PARSHVANATH CHARITABLE TRUST'S

A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)

Academic Year: 2025-26 **Semester: V**

Class / Branch: TE IT

Subject: Security Lab

Experiment No. 9

1. **Aim: To study and implement IPSEC in Linux .**
2. **Software Required** : Ubuntu 14.04 OS, Wireshark 2.6.1, Strongswan
3. **Theory :**

IPsec :

IPsec, also known as the Internet Protocol Security or IP Security protocol, defines the architecture for security services for IP network traffic. The IP Security (IPsec) architecture comprises a suite of protocols developed to ensure the integrity, confidentiality and authentication of data communications over an IP network. Also included in IPsec are protocols that define the cryptographic algorithms usedto encrypt, decrypt and authenticate packets, as well as the protocols needed for secure key exchange and key management.

IPsec may be used in three different security domains: virtual private networks, application-level security and routing security. At this time, IPsec is predominately used in VPNs. When used in application-level security or routing security, IPsec is not a complete solution and must be coupled with other security measures to be effective, hindering its deployment in these domains

StrongSwan

StrongSwan is basically a keying daemon, which uses the Internet Key Exchange protocols (IKEv1 and IKEv2) to establish security associations (SA) between two peers. IKE provides strong authentication of both peers and derives unique cryptographic session keys. Such an IKE session is often denoted IKE_SA. Besides authentication and key material IKE also provides the means to exchange configuration information and to negotiate IPsec SAs, which are often called CHILD_SAs. IPsec SAs define which network traffic is to be secured and how it has to be encrypted and authenticated.

To ensure that the peer with which an IKE_SA is established is really who it claims to be it has to be authenticated.

strongSwan provides several methods to do this:

To ensure that the peer with which an IKE_SA is established is really who it claims to be it has to be authenticated. strongSwan provides several methods to do this:

Pre-Shared-Key (PSK): A pre-shared-key is an easy to deploy option but it requires strong secrets to be secure. If the PSK is known to many users (which is often the case with IKEv1 XAuth with PSK) any user who knows the secret could impersonate the gateway. Therefore this method is not recommended for large scale deployments.

Configuration Files

strongSwan is usually controlled with the ipsec command. ipsec start will start the starter daemon which in turn starts and configures the keying daemon charon Connections defined as conn sections in ipsec.conf can be started on three different occasions:

On startup: Connections configured with auto=start will automatically be established when the daemon is started. They are not automatically restarted when they go down for some reason. You need to specify other configuration settings (dpdaction and/or closeaction) to restart them automatically, but even then, the setup is not bullet-proof and will potentially leak packets. You are encouraged to use auto=route and read the SecurityRecommendations to take care of any problems.

On traffic: If auto=route is used, IPsec trap policies for the configured traffic (left|rightsubnet) will be installed and traffic matching these policies will trigger acquire events that cause the daemon to establish the connection.

Manually: A connection that uses auto=add has to be established manually with ipsec up <name> or by a peer.

After an SA has been established ipsec down may be used to tear down the IKE_SA or individual CHILD_SAs.

Whenever the ipsec.conf file is changed it may be reloaded with ipsec update or ipsec reload. Already established connections are not affected by these commands, if that is required ipsec restart must be used.

Phase 1 of IKE Tunnel Negotiation

Phase1 of an AutoKey Internet Key Exchange (IKE) tunnel negotiation consists of the exchange of proposals for how to authenticate and secure the channel. The participants exchange proposals for acceptable security services such as:

- Encryption algorithms—Data Encryption Standard (DES), triple Data Encryption Standard (3DES), and Advanced Encryption Standard (AES).
- Authentication algorithms—Message Digest 5 (MD5) and Secure Hash Algorithm (SHA).
- Diffie-Hellman (DH) group.
- Preshared key or RSA/DSA certificates.

A successful Phase1 negotiation concludes when both ends of the tunnel agree to accept at least one set of the Phase 1 security parameters proposed and then process them. accept.

Phase2 of IKE Tunnel Negotiation

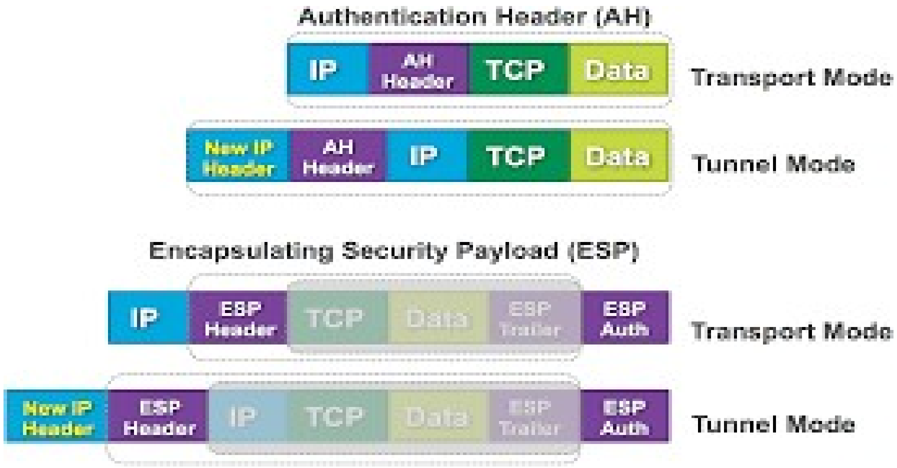
After the participants have established a secure and authenticated channel, they proceed through Phase2, in which they negotiate security associations (SAs) to secure the data to be transmitted through the IPsec tunnel.

Similar to the process for Phase 1, the participants exchange proposals to determine which security parameters to employ in the SA. A Phase 2 proposal also includes a security protocol—either Encapsulating Security Payload (ESP) or Authentication Header (AH)—and selected encryption and authentication algorithms. The proposal can also specify a Diffie-Hellman (DH) group, if Perfect Forward Secrecy (PFS) is desired.

ESP and AH Headers :

The **AH protocol** provides a mechanism for authentication only. AH provides data integrity, data origin authentication, and an optional replay protection service. Data integrity is ensured by using a message digest that is generated by an algorithm such as HMAC-MD5 or HMAC-SHA. AH authenticates IP headers and their payloads, with the exception of certain header fields that can be legitimately changed in transit, such as the Time To Live (TTL) field.

The **ESP protocol** provides data confidentiality (encryption) and authentication (data integrity, data origin authentication, and replay protection). ESP can be used with confidentiality only, authentication only, or both confidentiality and authentication. When ESP provides authentication functions, it uses the same algorithms as AH, but the coverage is different. AH-style authentication authenticates the entire IP packet, including the outer IP header, while the ESP authentication mechanism authenticates only the IP datagram portion of the IP packet.

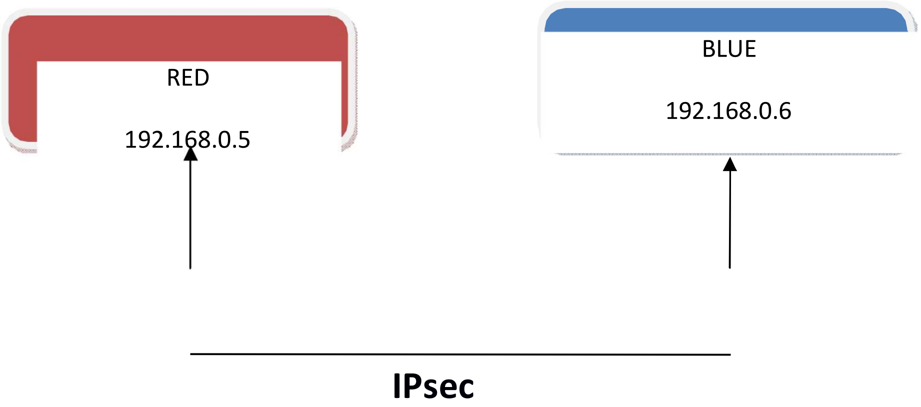


The IPsec standards define two distinct modes of IPsec operation,**transport mode** and **tunnel mode**. The modes do not affect the encoding of packets. The packets are protected by AH, ESP, or both in each mode. The modes differ in policy application when the inner packet is an IP packet, as follows:

- In transport mode, the outer header determines the IPsec policy that protects the inner IP packet.

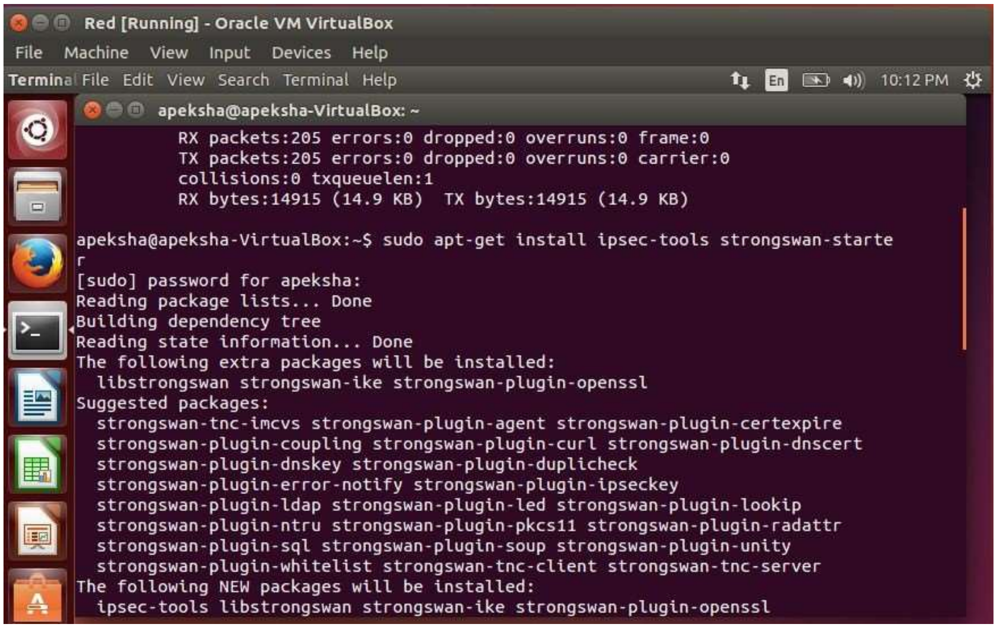
- In tunnel mode, the inner IP packet determines the IPsec policy that protects its contents.

In this lab we implement IPSec between two virtual machines like below :



Step 1. Install strongswan on both the machines.

```
sudo apt-get update
sudo apt-get install ipsec-tools strongswan-starter
```

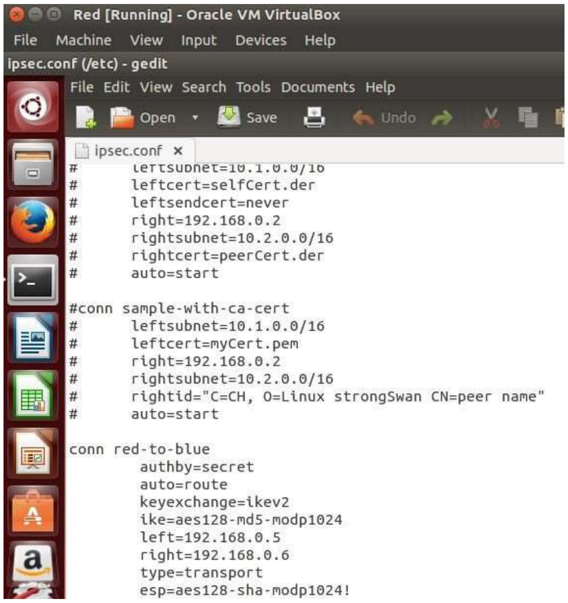

Red [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal File Edit View Search Terminal Help
En
10:12 PM
apeksha@apeksha-VirtualBox: ~

```
          RX packets:205 errors:0 dropped:0 overruns:0 frame:0
          TX packets:205 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:14915 (14.9 KB)  TX bytes:14915 (14.9 KB)

apeksha@apeksha-VirtualBox:~$ sudo apt-get install ipsec-tools strongswan-starte
r
[sudo] password for apeksha:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libstrongswan strongswan-ike strongswan-plugin-openssl
Suggested packages:
  strongswan-tnc-imcvs strongswan-plugin-agent strongswan-plugin-certexpire
  strongswan-plugin-coupling strongswan-plugin-curl strongswan-plugin-dnscert
  strongswan-plugin-dnskey strongswan-plugin-duplicheck
  strongswan-plugin-error-notify strongswan-plugin-ipseckey
  strongswan-plugin-ldap strongswan-plugin-led strongswan-plugin-lookip
  strongswan-plugin-ntru strongswan-plugin-pkcs11 strongswan-plugin-radattr
  strongswan-plugin-sql strongswan-plugin-soup strongswan-plugin-unity
  strongswan-plugin-whitelist strongswan-tnc-client strongswan-tnc-server
The following NEW packages will be installed:
  ipsec-tools libstrongswan strongswan-ike strongswan-plugin-openssl
```

t

the configuration files on both Red and Blue server and save the new connection policy.

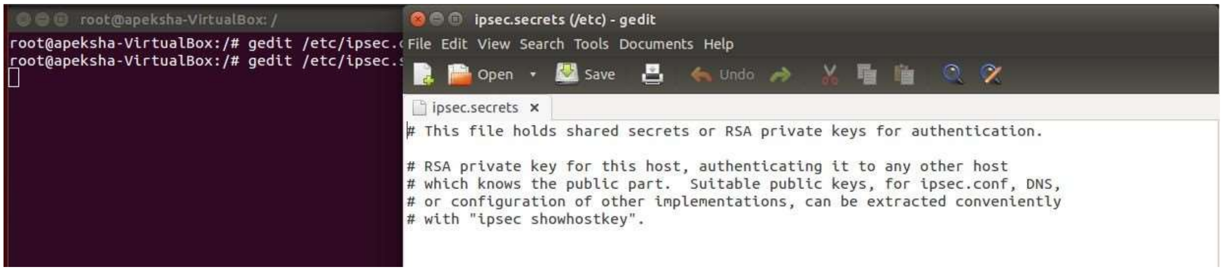
sudo edit etc/ipsec.conf


Red [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
ipsec.conf (/etc) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
ipsec.conf

```
#       leftsubnet=10.1.0.0/16
#       leftcert=selfCert.der
#       leftsendcert=never
#       right=192.168.0.2
#       rightsubnet=10.2.0.0/16
#       rightcert=peerCert.der
#       auto=start

#conn sample-with-ca-cert
#       leftsubnet=10.1.0.0/16
#       leftcert=myCert.pem
#       right=192.168.0.2
#       rightsubnet=10.2.0.0/16
#       rightid="C=CH, O=Linux strongSwan CN=peer name"
#       auto=start

conn red-to-blue
        authby=secret
        auto=route
        keyexchange=ikev2
        ike=aes128-md5-modp1024
        left=192.168.0.5
        right=192.168.0.6
        type=transport
        esp=aes128-sha-modp1024!
```

Step 3. Edit the secret file

sudo gedit /etc/ipsec.secret



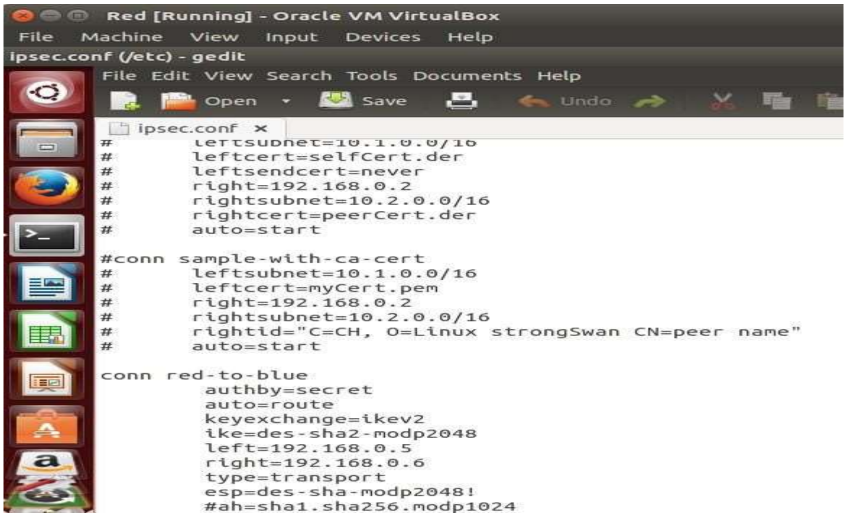
root@apeksha-VirtualBox: /
ipsec.secrets (/etc) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
ipsec.secrets

```
root@apeksha-VirtualBox:/# gedit /etc/ipsec.c
root@apeksha-VirtualBox:/# gedit /etc/ipsec.s
```

```
# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.  Suitable public keys, for ipsec.conf, DNS,
# or configuration of other implementations, can be extracted conveniently
# with "ipsec showhostkey".
```

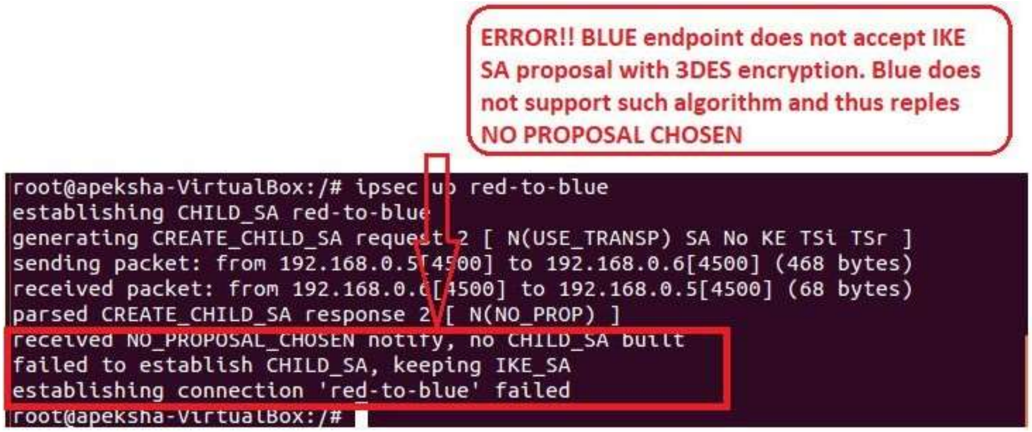
Step 4. Test the connection

sudo ipsec up connection name



Red [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
ipsec.conf (/etc) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
ipsec.conf

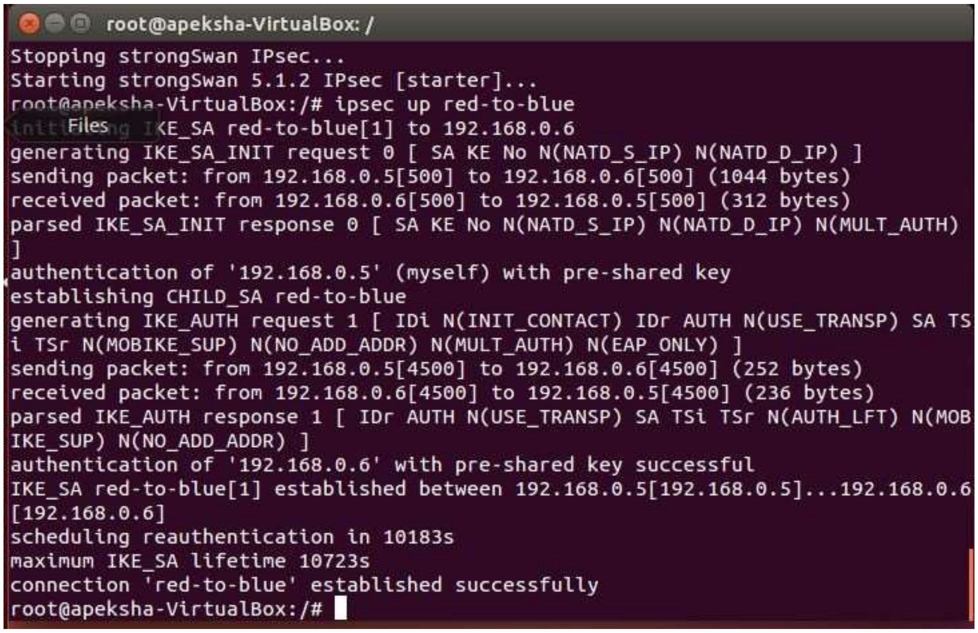
```
#       leftsubnet=10.1.0.0/16
#       leftcert=selfCert.der
#       leftsendcert=never
#       right=192.168.0.2
#       rightsubnet=10.2.0.0/16
#       rightcert=peerCert.der
#       auto=start

#conn sample-with-ca-cert
#       leftsubnet=10.1.0.0/16
#       leftcert=myCert.pem
#       right=192.168.0.2
#       rightsubnet=10.2.0.0/16
#       rightid="C=CH, O=Linux strongSwan CN=peer name"
#       auto=start

conn red-to-blue
        authby=secret
        auto=route
        keyexchange=ikev2
        ike=des-sha2-modp2048
        left=192.168.0.5
        right=192.168.0.6
        type=transport
        esp=des-sha-modp2048!
        #ah=sha1.sha256.modp1024
```



ERROR!! BLUE endpoint does not accept IKE SA proposal with 3DES encryption. Blue does not support such algorithm and thus reples NO PROPOSAL CHOSEN

```
root@apeksha-VirtualBox:/# ipsec up red-to-blue
establishing CHILD_SA red-to-blue
generating CREATE_CHILD_SA request 2 [ N(USE_TRANSP) SA No KE TSi TSr ]
sending packet: from 192.168.0.5[4500] to 192.168.0.6[4500] (468 bytes)
received packet: from 192.168.0.6[4500] to 192.168.0.5[4500] (68 bytes)
parsed CREATE_CHILD_SA response 2 [ N(NO_PROP) ]
received NO_PROPOSAL_CHOSEN notify, no CHILD_SA built
failed to establish CHILD_SA, keeping IKE_SA
establishing connection 'red-to-blue' failed
root@apeksha-VirtualBox:/#
```

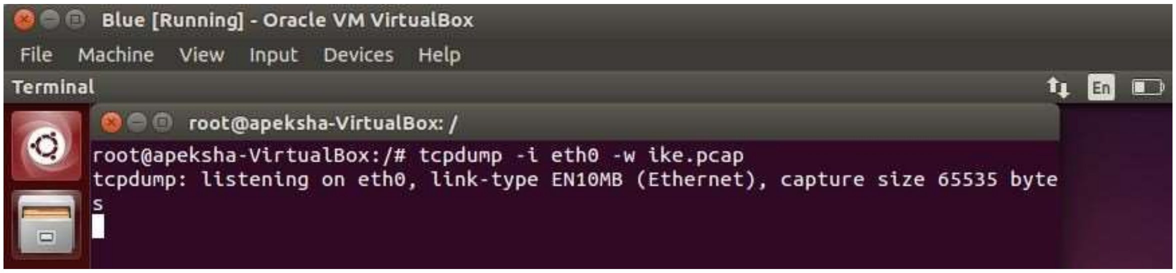
The connection is not established due to inconsistency in encryption algorithms. We again go ahead and reflect the same algorithms as on Blue server.



```
Stopping strongSwan IPsec...
Starting strongSwan 5.1.2 IPsec [starter]...
root@apeksha-VirtualBox:/# ipsec up red-to-blue
initiating IKE_SA red-to-blue[1] to 192.168.0.6
generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) ]
sending packet: from 192.168.0.5[500] to 192.168.0.6[500] (1044 bytes)
received packet: from 192.168.0.6[500] to 192.168.0.5[500] (312 bytes)
parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(MULT_AUTH)
]
authentication of '192.168.0.5' (myself) with pre-shared key
establishing CHILD_SA red-to-blue
generating IKE_AUTH request 1 [ IDi N(INIT_CONTACT) IDr AUTH N(USE_TRANSP) SA TS
i TSr N(MOBIKE_SUP) N(NO_ADD_ADDR) N(MULT_AUTH) N(EAP_ONLY) ]
sending packet: from 192.168.0.5[4500] to 192.168.0.6[4500] (252 bytes)
received packet: from 192.168.0.6[4500] to 192.168.0.5[4500] (236 bytes)
parsed IKE_AUTH response 1 [ IDr AUTH N(USE_TRANSP) SA TSi TSr N(AUTH_LFT) N(MOB
IKE_SUP) N(NO_ADD_ADDR) ]
authentication of '192.168.0.6' with pre-shared key successful
IKE_SA red-to-blue[1] established between 192.168.0.5[192.168.0.5]...192.168.0.6
[192.168.0.6]
scheduling reauthentication in 10183s
maximum IKE_SA lifetime 10723s
connection 'red-to-blue' established successfully
root@apeksha-VirtualBox:/#
```

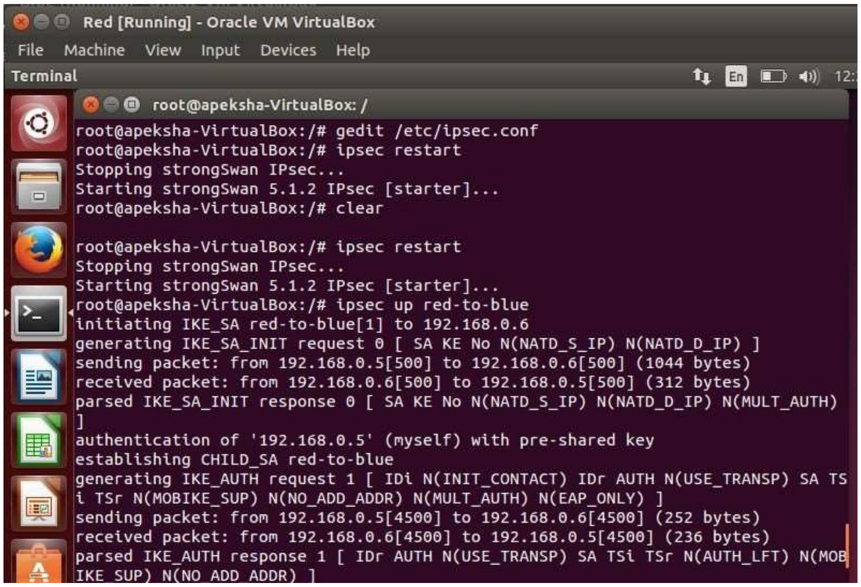
We now capture the packets on Blue server and try to analyze them.

We keep tcpdump in listening mode on Blue server


Blue [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
En
root@apeksha-VirtualBox: /

```
root@apeksha-VirtualBox:/# tcpdump -i eth0 -w ike.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 byte
s
```

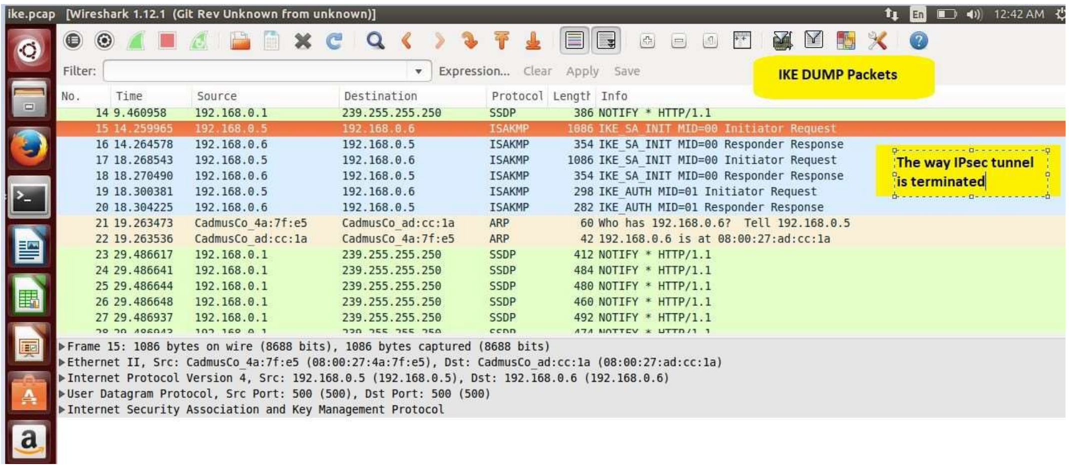
We now restart ipsec on Red server


Red [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
En
12:
root@apeksha-VirtualBox: /

```
root@apeksha-VirtualBox:/# gedit /etc/ipsec.conf
root@apeksha-VirtualBox:/# ipsec restart
Stopping strongSwan IPsec...
Starting strongSwan 5.1.2 IPsec [starter]...
root@apeksha-VirtualBox:/# clear

root@apeksha-VirtualBox:/# ipsec restart
Stopping strongSwan IPsec...
Starting strongSwan 5.1.2 IPsec [starter]...
root@apeksha-VirtualBox:/# ipsec up red-to-blue
initiating IKE_SA red-to-blue[1] to 192.168.0.6
generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) ]
sending packet: from 192.168.0.5[500] to 192.168.0.6[500] (1044 bytes)
received packet: from 192.168.0.6[500] to 192.168.0.5[500] (312 bytes)
parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(MULT_AUTH)
]
authentication of '192.168.0.5' (myself) with pre-shared key
establishing CHILD_SA red-to-blue
generating IKE_AUTH request 1 [ IDi N(INIT_CONTACT) IDr AUTH N(USE_TRANSP) SA TS
i TSr N(MOBIKE_SUP) N(NO_ADD_ADDR) N(MULT_AUTH) N(EAP_ONLY) ]
sending packet: from 192.168.0.5[4500] to 192.168.0.6[4500] (252 bytes)
received packet: from 192.168.0.6[4500] to 192.168.0.5[4500] (236 bytes)
parsed IKE_AUTH response 1 [ IDr AUTH N(USE_TRANSP) SA TSi TSr N(AUTH_LFT) N(MOB
IKE_SUP) N(NO_ADD_ADDR) ]
```

edit

We now open the captured file in wireshark on Blue server

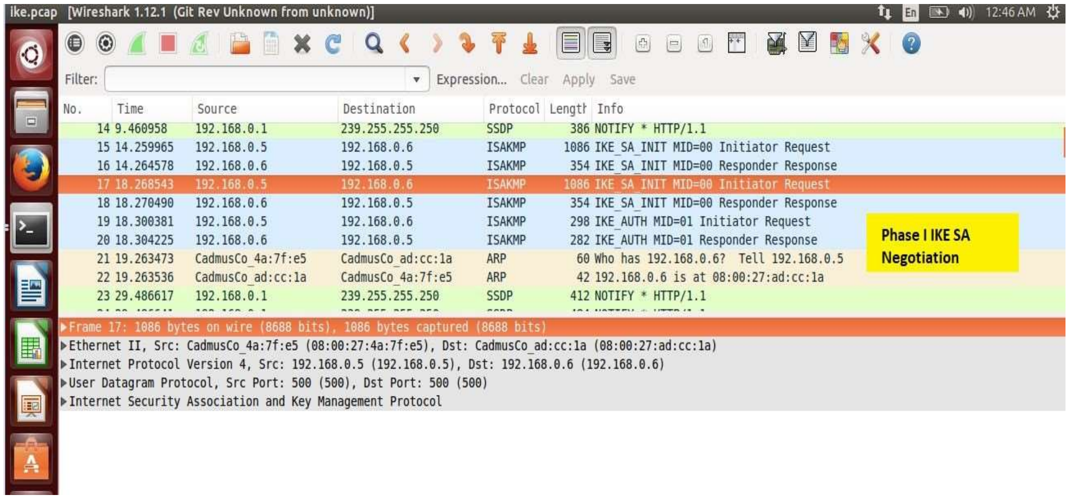


IKE DUMP Packets

The way IPsec tunnel is terminated

```
No.   Time       Source             Destination        Protocol Length Info
   14 9.460958   192.168.0.1        239.255.255.250    SSDP      386 NOTIFY * HTTP/1.1
   15 14.259965  192.168.0.5        192.168.0.6        ISAKMP   1086 IKE_SA_INIT MID=00 Initiator Request
   16 14.264578  192.168.0.6        192.168.0.5        ISAKMP    354 IKE_SA_INIT MID=00 Responder Response
   17 18.268543  192.168.0.5        192.168.0.6        ISAKMP   1086 IKE_SA_INIT MID=00 Initiator Request
   18 18.270490  192.168.0.6        192.168.0.5        ISAKMP    354 IKE_SA_INIT MID=00 Responder Response
   19 18.300381  192.168.0.5        192.168.0.6        ISAKMP    298 IKE_AUTH MID=01 Initiator Request
   20 18.304225  192.168.0.6        192.168.0.5        ISAKMP    282 IKE_AUTH MID=01 Responder Response
   21 19.263473  CadmusCo_4a:7f:e5  CadmusCo_ad:cc:1a  ARP        60 Who has 192.168.0.6?  Tell 192.168.0.5
   22 19.263536  CadmusCo_ad:cc:1a  CadmusCo_4a:7f:e5  ARP        42 192.168.0.6 is at 08:00:27:ad:cc:1a
   23 29.486617  192.168.0.1        239.255.255.250    SSDP      412 NOTIFY * HTTP/1.1
   24 29.486641  192.168.0.1        239.255.255.250    SSDP      484 NOTIFY * HTTP/1.1
   25 29.486644  192.168.0.1        239.255.255.250    SSDP      480 NOTIFY * HTTP/1.1
   26 29.486648  192.168.0.1        239.255.255.250    SSDP      460 NOTIFY * HTTP/1.1
   27 29.486937  192.168.0.1        239.255.255.250    SSDP      492 NOTIFY * HTTP/1.1
```

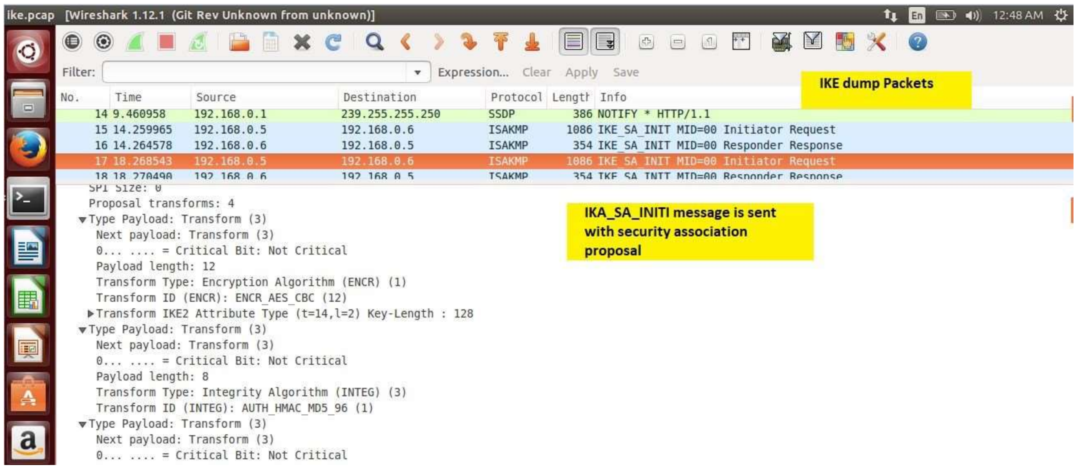
```
▶Frame 15: 1086 bytes on wire (8688 bits), 1086 bytes captured (8688 bits)
▶Ethernet II, Src: CadmusCo_4a:7f:e5 (08:00:27:4a:7f:e5), Dst: CadmusCo_ad:cc:1a (08:00:27:ad:cc:1a)
▶Internet Protocol Version 4, Src: 192.168.0.5 (192.168.0.5), Dst: 192.168.0.6 (192.168.0.6)
▶User Datagram Protocol, Src Port: 500 (500), Dst Port: 500 (500)
▶Internet Security Association and Key Management Protocol
```



Phase I IKE SA Negotiation

```
No.   Time       Source             Destination        Protocol Length Info
   14 9.460958   192.168.0.1        239.255.255.250    SSDP      386 NOTIFY * HTTP/1.1
   15 14.259965  192.168.0.5        192.168.0.6        ISAKMP   1086 IKE_SA_INIT MID=00 Initiator Request
   16 14.264578  192.168.0.6        192.168.0.5        ISAKMP    354 IKE_SA_INIT MID=00 Responder Response
   17 18.268543  192.168.0.5        192.168.0.6        ISAKMP   1086 IKE_SA_INIT MID=00 Initiator Request
   18 18.270490  192.168.0.6        192.168.0.5        ISAKMP    354 IKE_SA_INIT MID=00 Responder Response
   19 18.300381  192.168.0.5        192.168.0.6        ISAKMP    298 IKE_AUTH MID=01 Initiator Request
   20 18.304225  192.168.0.6        192.168.0.5        ISAKMP    282 IKE_AUTH MID=01 Responder Response
   21 19.263473  CadmusCo_4a:7f:e5  CadmusCo_ad:cc:1a  ARP        60 Who has 192.168.0.6?  Tell 192.168.0.5
   22 19.263536  CadmusCo_ad:cc:1a  CadmusCo_4a:7f:e5  ARP        42 192.168.0.6 is at 08:00:27:ad:cc:1a
   23 29.486617  192.168.0.1        239.255.255.250    SSDP      412 NOTIFY * HTTP/1.1
```

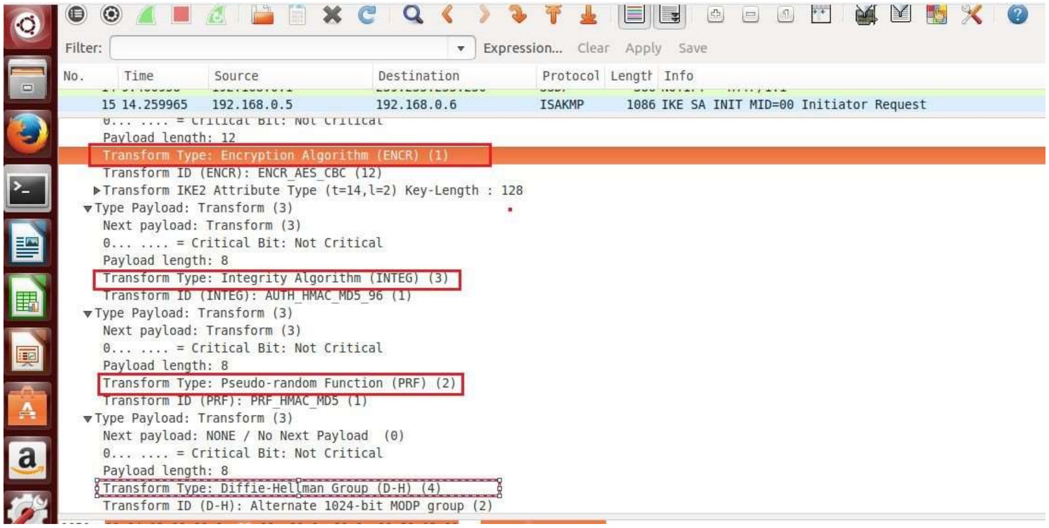
```
▶Frame 17: 1086 bytes on wire (8688 bits), 1086 bytes captured (8688 bits)
▶Ethernet II, Src: CadmusCo_4a:7f:e5 (08:00:27:4a:7f:e5), Dst: CadmusCo_ad:cc:1a (08:00:27:ad:cc:1a)
▶Internet Protocol Version 4, Src: 192.168.0.5 (192.168.0.5), Dst: 192.168.0.6 (192.168.0.6)
▶User Datagram Protocol, Src Port: 500 (500), Dst Port: 500 (500)
▶Internet Security Association and Key Management Protocol
```


ike.pcap [Wireshark 1.12.1 (Git Rev Unknown from unknown)]
12:48 AM
Filter:
Expression... Clear Apply Save

IKE dump Packets

```
No.   Time          Source        Destination       Protocol  Length Info
   14 9.460958      192.168.0.1   239.255.255.250   SSDP        386 NOTIFY * HTTP/1.1
   15 14.259965     192.168.0.5   192.168.0.6       ISAKMP     1086 IKE_SA_INIT MID=00 Initiator Request
   16 14.264578     192.168.0.6   192.168.0.5       ISAKMP      354 IKE_SA_INIT MID=00 Responder Response
   17 18.268543     192.168.0.5   192.168.0.6       ISAKMP     1086 IKE_SA_INIT MID=00 Initiator Request
   18 18.270490     192.168.0.6   192.168.0.5       ISAKMP      354 IKE_SA_INIT MID=00 Responder Response
    SPI Size: 0
    Proposal transforms: 4
   ▼Type Payload: Transform (3)
     Next payload: Transform (3)
     0... .... = Critical Bit: Not Critical
     Payload length: 12
     Transform Type: Encryption Algorithm (ENCR) (1)
     Transform ID (ENCR): ENCR_AES_CBC (12)
    ▶Transform IKE2 Attribute Type (t=14,l=2) Key-Length : 128
   ▼Type Payload: Transform (3)
     Next payload: Transform (3)
     0... .... = Critical Bit: Not Critical
     Payload length: 8
     Transform Type: Integrity Algorithm (INTEG) (3)
     Transform ID (INTEG): AUTH_HMAC_MD5_96 (1)
   ▼Type Payload: Transform (3)
     Next payload: Transform (3)
     0... .... = Critical Bit: Not Critical
```

IKA_SA_INITI message is sent with security association proposal

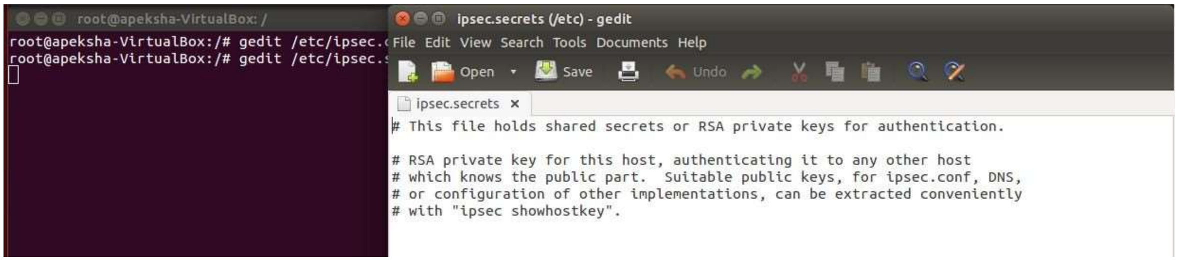


```
No.   Time          Source        Destination       Protocol  Length Info
   15 14.259965     192.168.0.5   192.168.0.6       ISAKMP     1086 IKE SA INIT MID=00 Initiator Request
     0... .... = Critical Bit: Not Critical
     Payload length: 12
     Transform Type: Encryption Algorithm (ENCR) (1)
     Transform ID (ENCR): ENCR_AES_CBC (12)
    ▶Transform IKE2 Attribute Type (t=14,l=2) Key-Length : 128
   ▼Type Payload: Transform (3)
     Next payload: Transform (3)
     0... .... = Critical Bit: Not Critical
     Payload length: 8
     Transform Type: Integrity Algorithm (INTEG) (3)
     Transform ID (INTEG): AUTH_HMAC_MD5_96 (1)
   ▼Type Payload: Transform (3)
     Next payload: Transform (3)
     0... .... = Critical Bit: Not Critical
     Payload length: 8
     Transform Type: Pseudo-random Function (PRF) (2)
     Transform ID (PRF): PRF_HMAC_MD5 (1)
   ▼Type Payload: Transform (3)
     Next payload: NONE / No Next Payload  (0)
     0... .... = Critical Bit: Not Critical
     Payload length: 8
     Transform Type: Diffie-Hellman Group (D-H) (4)
     Transform ID (D-H): Alternate 1024-bit MODP group (2)
```

Next we understand the Pre-shared Key.

On Red server open secret file.

sudo gedit /etc/secret

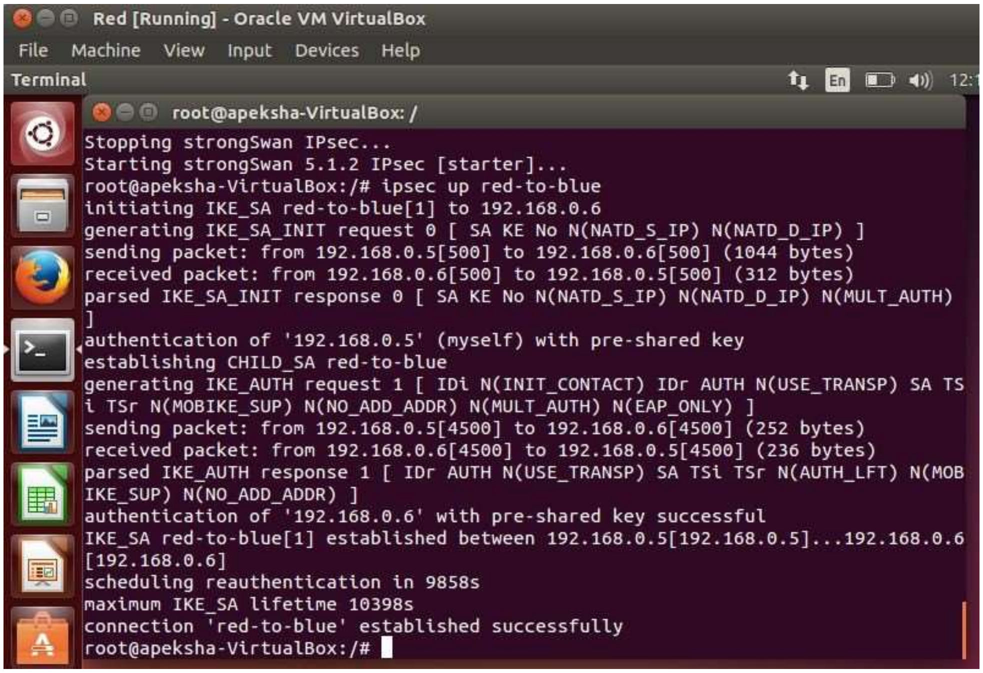


root@apeksha-VirtualBox: /
ipsec.secrets (/etc) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
ipsec.secrets

```
root@apeksha-VirtualBox:/# gedit /etc/ipsec.c
root@apeksha-VirtualBox:/# gedit /etc/ipsec.s
```

```
# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.  Suitable public keys, for ipsec.conf, DNS,
# or configuration of other implementations, can be extracted conveniently
# with "ipsec showhostkey".
```

If invalid PSK is configured the connection is failed.Recorrect the PSK and test again. Connection established.

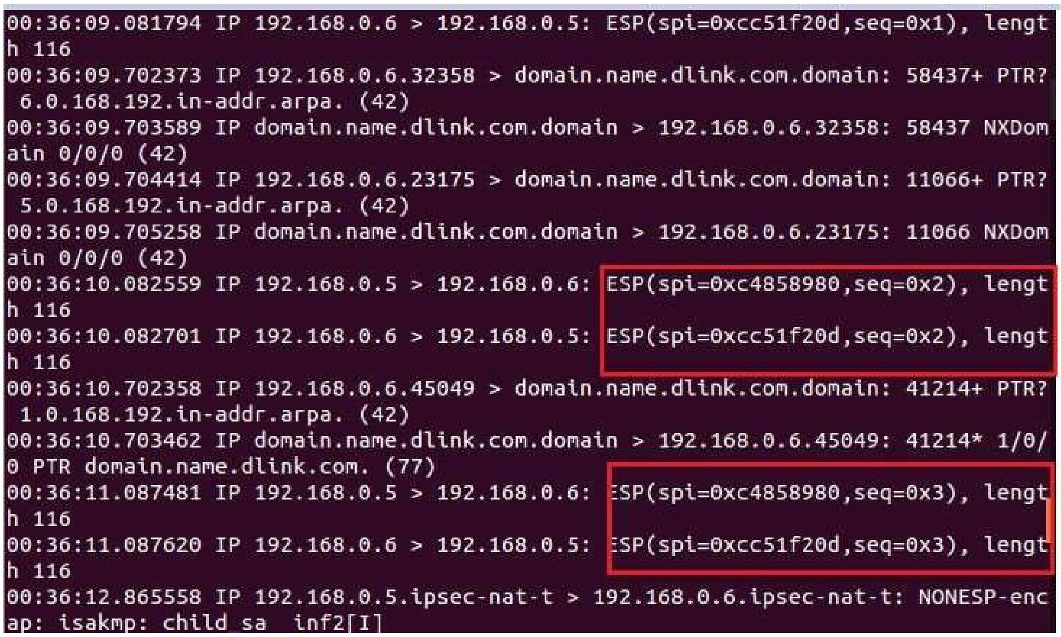


Red [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
En
12:1
root@apeksha-VirtualBox: /

```
Stopping strongSwan IPsec...
Starting strongSwan 5.1.2 IPsec [starter]...
root@apeksha-VirtualBox:/# ipsec up red-to-blue
initiating IKE_SA red-to-blue[1] to 192.168.0.6
generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) ]
sending packet: from 192.168.0.5[500] to 192.168.0.6[500] (1044 bytes)
received packet: from 192.168.0.6[500] to 192.168.0.5[500] (312 bytes)
parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(MULT_AUTH)
]
authentication of '192.168.0.5' (myself) with pre-shared key
establishing CHILD_SA red-to-blue
generating IKE_AUTH request 1 [ IDi N(INIT_CONTACT) IDr AUTH N(USE_TRANSP) SA TS
i TSr N(MOBIKE_SUP) N(NO_ADD_ADDR) N(MULT_AUTH) N(EAP_ONLY) ]
sending packet: from 192.168.0.5[4500] to 192.168.0.6[4500] (252 bytes)
received packet: from 192.168.0.6[4500] to 192.168.0.5[4500] (236 bytes)
parsed IKE_AUTH response 1 [ IDr AUTH N(USE_TRANSP) SA TSi TSr N(AUTH_LFT) N(MOB
IKE_SUP) N(NO_ADD_ADDR) ]
authentication of '192.168.0.6' with pre-shared key successful
IKE_SA red-to-blue[1] established between 192.168.0.5[192.168.0.5]...192.168.0.6
[192.168.0.6]
scheduling reauthentication in 9858s
maximum IKE_SA lifetime 10398s
connection 'red-to-blue' established successfully
root@apeksha-VirtualBox:/# 
```

To see the analyze the packets we again keep tcpdump on Blue server in listening state. Perform a simple ping to Blue server from Red Server.

Ping 192.168.0.6



```
00:36:09.081794 IP 192.168.0.6 > 192.168.0.5: ESP(spi=0xcc51f20d,seq=0x1), lengt
h 116
00:36:09.702373 IP 192.168.0.6.32358 > domain.name.dlink.com.domain: 58437+ PTR?
 6.0.168.192.in-addr.arpa. (42)
00:36:09.703589 IP domain.name.dlink.com.domain > 192.168.0.6.32358: 58437 NXDom
ain 0/0/0 (42)
00:36:09.704414 IP 192.168.0.6.23175 > domain.name.dlink.com.domain: 11066+ PTR?
 5.0.168.192.in-addr.arpa. (42)
00:36:09.705258 IP domain.name.dlink.com.domain > 192.168.0.6.23175: 11066 NXDom
ain 0/0/0 (42)
00:36:10.082559 IP 192.168.0.5 > 192.168.0.6: ESP(spi=0xc4858980,seq=0x2), lengt
h 116
00:36:10.082701 IP 192.168.0.6 > 192.168.0.5: ESP(spi=0xcc51f20d,seq=0x2), lengt
h 116
00:36:10.702358 IP 192.168.0.6.45049 > domain.name.dlink.com.domain: 41214+ PTR?
 1.0.168.192.in-addr.arpa. (42)
00:36:10.703462 IP domain.name.dlink.com.domain > 192.168.0.6.45049: 41214* 1/0/
0 PTR domain.name.dlink.com. (77)
00:36:11.087481 IP 192.168.0.5 > 192.168.0.6: ESP(spi=0xc4858980,seq=0x3), lengt
h 116
00:36:11.087620 IP 192.168.0.6 > 192.168.0.5: ESP(spi=0xcc51f20d,seq=0x3), lengt
h 116
00:36:12.865558 IP 192.168.0.5.ipsec-nat-t > 192.168.0.6.ipsec-nat-t: NONESP-enc
ap: isakmp: child_sa  inf2[I]
```

4. Conclusion:

IPsec incorporates all of the most commonly employed security services, including authentication, integrity, confidentiality, encryption and non repudiation. However, the major drawbacks to IPsec are its complexity and the confusing nature of its associated documentation. In spite of these various drawbacks, IPsec is believed by many to be one of the best security systems available.