INFORMATION MANAGEMENT AND CRYPTOGRAPHY

COMP 43

LAB 6

KARAN TANK

10196998

Internet Protocol security (Ipsec) is a protocol security which authenticates and gives you end to end encryption of packets sent over the internet. It can be used for Virtual private networks.

In this lab we are going to construct an IPsec, host-to-host tunnel mode vpn connection between two Linux computers in the lab.

For this we will install ipsec in both Linux. For this we will use command ap-get install strongswan in the command line. You must also install apt-get install strongswan-pki. \**refer figure 1 and 2\**

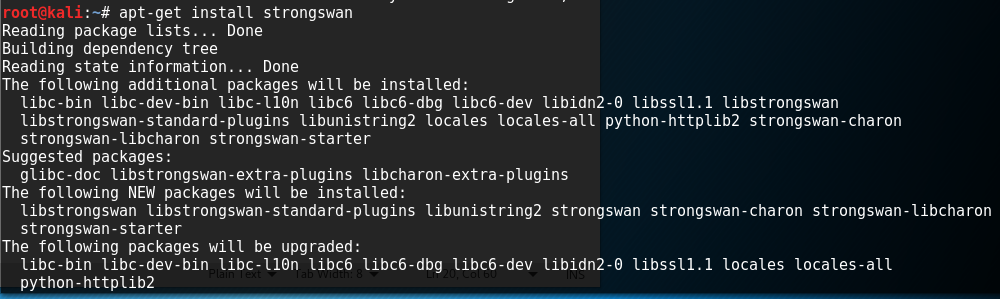


Figure 1

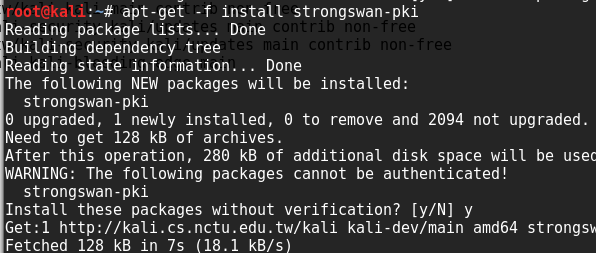


Figure 2

Ipsec uses public key cryptography to authenticate endpoints. So we need to create public keys. The way we learned to create keys in the previous lab was using openssl or by using CA.pl.

Ipse(strongswan) includes pki that we installed, that can also be used for generating public certificates.

We will use the command **ipsec pki –gen >mycakey.der ,** in this the –gen switch is used to create x509 certificates. This command generates mycakey.der.



Figure 3

We can sign the certificates as well using ipsec pki, for this we will use the command

**Ipsec pki –self –in mycakey.der –dn “0=SSFC, CN=myCA” –ca > mycaCert.der**

This command is used to self-sign the certificates with the help of –self switch. With –in switch you can specify the cakey.der to input which is used to sign this certificate and with the help of –out switch you can specify the new self-signed certificated to be generated.

--dn key is used to extract the subject DN of an x509 certificate.

For the ipsec tunnel we need to create right key and left key as in the ipsec configuration file right and left end are described arbitrarily. So you need to select one linux as right and the other one as left and create the certificates and sign them respectively. This is because when the remote machine tries to make a connection using the x509 certificates, ipsec first goes through the ipsec.cnf file to check the ip address and verify. If it verifies the ip address the connection is established. For this you will sign the certificate with the identification of the expected output.

You can use any other identification, but the easiest way is by using ip address so that we know which machine is right or left and It can make our work easy, hence we will use ip address for this process. You should also have the flag setting configured to make them eligible for the identification and authentication or server endpoints. For generating the certificate, we can use the same old command **ipsec pki –gen > leftkey.der or rightkey.der**

But to sign these certificates and set the flags and to create the identification ids, we will use the command

**ipsec pki –pub –in leftkey.der | ipsec.pki –issue –cacert mycaCert.der –cakey mycakey.der –dn “O=SSFC, CN=left ip” –flag ikeIntermediate –flag serverAuth > leftCert.der**

in the above command, --pub switch is used to extract the public key from the private key, --in will take the input file, --issue switch is used to issue a certificate using CA certificate and cakey, to specify ca certificate we will use –cacert switch and to specify key we will use –cakey switch. –dn switch is used to specify the subject and then we specify the flags using –flag switch. And the we output everything using > switch to get the self-signed certificate with the identification and flags.

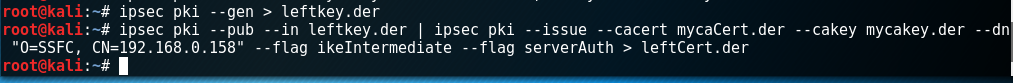


Figure 4

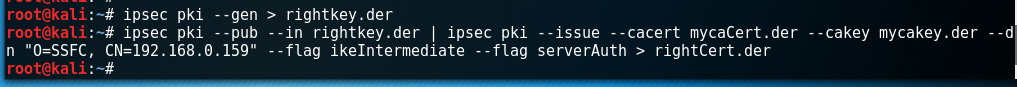


Figure 5

We will issue the command twice to create and self sign a Right key and a Left key. *Refer figure 4 and 5*

The above command will create files like leftcakey.der and leftcert.der, same for right too.

After generating all these certificates, we will move the left certificates to the left machine and right certificates to the right machine. I created and signed all the certificates on left side, so I will transfer the right certificates to right side using netcat. We will also need the root certificate, mycaCert.der on both sides.

For this on the sender side we will use the command **nc ip address port < filename** and on the receiving side we will use command **nc -l -p port > output file**

*Refer figure 6 for sender side*

*Refer figure 7 for receiver side.*

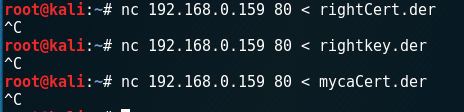


Figure 6

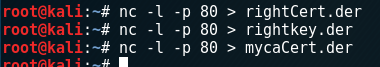


Figure 7

After we have transferred left files to left side and right files to right side, we will load the certificates as follows.

etc/ipsec.d/private/  should contain the private key for the left endpoint **(leftkey.der)** /etc/ipsec.d/certs/      should contain the signed certificate for the left endpoint **(leftCert.der)** /etc/ipsec.d/cacerts/   should contain the CA certificate that signed the other certificates (**mycaCert.der)**

We will do this same on the right side as well.

.

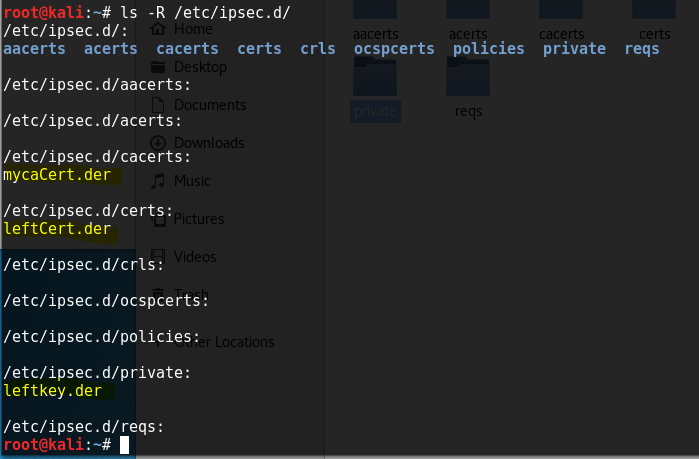


Figure 8

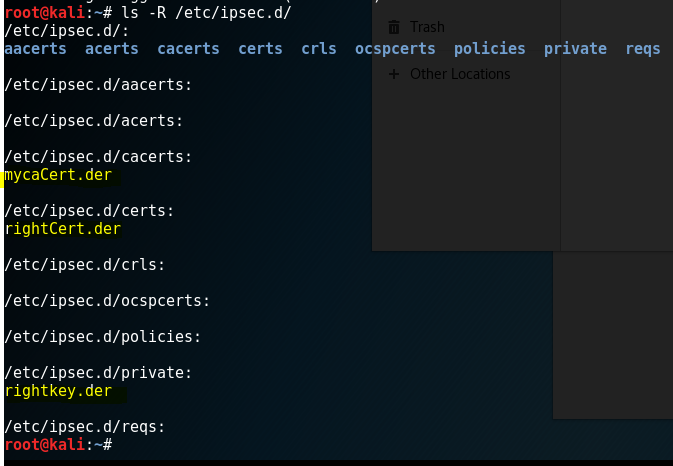
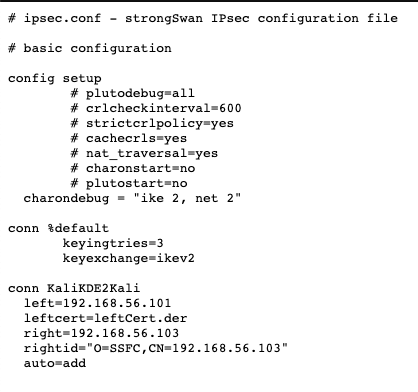


Figure 9

After this we need to configure the ipsec.cnf file as well as ipsec.secrets on both sides.



We can use the configuration shown above and use the proper identification ids(ip address) for left and right.

The ipsec.cnf file consists of three parts. The setup, the connection defaults and connection name and details the connection setup, For the conn name, you can give any name, I have given kali2kali as i am configuring tunnel between two kali linux.

In left = ip address, I will enter the left ip address while on the right config file I will fill in the right machine address.

We will specigy the common name and the cert files and then at last there is auto=add which tells the ipsec to add the connection to its local connection database but not start until it is told so. Figure 10 shows the left side ipsec.cnf file.

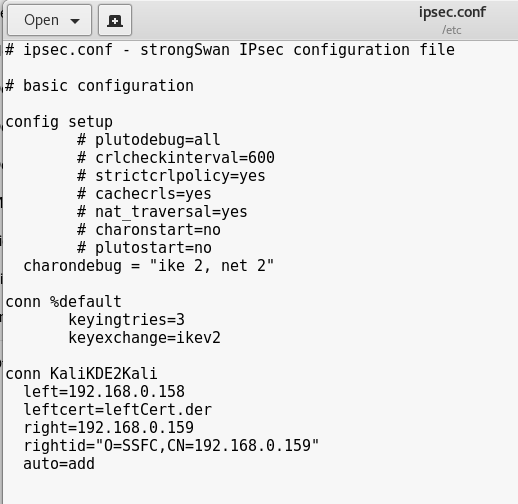


Figure 10

After configuring the ipsec.cnf file, we also need to configure the ipsec.secrets file in which we need to add a line

**: RSA /path/to/key.der**

For left machine we will specify left key and for right machine we will specify right key.

After both files are configured we will start the tunneling, but before we start tunneling we will open a new terminal and run the command **tail -f /var/log/syslog** to see the logs of the connection. We will capture the network packets in the wireshark as well

And then to start the ipsec using **ipsec start** command

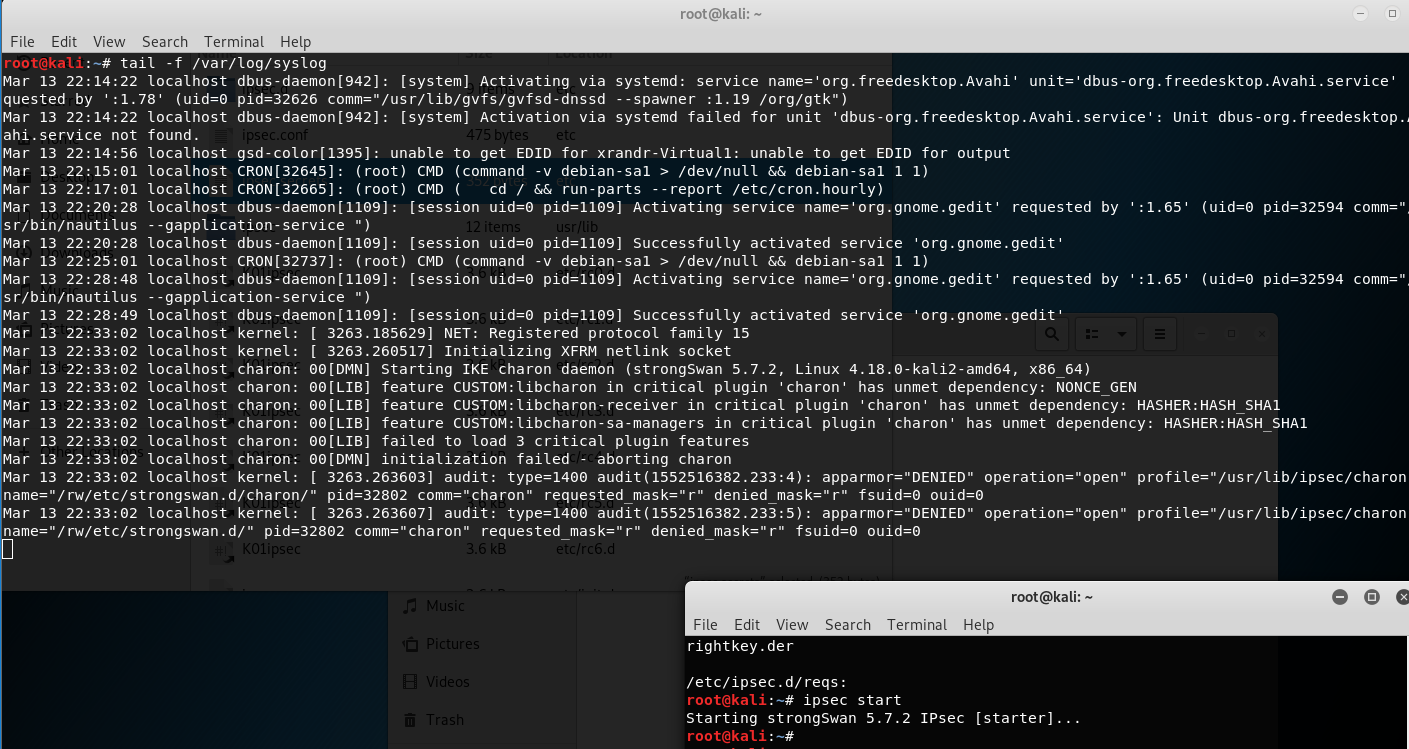


Figure 11

You should see some logs in the other terminal and if nothing is shown failed, we can move forward and start the ipsec on the other machine as well.

after both sides are started with ipsec, you can see see the status using **ipsec statusall.**

After both sides are up, we will start up the tunnel using the command **ipsec up kali2kali** using the common name we specified in the config file.

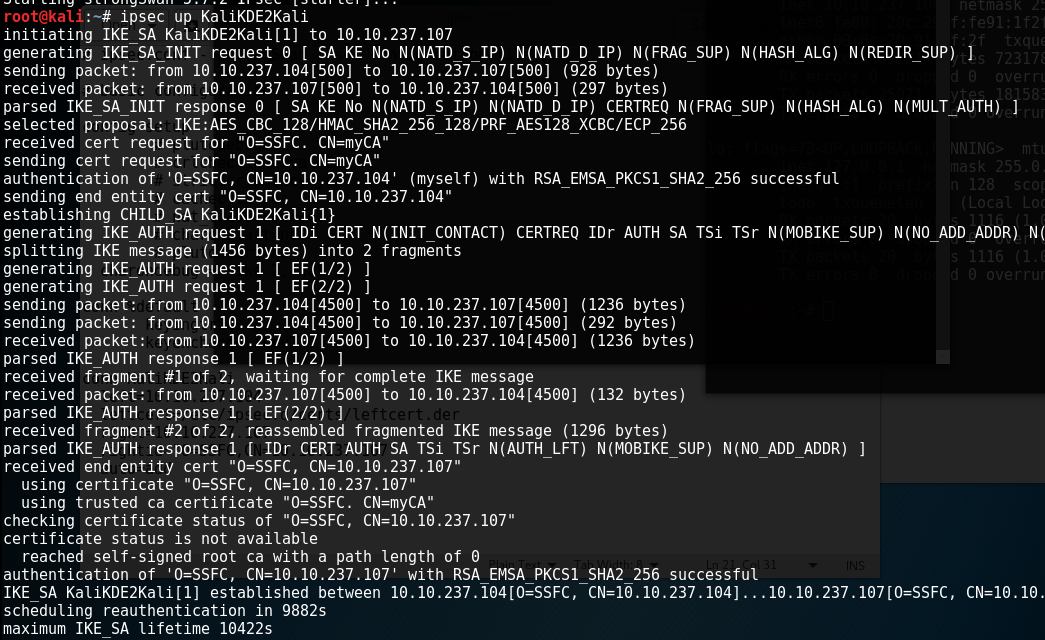


Figure 12

After the connection is up, you can see the logs where it says the connection is established successfully to confirm.

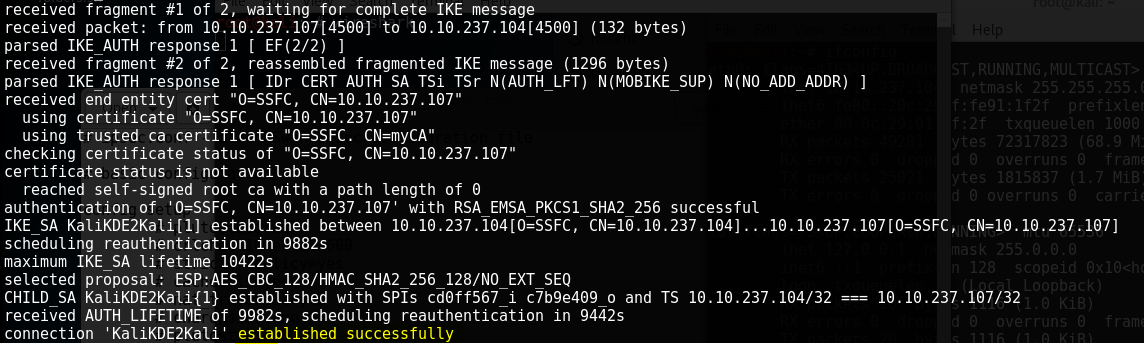


Figure 13

After the connection is up, you can see the status of the connection

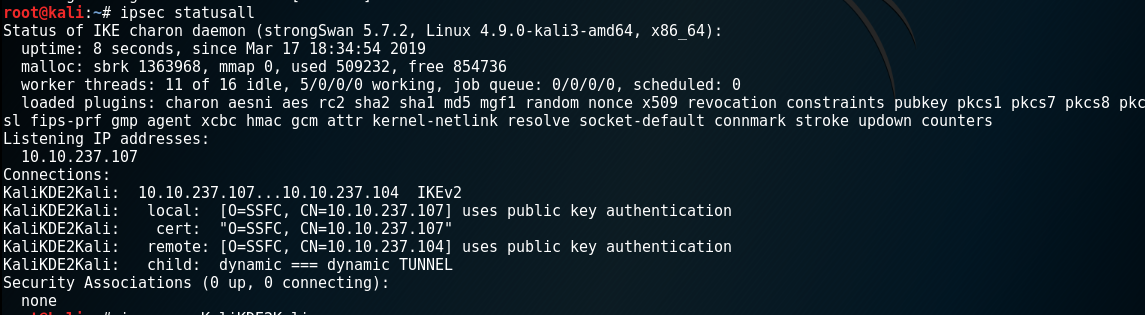


Figure 14

We can even try pinging the other machine and the ping should be successful as the tunnel is connected and it should send esp packets.

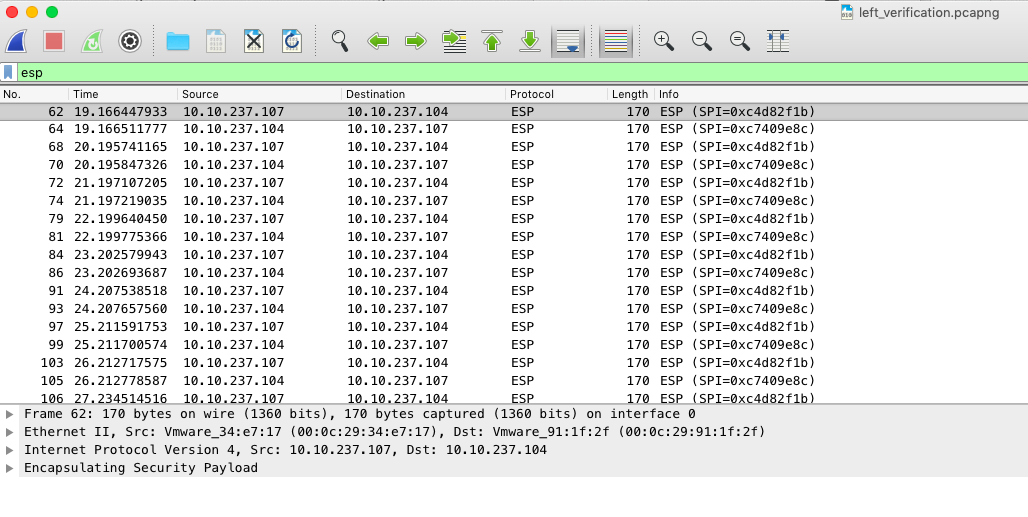


Figure 15

***You will find out two ip network address 192.168.0.\* and 10.10.237.\* cause I was. Running kali live, and in between it turned off cause of an error, so I had to setup a new one.***

In Ipsec tunnel, it has two keys, encryption and decryption key. The sending side uses the encryption key to encrypt the packets send over the tunnel and the reciving end has a decryption key, which is signed by the same root key as the encrypted key to decrypt the data packets over the internet tunnel. The x509 certificates we made are used as public and private keys.

This is done by crating the SA in the IKE where we use the UDP port 500. ESP provides an encapsulation for the encryption and authentication of the user traffics. ESP header is 4 layer while the ip address are 3 layers. The extra layer of the ESP provides the encrypted payloads.

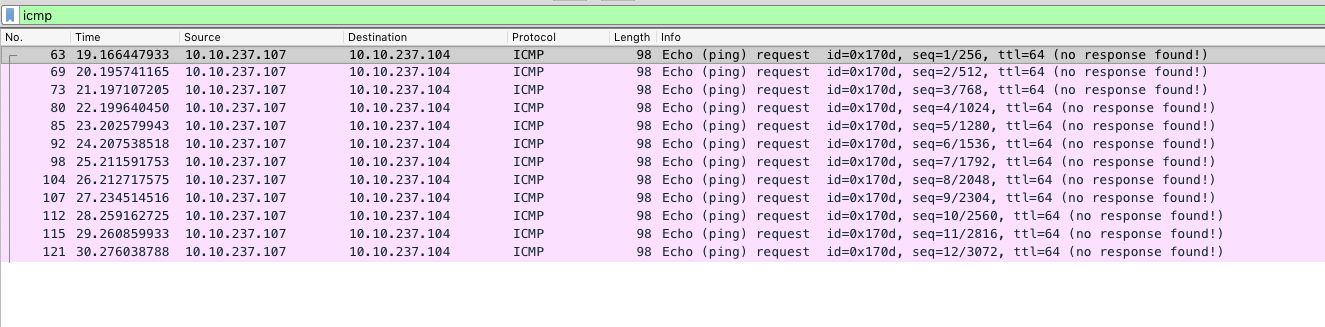
You will see the esp packets going out and coming and icmp packets appear to be coming from the other machine. If you were to do a MITM attack, you wont be able to see icmp packets, just the esp packets. Ipsec takes the incoming packets and unpacks it and decrypts it back in the stream to look like it came from the wire and is ready to process the receiving end. You won’t see any icmp packets outgoing. 

Figure 16

After that we will stop the ipsec sing **ipsec stop** command. And then start it again and use the ipsec statusall to check th status to confirm that there is no SA been created. And then ping the other machine to see that the connection is with plain packets. As when the tunnel is stopped, there is no ESP header to encrypt the packets on the IP header and it sends and receives the plain text ICMP packets.

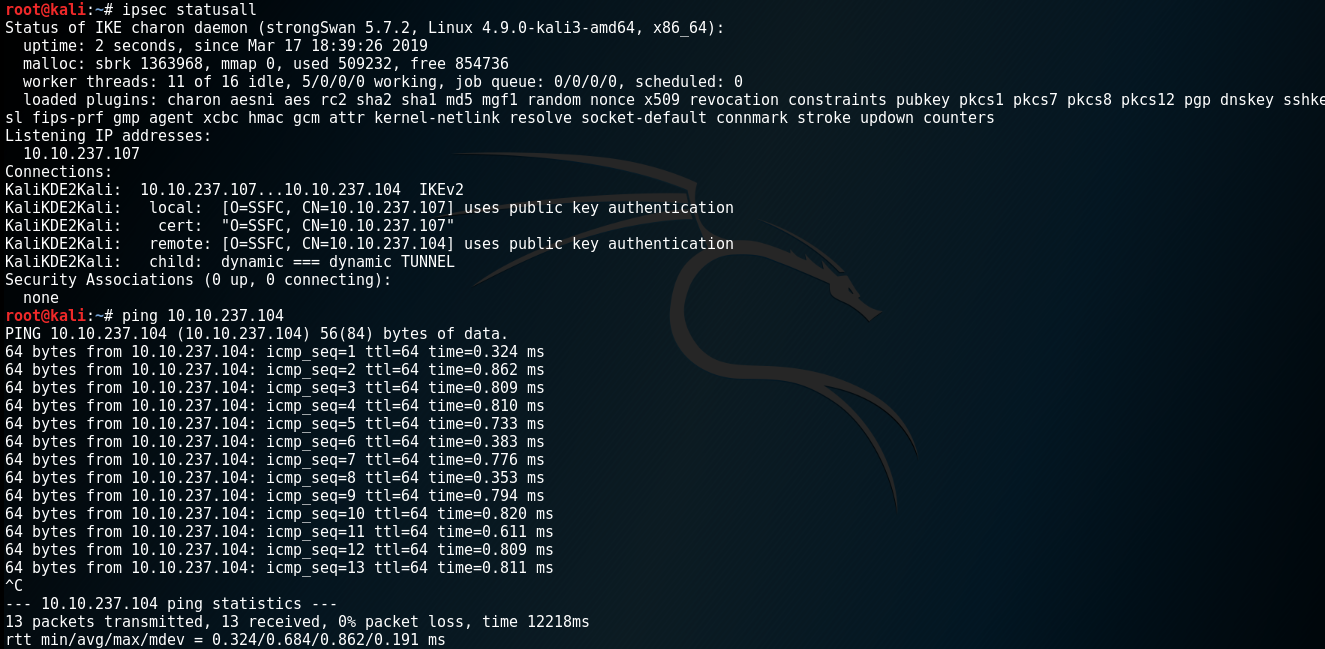


Figure 17

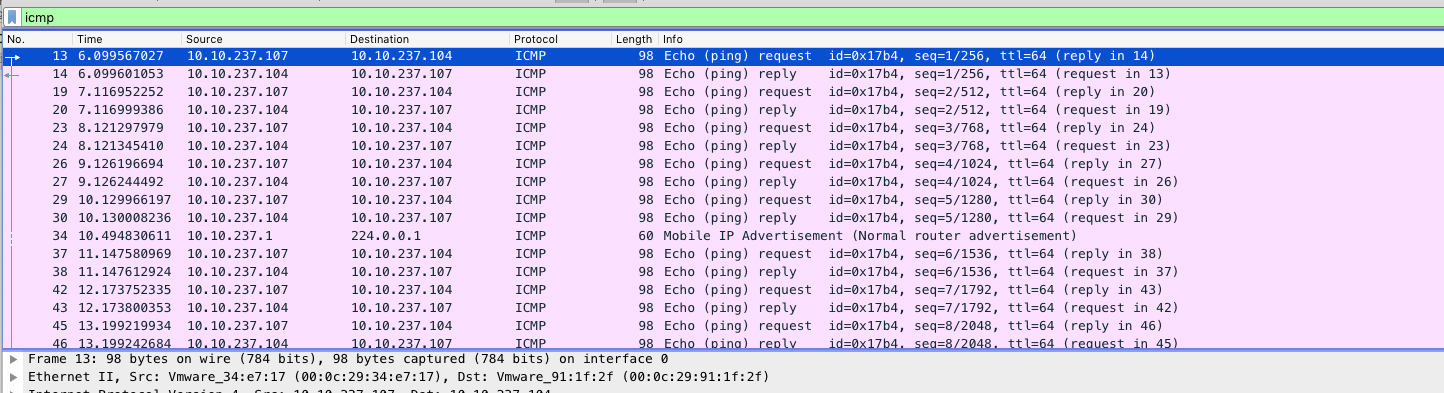


Figure 18

Now we will route the connection using ipsec route conn-name to record the connection in policy database.



Figure 19

We will check the ipsec status to see if there is no SA created

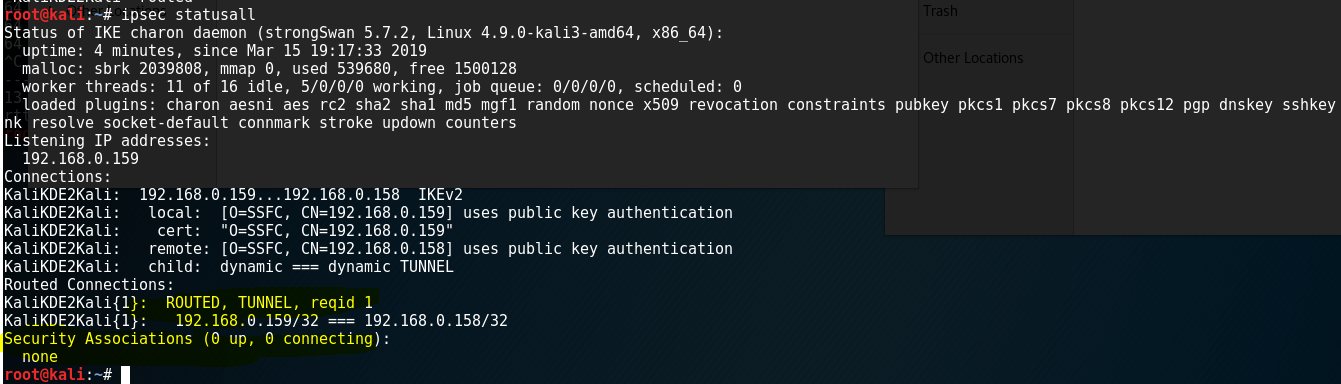


Figure 20

Now we will try to ping the other pc, it will not go through as we have not started the connection.

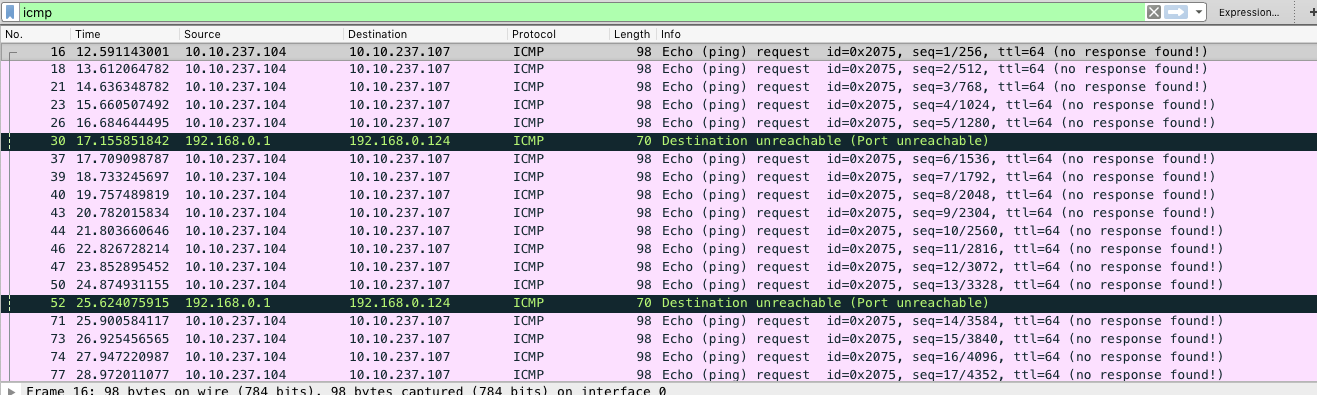


Figure 2132