

Simulazione connessione due macchine

Nell'esercizio di oggi metteremo insieme le competenze acquisite finora. Lo studente verrà valutato sulla base della risoluzione al problema seguente.

Requisiti e servizi:

- Kali Linux □ IP 192.168.32.100
- Windows 7 □ IP 192.168.32.101
- HTTPS server: attivo
- Servizio DNS per risoluzione nomi di dominio: attivo

Traccia:

Simulare, in ambiente di laboratorio virtuale, un'architettura client server in cui un client con indirizzo 192.168.32.101 richiede tramite web browser una risorsa all'hostname epicode.internal che risponde all'indirizzo 192.168.32.100.

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 differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS. Spiegare, motivandole, le principali differenze.

Se si volesse seguire una scaletta con un approccio strutturato, potremmo procedere alla risoluzione del problema come segue:

- Configurazione IP statici di client e server
- Configurazione servizi HTTPS e DNS
- Abilitazione servizi tramite inetsim
- Test connessione con client W7
- Prima cattura con Wireshark (comunicazione HTTPS)
- Disabilitazione servizio HTTPS
- Configurazione servizio HTTP
- Abilitazione servizio HTTP
- Test connessione con client W7
- Seconda cattura con Wireshark (comunicazione HTTP)
- Valutazione delle macro differenze tra le due catture.

Possiamo assegnare ip statici:

- Modificando il file /etc/network/interfaces sui sistemi Linux (inserendo gli IP richiesti dall'esercizio);
- Modificando impostazioni tcp/ip della scheda di rete su windows. Per la macchina Windows si deve modificare anche il puntamento del DNS, come in figura.

The image shows two overlapping windows. The background window is a terminal on a Kali Linux system, displaying the nano text editor editing the file /etc/network/interfaces. The terminal output shows the configuration for the loopback interface 'lo' and the ethernet interface 'eth0'. The 'eth0' interface is configured with a static IP address of 192.168.50.100, a subnet mask of 255.255.255.0, and a default gateway of 192.168.50.1.

The foreground window is a Windows network configuration dialog box. It has a title bar that says 'kali@kali: ~'. The dialog box contains a message: 'You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.' Below this message, there are two radio buttons: 'Obtain an IP address automatically' (which is selected) and 'Use the following IP address:'. The 'Use the following IP address:' option is selected, and the fields for IP address, Subnet mask, and Default gateway are filled with the values: 192 . 168 . 50 . 102, 255 . 255 . 255 . 0, and 192 . 168 . 50 . 1, respectively. Below these fields, there are two more radio buttons: 'Obtain DNS server address automatically' (which is selected) and 'Use the following DNS server addresses:'. The 'Use the following DNS server addresses:' option is selected, and the fields for Preferred DNS server and Alternate DNS server are filled with the values: 192 . 168 . 50 . 103 and . . ., respectively. At the bottom of the dialog box, there is a checkbox labeled 'Validate settings upon exit' and an 'Advanced...' button. At the very bottom, there are 'OK' and 'Cancel' buttons.

```
File Actions Edit View Help
GNU nano 6.0 /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.50.100/24
gateway 192.168.50.1
```

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

☐ Obtain an IP address automatically

☒ Use the following IP address:

IP address: 192 . 168 . 50 . 102

Subnet mask: 255 . 255 . 255 . 0

Default gateway: 192 . 168 . 50 . 1

☐ Obtain DNS server address automatically

☒ Use the following DNS server addresses:

Preferred DNS server: 192 . 168 . 50 . 103

Alternate DNS server: . . .

☐ Validate settings upon exit

Advanced...

OK Cancel

Configurazione servizi HTTPS e DNS

I servizi HTTPS e DNS possiamo configurarli ed attivarli utilizzando InetSim su Kali Linux. Oggi utilizzeremo l'ip di Kali così da rendere i servizi disponibili esternamente alla macchina. Per farlo, eliminiamo il carattere «#» alla riga «service_bind_address» e modifichiamo il valore con l'ip di Kali Linux. Per quanto riguarda il DNS, notare che si è richiesta l'associazione tra epicode.internal e l'ip di Kali (192.168.32.100). Questa entry va inserita al posto degli esempi sotto a destra (www.foo.com IP)

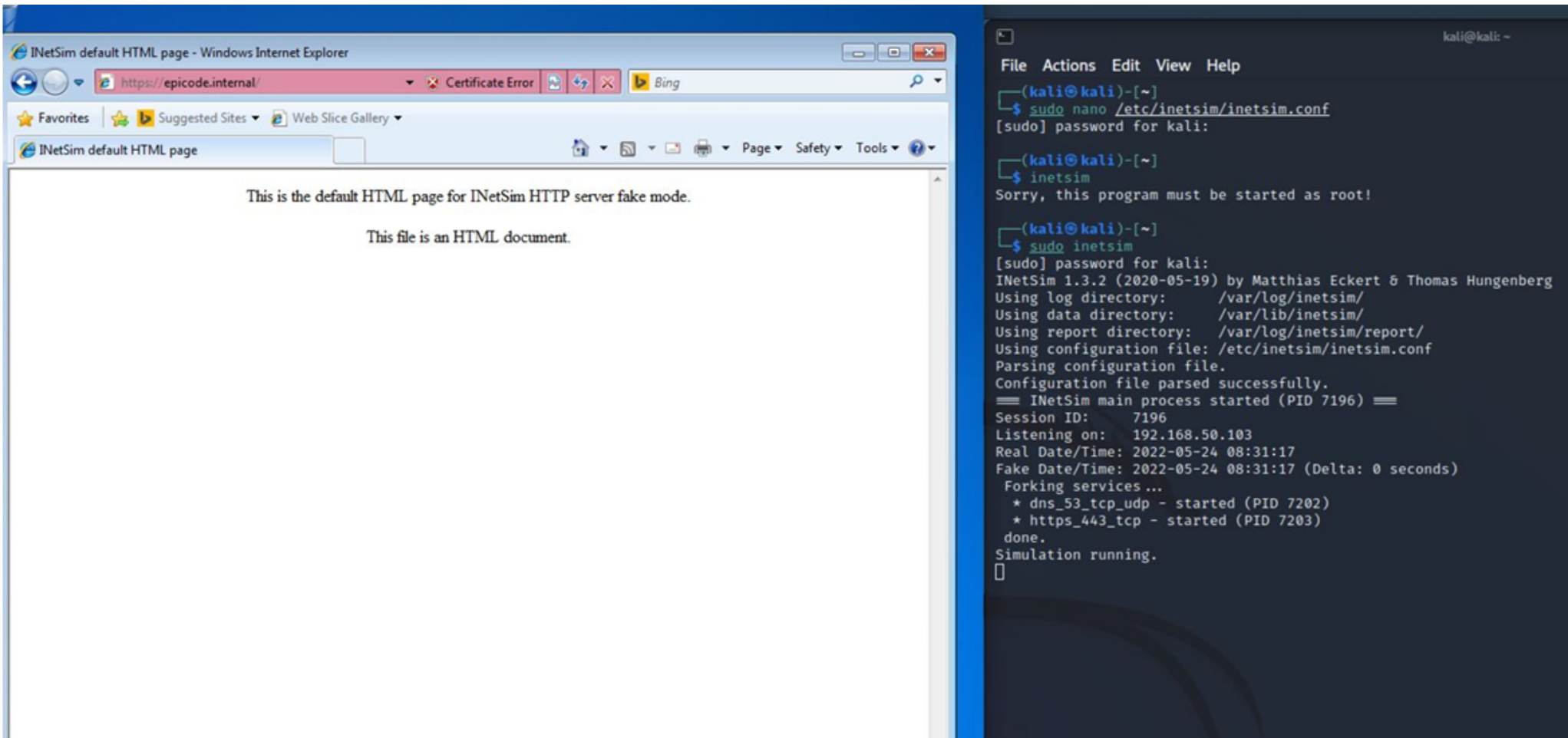
```
#####  
# dns_static  
#  
# Static mappings for DNS  
#  
# Syntax: dns_static <fqdn hostname> <IP address>  
#  
# Default: none  
#  
#dns_static www.foo.com 10.10.10.10  
#dns_static ns1.foo.com 10.70.50.30  
#dns_static ftp.bar.net 10.10.20.30  
  
#####  
# dns_version  
#
```

```
#start_service quotd_tcp  
#start_service quotd_udp  
#start_service chargen_tcp  
#start_service chargen_udp  
#start_service dummy_tcp  
#start_service dummy_udp  
  
#####  
# service_bind_address  
#  
# IP address to bind services to  
#  
# Syntax: service_bind_address <IP address>  
#  
# Default: 127.0.0.1  
#  
#service_bind_address 10.10.10.1  
  
#####  
# service_run_as_user  
#  
# User to run services  
#  
# Syntax: service_run_as_user <username>  
#  
# Default: inetsim  
#  
#service_run_as_user nobody
```

Abilitazione servizi tramite inetsim

Test connessione con client W7

A questo punto, non ci resta che attivare i servizi lanciando il comando inetsim dal terminale, accedere alla macchina Windows e testare la connettività al servizio HTTPS verso <https://epicode.internal>. La figura sotto mostra il risultato atteso (client a sinistra, server HTTPS a destra)



Prima cattura con Wireshark (comunicazione HTTPS)

A valle del test, facciamo partire Wireshark, ed iniziamo la cattura dei pacchetti, avendo cura di inviare almeno un'altra richiesta al Web Server. Fermiamo e salviamo la cattura, in modo tale da poterla paragonare alla cattura che faremo in HTTP.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_99:b1:5f	Broadcast	ARP	60	Who has 192.168.50.103? Tell 192.168.50.102
2	0.000017833	PcsCompu_39:7d:fe	PcsCompu_99:b1:5f	ARP	42	192.168.50.103 is at 08:00:27:39:7d:fe
3	0.000372796	192.168.50.102	192.168.50.103	TCP	66	49188 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
4	0.000391626	192.168.50.103	192.168.50.102	TCP	66	443 → 49188 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
5	0.000822528	192.168.50.102	192.168.50.103	TCP	60	49188 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6	0.001150186	192.168.50.102	192.168.50.103	TLSv1	217	Client Hello
7	0.001159146	192.168.50.103	192.168.50.102	TCP	54	443 → 49188 [ACK] Seq=1 Ack=164 Win=64128 Len=0
8	0.007944913	192.168.50.103	192.168.50.102	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
9	0.023255727	192.168.50.102	192.168.50.103	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
10	0.023304832	192.168.50.103	192.168.50.102	TCP	54	443 → 49188 [ACK] Seq=1320 Ack=298 Win=64128 Len=0
11	0.023674555	192.168.50.103	192.168.50.102	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
12	0.029204353	PcsCompu_99:b1:5f	Broadcast	ARP	60	Who has 192.168.50.1? Tell 192.168.50.102
13	0.228970078	192.168.50.102	192.168.50.103	TCP	60	49188 → 443 [ACK] Seq=298 Ack=1379 Win=64320 Len=0
14	1.009327540	PcsCompu_99:b1:5f	Broadcast	ARP	60	Who has 192.168.50.1? Tell 192.168.50.102
15	2.009891554	PcsCompu_99:b1:5f	Broadcast	ARP	60	Who has 192.168.50.1? Tell 192.168.50.102
16	3.172837422	fe80::256b:4013:414...	ff02::1:3	LLMNR	84	Standard query 0x2b9d A wpa
17	3.174270198	192.168.50.102	224.0.0.252	LLMNR	64	Standard query 0x2b9d A wpa
18	3.275280859	fe80::256b:4013:414...	ff02::1:3	LLMNR	84	Standard query 0x2b9d A wpa
19	3.275281216	192.168.50.102	224.0.0.252	LLMNR	64	Standard query 0x2b9d A wpa
20	3.477979434	192.168.50.102	192.168.50.255	NBNS	92	Name query NB WPA<00>
21	4.227556042	192.168.50.102	192.168.50.255	NBNS	92	Name query NB WPA<00>
22	4.978324856	192.168.50.102	192.168.50.255	NBNS	92	Name query NB WPA<00>
23	5.167002330	PcsCompu_39:7d:fe	PcsCompu_99:b1:5f	ARP	42	Who has 192.168.50.102? Tell 192.168.50.103
24	5.167474847	PcsCompu_99:b1:5f	PcsCompu_39:7d:fe	ARP	60	192.168.50.102 is at 08:00:27:99:b1:5f
25	5.735722809	PcsCompu_99:b1:5f	Broadcast	ARP	60	Who has 192.168.50.1? Tell 192.168.50.102
26	6.509542606	PcsCompu_99:b1:5f	Broadcast	ARP	60	Who has 192.168.50.1? Tell 192.168.50.102
27	7.508828248	PcsCompu_99:b1:5f	Broadcast	ARP	60	Who has 192.168.50.1? Tell 192.168.50.102
28	8.875275529	fe80::256b:4013:414...	ff02::1:3	LLMNR	84	Standard query 0xb418 A wpa
29	8.875275790	192.168.50.102	224.0.0.252	LLMNR	64	Standard query 0xb418 A wpa
30	8.978185211	fe80::256b:4013:414...	ff02::1:3	LLMNR	84	Standard query 0xb418 A wpa
31	8.978185485	192.168.50.102	224.0.0.252	LLMNR	64	Standard query 0xb418 A wpa
32	9.181669495	192.168.50.102	192.168.50.255	NBNS	92	Name query NB WPA<00>
33	9.931957289	192.168.50.102	192.168.50.255	NBNS	92	Name query NB WPA<00>
34	10.681937981	192.168.50.102	192.168.50.255	NBNS	92	Name query NB WPA<00>
35	12.457064667	PcsCompu_99:b1:5f	Broadcast	ARP	60	Who has 192.168.50.1? Tell 192.168.50.102
36	13.009487720	PcsCompu_99:b1:5f	Broadcast	ARP	60	Who has 192.168.50.1? Tell 192.168.50.102
37	14.009887571	PcsCompu_99:b1:5f	Broadcast	ARP	60	Who has 192.168.50.1? Tell 192.168.50.102

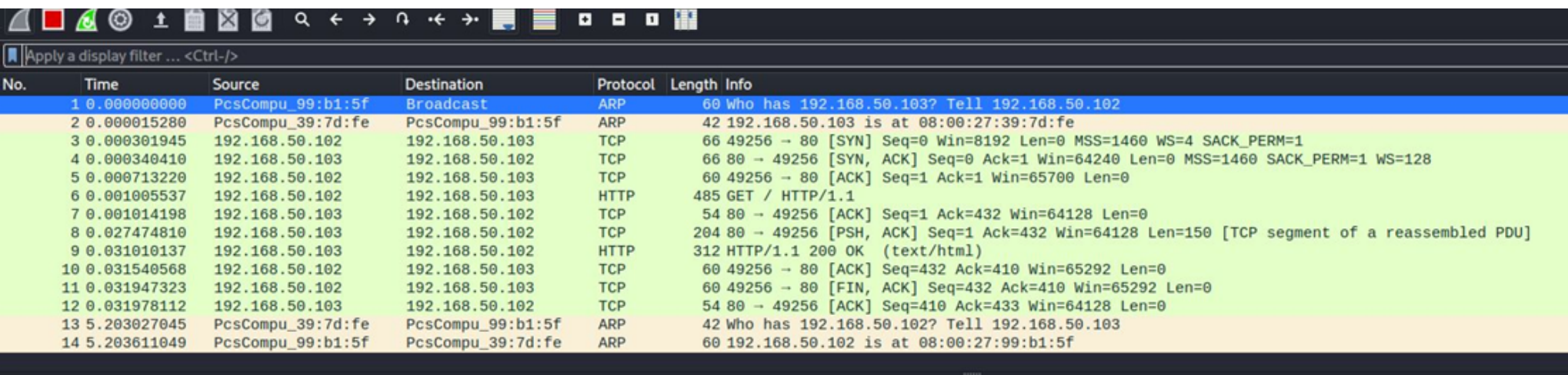
Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth1, id 0

Ethernet II, Src: PcsCompu_99:b1:5f (08:00:27:99:b1:5f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

Address Resolution Protocol (request)

- Disabilitazione servizio HTTPS
- Configurazione servizio HTTP
- Abilitazione servizio HTTP
- Test connessione con client W7
- Seconda cattura con Wireshark (comunicazione HTTP)

Completiamo il secondo blocco di attività, disattivando il servizio HTTPS, e sostituendolo con il servizio HTTP. Lanciamo una nuova cattura con Wireshark, sempre dopo aver verificato la presenza di comunicazione tra client e server. Al netto degli IP source e destination (noi abbiamo utilizzato altri IP il 192.168.50.102 e 103) la vostra cattura dovrà essere simile a quella sotto.



Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_99:b1:5f	Broadcast	ARP	60	Who has 192.168.50.103? Tell 192.168.50.102
2	0.000015280	PcsCompu_39:7d:fe	PcsCompu_99:b1:5f	ARP	42	192.168.50.103 is at 08:00:27:39:7d:fe
3	0.000301945	192.168.50.102	192.168.50.103	TCP	66	49256 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
4	0.000340410	192.168.50.103	192.168.50.102	TCP	66	80 → 49256 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
5	0.000713220	192.168.50.102	192.168.50.103	TCP	60	49256 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6	0.001005537	192.168.50.102	192.168.50.103	HTTP	485	GET / HTTP/1.1
7	0.001014198	192.168.50.103	192.168.50.102	TCP	54	80 → 49256 [ACK] Seq=1 Ack=432 Win=64128 Len=0
8	0.027474810	192.168.50.103	192.168.50.102	TCP	204	80 → 49256 [PSH, ACK] Seq=1 Ack=432 Win=64128 Len=150 [TCP segment of a reassembled PDU]
9	0.031010137	192.168.50.103	192.168.50.102	HTTP	312	HTTP/1.1 200 OK (text/html)
10	0.031540568	192.168.50.102	192.168.50.103	TCP	60	49256 → 80 [ACK] Seq=432 Ack=410 Win=65292 Len=0
11	0.031947323	192.168.50.102	192.168.50.103	TCP	60	49256 → 80 [FIN, ACK] Seq=432 Ack=410 Win=65292 Len=0
12	0.031978112	192.168.50.103	192.168.50.102	TCP	54	80 → 49256 [ACK] Seq=410 Ack=433 Win=64128 Len=0
13	5.203027045	PcsCompu_39:7d:fe	PcsCompu_99:b1:5f	ARP	42	Who has 192.168.50.102? Tell 192.168.50.103
14	5.203611049	PcsCompu_99:b1:5f	PcsCompu_39:7d:fe	ARP	60	192.168.50.102 is at 08:00:27:99:b1:5f

Valutazione delle macro differenze tra le due catture

La differenza più grande tra le due catture è la presenza di un tentativo di creazione di canale cifrato in HTTPS (è possibile filtrare la cattura Wireshark per protocollo TLS per seguirne la sequenza) che invece non trovate nella cattura HTTP (è un protocollo in chiaro). Se controllate il pacchetto dati, HTTP vi farà vedere il contenuto della richiesta in chiaro, mentre HTTPS, grazie al TLS, creerà un tunnel cifrato prima di inviare i flussi di dati.