

Progetto finale Benchmark M1 di Pier Francesco Monaco

Inizio il progetto finale del modulo M1 scusandomi per aver consegnato in ritardo e consapevole che questo inciderà sulla votazione finale. Il lavoro che faccio (tecnico dei pos per Nexi e Siapay) è aumentato considerevolmente nelle ultime settimane, per quanto ci stia provando non sto riuscendo a trovare il tempo necessario da dedicare agli esercizi da consegnare. Farò del mio meglio per mettermi in riga e rimettermi in pari con il resto della classe.

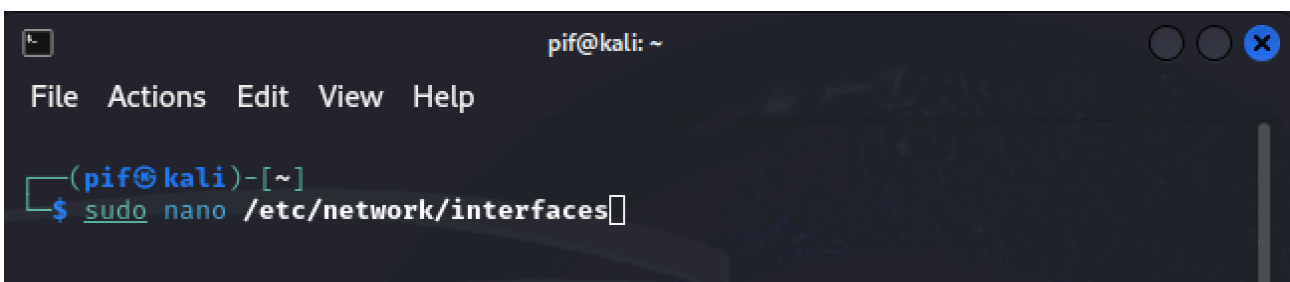
Traccia: Simulare, in ambiente di laboratorio virtuale, un'architettura client server in cui un client con indirizzo 192.168.32.101 (Windows) richiede tramite web browser una risorsa all'hostname epicode.internal che risponde all'indirizzo 192.168.32.100 (Kali). Si intercetti poi la comunicazione con Wireshark, evidenziando i MAC address di sorgente e destinazione ed il contenuto della richiesta HTTPS. Ripetere l'esercizio, sostituendo il server HTTPS, con un server HTTP. Si intercetti nuovamente il traffico, evidenziando le eventuali differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS.

Svolgimento

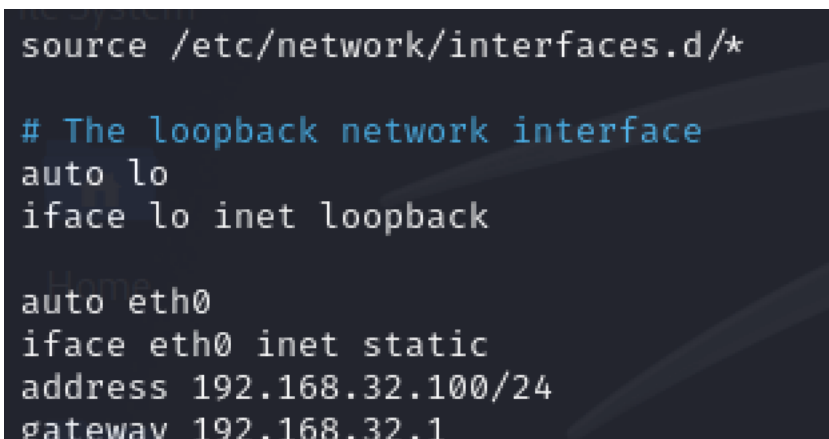
Apro su Utm* le due macchine virtuali necessarie per l'esercizio, cioè Windows 7 e Kali Linux.

Avviate le due macchine andremo a cambiare gli indirizzi IP seguendo quanto richiesto dalla traccia.

Per Kali Linux apro la pagina di comando : e invio il comando “ `sudo nano/etc/network/interfaces` “



Aperta la pagina di configurazione dei network, cambio l'indirizzo IP seguendo quanto richiesto dalla traccia.

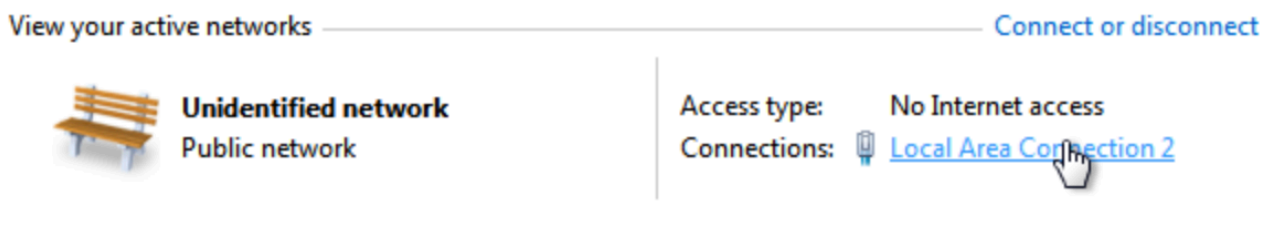


```
source /etc/network/interfaces.d/*

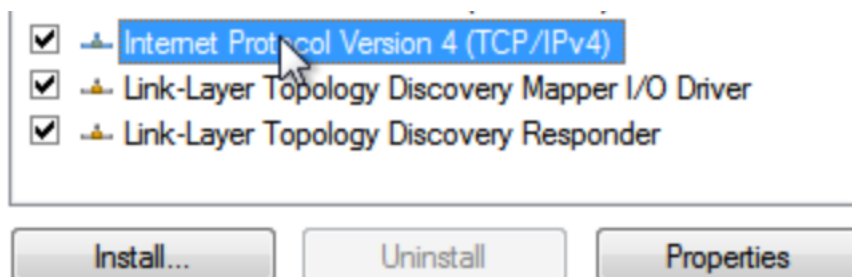
# The loopback network interface
auto lo
iface lo inet loopback

#eth0
auto eth0
iface eth0 inet static
address 192.168.32.100/24
gateway 192.168.32.1
```

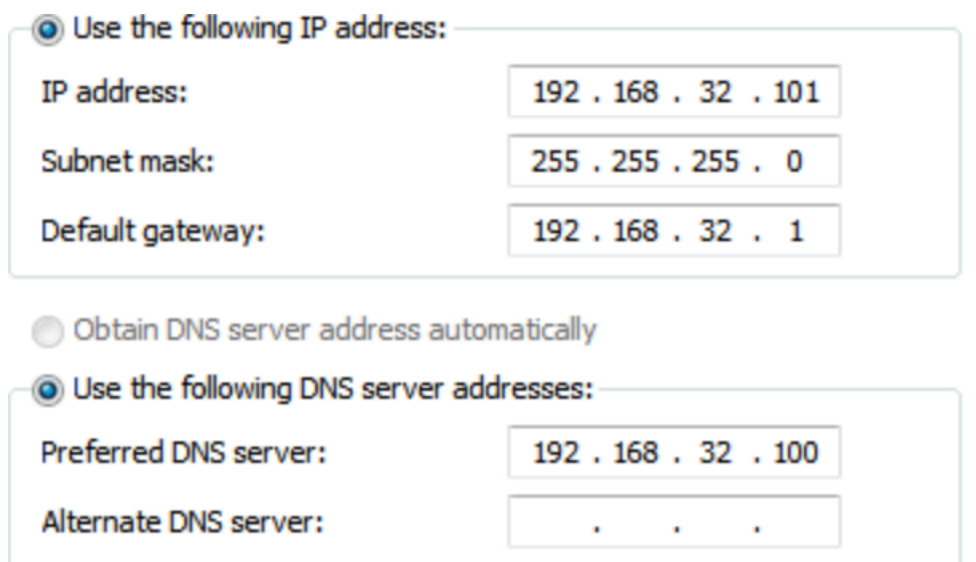
Per la macchina virtuale di Windows invece andremo su Control Panel, Network and Internet, Network and Sharing Center e andiamo a selezionare la connessione attiva, in questo caso Local Area Connection 2



Aperta questa selezioniamo la casella “Properties” e selezionando IPv4 apriamo di nuovo la pagina “Properties “ di questa voce.

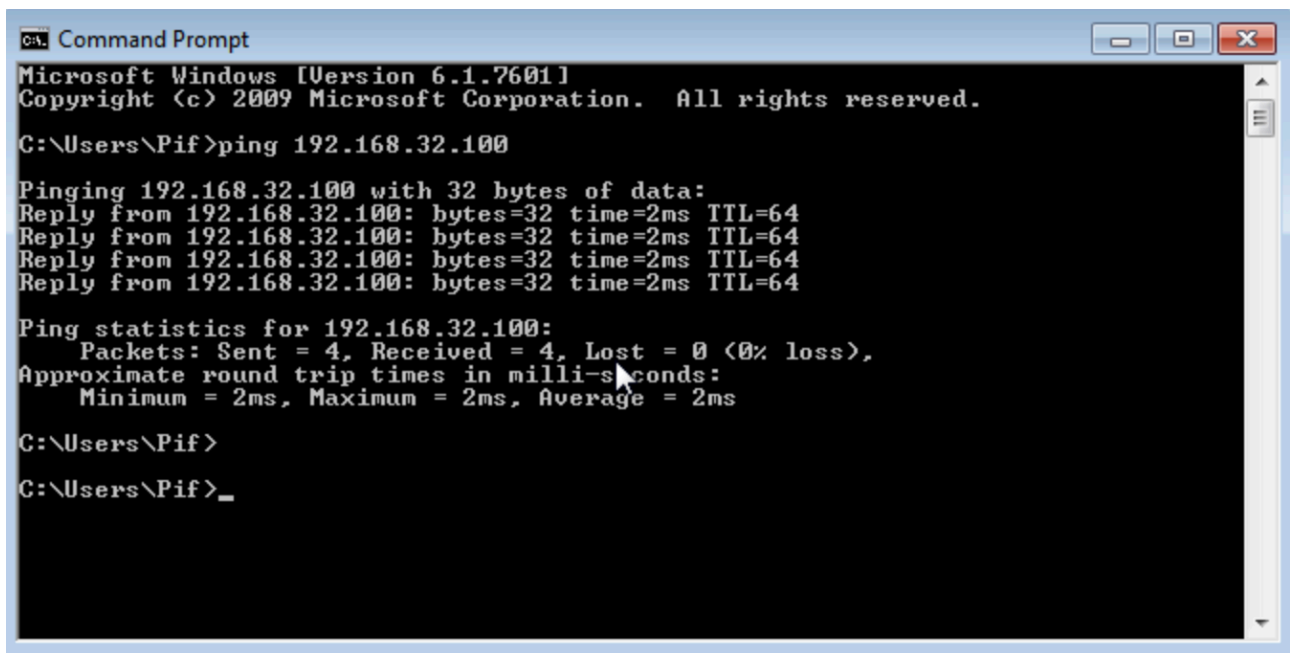


Arrivati a questo punto si possono selezionare l'indirizzo ip e i parametri che a noi interessando per lo svolgimento dell'esercizio.



Per controllare che tra le due macchine ci sia connessione svolgiamo un comando di Ping che altro non è che un pacchetto dati inviato da una macchina all'altra. Se correttamente configurate le macchine, i dati verranno trasmessi correttamente senza perdita di pacchetti.

Per effettuare il ping tra le due macchine apriamo il centro comandi della macchina Windows e digitiamo il comando : “ ping 192.168.32.100 (che corrisponde all’indirizzo Ip che abbiamo configurato sulla macchina virtuale Kali Linux)



```
C:\Users\Pif>ping 192.168.32.100

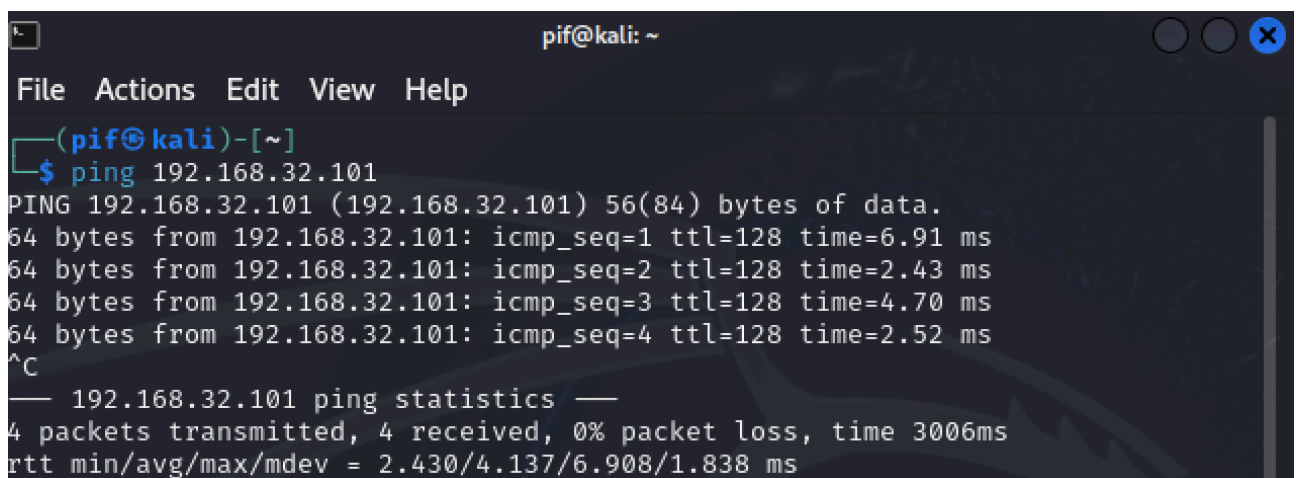
Pinging 192.168.32.100 with 32 bytes of data:
Reply from 192.168.32.100: bytes=32 time=2ms TTL=64
Reply from 192.168.32.100: bytes=32 time=2ms TTL=64
Reply from 192.168.32.100: bytes=32 time=2ms TTL=64
Reply from 192.168.32.100: bytes=32 time=2ms TTL=64

Ping statistics for 192.168.32.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 2ms, Average = 2ms

C:\Users\Pif>
C:\Users\Pif>_
```

Come osservato figura sono stati inviati 4 pacchetti, ricevuti 4 e persi 0, per cui Windows riesce a comunicare con Kali correttamente.

Ora per controllare che anche Kali riesca a comunicare correttamente con Windows apriamo la pagina di comando di Kali e come fatto precedentemente inviamo un comando Ping all’indirizzo Ip su cui abbiamo impostato Windows cioè 192.168.32.101



```
pif@kali: ~
File Actions Edit View Help
(pif@kali)-[~]
$ ping 192.168.32.101
PING 192.168.32.101 (192.168.32.101) 56(84) bytes of data.
64 bytes from 192.168.32.101: icmp_seq=1 ttl=128 time=6.91 ms
64 bytes from 192.168.32.101: icmp_seq=2 ttl=128 time=2.43 ms
64 bytes from 192.168.32.101: icmp_seq=3 ttl=128 time=4.70 ms
64 bytes from 192.168.32.101: icmp_seq=4 ttl=128 time=2.52 ms
^C
— 192.168.32.101 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 2.430/4.137/6.908/1.838 ms
```

Anche in questo caso i pacchetti inviati sono stati 4 e tutti correttamente trasmessi senza perdita di nessun dato, da questo evinciamo che le due macchine sono correttamente impostate e riescono a comunicare tra di loro.

Una volta impostato il nostro DNS vado su Inetsim, un programma nativo di Kali che permette di simulare vari servizi internet, tra cui quelli che a noi servono per l'esercizio cioè HTTP e HTTPS.

Per aprire inetsim su Kali apriamo la pagina di comando e inseriamo il comando `“ sudo nano /etc/inetsim/inetsim.conf “`

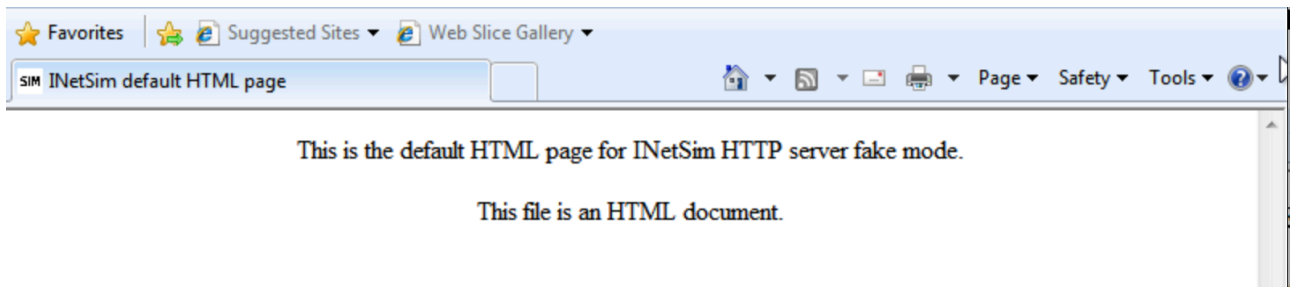
Aperta la pagina dell'applicazione avremo una lista dei servizi che si possono emulare, per permettere a Kali di avviare solo quello che interessa a noi, in questo caso HTTP andiamo a de-commentare la riga relativa al servizio (rimuovere il cancelletto dalla linea iniziale di comando) lasciando tutto il resto delle righe commentare per cui Kali le ignorerà.

```
start_service http
#start_service https
```

L'ultima cosa da fare in Inetsim è modificare un altro parametro, de-commentare la linea `service_bind_address` e modificare l'indirizzo ip in modo che rifletta 192.168.32.100, dopo questo passo salviamo le modifiche e per controllare che il servizio HTTP sia in ascolto sulla porta 80 inseriamo nella linea di comando `sudo inetsim` che ci restituirà le impostazioni correnti di inetsim.

```
(pif@kali)-[~]
$ sudo inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== INetSim main process started (PID 7426) ==
Session ID: 7426
Listening on: 192.168.32.100
Real Date/Time: 2025-03-27 21:37:51
Fake Date/Time: 2025-03-27 21:37:51 (Delta: 0 seconds)
Forking services ...
* dns_53_tcp_udp - started (PID 7428)
deprecated method; prefer start_server() at /usr/share/perl5/INetSim/DNS.pm line 69.
Attempt to start Net::DNS::Nameserver in a subprocess at /usr/share/perl5/INetSim/DNS.pm line 69.
* http_80_tcp - started (PID 7429)
done.
Simulation running.
```

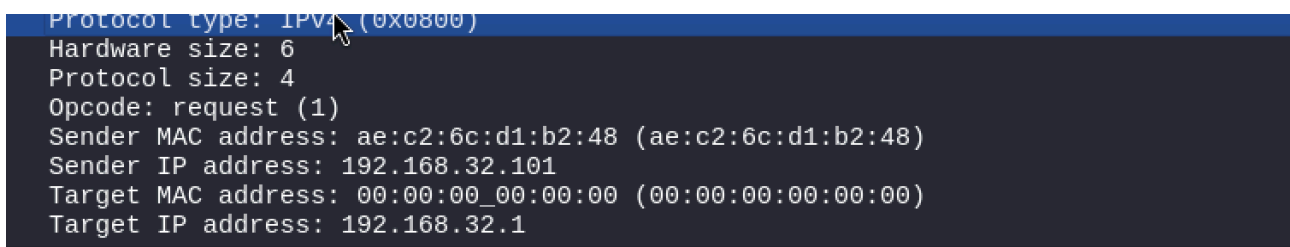
Ora che abbiamo impostato tutti i parametri dobbiamo controllare che funzionino correttamente per cui apriamo Windows, Internet Explorer e digitiamo sulla barra degli indirizzi epicode.internal, una volta digitato l'indirizzo il browser mi restituisce questa immagine che ci indica che la configurazione è stata eseguita correttamente.



Ora apriamo un programma su Kali chiamato Wireshark, questo programma ci permette di intercettare e analizzare i pacchetti dati tra le due macchine virtuali. Una volta aperto il programma selezioniamo eth0 e avviamo la cattura dei pacchetti.

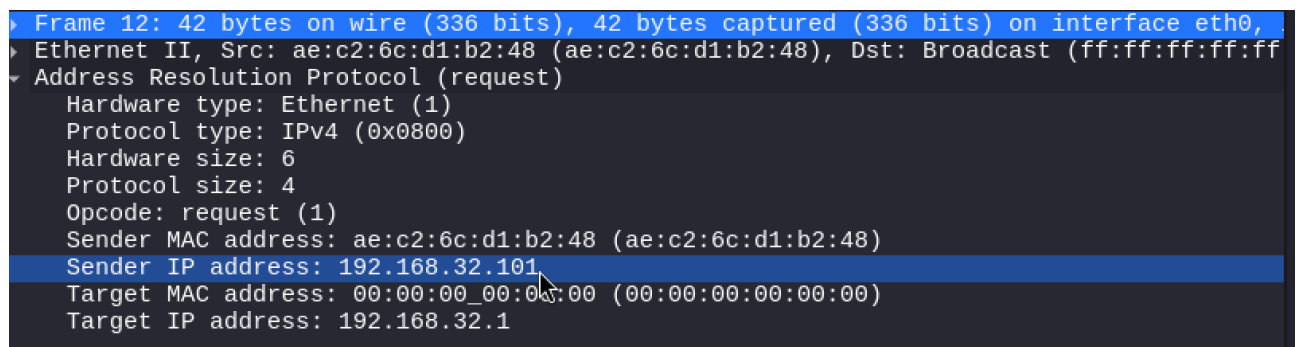
1	0.000000000	0a:1f:f3:5d:73:c2	Broadcast	ARP	42 Who has 192.168.32.1? Tell 192.168.32.100
2	1.024409334	0a:1f:f3:5d:73:c2	Broadcast	ARP	42 Who has 192.168.32.1? Tell 192.168.32.100
3	2.047715043	0a:1f:f3:5d:73:c2	Broadcast	ARP	42 Who has 192.168.32.1? Tell 192.168.32.100
4	3.073679002	fe80::81f:f3ff:fe5d...	fe80::842f:57ff:fe4...	DNS	101 Standard query 0xfb4a A 1.debian.pool.ntp.org
5	3.073738377	fe80::81f:f3ff:fe5d...	fe80::842f:57ff:fe4...	DNS	101 Standard query 0x6449 AAAA 1.debian.pool.ntp.org
6	3.101955043	fe80::842f:57ff:fe4...	fe80::81f:f3ff:fe5d...	DNS	156 Standard query response 0x6449 AAAA 1.debian.pool.ntp.org
7	3.121634502	fe80::842f:57ff:fe4...	fe80::81f:f3ff:fe5d...	DNS	165 Standard query response 0xfb4a A 1.debian.pool.ntp.org
8	3.122286085	0a:1f:f3:5d:73:c2	Broadcast	ARP	42 Who has 192.168.32.1? Tell 192.168.32.100
9	4.128051044	0a:1f:f3:5d:73:c2	Broadcast	ARP	42 Who has 192.168.32.1? Tell 192.168.32.100
10	5.151215961	0a:1f:f3:5d:73:c2	Broadcast	ARP	42 Who has 192.168.32.1? Tell 192.168.32.100
11	13.251251048	0a:1f:f3:5d:73:c2	Broadcast	ARP	42 Who has 192.168.32.1? Tell 192.168.32.100
12	14.275707215	0a:1f:f3:5d:73:c2	Broadcast	ARP	42 Who has 192.168.32.1? Tell 192.168.32.100
13	15.295237258	0a:1f:f3:5d:73:c2	Broadcast	ARP	42 Who has 192.168.32.1? Tell 192.168.32.100
14	23.500011262	0a:1f:f3:5d:73:c2	Broadcast	ARP	42 Who has 192.168.32.1? Tell 192.168.32.100

Una volta stoppato il programma selezioniamo uno dei pacchetti con protocollo ARP, aprendolo potremo controllare il sender MAC Address e il target MAC Address



Ora che abbiamo i MAC Address dello scambio dati su servizio HTTP apriamo Inetsim sulla macchina Kali, decommentiamo il servizio HTTPS (attivandolo) e commentiamo il servizio HTTP (disattivandolo). Sulla macchina Windows apriamo di nuovo il web browser e digitiamo di nuovo `epicode.internal`, non avendo creato un certificato di sicurezza il browser cerca di bloccare l'accesso, ignorando l'avviso riusciamo nuovamente ad accedere alla pagina web.

Riapriamo Wireshark e rianalizziamo lo scambio di pacchetti dati tra le due macchine, analizzando il pacchetto dati in protocollo ARP notiamo che i MAC Address sono cambiati rispecchiando il protocollo di sicurezza che denota la natura privata e criptata del certificato HTTPS



```
Frame 12: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0,
Ethernet II, Src: ae:c2:6c:d1:b2:48 (ae:c2:6c:d1:b2:48), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: ae:c2:6c:d1:b2:48 (ae:c2:6c:d1:b2:48)
  Sender IP address: 192.168.32.101
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.32.1
```

FINE