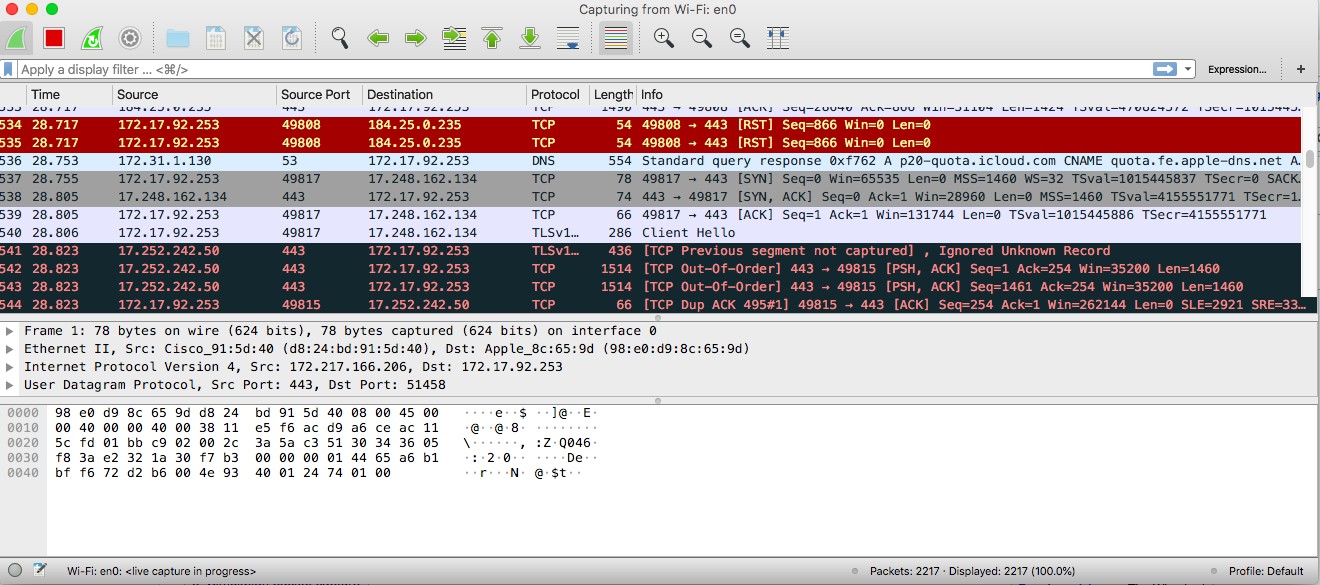
DETECTION OF MALICIOUS TRAFFIC USING PYTHON AND WIRESHARK

# WIRESHARK

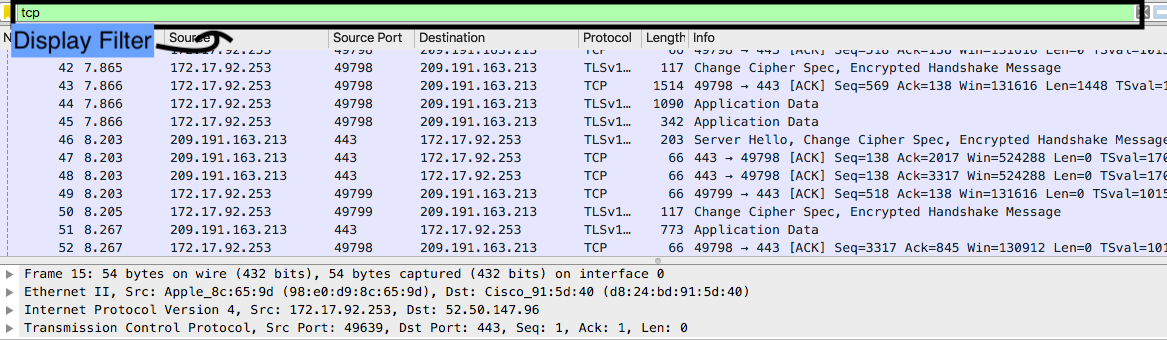
**Wireshark** is a [free](https://en.wikipedia.org/wiki/Free_software) and [open-source](https://en.wikipedia.org/wiki/Open-source_software) [packet analyzer](https://en.wikipedia.org/wiki/Packet_analyzer). It is used for [network](https://en.wikipedia.org/wiki/Computer_network) troubleshooting, analysis, software and [communications protocol](https://en.wikipedia.org/wiki/Communications_protocol) development, and education.

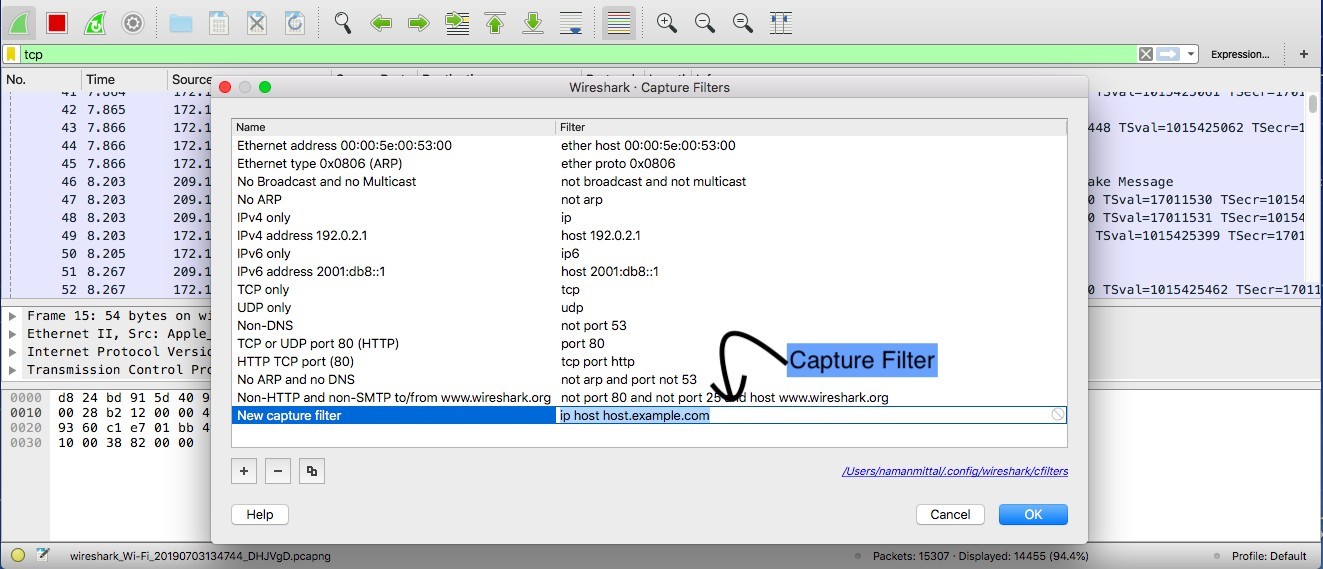
Wireshark is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), using the [Qt](https://en.wikipedia.org/wiki/Qt_(software)) [widget toolkit](https://en.wikipedia.org/wiki/Widget_toolkit) in current releases to implement its user interface, and using [pcap](https://en.wikipedia.org/wiki/Pcap) to capture packets; it runs on [Linux](https://en.wikipedia.org/wiki/Linux), [macOS](https://en.wikipedia.org/wiki/MacOS), [BSD](https://en.wikipedia.org/wiki/BSD), [Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system)), some other [Unix-like](https://en.wikipedia.org/wiki/Unix-like) operating systems, and [Microsoft](https://en.wikipedia.org/wiki/Microsoft_Windows) [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows). There is also a terminal-based (non-GUI) version called TShark.

Wireshark lets the user put [network interface controllers](https://en.wikipedia.org/wiki/Network_interface_controller) into [promiscuous mode](https://en.wikipedia.org/wiki/Promiscuous_mode) (if supported by the [network interface controller](https://en.wikipedia.org/wiki/Network_interface_controller)), so they can see all the traffic visible on that interface including unicast traffic not sent to that network interface controller's [MAC address](https://en.wikipedia.org/wiki/MAC_address).



Wiresharks lets the user filter out the packets captured for analysing, it has two filters, a display filter, which filters the packets captured and shows the relevant ones and a capture filter, which only catches the packets that are relevant according to the filter. Both the filters work differently and have a different syntax.





Wireshark can color packets based on rules that match particular fields in packets, to help the user identify the types of traffic at a glance. A default set of rules is provided; users can change existing rules for coloring packets, add new rules, or remove rules

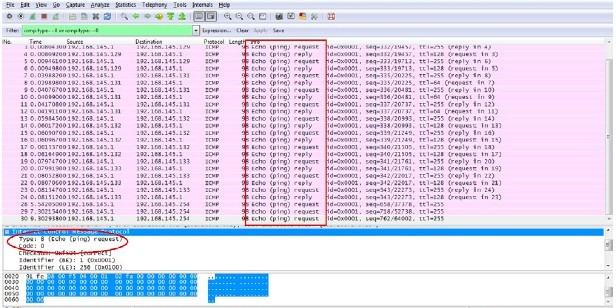
# Ping Sweep

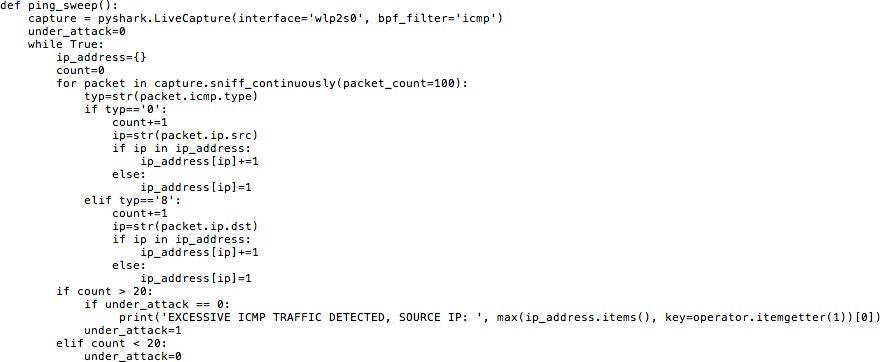
This scan is helpful to find out which IPs are active in the network. Ping Sweep can be performed using ICMP, TCP or UDP, the most popular one is ICMP Ping Sweep. In this ICMP type 8, ECHO request is followed by ICMP type 0, ECHO reply packets are being used while in TCP/UDP ping sweep packets are destined to TCP/UDP port 7, The ECHO port.

If that target host doesn’t support ECHO service then this TCP/UDP ping sweep will not work. Thus ICMP ping sweep is mostly used, but if there is a firewall in between which is configured to block ICMP packet then even ICMP ping sweep is useless. In this situation, ARP scan/ARP sweep can be used.

To detect ICMP ping sweep in Wireshark apply simple filter icmp.type==8 or icmp.type==0. TCP ping sweep can be detected with tcp.dstport==7 filter and for UDP ping sweep udp.dstport==7 filter can be used. After applying these filters if we are getting more than expected packets then it’s possible that ping sweep is going on in our network.

We need to be careful about the volume of such traffic as it might be normal ping traffic. It should be considered as a scan signature only if you are getting unexpected increase in ICMP traffic.



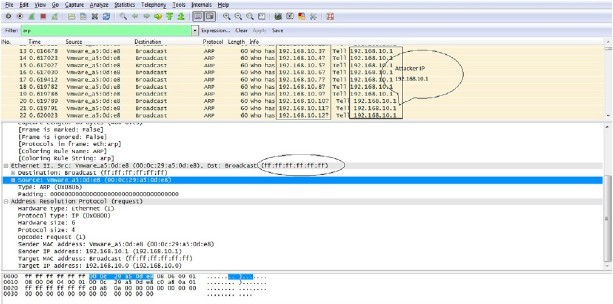
Python code:

# ARP Sweep/ARP Scan

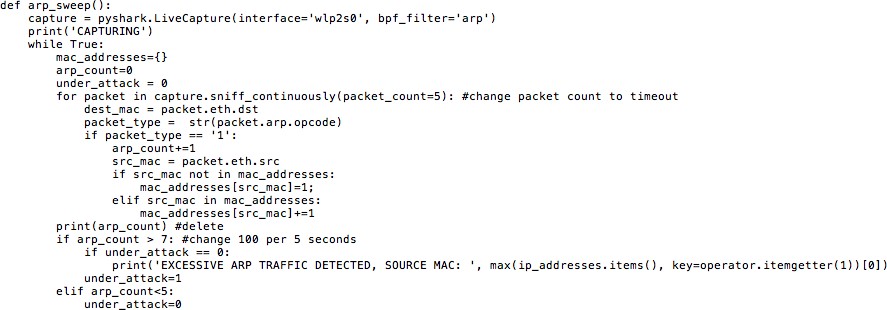
If a firewall is implemented in between and ICMP is blocked then we can’t use ICMP ping sweep. In such a situation, ARP scan is helpful to find out active IPs in the network.

Here, attacker sends ARP broadcast (for broadcast, destination MAC will be 0xff:ff:ff:ff:ff:ff) for each and every possible IP in selected subnet and if he gets ARP response then it shows that IP is active.

Advantage of this scan is that ARP communication can’t be filtered or disabled because all TCP/IP communication is based on it. Blocking or disabling ARP communication will break TCP/IP communication or it will force static ARP entries and disadvantage of this scan is that it can’t cross layer 3 Devices. This scan can be easily detected with filter ARP. After applying this filter if we are getting unexpected no. of ARP queries as shown in the picture, it is a sign for ARP scan or ARP sweep



Python code:

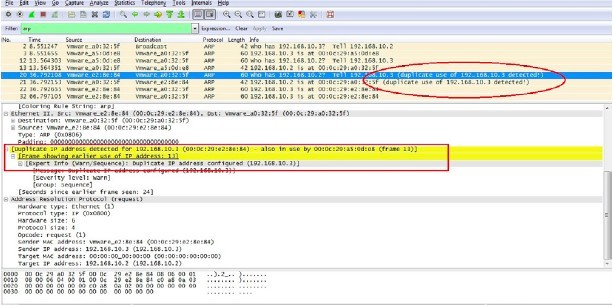


# ARP Poisoning

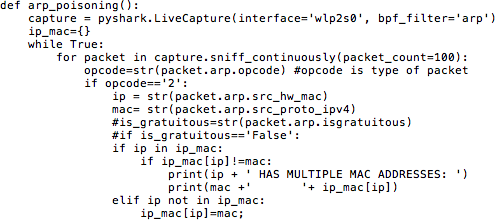
Address Resolution Protocol poisoning (ARP poisoning) is a form of attack in which an attacker changes the Media Access Control (MAC) address and attacks an Ethernet LAN by changing the target computer's ARP cache with a forged ARP request and reply packets.

This modifies the layer -Ethernet MAC address into the hacker's known MAC address to monitor it. Because the ARP replies are forged, the target computer unintentionally sends the frames to the hacker's computer first instead of sending it to the original destination. As a result, both the user's data and privacy are compromised. An effective ARP poisoning attempt is undetectable to the user.

ARP poisoning is a layer 2 redirection technique which can be easily identified by Wireshark. If more than one MAC addresses claim to have the same IP address it will highlight that packet as Duplicate IP Address Detected.



Python code:



# SYN Flooding

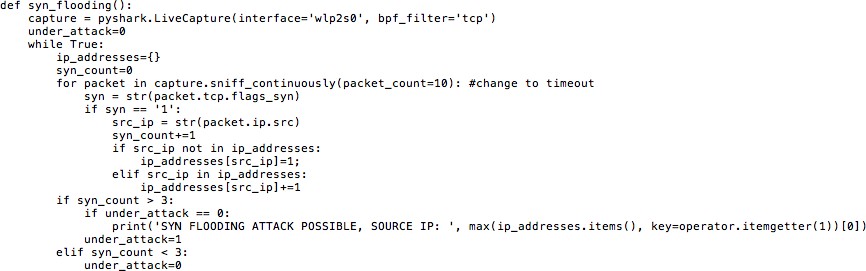
A SYN flood (half-open attack) is a type of denial of service (DOS) which aims to make a server unavailable to legitimate traffic by consuming all available server resources.

By repeatedly sending initial connection request (TCP-SYN) packets, the attacker is able to overwhelm all available ports on a targeted server machine, causing the targeted device to respond to legitimate traffic sluggishly or not at all. In a SYN flood attack, the attacker sends repeated SYN packets to every port on the targeted server, often using a fake IP address. The server, unaware of the attack, receives multiple, apparently legitimate requests to establish communication. It responds to each attempt with a SYN-ACK packet from each open port.

The malicious client either does not send the expected ACK, or—if the IP address is spoofed—never receives the SYN-ACK in the first place. Either way, the server under attack will wait for acknowledgement of its SYN-ACK packet for some time.

After filtering TCP SYN packets, if we are getting unnexected amount of traffic, it may be a sign of SYN flooding attack.

Python Code:

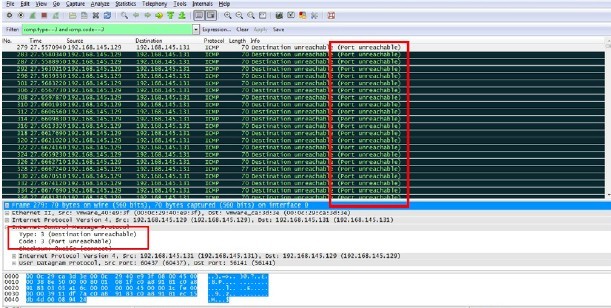


Output:

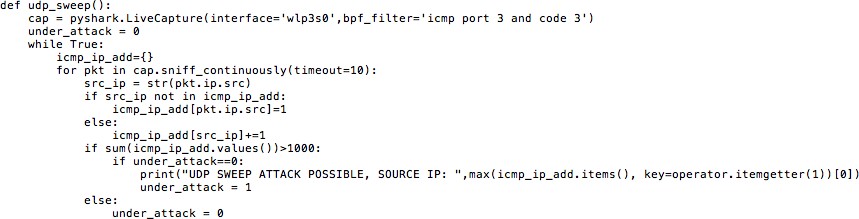


# UDP Scan

In UDP scan attacker sends a UDP packet (contains no meaningful data) on the target port and if that target responds with ICMP Type 3 Code 3 port is unavailable but if there is no response then it might be open or filtered. After capturing packets in Wireshark if you are getting high no. of packets with ICMP type 3 Code 3, it is a sign of UDP Scan. We can use filter icmp.type==3 and icmp.code==3 to detect UDP scan in Wireshark.



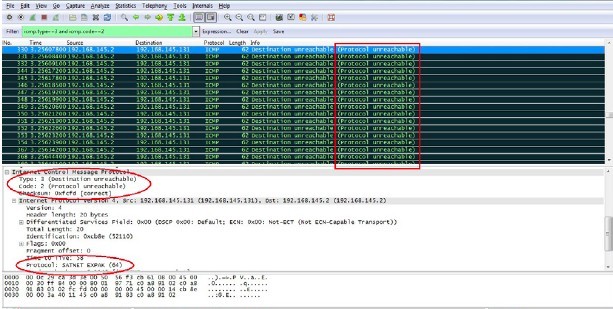
Python Code:-



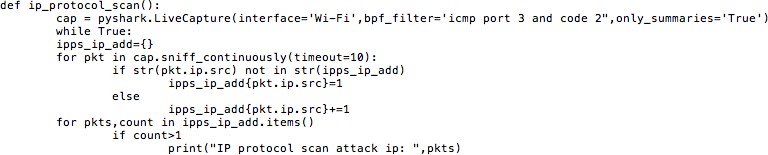
IP Protocol Scan

IP Protocol Scan is helpful in finding out protocols running over IP. To detect this attacker sends packet with different protocol nos., if he gets ICMP type 3 Code 2 Packet as a response then it means that this

protocol is not running on the target system while no response means protocol is there or filtered. To detect this scan in Wireshark, we can apply icmp. type==3 and icmp.code==2 as a filter.

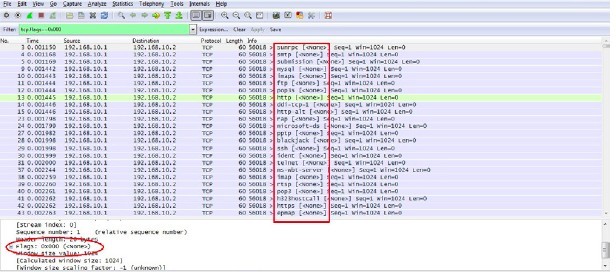


Python Code:-



Null Scan

In this scan attacker sends a TCP packet without setting any flag on it and as a response if he is getting RST packet it means the port is closed. There will be no response to null scan if the port is open or filtered and if he is getting ICMP Type 3 Code 1,2,3,9,10 or 13 packet then port seems to be firewalled.To detect Null Scan in Wireshark, we can use a simple filter TCP.flags==0x000. It will filter all TCP packets moving without Flag.



Python Code:-

