

Progetto d'esame per il corso Reti di calcolatori

Alessandro Pistola

Novembre 2020

Sviluppo di un software per la rilevazione di host e dispositivi di rete basato sui protocolli UDP e LLDP e di un software per attacchi di tipo "ARP cache poisoning".

Indice contenuti

1	Introduzione	3
2	Host e device discovery (UDP e LLDP)	4
2.1	Introduzione	4
2.2	Protocollo ICMP	4
2.3	Approccio scelto	4
2.4	Perchè UDP?	5
2.5	Problematiche	5
2.6	Protocollo LLDP	5
2.7	Analisi network_scanner.py	6
2.8	Test d'uso	7
3	ARP cache poisoning	9
3.1	Introduzione	9
3.2	Protocollo ARP	9
3.3	Attacco ARP cache poisoning	10
3.4	Scelte progettuali	10
3.5	Analisi arp_poisoning.py	11
3.6	Test d'uso	12
4	Conclusione	14

1 Introduzione

Il progetto risulta essere diviso principalmente in due aree. La prima riguarda la parte progettuale comprendente gli strumenti di rilevazione degli host e dei dispositivi in una rete, mentre, la seconda parte fa riferimento ad una specifica tipologia di attacco, il cosiddetto avvelenamento della cache ARP.

Per lo sviluppo del software si è scelto di utilizzare *python* essendo un linguaggio di scripting molto versatile, ben documentato e con una grande quantità di librerie da poter utilizzare.

Nello sviluppo dello script riguardante l'attacco di tipo ARP cache poisoning ci si avvale della libreria *Scapy* essendo il focus dello script nell'attacco piuttosto che nella sua implementazione, mentre, per lo sviluppo dello script per la rilevazione di host e dispositivi di rete si è voluto mantenere il focus nell'implementazione e proprio per questo non viene utilizzata nessuna libreria ed ogni livello (di interesse) viene quindi destrutturato per ottenere le informazioni ricercate.

2 Host e device discovery (UDP e LLDP)

2.1 Introduzione

La mappatura della rete è una delle prime azioni che intraprende un attaccante (malevolo) al fine di ottenere informazioni sulla topologia della rete stessa, dettagli sugli host e sui dispositivi di rete che la compongono e infine cercare dispositivi vulnerabili da attaccare.

Esistono vari approcci e vari tool (uno dei più famosi è *nmap*), più o meno aggressivi ed efficaci.

Si è deciso di sfruttare un noto comportamento di molti sistemi operativi quando devono gestire delle porte UDP chiuse per determinare se un host è attivo e raggiungibile ad uno specifico indirizzo IP.

Per capire a pieno il metodo utilizzato è necessario prima introdurre il protocollo ICMP.

2.2 Protocollo ICMP

Il protocollo ICMP (*Internet control message protocol*, RFC792) viene usato da host e router per scambiarsi informazioni a livello di rete, uno degli utilizzi più frequenti è la notifica degli errori.

I messaggi ICMP vengono trasportati come payload di IP rendendolo dal punto di vista dell'architettura esattamente sopra quest'ultimo (come UDP e TCP), inoltre hanno un campo tipo e un campo codice e contengono l'intestazione e i primi 8 byte del datagramma IP che ha provocato la generazione del messaggio, in modo che il mittente possa determinare il datagramma che ha causato l'errore.

	Bit 0-7	Bit 8-15	Bit 16-23	Bit 24-31
0	Tipo	Codice	Checksum	
32	Dati dell'header			

Figure 1: Struttura di pacchetto ICMP

2.3 Approccio scelto

Come brevemente spiegato nell'introduzione, quando si invia un pacchetto UDP verso una porta chiusa in un host, questo generalmente risponde inviando un pacchetto ICMP indicando che la porta specificata è irraggiungibile.

Tuttavia, tale pacchetto ICMP sta ad indicare che vi è un host attivo poichè, nel caso non vi fosse nessun host attivo, non dovremmo ricevere nessuna risposta al pacchetto UDP inviato.

L'essenza di questo tipo di scanner risiede nello scegliere una porta che nella maggior parte dei casi non viene utilizzata. Per ottenere una maggiore copertura, è possibile provare ad inviare il pacchetto UDP su più porte di un host per essere sicuri di non star inviando pacchetti a porte UDP attive ed in uso.

2.4 Perché UDP?

Il livello di trasporto di Internet trasferisce i messaggi del livello di applicazione tra i punti periferici gestiti dalle applicazioni. I due protocolli più utilizzati sono UDP e TCP.

Al contrario del protocollo TCP, il protocollo UDP fornisce alle proprie applicazioni un servizio minimale, non orientato alla connessione, senza affidabilità, nè controllo di flusso e della congestione. Nella pratica UDP prende i messaggi del processo applicativo, aggiunge il numero di porta di origine e di destinazione per il multiplexing/demultiplexing, aggiunge altri due piccoli campi (lunghezza e checksum) e passa il segmento risultante al livello di rete.

Nello specifico le seguenti caratteristiche rendono UDP perfetto per la rilevazione di host nella rete:

- Invio quasi istantaneo del segmento al livello di rete (nessun ritardo dovuto al controllo della congestione);
- Non viene introdotto nessun ritardo nello stabilire una connessione (3-way handshake su TCP);
- Nessuno stato di connessione da mantenere;
- Minor spazio utilizzato per l'intestazione, TCP aggiunge 20 byte mentre UDP solo 8.

In conclusione, l'utilizzo di UDP non sovraccarica la rete.

2.5 Problematiche

Allo stato attuale, risulterà evidente che lo scanner non è in grado di rilevare tutti i dispositivi nella rete, infatti tutti i dispositivi che agiscono al di sotto del livello di rete (es. switch) non verranno rilevati.

Per ampliare la copertura dello scanner è stato implementato un rilevatore di pacchetti LLDP, che scambia frame a livello di collegamento.

2.6 Protocollo LLDP

Il protocollo LLDP (*Link Layer Discovery Protocol*) è un protocollo al livello di collegamento, neutrale, sviluppato dalla comunità informatica, è utilizzato dai dispositivi di rete per informare gli host e gli apparati di rete vicini (all'interno

di una rete LAN generica *IEEE 802.x*) la loro identità e le loro capacità. Formalmente, il protocollo è standardizzato all'interno del documento *IEEE 802.1AB, Station and Media Access Control Connectivity Discovery* e le sue proprietà sono in parte conformi ad altri protocolli proprietari come CDP (Cisco Discovery Protocol), FDP (Foundry Discovery Protocol). Le informazioni scambiate attraverso questo protocollo comprendono:

- Nome e descrizione del sistema
- Interfaccia di collegamento (nome e indirizzo MAC)
- Indirizzo IP
- Funzioni del sistema come: Bridge, Router, Wlan, Station, Power over Ethernet

Queste informazioni sono spedite da un device da ognuna delle sue interfacce ad uno specifico intervallo di tempo sottoforma di frame Ethernet. Ogni frame contiene un LLDP Data Unit (LLDPDU), il quale è una sequenza di TLV (type-length-value).

I frame ethernet hanno uno speciale indirizzo MAC di destinazione multicast (01:80:c2:00:00:0e) cosicchè i bridge che rispettano lo standard 802.1D possano non inoltrare il frame. Il campo EtherType è fisso a 0x88c.

Ogni frame LLDP inizia con i seguenti TLV obbligatori: *Chassis ID* (*Type=1*), *Port ID* (*Type=2*), *Time-to-Live* (*Type=3*), tali TLV sono seguiti da un numero arbitrario di TLV opzionali impostati. Il frame termina con uno speciale TLV con *Type=0* e *Lenght=0*.

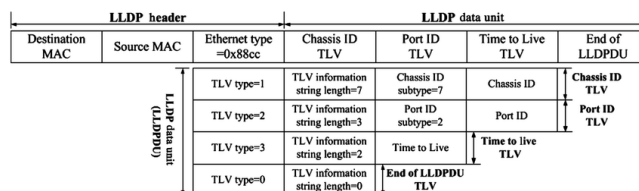


Figure 2: Struttura di un frame ethernet LLDP

2.7 Analisi network_scanner.py

Network_scanner.py è lo script in cui sono stati implementati i metodi di discovery sopra riportati. All'inizio dello script, sono inizializzati "hard-coded" l'host da cui si sta "sniffando" i pacchetti in rete, la sua sottorete di azione e la variabile signature. Tale valore viene inviato nel pacchetto UDP originario cosicchè si possano validare i pacchetti ICMP reply accedendo al loro payload e verificando che siano pacchetti di risposta generati dal pacchetto UDP inviato

dallo script.

E' possibile specificare la sottorete nel formato *192.168.1.0/24* grazie all'utilizzo della libreria python *netaddr* i cui moduli *IPNetwork* e *IPAddress* sono sfruttati per calcolare tutti i possibili indirizzi IP della sottorete.

Procedendo alla funzione *main*, in ordine:

1. Viene avviato il thread *udp_sender* utilizzato per inviare a tutti gli indirizzi IP della sottorete un datagramma UDP alla porta 65212 contenente la signature. Viene utilizzato un thread diverso per non interferire con le operazioni dello script che si dovrà poi mettere in ascolto sulla rete locale in attesa dei pacchetti ICMP e LLDP.
2. Viene creata una socket con parametri: *AddressFamily = AF_PACKET* (specifica l'utilizzo di un'interfaccia per ottenere pacchetti a basso livello), *SocketKind = RAW_SOCKET* (specifica la creazione di una socket RAW ovvero non orientata ad un particolare servizio), *PacketProtocol=0x0003* (specifica di catturare tutti i pacchetti).
3. All'interno di un ciclo while, si ricevono i pacchetti che vengono analizzati, finché lo script non viene interrotto dall'utente.

All'interno del ciclo while si procede alla decomposizione di ogni pacchetto. Si estrae l'header del frame Ethernet (*ethernet_head()*) e se il campo *EtherType* è uguale a 0x0800 allora il frame contiene un pacchetto IPv4 come payload, se invece è uguale a 0x88CC allora sta trasportando un pacchetto LLDP. Nel caso fosse un pacchetto IPv4, si procede estraendone l'header (*ipv4_head()*), controllando che il campo *prototype* sia uguale ad 1, ovvero stia trasportando un messaggio ICMP al suo interno.

A questo punto si procede ad estrarre l'header del pacchetto ICMP, verificando che il campo *type* ed il campo *code* siano uguali a 3, in quanto, tutti i messaggi di tipo 3 sono della famiglia *Destination unreachable* (destinazione non raggiungibile) e il codice 3 identifica il messaggio *Port unreachable* (porta non raggiungibile) e che la signature corrisponda, se così fosse, l'host viene aggiunto alla lista degli host attivi nella rete locale.

Nel caso il frame Ethernet sia di tipo 0x88CC si procede estraendo solamente l'header del pacchetto LLDP contenente la serie di TLV (essendo un protocollo che agisce a livello di collegamento) (*lldp_head()*). Tutte le funzioni di decomposizione di ogni specifico header si basano sulla struttura standard dei pacchetti, segmenti e frame così come descritta negli standard IEEE.

2.8 Test d'uso

Il test è svolto attraverso una macchina virtuale Ubuntu all'interno di una rete locale domestica a cui sono collegati più sistemi periferici.

Al fine di verificare la corretta rilevazione dei pacchetti LLDP, all'interno di un'ulteriore macchina virtuale che esegue un'istanza del sistema operativo Kali

Linux è stato attivato il daemon *lldpd*. Dopo averlo scaricato, per attivarlo è sufficiente avviare il servizio attraverso il comando:

```
sudo systemctl start lldpd
```

Il sistema linux ogni x secondi invierà un pacchetto lldp (configurabile) contenente le informazioni di sistema.

Avviando lo script `network_scanner.py` sulla macchina Ubuntu si ottiene tale output:

```
apistola@apistola-VirtualBox:~/Desktop/Reti$ sudo python3 network_scanner.py
[sudo] password for apistola:
[!] Starting the sender thread
[!] Now sniffing on interface.. [CTRL-C to stop]
[+] Host Up: 192.168.1.1 - 20:b0:01:c8:da:28 (ICMP reply)
[+] Host Up: 192.168.1.38 - 08:00:27:a1:88:25 (ICMP reply)
[+] Host Up: 192.168.1.176 - 22:b0:01:c8:da:31 (ICMP reply)
[+] Host Up: 192.168.1.223 - fc:3f:7c:c1:42:4a (ICMP reply)
[+] Device found (LLDP):
    ChassisID: network address 08:00:27:a1:88:25
    PortID: mac address 08:00:27:a1:88:25
    TTL: 120
    Sys name: kali-ap
    Sys Description: Kali GNU/Linux Rolling Linux 5.6.0-kali1-amd64 #1 SMP Debian 5.6.7-1kali1 (2020-05-12) x86_64
    Sys Capability: {'Bridge': 'off', 'Wlan AP': 'off', 'Router': 'off', 'Station': 'on'}
    Mgmt Addr: 192.168.1.38
    Port Description: eth0
[+] Device found (LLDP):
    ChassisID: network address 08:00:27:b4:31:21
    PortID: mac address 08:00:27:b4:31:21
    TTL: 120
    Sys name: apistola-VirtualBox
    Sys Description: Ubuntu 18.04.4 LTS Linux 5.3.0-62-generic #56~18.04.1-Ubuntu SMP Wed Jun 24 16:17:03 UTC 2020 x86_64
    Sys Capability: {'Bridge': 'off', 'Wlan AP': 'off', 'Router': 'off', 'Station': 'on'}
    Mgmt Addr: 192.168.1.153
    Port Description: enp0s3
^C
Quitting..
```

Figure 3: output dello script

3 ARP cache poisoning

3.1 Introduzione

I computer all'interno di una rete LAN comunicano tra di loro usufruendo delle schede di rete collegate agli switch, agli hub o direttamente al router.

Lo switch è un dispositivo in grado di instradare i pacchetti di dati in modo logico, ovvero distribuisce al singolo host i soli dati che ha richiesto (a livello di protocolli), evitando di fatto un invio su tutte le macchine (broadcast) come invece fa un hub.

E' chiaro che analizzando il traffico di un host collegato ad un hub, a differenza dello switch, è possibile rilevare dati sensibili destinati ad altri host della rete.

Per aggirare l'instradamento degli switch viene utilizzata una tipologia di attacco "man-in-the-middle", l'attacco ARP Cache Poisoning, utilizzato per redirigere o intercettare il traffico tra due computer/dispositivi all'interno di una rete LAN.

3.2 Protocollo ARP

Il compito di convertire gli indirizzi del livello di rete (come gli indirizzi IP di Internet) in indirizzi del livello di collegamento (indirizzi MAC) è affidato al protocollo di risoluzione degli indirizzi (ARP, *address resolution protocol* [RFC826]). Un modulo ARP in un nodo sorgente riceve in input un indirizzo IP della stessa LAN e restituisce l'indirizzo MAC corrispondente. Un modulo ARP, quindi, traduce un indirizzo IP in un indirizzo MAC.

Per molti aspetti è simile al DNS, che traduce i nomi degli host in indirizzi IP. Tuttavia, un'importante differenza risiede nel fatto che DNS esegue l'operazione per host localizzati in qualunque punto di Internet, mentre ARP risolve soltanto gli indirizzi IP per i nodi nella stessa sottorete.

Nella RAM dei nodi vi è una tabella ARP che contiene la corrispondenza tra gli indirizzi IP e MAC. La tabella solitamente contiene anche un valore TTL (*Time To Live*) che indica quando bisognerà eliminare una data voce dalla tabella.

Quando un nodo x vuole comunicare con un nodo y ma non è a conoscenza del suo indirizzo MAC bensì del solo indirizzo IP, il nodo trasmittente x costruisce uno speciale pacchetto, chiamato pacchetto ARP di richiesta (con formato identico a quello di risposta). Lo scopo di un pacchetto ARP di richiesta è interrogare tutti gli altri nodi della sottorete riguardo l'indirizzo MAC corrispondente all'indirizzo IP da risolvere. Ovviamente tale pacchetto è inviato in broadcast all'indirizzo FF:FF:FF:FF:FF:FF.

A questo punto, il frame contenente la richiesta ARP è ricevuto da tutte le schede di rete, ciascuna delle quali trasferisce il pacchetto ARP al proprio nodo che controlla se il proprio indirizzo IP corrisponde a quello di destinazione indicato nel pacchetto ARP. L'unico nodo che ha l'indirizzo corrispondente invia al nodo richiedente x un frame di risposta ARP con la corrispondenza desiderata.

3.3 Attacco ARP cache poisoning

L'avvelenamento o meglio ancora modifica della tabella ARP si basa sulla natura insicura del protocollo ARP: i dispositivi che utilizzano ARP accettano risposte in qualsiasi momento (anche se non è stata generata precedentemente una richiesta specifica) e questo significa che qualsiasi dispositivo può inviare delle risposte ARP ad altri dispositivi presenti in rete con l'intenzione di modificarli la tabella ARP a proprio piacimento, in particolare in modo tale da poter ascoltare il traffico passante sulla rete.

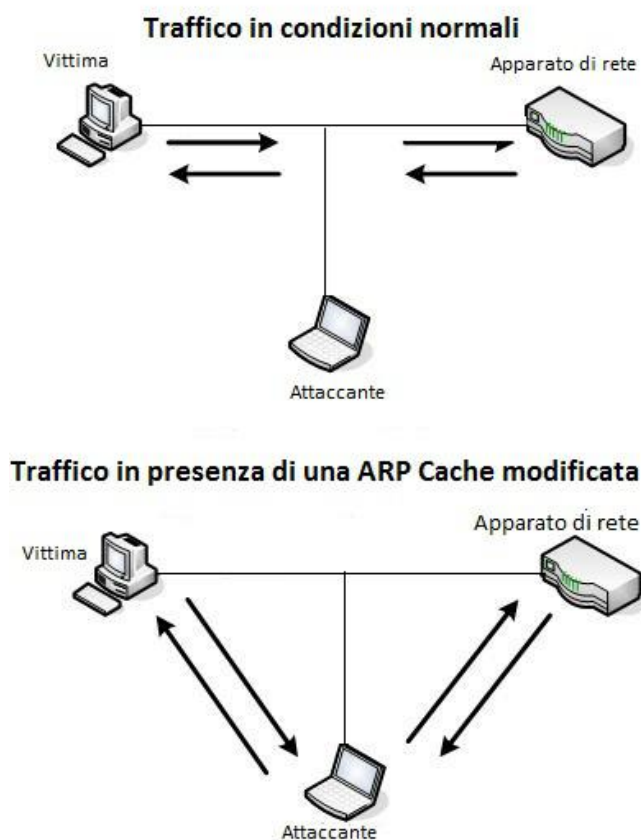


Figure 4: Teoria dell'attacco ARP cache poisoning

3.4 Scelte progettuali

Essendo il focus dello script sul protocollo ARP e il suo attacco tramite l'avvelenamento della tabella ARP, in fase di implementazione si è scelto di utilizzare la libreria

python Scapy.

Scapy è una libreria popolare creata per la manipolazione dei pacchetti di rete. Con scapy è possibile assemblare o decodificare i pacchetti di un ampio numero di protocolli, inviarli, catturarli e filtrare richieste e risposte.

Nello script `arp_poisoning.py` scapy è utilizzato per inviare e ricevere pacchetti ARP costruiti ad-hoc, effettuare la cattura di pacchetti e salvare tale cattura in un file con estensione `pcap` così da poter effettuare analizzare con cura i pacchetti catturati.

3.5 Analisi `arp_poisoning.py`

Nelle prime righe dello script, come per il precedente, si trovano le inizializzazioni dove vanno specificati gli indirizzi IP della vittima e del gateway, l'interfaccia di rete da utilizzare e il numero di pacchetti da catturare durante l'attacco. All'interno della funzione `main`, prima di tutto, si procede all'ottenimento degli indirizzi MAC del gateway e della vittima. Questo avviene sfruttando lo stesso protocollo ARP, infatti la funzione `get_mac(ip_addr)` non fa altro che inviare un pacchetto di richiesta ARP a tutte le schede di rete nella LAN.

```
def get_mac(ip_address):
    resp, unans = sr(ARP(op=1, hwdst="ff:ff:ff:ff:ff:ff",
        pdst=ip_address), retry=2, timeout=10)

    # return the MAC address from a response
    for s,r in resp:
        return r[ARP].hwsrc
    return None
```

La funzione `sr` (send and receive) di Scapy permette di inviare il pacchetto specificato (che in questo caso è di tipo ARP) e attendere eventuali risposte in ingresso.

Ottenuti gli indirizzi MAC, viene avviato il thread responsabile dell'avvelenamento della tabella ARP della vittima mentre lo script si mette in ascolto della rete aspettando di catturare i pacchetti che poi salverà sul file *sniffed.pcap*,

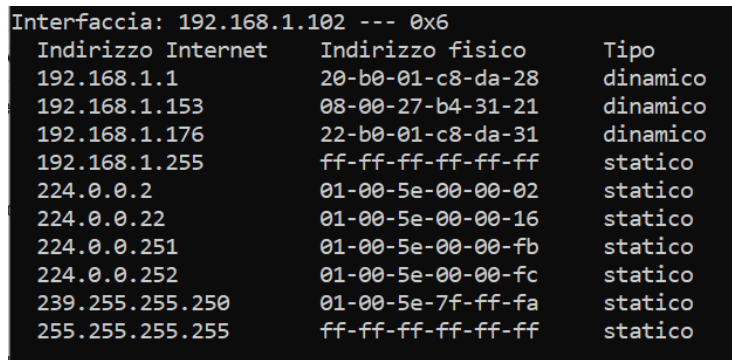
Nota: nel processo di creazione dello sniffer scapy permette di specificare le regole di filtraggio proprio come avviene in Wireshark.

Il thread responsabile dell'avvelenamento manda in ripetizione due pacchetti ARP, uno alla vittima e uno al gateway. Manda alla vittima un pacchetto ARP con destinazione IP e MAC gli indirizzi corretti della vittima ma con indirizzo IP sorgente del gateway, così la vittima assocerà all'indirizzo IP del gateway l'indirizzo MAC dell'attaccante; manda al gateway un pacchetto ARP con destinazione IP e MAC gli indirizzi corretti del gateway ma con indirizzo IP sorgente della vittima così che il gateway assocerà all'indirizzo IP della vittima

l'indirizzo MAC dell'attaccante.

3.6 Test d'uso

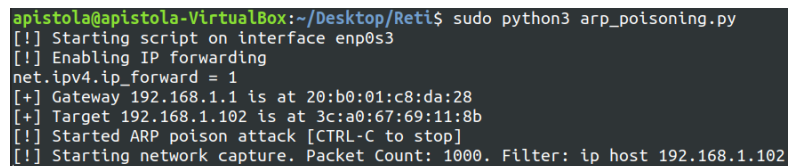
Per testare il corretto funzionamento dello script python implementato nel file *arp_poisoning.py* è stato riprodotto un attacco. L'attaccante che eseguirà lo script utilizza una macchina virtuale Linux mentre la vittima è una macchina Windows. Di seguito è riportato il contenuto della tabella ARP della vittima prima dell'esecuzione dello script (ottenuta attraverso il comando "arp -a"):



Indirizzo Internet	Indirizzo fisico	Tipo
192.168.1.1	20-b0-01-c8-da-28	dinamico
192.168.1.153	08-00-27-b4-31-21	dinamico
192.168.1.176	22-b0-01-c8-da-31	dinamico
192.168.1.255	ff-ff-ff-ff-ff-ff	statico
224.0.0.2	01-00-5e-00-00-02	statico
224.0.0.22	01-00-5e-00-00-16	statico
224.0.0.251	01-00-5e-00-00-fb	statico
224.0.0.252	01-00-5e-00-00-fc	statico
239.255.255.250	01-00-5e-7f-ff-fa	statico
255.255.255.255	ff-ff-ff-ff-ff-ff	statico

Figure 5: Tabella ARP prima dell'attacco

La macchina attaccante è identificata dalla seguente coppia IP-MAC: (192.168.1.153 - 08:00:27:b4:31:21) Il gateway della LAN è identificato dalla seguente coppia IP-MAC: (192.168.1.1 - 20:b0:01:c8:da:28) Lo script in esecuzione genera il seguente output:



```
apistola@apistola-VirtualBox:~/Desktop/Reti$ sudo python3 arp_poisoning.py
[!] Starting script on interface enp0s3
[!] Enabling IP forwarding
net.ipv4.ip_forward = 1
[+] Gateway 192.168.1.1 is at 20:b0:01:c8:da:28
[+] Target 192.168.1.102 is at 3c:a0:67:69:11:8b
[!] Started ARP poison attack [CTRL-C to stop]
[!] Starting network capture. Packet Count: 1000. Filter: ip host 192.168.1.102
```

Figure 6: Output durante l'attacco

Di seguito la tabella ARP durante l'attacco, si può notare, ora, che all'indirizzo IP del gateway è associato l'indirizzo MAC dell'attaccante.

```
Interfaccia: 192.168.1.102 --- 0x6
Indirizzo Internet    Indirizzo fisico    Tipo
192.168.1.1          08-00-27-b4-31-21  dinamico
192.168.1.153        08-00-27-b4-31-21  dinamico
192.168.1.176        22-b0-01-c8-da-31  dinamico
192.168.1.255        ff-ff-ff-ff-ff-ff  statico
224.0.0.2            01-00-5e-00-00-02  statico
224.0.0.22          01-00-5e-00-00-16  statico
224.0.0.251         01-00-5e-00-00-fb  statico
224.0.0.252         01-00-5e-00-00-fc  statico
239.255.255.250     01-00-5e-7f-ff-fa  statico
255.255.255.255     ff-ff-ff-ff-ff-ff  statico
```

Figure 7: Tabella ARP durante l'attacco

4 Conclusione

In conclusione, nel primo script python si è voluta porre l'attenzione nell'analisi tecnica del processo di decomposizione di un pacchetto, dei suoi campi e dei vari standard IEEE con la finalità di effettuare una rilevazione degli host e degli apparati di rete all'interno di una LAN.

Le difficoltà principali risiedono appunto nella decomposizione dei campi dei vari pacchetti e nella decodifica dei valori ma tale processo senza l'ausilio di librerie come Scapy, permette di comprendere a pieno la complessità dei vari livelli e dei vari protocolli.

Nel secondo script python l'attenzione è stata rivolta ad un particolare protocollo che agisce tra il livello di rete e quello di collegamento, cercando di sfruttare il suo comportamento per portare a termine un attacco MITM (man-in-the-middle).

Sebbene l'attacco di tipo ARP cache poisoning sia tutt'ora efficace, la diffusione di switch con funzioni di *port security* mitiga completamente l'attacco, poichè tali switch non permettono indirizzi MAC duplicati sulle stesse porte.

Un'ulteriore difesa non trattata in questo progetto, è senz'altro l'utilizzo del protocollo SARP ovvero Secure ARP, un'estensione del protocollo ARP che si basa sulla crittografia asimmetrica, così da poter autenticare il mittente.