for a specific VLAN and a Trunk port which will carry traffic from all VLAN's, thereby forming a layer of security on Layer 2.

VLANs work by tagging packets with an identification header and then restricting the ports that the tagged packets can be received on to those that are part of the VLAN. The two most prevalent VLAN tagging techniques are the IEEE 802.1q tag and the Cisco Inter-Switch Link (ISL) tag.

The VLAN header is inserted at Layer 2 and the information contained within the tags is used to identify which VLAN the traffic belongs to. Only those ports that belong to the VLAN specified in the header are capable of receiving the traffic. The destination address then further specifies the particular port within the VLAN where the traffic is destined.

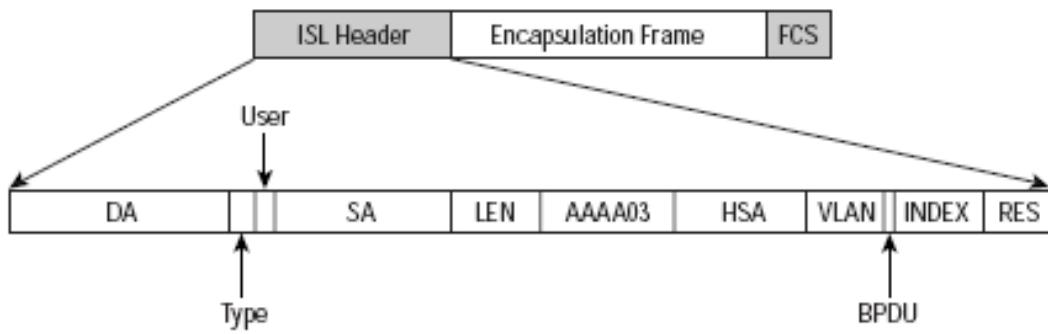


Fig 10. ISL Header and Frame

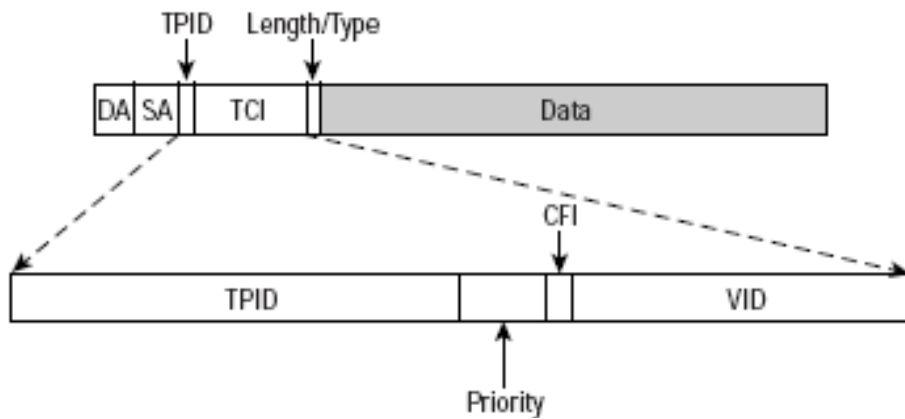


Fig 11. 802.1q Header and Frame

VLAN Hopping

VLAN hopping is a network attack whereby an end system sends out packets destined for a system on a different VLAN that cannot normally be reached by the end system. This traffic is tagged with a different VLAN ID to which the end system belongs (Double Tagging) . Or, the attacking system may be trying to behave like a switch and negotiate trunking so that the attacker can send and receive traffic between other VLANs (Switch Spoofing).

Switch Spoofing— Attacker tries to appear as a switch with trunk enabled. In order to successfully perform this attack, attacker must be capable of trunk, 802.1q, ISL signaling. This attack can allow attacker to see traffic from all VLAN's

Double Tagging— This attack involves encapsulating two 802.1q headers and forward it to wrong VLAN when the first switch encounters this frame it strips of the first tag and forwards the frame with second tag to all the port including the truck ports, when second switch receives this frame it will forward it to necessary destination.

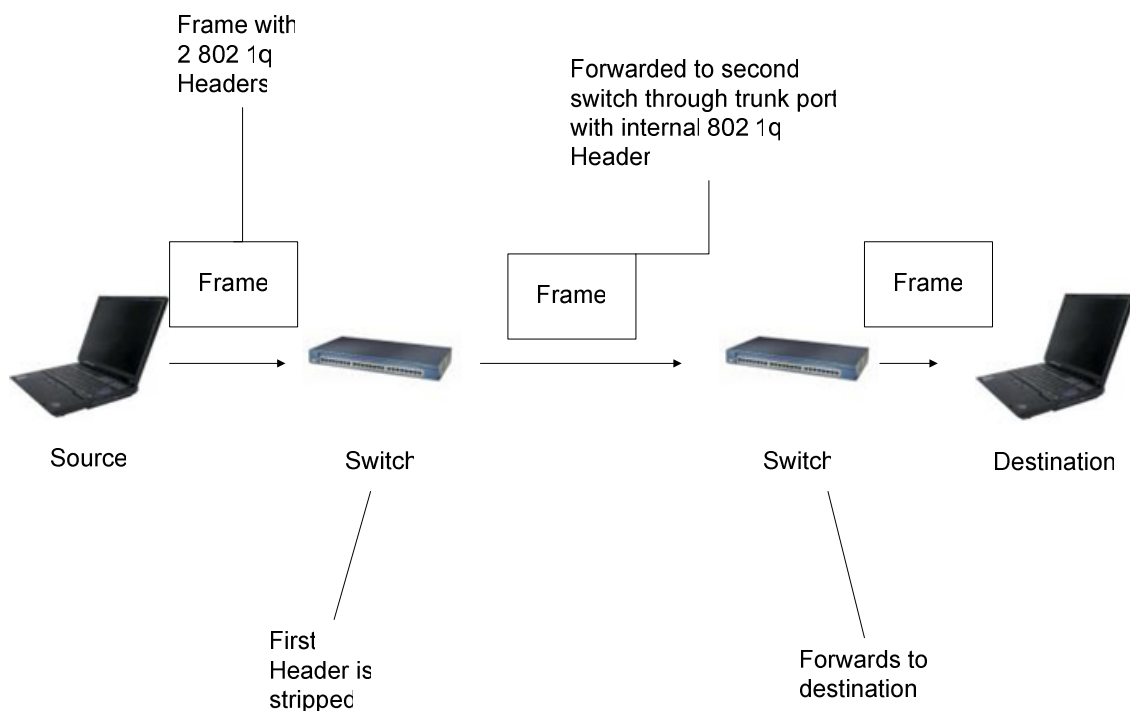


Fig 12. Double Tagging

Attack 4 - Spanning tree protocol based attack

Spanning tree is a link management protocol between switches of an extended network. Its main purpose is to choose a path between two switches (i.e. any two hosts) and remove any redundant paths which will be designated to a stand-by state (blocked). If any of the segments becomes unreachable the paths are recalculated and tree is reformed now with Standby paths in consideration. If there are redundant paths between two systems on a LAN it will lead to a duplication of the messages because switch will have multiple paths through which a host is visible to it. All paths are calculated with respect to a root switch.

Election of a Root Switch

Switches exchange Bridge Protocol Data Units (BPDU) to learn about other switches in the network these messages result in:

- One switch is elected as the root switch.
- The shortest distance to the root switch is calculated for each switch.
- A designated switch is selected. This is the switch closest to the root switch through which frames will be forwarded to the root.
- A port for each switch is selected. This is the port providing the best path from the switch to the root switch.
- Ports included in the Spanning-Tree Protocol are selected.

Usually the switch with the Lowest MAC address is elected as a Root which might not always be a good idea this can be altered by giving priority to the switches.

Attack

In order to successfully conduct this attack, attacker must be connected to atleast two switches. He then acts as a switch with lowest priority thereby advertising as a root, if successful this will result in the recalculatation of the paths and attacker will be elected as a root so having the access to all the traffic on the LAN.

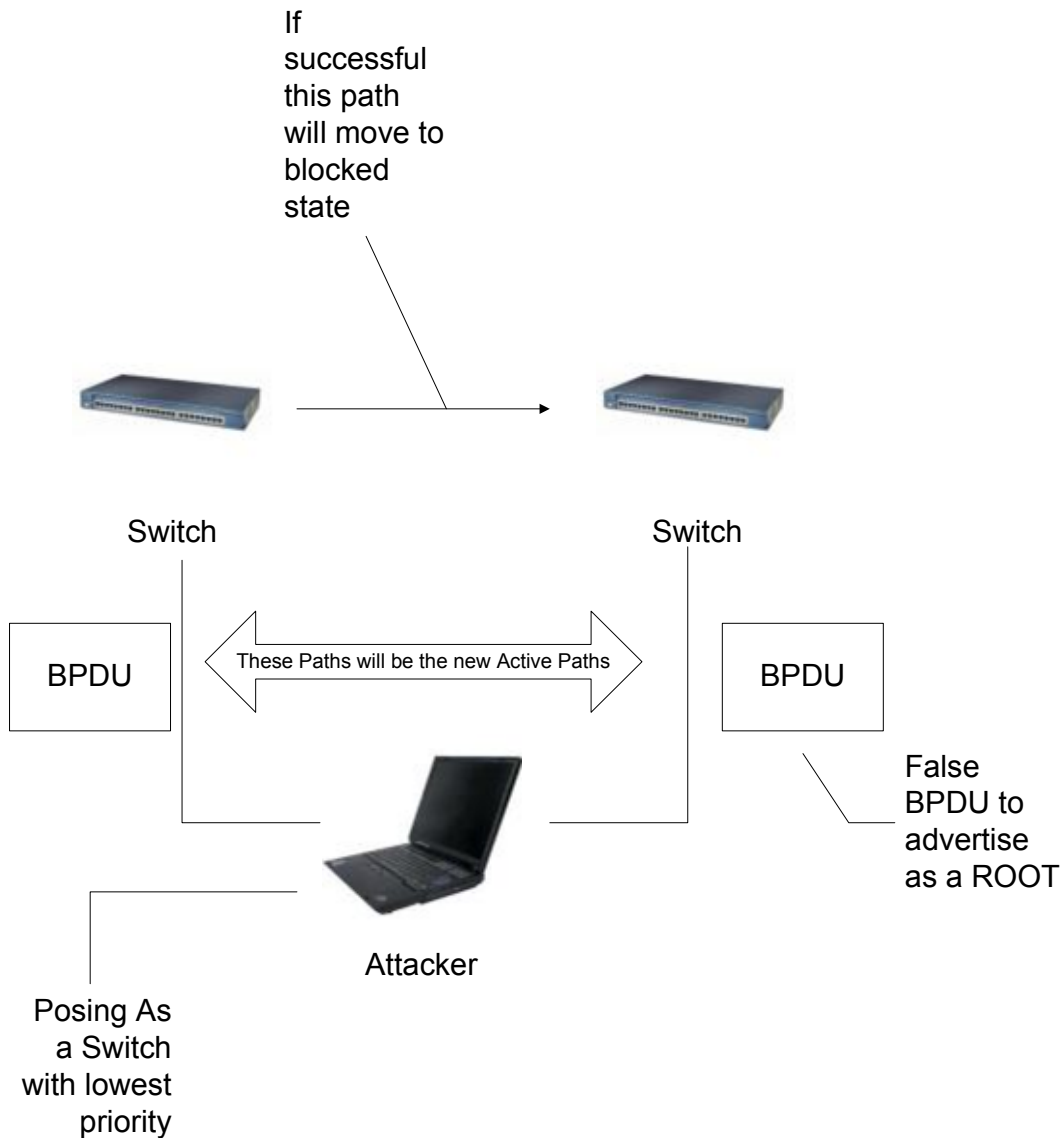


Fig 13. STP based Attack

Mitigation

To mitigate Spanning-Tree Protocol manipulation use the root guard and the BPDU guard enhancement commands to enforce the placement of the root bridge in the network as well as enforce the Spanning-Tree Protocol domain borders.

Attack 5 - DHCP Starvation Attack

In this attack Attacker sends a large number of DHCP requests to the DHCP server, can be accomplished by tools like globber, this will exhaust the total allocatable address space of the server and legitimate requests will go unanswered.

Mitigation

It's easy to mitigate this attack, by configuring the maximum number of MAC entries allowed on a switch port.

Epilogue

These attacks are reality though they might need an attacker to be present on the LAN but considering the fact that most of the breaches are from employees and a wide availability of tools like dsniff and ettercap these attacks should be taken with utmost seriousness.

None of the above attacks demanded a very high skill level from the attacker so the effort to output ratio is extremely significant.

References

1. Raghu Chinthoju – Network Specialist, Google.
2. www.cisco.com
3. www.securityfocus.com
4. www.monkey.org/~dugsong/dsniff/
5. A Middleware Approach to Asynchronous and Backward Compatible Detection and Prevention of ARP Cache Poisoning by Mahesh V. Tripunitara and Partha Dutta.
6. phrack