LAB – 2 ASSIGNMENT

ET 2595 Network and System Security

Jan 2024

Name : NIKHIL CHALIKONDA

Id No:

Mail: nich20@student.bth.se

I. *Tasks Performing*

Objective 1: [v3_ca]

This section enumerates the parameters linked to the [v3_ca] extension.:

```
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always, issuer
basicConstraints        = critical, CA:true
keyUsage                = critical, digitalSignature, cRLSign, keyCertsign
```

- 1. **SubjectKeyIdentifier Extension**: This identifies certificates with a specific public key using a 'hash' value or a hexadecimal string. OpenSSL calculates this hash as outlined in RFC 5280.

- 2. **AuthorityKeyIdentifier Extension**: This extension, when set as 'keyid:always, issuer', links a private key to its corresponding public key. The process involves the utilization of two distinct values: 'keyid' to replicate the subject key identifier from the issuer certificate, and 'issuer' to duplicate the distinguished name (DN) and serial number of the issuer from their respective certificate.

- 3. **BasicConstraints Extension**: When set to 'critical, CA:true', this extension helps determine if the certificate is a certificate authority (CA). Setting it as 'CA:true' means the certificate is a certificate authority. Marking it as 'critical' is essential as it marks the extension as critical.

- 4. **KeyUsage Extension**: This extension, set as 'critical, digitalSignature, cRLSign, keyCertSign', defines how the key associated with the certificate can be used. Specifically, it permits the key to verify signatures on both public key certificates and Certificate Revocation Lists (CRLs)..

Objective 2: [v3_intermediate_ca]

Within this section, you'll locate the parameters associated with the [v3_intermediate_ca] extension.

```
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always, issuer
basicConstraints        = critical, CA:true, pathlen:0
keyUsage                = critical, digitalSignature, cRLSign, keyCertsign
```

In Task 2, the parameters remain consistent with those detailed in Task 1, save for a distinctive alteration involving the assignment of 'pathlen:0'. This specific modification exclusively applies to the endorsement of concluding user certificates. The imperative criterion stipulates that the 'pathlen' value necessitates being either precisely 0 or a numerical value surpassing 0 to ensure compliance.

Performing of Tasks 3: [usr_cert]

This part details the settings associated with the [usr_cert] extension:

```
basicConstraints        = CA:FALSE
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid, issuer
extendedKeyUsage        = clientAuth, emailProtection
keyUsage                = critical, nonRepudiation, digitalSignature,
                          keyEncipherment
```

- **basicConstraints:** When given the value CA:FALSE, it indicates that the certificate is not a Certificate Authority (CA) certificate.
- **nscomment = "OpenSSL Generated Certificate":** This is assigned so that it appears in Netscape's comment listbox.
- **subjectKeyIdentifier=hash:** This is a way of identifying certificates using their public keys.
- **authorityKeyIdentifier=keyid,issuer:** This helps in identifying the public key that can be traced back to the private key used to sign the certificate.

Performing of Task 4: [server_cert]

Within this segment, you'll discover specific information outlining the parameters attributed to the [server_cert] extension.

```
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid, issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
```

- **basicConstraints=CA:FALSE**: This indicates it's not a certificate used for being a Certificate Authority (CA).
- **subjectKeyIdentifier=hash**: This is a way to spot certificates based on their public keys.
- **authorityKeyIdentifier=keyid,issuer**: It helps locate the public key connected to the private key used for signing the certificate.
- **keyUsage**: This section lists the allowed uses for the key. It can include types like digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment, keyAgreement, keyCertSign, cRLSign, encipherOnly, and decipherOnly.

    - **keyEncipherment**: Used when a public key carries private keys for transport.
    - **digitalSignature**: Used to check digital signatures on things other than certificates and lists. These bits confirm things like entity authentication and integrity services.

Performing of Task 5: [policy_match] and [policy_anything]

```
[ policy_match ]
    countryName             = match
    stateOrProvinceName     = match
    organizationName        = match
    organizationalUnitName  = optional
    commanName              = supplied
    emailAddress            = optional

[ policy_anything ]
    countryName             = optional
    stateOrProvinceName     = optional
    organizationName        = optional
    organizationalUnitName  = optional
    commanName              = supplied
    emailAddress            = optional
```

Certainly! In the "policy_match" policy, if a field is labeled as "match," it has to contain the exact information as that field in the CA's details. Fields marked as "supplied" must be present, while those labeled as "optional" don't have to be there, but they're allowed. Essentially, this policy mandates that any certificate issued by it must have the same country, state, and organization name as the CA.

Performing of Task 6:

Explore the method for generating a self-signed certificate through a command.

openssl req -config openssl.cnf -key private/root.key.pem -new -x509 -days 7300 -sha256 -extensions v3_ca -out certs/root.cert.pem

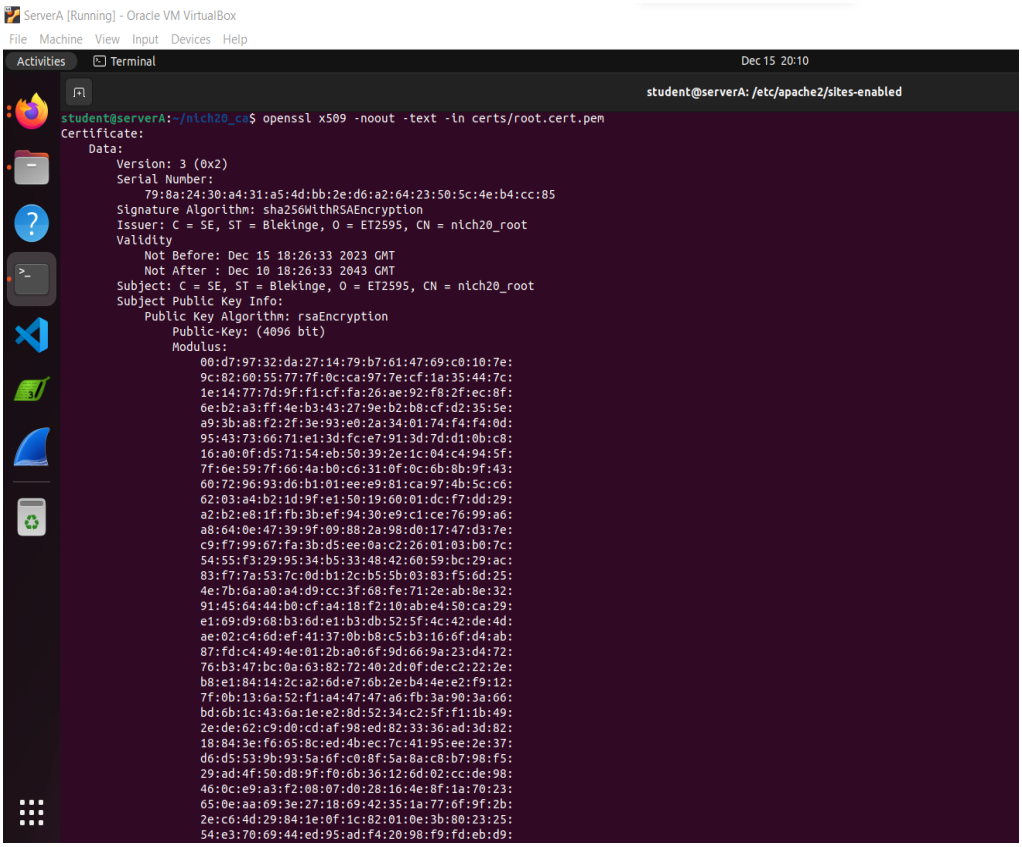
These directives serve distinct functions:

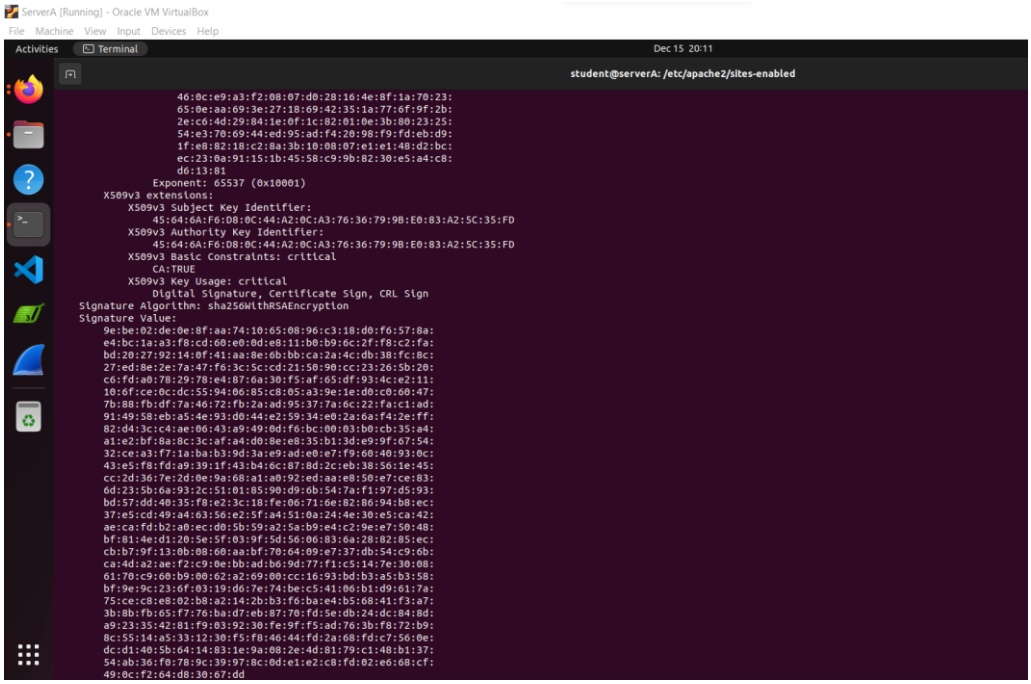
- 'config' designates a configuration file, such as openssl.cnf.
- 'key' denotes the file containing the private key.
- 'new' signals the creation of a new certificate request.
- 'x509' is used for self-signed certificate generation.
- 'days' permits setting the certificate's validity duration.
- 'extensions' indicates the inclusion of extensions in the certificate.
- 'out' specifies the file path for storing the resulting certificate.

Performing of Task 7:

The instruction employed to authenticate the root certificate.

openssl x509 -noout -text -in certs/root.cert.pem



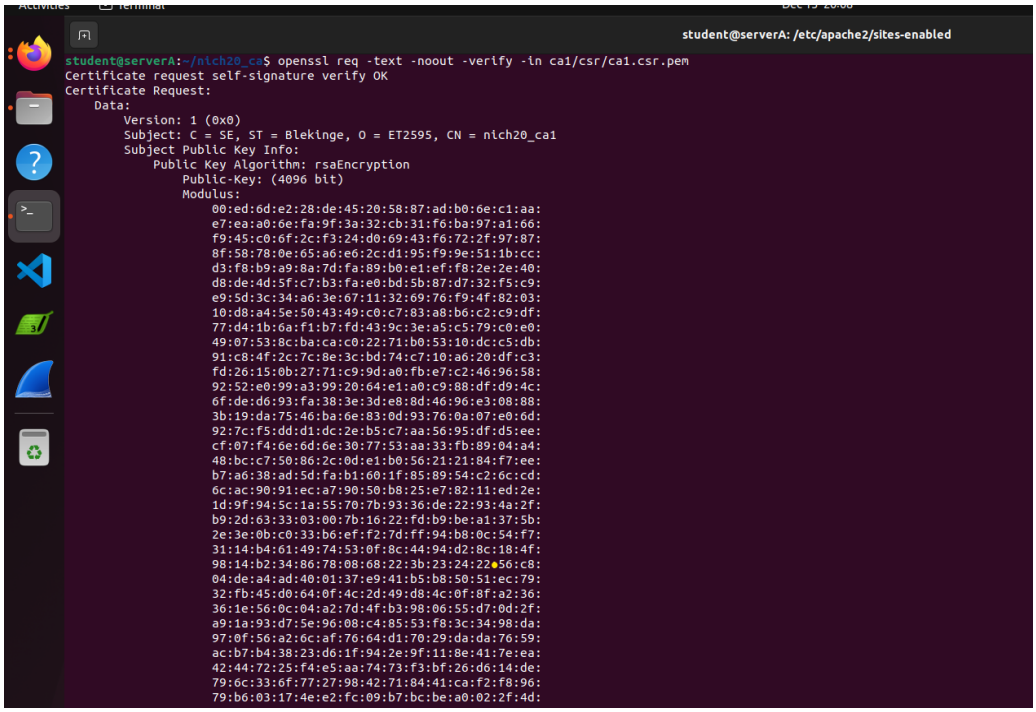


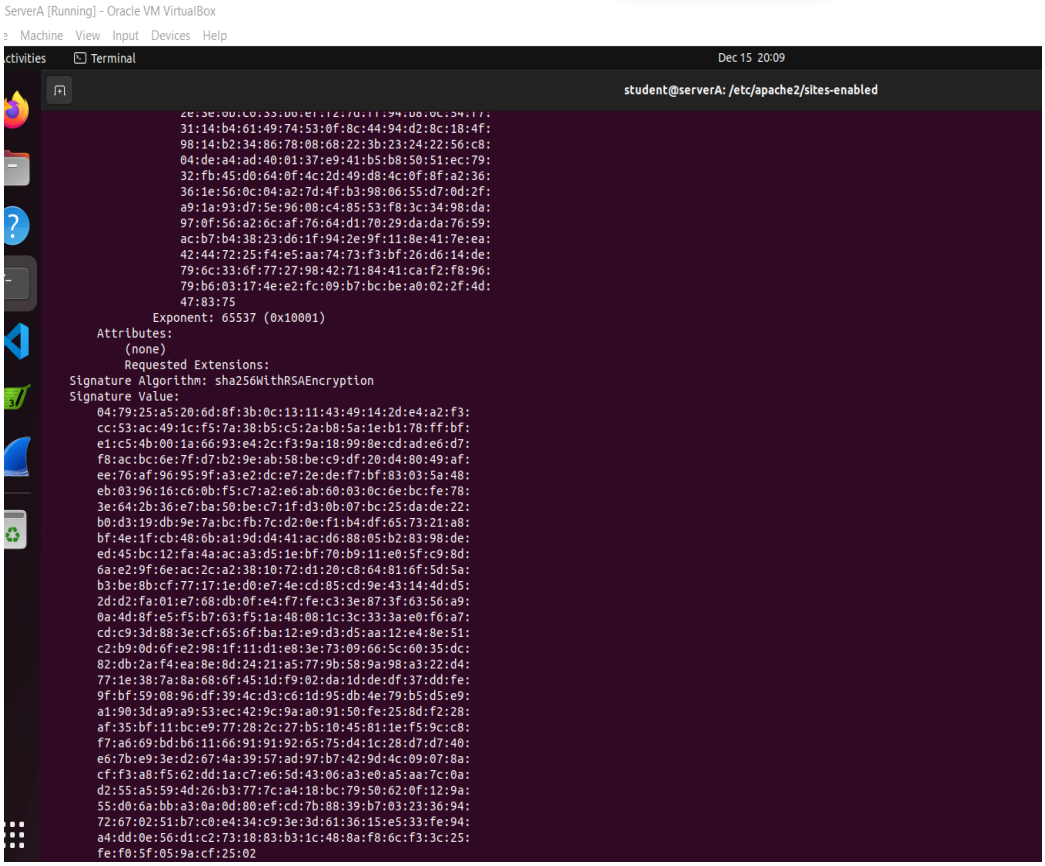
Performing of Task 8:

The command is utilized to verify the Certificate Signing Request (CSR).

openssl req -text -noout -verify -in ca1/csr/ca1.csr.pem

The Certificate Signing Request (CSR) is created using the RSA private key, and it appears as follows:





Performing of Task 9:

This command enables the generation of the CA1 certificate utilizing a Certificate Signing Request (CSR):

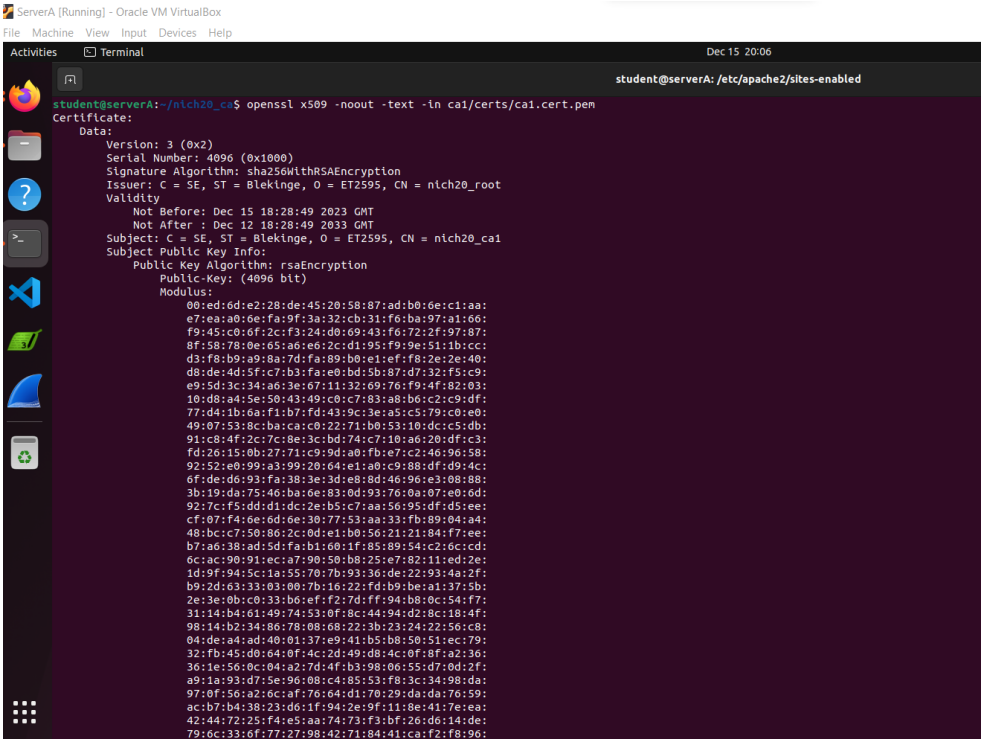
openssl ca -config openssl.cnf -extensions v3 intermediate ca -days 3650 -notext -md sha256 -in ca1/csr/ca1.csr.pem -out ca1/certs/ca1.cert.pem

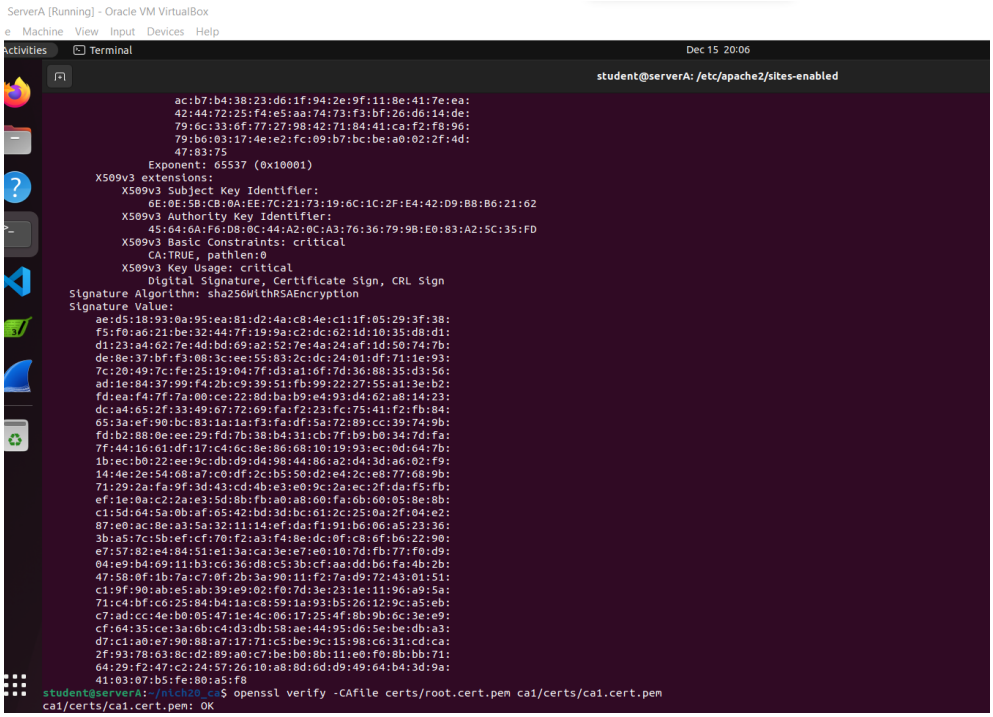
- 'config': It points to the 'openssl.cnf' configuration file necessary for the CA1 certificate signing process.
- 'extensions': Specifies the 'v3_intermediate_ca' extension, enabling the CA1 certificate to function as an intermediate certificate authority with essential capabilities.
- 'days 3650': Sets the certificate's validity period to 3650 days, equivalent to a 10-year duration.
- 'notext': Instructs the command to abstain from generating text-based output.
- 'md sha256': Specifies the utilization of the SHA-256 encryption algorithm for the certificate's signature generation.
- 'in': Identifies the input file.
- 'out': Designates the destination path for the resulting output file storage.

Performing of Task 10:

A sequence of commands is utilized to validate the authenticity and legitimacy of the CA1 certificate.

openssl x509 -noout -text -in ca1/certs/ca1.cert.pem
openssl verify -CAfile certs/root.cert.pem ca1/certs/ca1.cert.pem



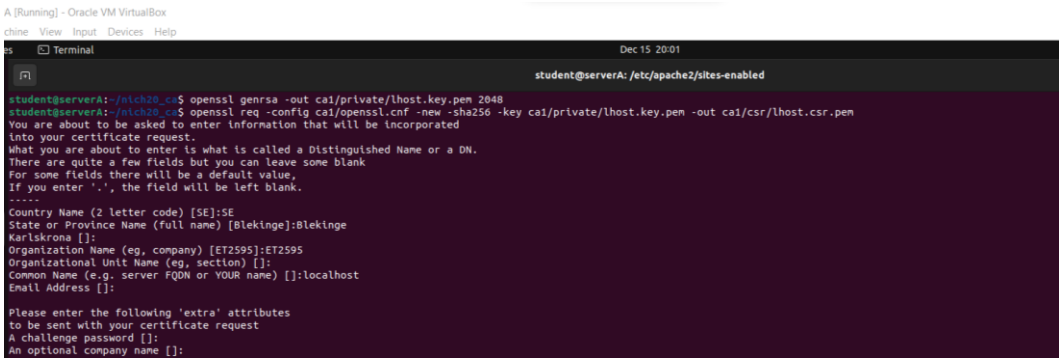


Performing of Task 11:

Below is the process used to generate a server certificate:

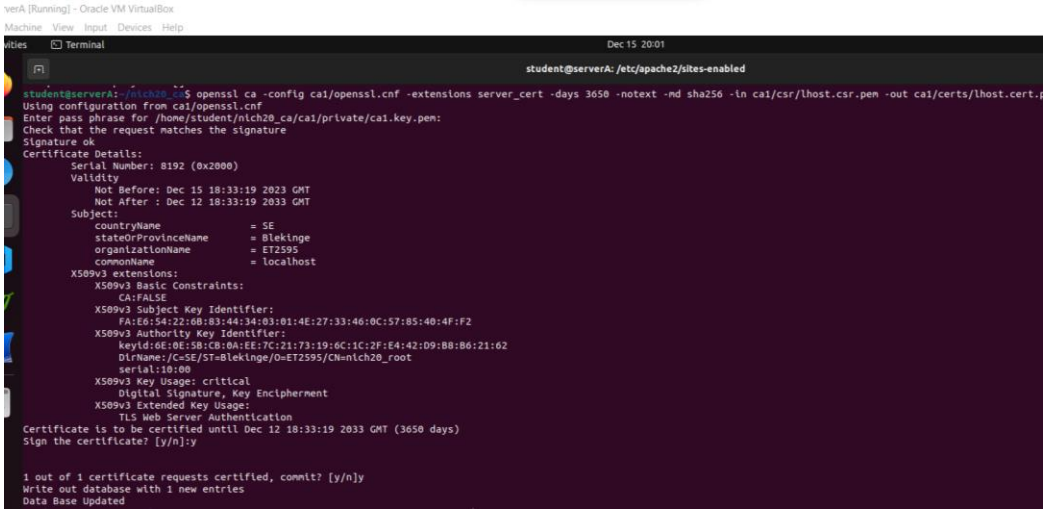
- Initially, an RSA private key was produced using a particular command. Following that, a Certificate Signing Request (CSR) was generated by employing the previously created RSA private key named lhost.key.pem.

openssl genrsa -out ca1/private/lhost.key.pem 2048
openssl req -config ca1/openssl.cnf -new -sha256 -key ca1/private/lhost.key.pem -out ca1/csr/lhost.csr.pem



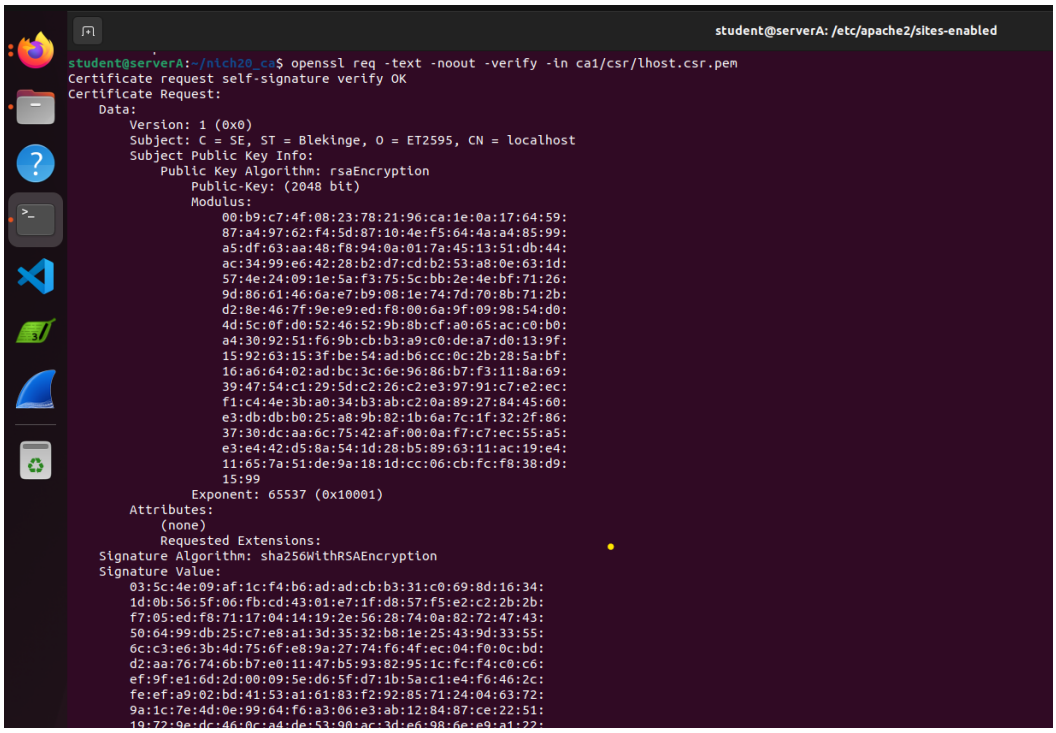
- Ultimately, we crafted a CA1 certificate using the command provided below:

openssl ca -config ca1/openssl.cnf -extensions server_cert -days 3650 -notext -md sha256 -in ca1/csr/lhost.csr.pem -out ca1/certs/lhost.cert.pem



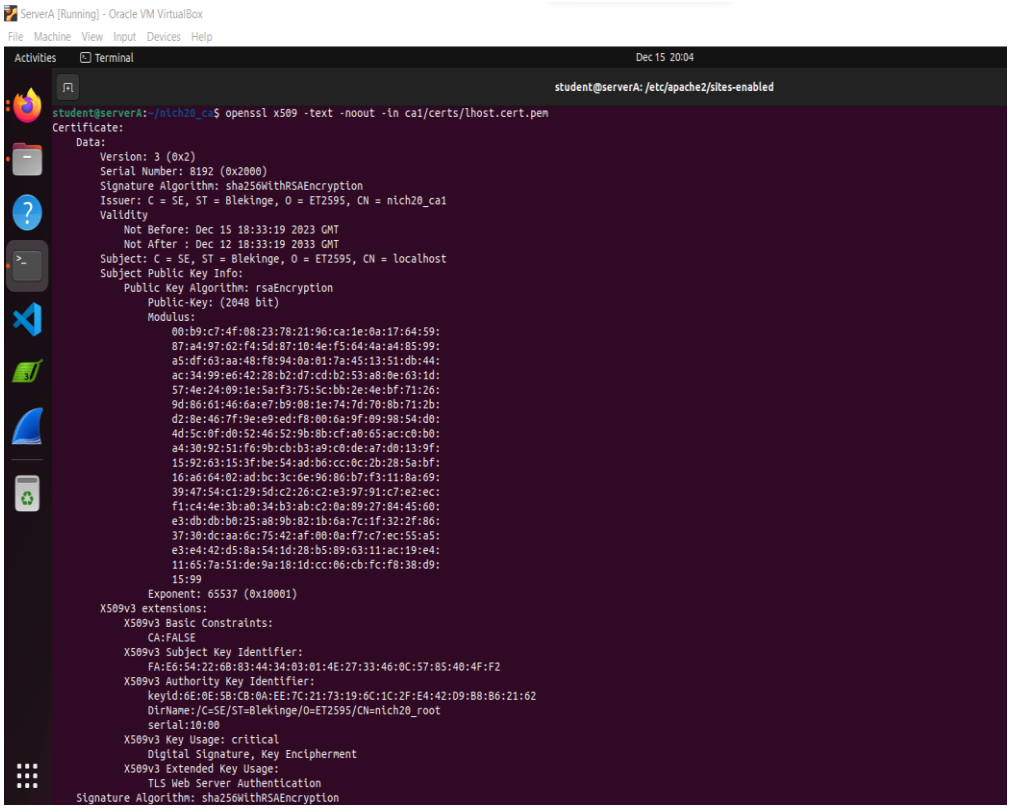
- During this phase, we proceed with the validation of the Certificate Signing Request (CSR) by employing a specific command dedicated to this verification process.

openssl req -text -noout -verify -in ca1/csr/lhost.csr.pem



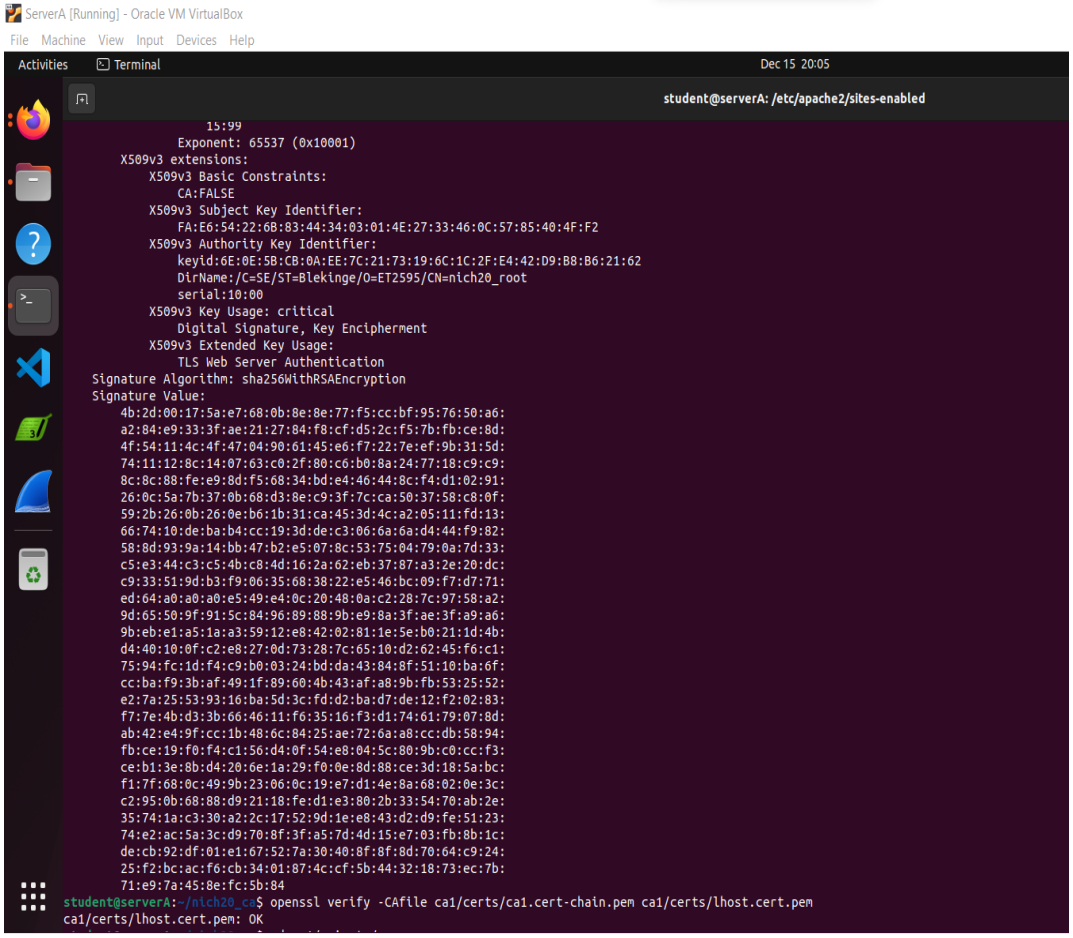
- We confirm the validation of the server certificate (localhost) generated by employing a dedicated command designed for this verification process.

  openssl x509 -text -noout -in ca1/certs/lhost.cert.pem



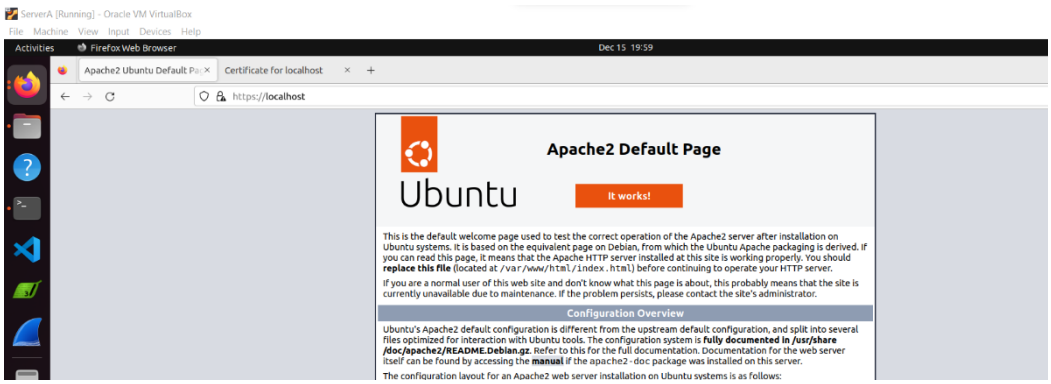
- In this stage, we verify the server certificate against the certificate chain using the specified command, as depicted in the accompanying figure.

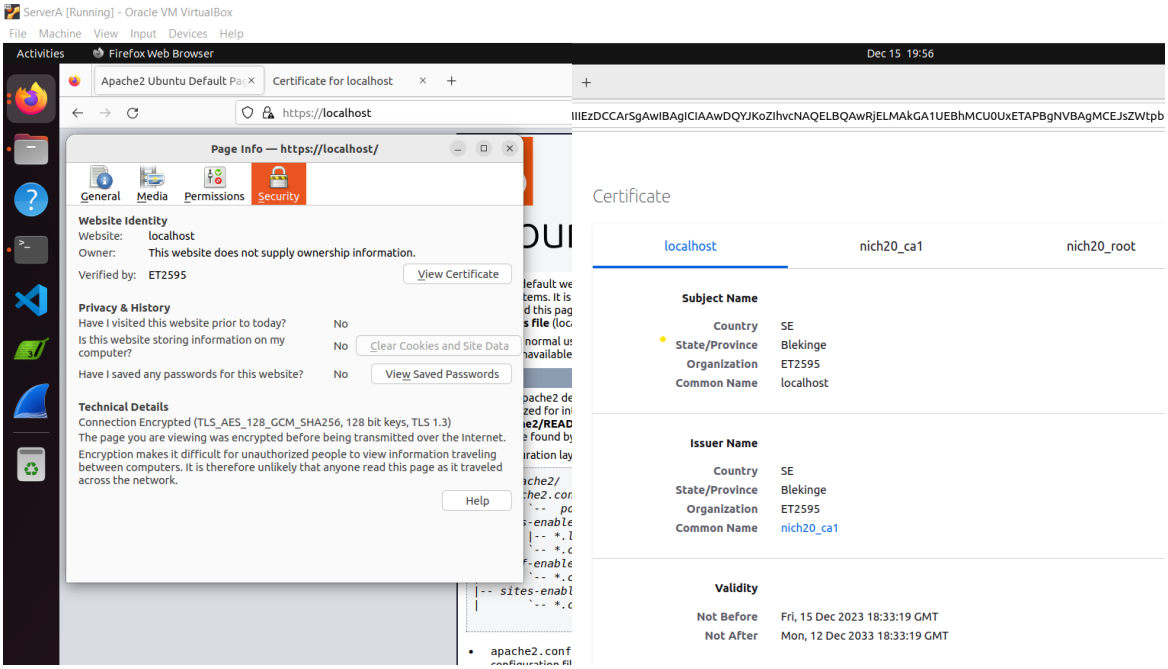
openssl verify -CAfile ca1/certs/ca1.cert-chain.pem ca1/certs/lhost.cert.pem



Performing Task 12:

Once the suggested settings for the firewall browser preferences, as described in the lab manual, have been applied, the subsequent images illustrate the sequential process. These visuals serve as a guide to follow the provided steps meticulously for a comprehensive examination.

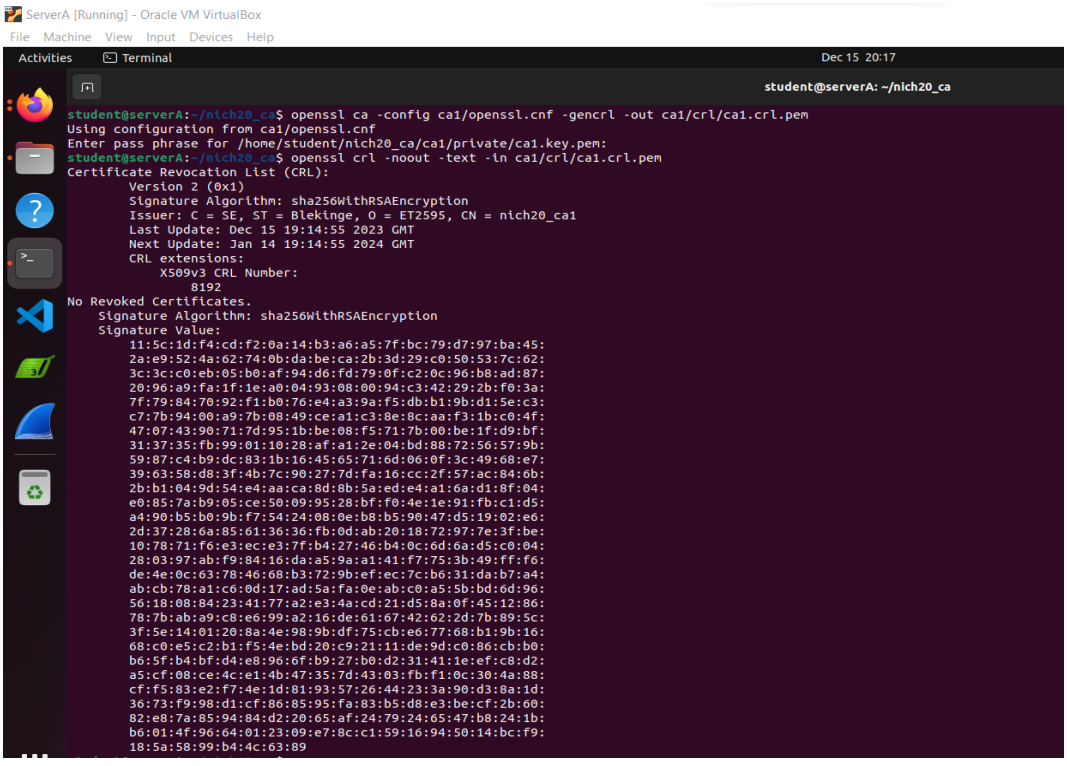




Performing of Task 13:

The steps we executed involved the use of a sequence of specific commands to generate the Certificate Revocation List (CRL) tailored for CA1.

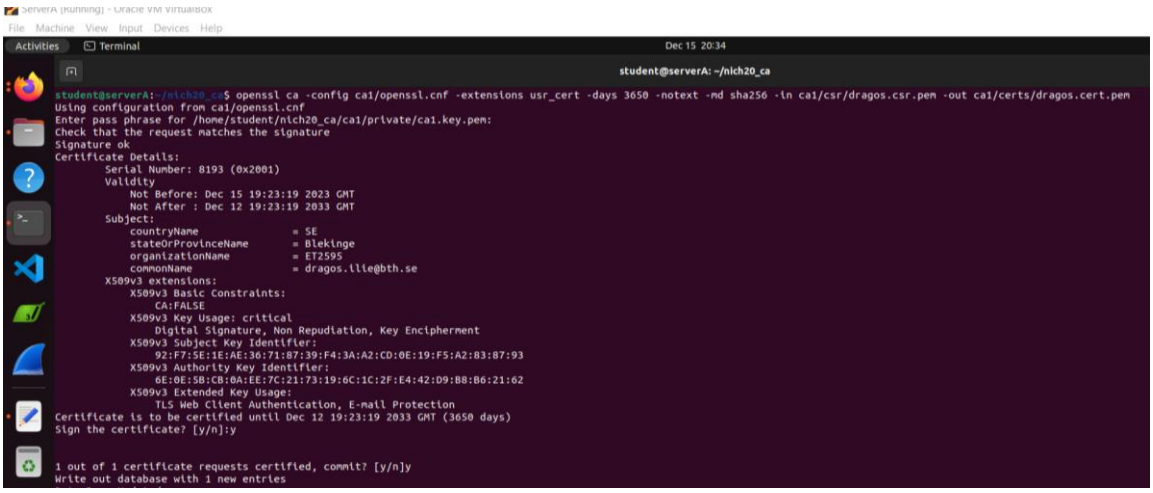
openssl ca -config ca1/openssl.cnf -gencrl -out ca1/crl/ca1.crl.pem
openssl crl -in ca1/crl/ca1.crl.pem -noout –text



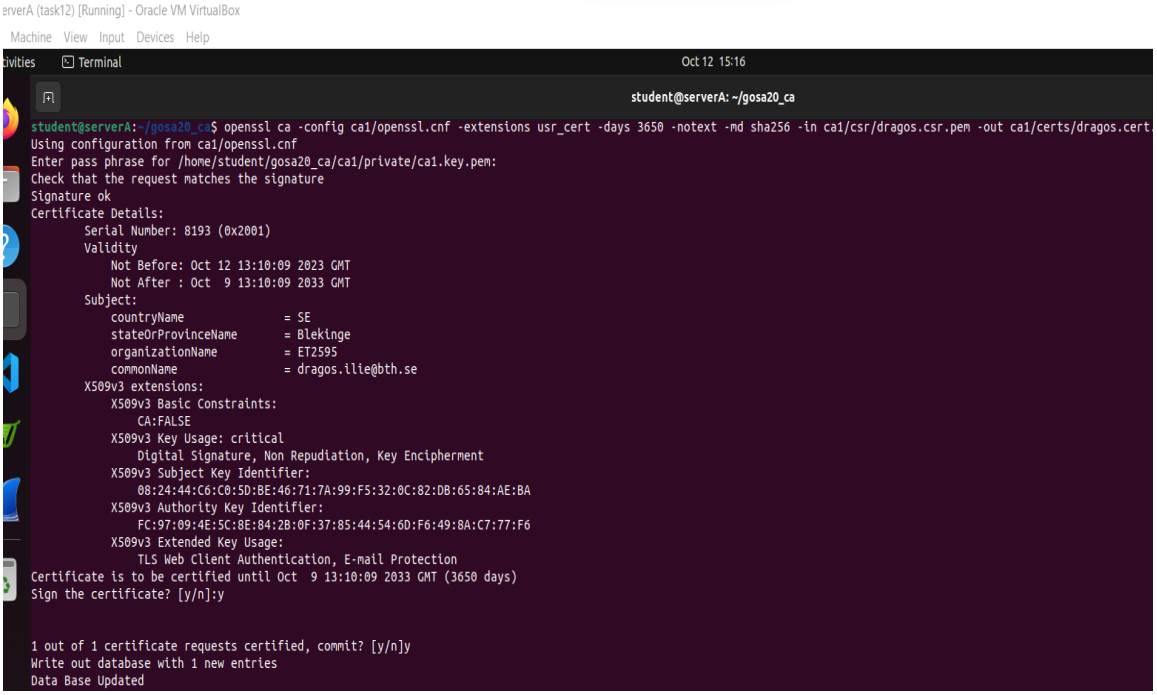
Performing of Task 14 :

We employed the subsequent commands to revoke a certificate. Initially, a user certificate was generated under the name 'dragos,' following which it was revoked and subsequently removed.

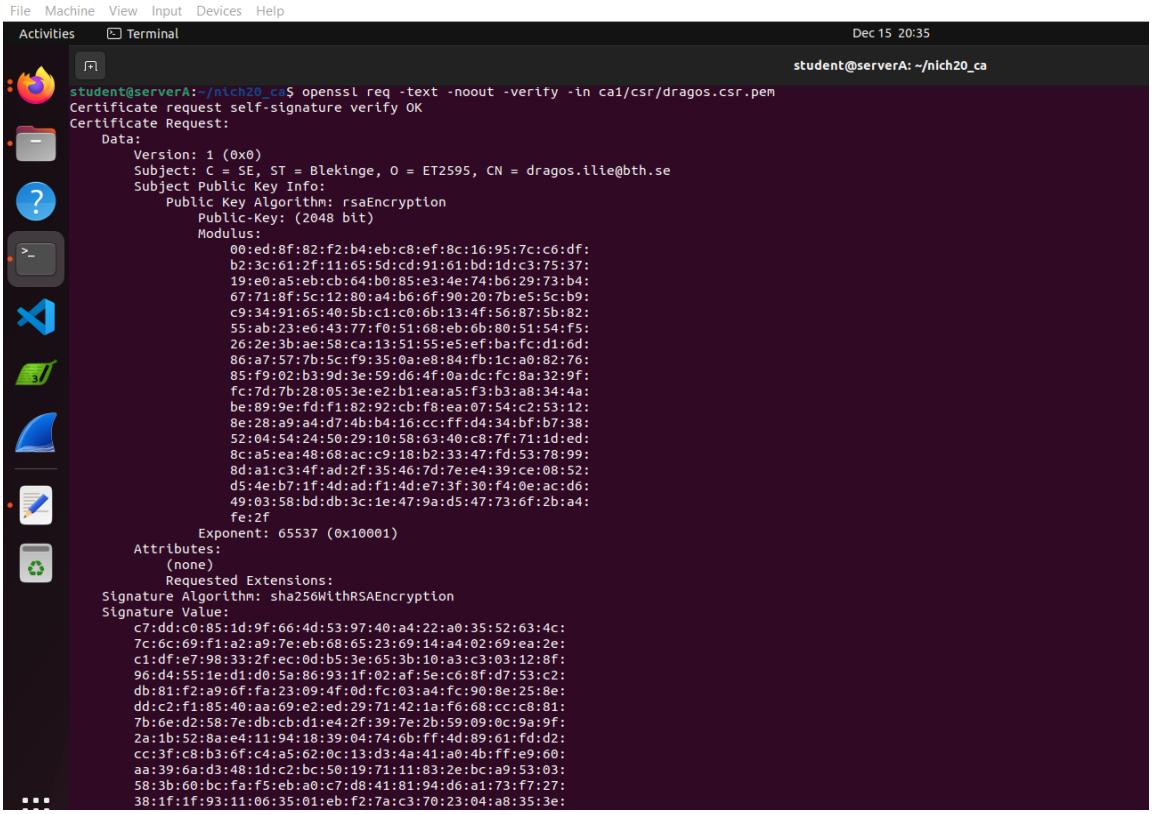
openssl genrsa -out ca1/private/usrkey.key.pem 2048
openssl req -config ca1/openssl.cnf -new -sha256 -key ca1/private/dragos.key.pem -out ca1/csr/dragos.csr.pem



openssl ca -config ca1/openssl.cnf -extensions usr cert -days 3650 -notext -md sha256 -in ca1/csr/dragos.csr.pem -out ca1/certs/dragos.cert.pem

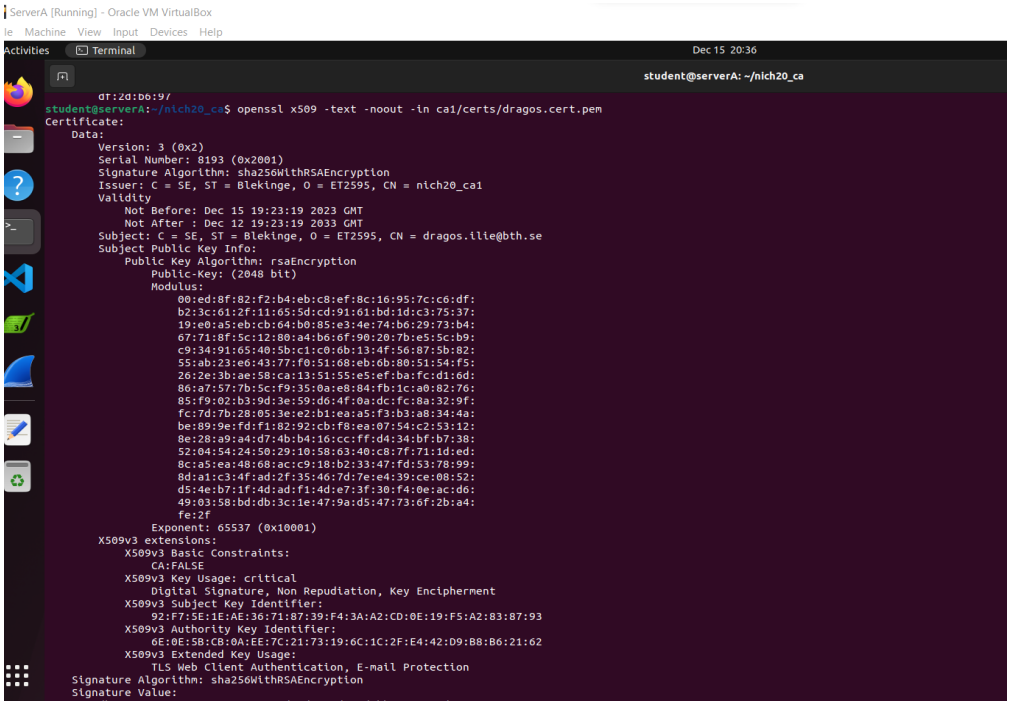


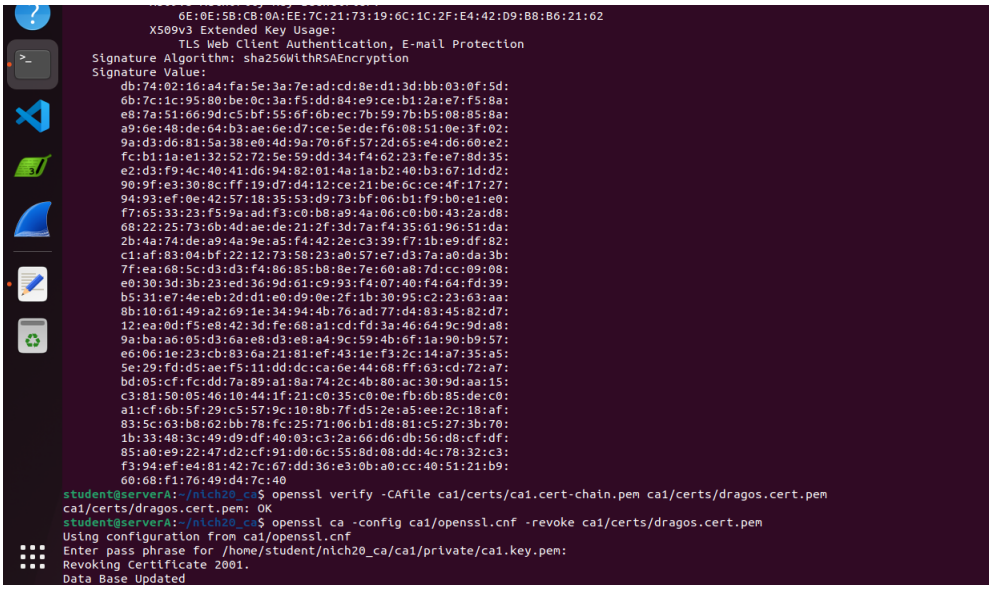
openssl req -text -noout -verify -in ca1/csr/usrkey.csr.pem



openssl x509 -text -noout -in ca1/certs/usrkey.cert.pem
openssl verify -CAfile ca1/certs/ca1.cert-chain.pem ca1/certs/usrkey.cert.pem

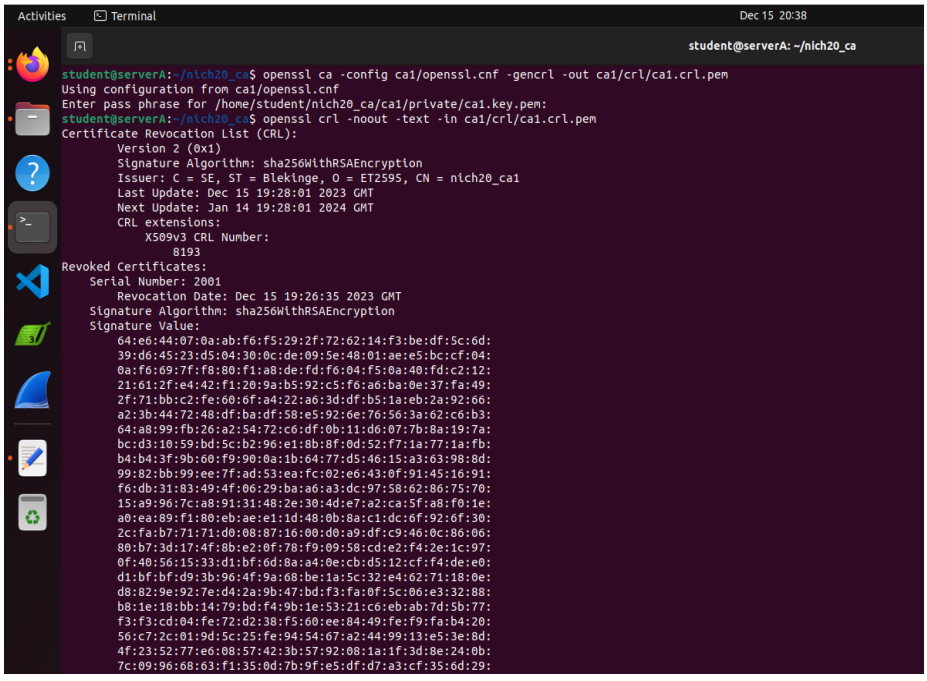
To revoke we used:
openssl ca -config ca1/openssl.cnf -revoke ca1/certs/usrkey.cert.pem





In the 'index.txt' file, as you navigate through its contents, you'll readily discern the distinct marker denoting the revoked certificate—specifically represented by the letter 'R.' This symbol serves as a clear indication within the file structure, signifying the certificate's revocation status.

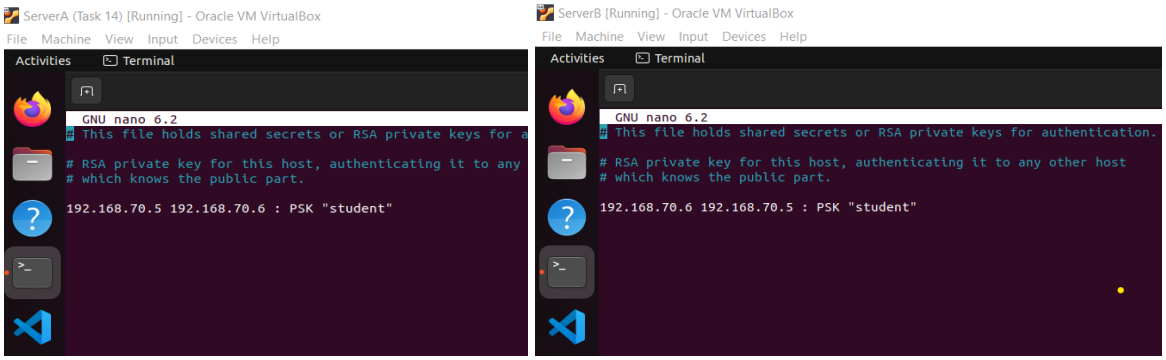
By regenerating the Certificate Revocation List (CRL), it becomes possible to access information about the revoked certificate.



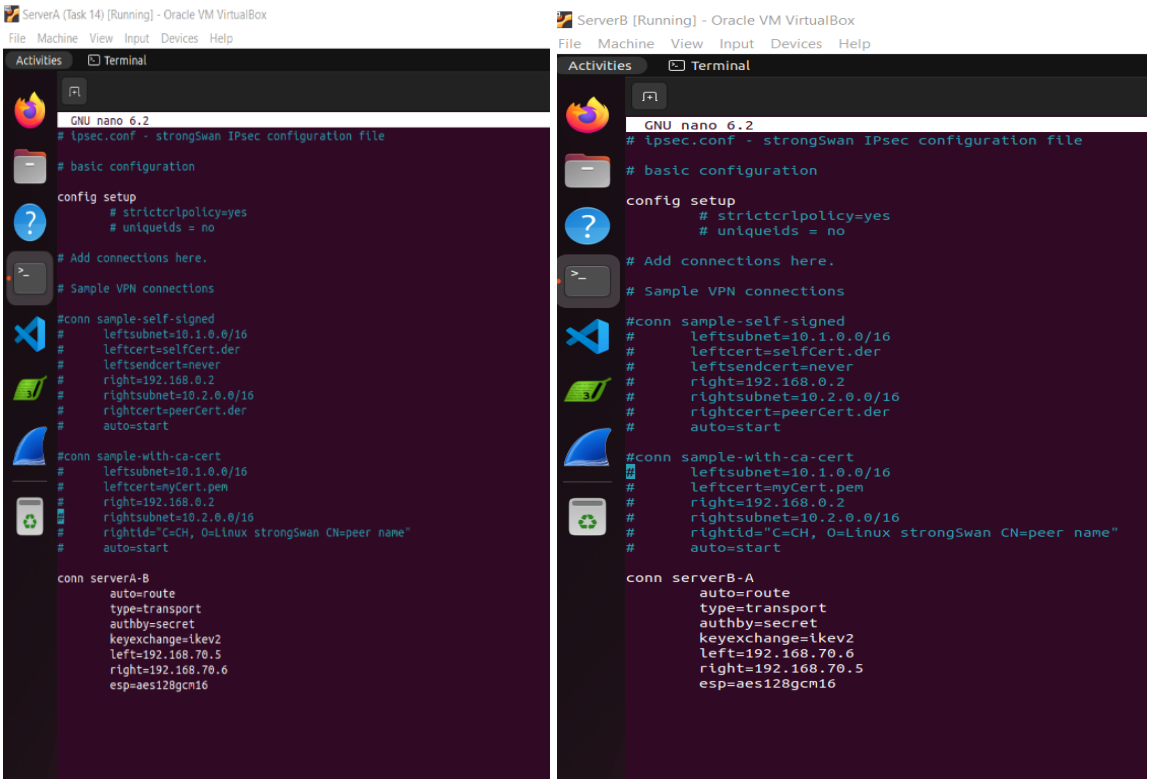
Performing of Task 15:

To facilitate this specific task, we configured Server A and Server B. And implementing specific adjustments within the ipsec.conf and ipsec.secrets files. These modifications were aimed at establishing a host-to-host transport VPN utilizing (PSK) and IKEv2 protocols. The details of these configurations are visually depicted in the accompanying figures for reference and verification.

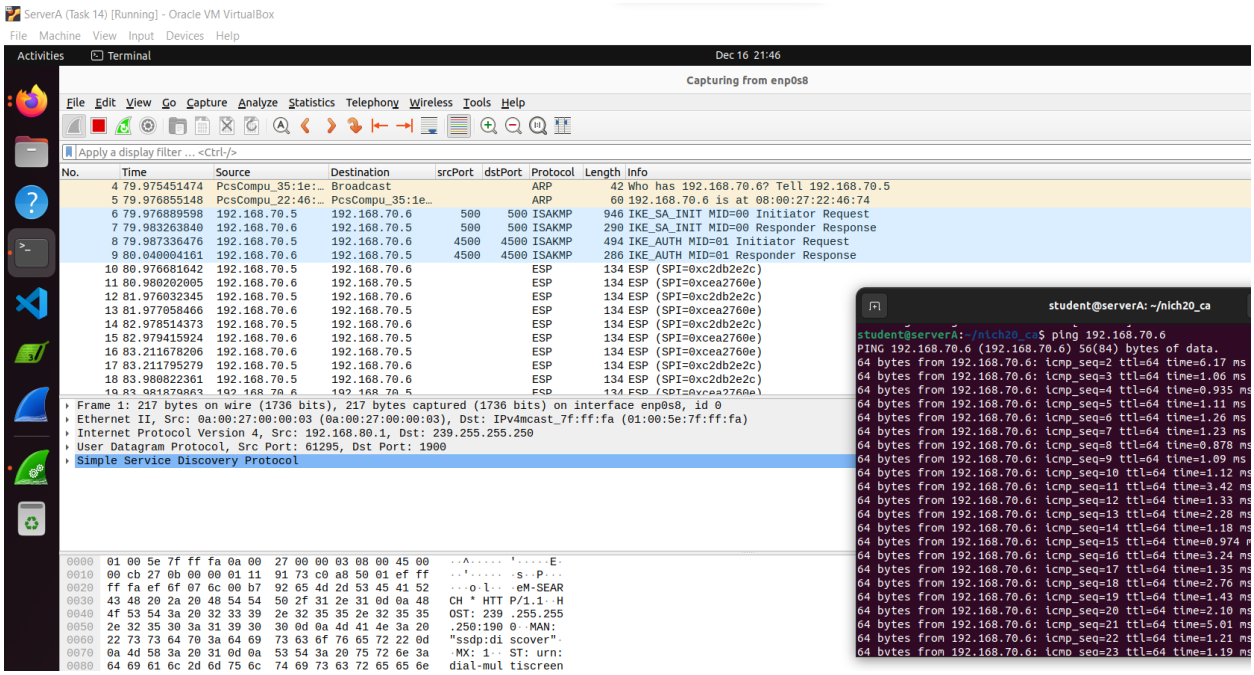
The instructions listed below are part of the configuration in the Server A ipsec.conf file, The instructions listed below are part of the configuration in the Server B ipsec.conf file:

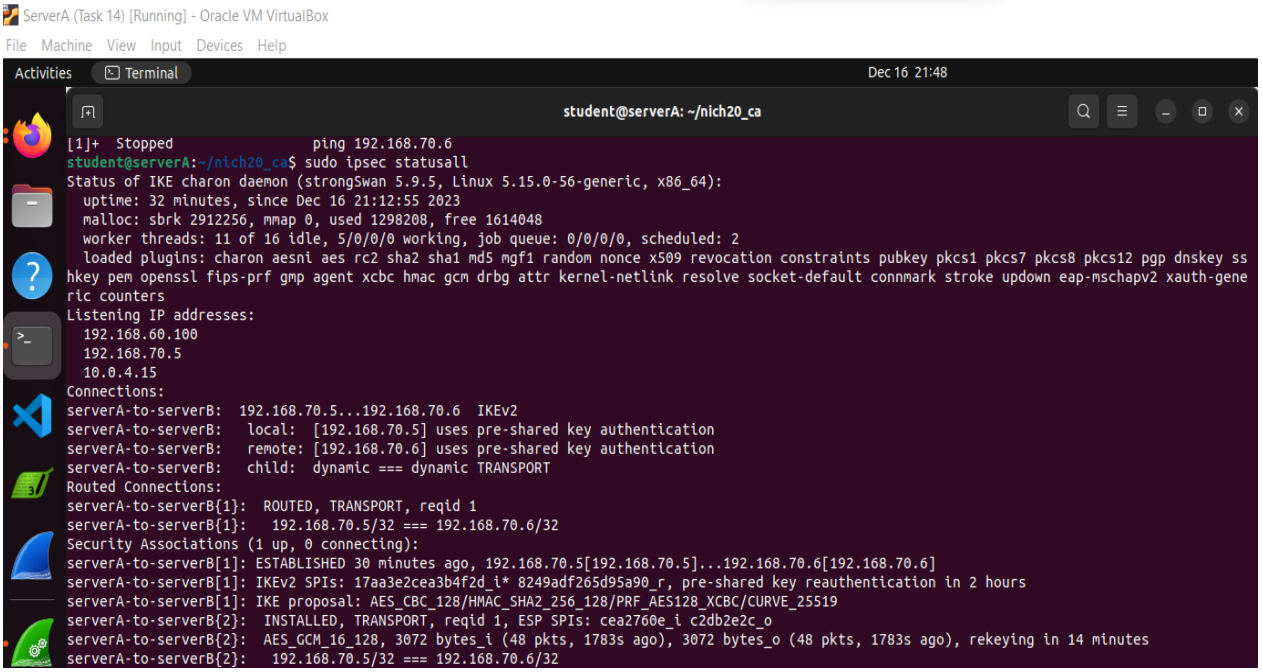


The instructions provided below represent the content incorporated within the configuration of the Server A ipsec.secrets file. The instructions provided below represent the content incorporated within the configuration of the Server B ipsec.secrets file.



The recorded network traffic, encompassing ISKAMP packets, has been meticulously captured and documented through the Wireshark application specifically on Server A. This comprehensive data provides insight into the exchanged ISKAMP packets within the network environment, facilitating detailed analysis and troubleshooting, if necessary.



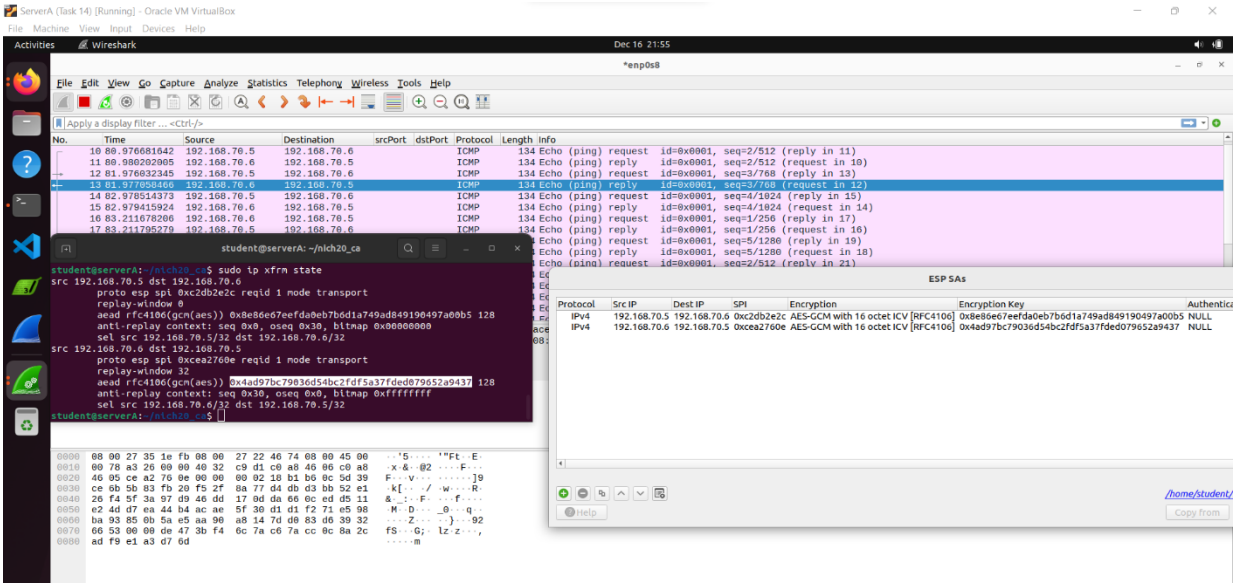


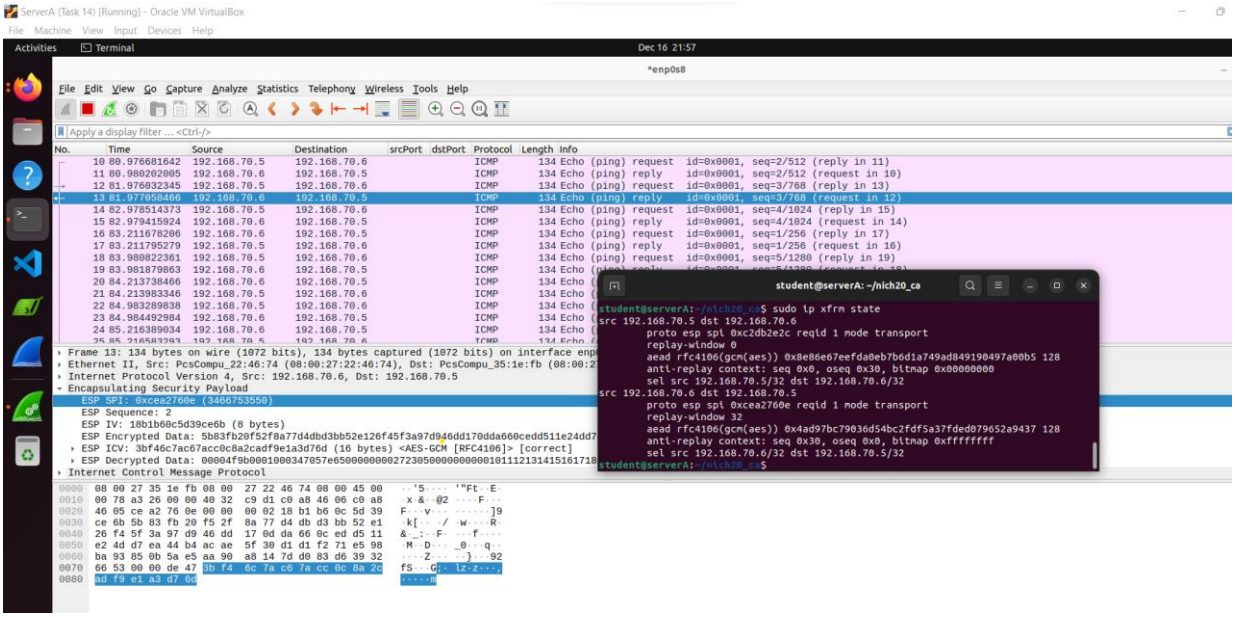
Performing of Task 16:

For accessing details regarding established connections, such as Security Parameter Indexes (SPIs), encryption and authentication keys, and related information, we employ the following command:

sudo ip xfrm state

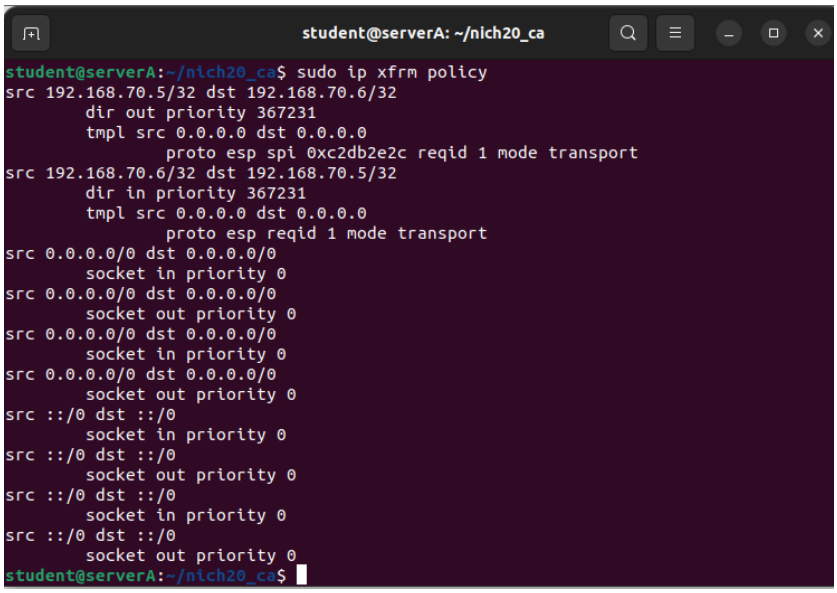
```
student@serverA: ~/nich20_ca
student@serverA:~/nich20_ca$ sudo ip xfrm state
src 192.168.70.5 dst 192.168.70.6
        proto esp spi 0xc2db2e2c reqid 1 mode transport
        replay-window 0
        aead rfc4106(gcm(aes)) 0x8e86e67eefda0eb7b6d1a749ad849190497a00b5 128
        anti-replay context: seq 0x0, oseq 0x30, bitmap 0x00000000
        sel src 192.168.70.5/32 dst 192.168.70.6/32
src 192.168.70.6 dst 192.168.70.5
        proto esp spi 0xcea2760e reqid 1 mode transport
        replay-window 32
        aead rfc4106(gcm(aes)) 0x4ad97bc79036d54bc2fdf5a37fded079652a9437 128
        anti-replay context: seq 0x30, oseq 0x0, bitmap 0xffffffff
        sel src 192.168.70.6/32 dst 192.168.70.5/32
student@serverA:~/nich20_ca$
```





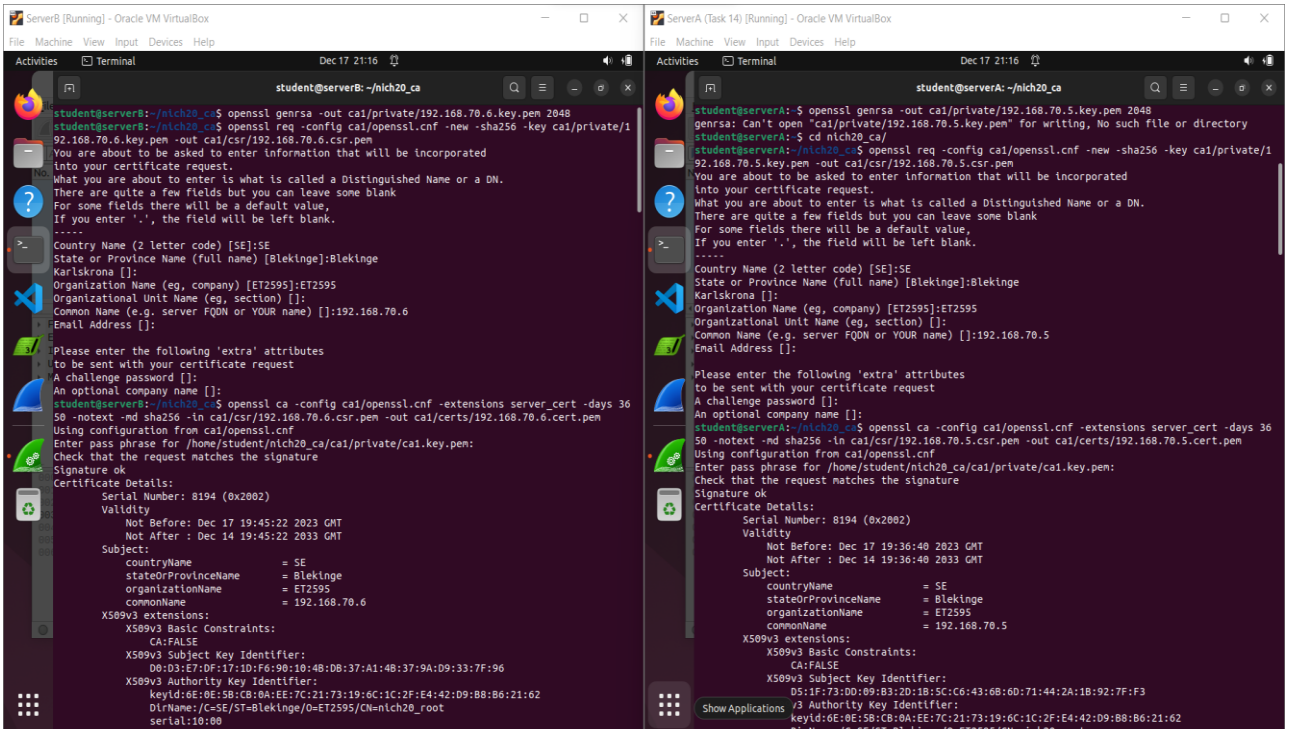
Performing of Task 17:

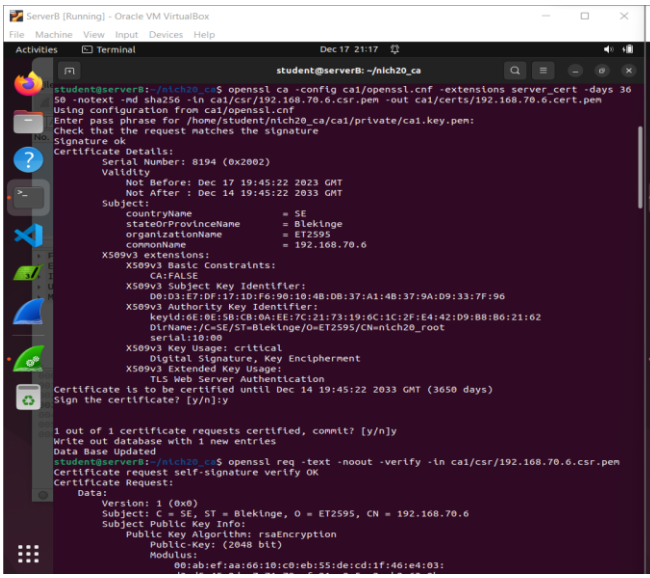
When executing the command *'sudo ip xfrm policy*,' it generates the subsequent output:

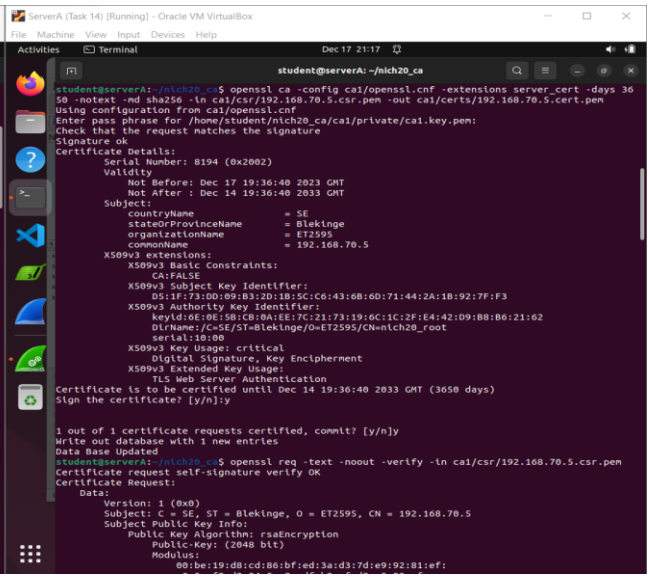


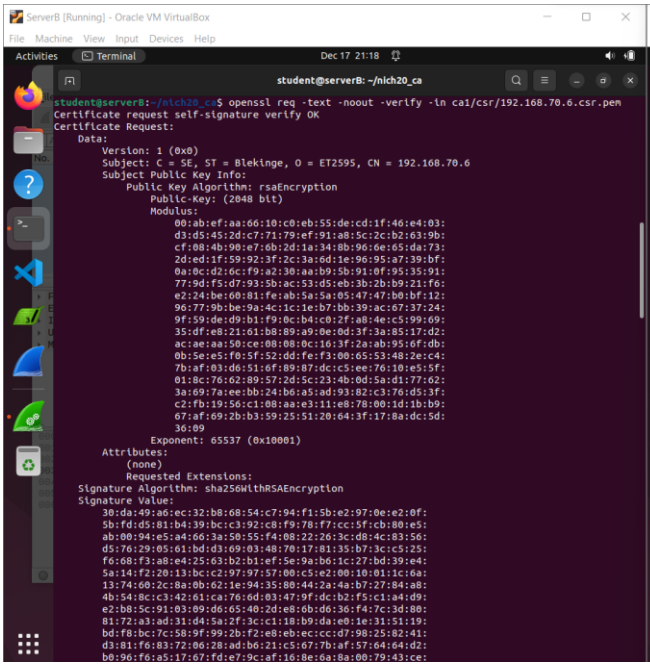
Performing of Task 18:

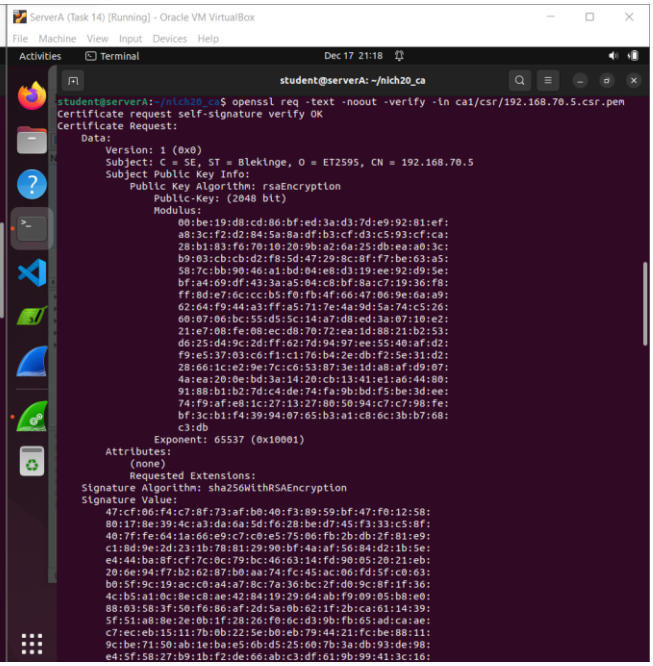
Certificates have been crafted for Server A and Server B employing a methodology reminiscent of the techniques employed in previous tasks, drawing parallels with the procedures outlined in both Task 11 and Task 14. The generation of these certificates for the respective servers was conducted through a similar approach, ensuring compliance with established protocols and standards.

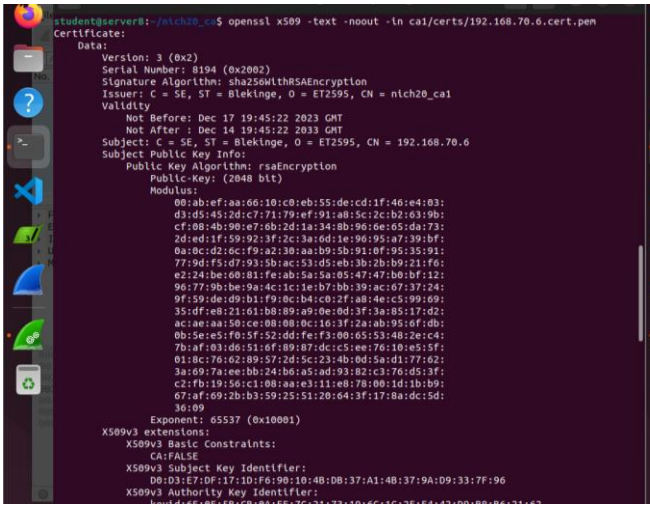


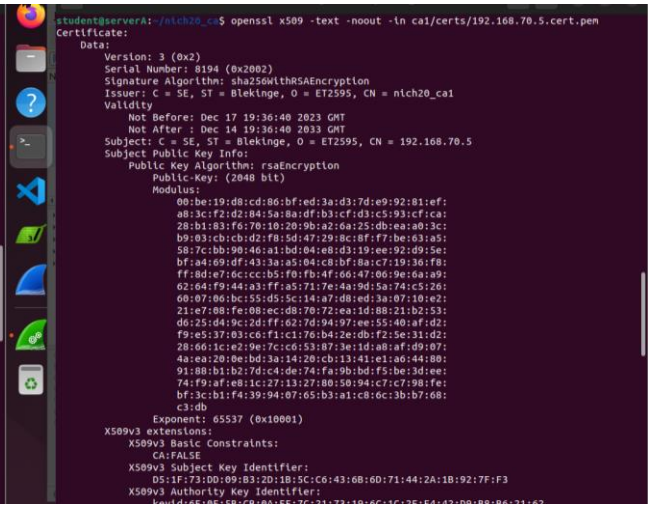

ServerB [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Dec 17 21:17
student@serverB: ~/nich20_ca
student@serverB:~/nich20_ca$ openssl ca -config ca1/openssl.cnf -extensions server_cert -days 3650 -notext -md sha256 -in ca1/csr/192.168.70.6.csr.pem -out ca1/certs/192.168.70.6.cert.pem
Using configuration from ca1/openssl.cnf
Enter pass phrase for /home/student/nich20_ca/ca1/private/ca1.key.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
Serial Number: 8194 (0x2002)
Validity
Not Before: Dec 17 19:45:22 2023 GMT
Not After : Dec 14 19:45:22 2033 GMT
Subject:
countryName = SE
stateOrProvinceName = Blekinge
organizationName = ET2595
commonName = 192.168.70.6
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
X509v3 Subject Key Identifier:
D0:D3:E7:DF:17:1D:F6:90:10:4B:DB:37:A1:4B:37:9A:D9:33:7F:96
X509v3 Authority Key Identifier:
keyid:6E:0E:58:CB:0A:EE:7C:21:73:19:6C:1C:2F:E4:42:D9:B8:B6:21:62
DirName:/C=SE/ST=Blekinge/O=ET2595/CN=nich20_root
serial:10:00
X509v3 Key Usage: critical
Digital Signature, Key Encipherment
X509v3 Extended Key Usage:
TLS Web Server Authentication
Certificate is to be certified until Dec 14 19:45:22 2033 GMT (3650 days)
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
student@serverB:~/nich20_ca$ openssl req -text -noout -verify -in ca1/csr/192.168.70.6.csr.pem
Certificate request self-signature verify OK
Certificate Request:
Data:
Version: 1 (0x0)
Subject: C = SE, ST = Blekinge, O = ET2595, CN = 192.168.70.6
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
00:ab:ef:aa:66:10:c0:eb:55:de:cd:1f:46:e4:03:

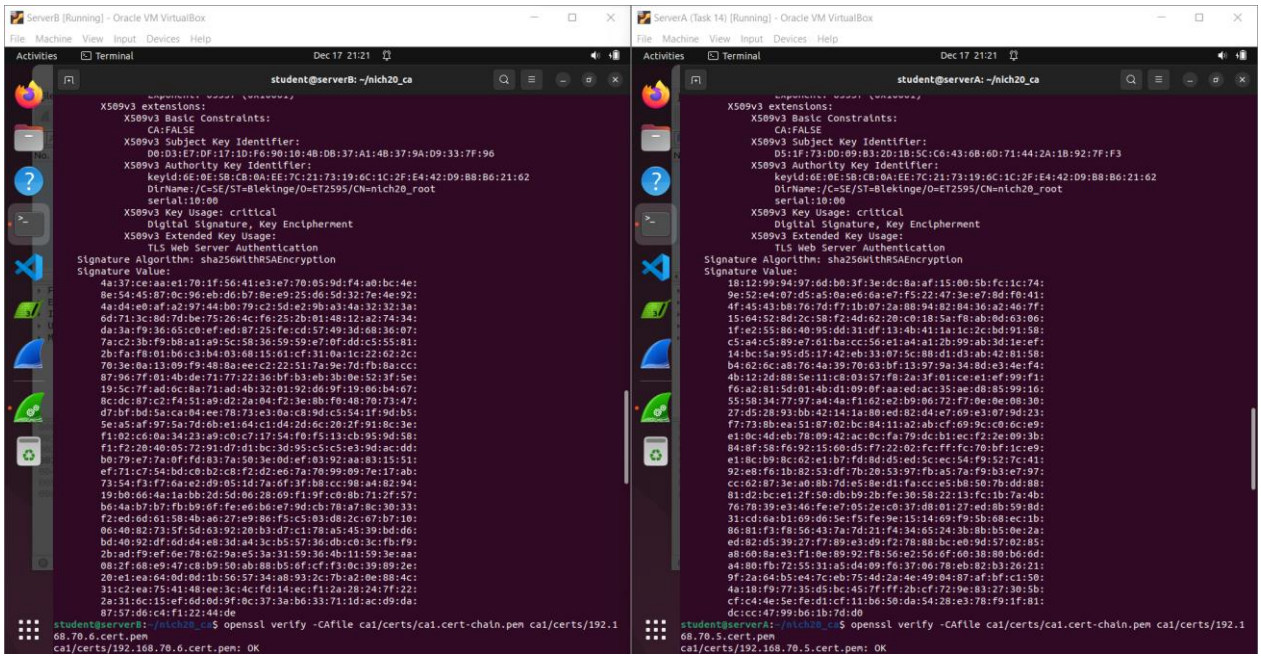

ServerA (Task 14) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Dec 17 21:17
student@serverA: ~/nich20_ca
student@serverA:~/nich20_ca$ openssl ca -config ca1/openssl.cnf -extensions server_cert -days 3650 -notext -md sha256 -in ca1/csr/192.168.70.5.csr.pem -out ca1/certs/192.168.70.5.cert.pem
Using configuration from ca1/openssl.cnf
Enter pass phrase for /home/student/nich20_ca/ca1/private/ca1.key.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
Serial Number: 8194 (0x2002)
Validity
Not Before: Dec 17 19:36:40 2023 GMT
Not After : Dec 14 19:36:40 2033 GMT
Subject:
countryName = SE
stateOrProvinceName = Blekinge
organizationName = ET2595
commonName = 192.168.70.5
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
X509v3 Subject Key Identifier:
D5:1F:73:DD:09:B3:2D:1B:5C:C6:43:6B:6D:71:44:2A:1B:92:7F:F3
X509v3 Authority Key Identifier:
keyid:6E:0E:58:CB:0A:EE:7C:21:73:19:6C:1C:2F:E4:42:D9:B8:B6:21:62
DirName:/C=SE/ST=Blekinge/O=ET2595/CN=nich20_root
serial:10:00
X509v3 Key Usage: critical
Digital Signature, Key Encipherment
X509v3 Extended Key Usage:
TLS Web Server Authentication
Certificate is to be certified until Dec 14 19:36:40 2033 GMT (3650 days)
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
student@serverA:~/nich20_ca$ openssl req -text -noout -verify -in ca1/csr/192.168.70.5.csr.pem
Certificate request self-signature verify OK
Certificate Request:
Data:
Version: 1 (0x0)
Subject: C = SE, ST = Blekinge, O = ET2595, CN = 192.168.70.5
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
00:be:19:d8:cd:86:bf:ed:3a:d3:7d:e9:92:81:ef:


ServerB [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Dec 17 21:18
student@serverB: ~/nich20_ca
student@serverB:~/nich20_ca$ openssl req -text -noout -verify -in ca1/csr/192.168.70.6.csr.pem
Certificate request self-signature verify OK
Certificate Request:
Data:
Version: 1 (0x0)
Subject: C = SE, ST = Blekinge, O = ET2595, CN = 192.168.70.6
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
00:ab:ef:aa:66:10:c0:eb:55:de:cd:1f:46:e4:03:
d3:d5:45:2d:c7:71:79:ef:91:a8:5c:2c:b2:63:9b:
cf:08:4b:90:e7:6b:2d:1a:34:8b:96:6e:65:da:73:
2d:ed:1f:59:92:3f:2c:3a:6d:1e:96:95:a7:39:bf:
0a:0c:d2:6c:f9:a2:30:aa:b9:5b:91:0f:95:35:91:
77:9d:f5:d7:93:5b:ac:53:d5:eb:3b:2b:b9:21:f6:
e2:24:be:60:81:fe:ab:5a:5a:05:47:47:b0:bf:12:
96:77:9b:be:9a:4c:1c:1e:b7:bb:39:ac:67:37:24:
9f:59:de:d9:b1:f9:0c:b4:c0:2f:a8:4e:c5:99:69:
35:df:e8:21:61:b8:89:a9:0e:0d:3f:3a:85:17:d2:
ac:ae:aa:50:ce:08:08:0c:16:3f:2a:ab:95:6f:db:
0b:5e:e5:f0:5f:52:dd:fe:f3:00:65:53:48:2e:c4:
7b:af:03:d6:51:6f:89:87:dc:c5:ee:76:10:e5:5f:
01:8c:76:62:89:57:2d:5c:23:4b:0d:5a:d1:77:62:
3a:69:7a:ee:bb:24:b6:a5:ad:93:82:c3:76:d5:3f:
c2:fb:19:56:c1:08:aa:e3:11:e8:78:00:1d:1b:b9:
67:af:69:2b:b3:59:25:51:20:64:3f:17:8a:dc:5d:
36:09
Exponent: 65537 (0x10001)
Attributes:
(none)
Requested Extensions:
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
30:da:49:a6:ec:32:b8:68:54:c7:94:f1:5b:e2:97:0e:e2:0f:
5b:fd:d5:81:b4:39:bc:c3:92:c8:f9:78:f7:cc:5f:cb:80:e5:
ab:00:94:e5:a4:66:3a:50:55:f4:08:22:26:3c:d8:4c:83:56:
d5:76:29:05:61:bd:d3:69:03:48:70:17:81:35:b7:3c:c5:25:
f6:68:f3:a8:e4:25:63:b2:b1:ef:5e:9a:b6:1c:27:bd:39:e4:
5a:14:f2:28:13:bc:c2:97:97:57:00:c5:e2:00:10:01:1c:6a:
13:74:60:2c:8a:0b:62:1e:94:35:80:44:2a:4a:b7:27:84:a8:
4b:54:8c:c3:42:61:ca:76:6d:03:47:9f:dc:b2:f5:c1:a4:d9:
e2:b8:5c:91:03:09:d6:65:40:2d:e8:6b:d6:36:f4:7c:3d:80:
81:72:a3:ad:31:d4:5a:2f:3c:c1:18:b9:da:e0:1e:31:51:19:
bd:f8:bc:7c:58:9f:99:2b:f2:e8:eb:ec:cc:d7:98:25:82:41:
d3:81:f6:83:72:86:28:ad:b6:21:c5:67:7b:af:57:64:64:d2:
b0:96:f6:a5:17:67:fd:e7:9c:af:16:8e:6a:8a:00:79:43:ce:


ServerA (Task 14) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Dec 17 21:18
student@serverA: ~/nich20_ca
student@serverA:~/nich20_ca$ openssl req -text -noout -verify -in ca1/csr/192.168.70.5.csr.pem
Certificate request self-signature verify OK
Certificate Request:
Data:
Version: 1 (0x0)
Subject: C = SE, ST = Blekinge, O = ET2595, CN = 192.168.70.5
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
00:be:19:d8:cd:86:bf:ed:3a:d3:7d:e9:92:81:ef:
a8:3c:f2:d2:84:5a:8a:df:b3:cf:d3:c5:93:cf:ca:
28:b1:83:f6:70:10:20:9b:a2:6a:25:db:ea:a0:3c:
b9:03:cb:cb:d2:f8:5d:47:29:8c:8f:f7:be:63:a5:
58:7c:bb:90:46:a1:bd:04:e8:d3:19:ee:92:d9:5e:
bf:a4:69:df:43:3a:a5:04:c8:bf:8a:c7:19:36:f8:
ff:8d:e7:6c:cc:b5:f0:fb:4f:66:47:06:9e:6a:a9:
62:64:f9:44:a3:ff:a5:71:7e:4a:9d:5a:74:c5:26:
60:07:06:bc:55:d5:5c:14:a7:d8:ed:3a:07:10:e2:
21:e7:08:fe:08:ec:d8:70:72:ea:1d:88:21:b2:53:
d6:25:d4:9c:2d:ff:62:7d:94:97:ee:55:40:af:d2:
f9:e5:37:03:c6:f1:c1:76:b4:2e:db:f2:5e:31:d2:
28:66:1c:e2:9e:7c:c6:53:87:3e:1d:a8:af:d9:07:
4a:ea:20:0e:bd:3a:14:20:cb:13:41:e1:a6:44:80:
91:88:b1:b2:7d:c4:de:74:fa:9b:bd:f5:be:3d:ee:
74:f9:af:e8:1c:27:13:27:80:50:94:c7:c7:98:fe:
bf:3c:b1:f4:39:94:07:65:b3:a1:c8:6c:3b:b7:68:
c3:db
Exponent: 65537 (0x10001)
Attributes:
(none)
Requested Extensions:
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
47:cf:06:f4:c7:8f:73:af:b0:40:f3:89:59:bf:47:f0:12:58:
80:17:8e:39:4c:a3:da:6a:5d:f6:28:be:d7:45:f3:33:c5:8f:
40:7f:fe:64:1a:66:e9:c7:c0:e5:75:06:fb:2b:db:2f:81:e9:
c1:8d:9e:2d:23:1b:78:81:29:90:bf:4a:af:56:84:d2:1b:5e:
e4:44:ba:8f:cf:7c:0c:79:bc:46:63:14:fd:90:05:20:21:eb:
20:6e:94:f7:b2:62:87:b0:e4:74:fc:45:ac:06:fd:5f:c0:63:
b0:5f:9c:19:ac:c0:a4:a7:8c:7a:36:bc:2f:d0:9c:8f:1f:36:
4c:b5:a1:0c:8e:c8:ae:42:84:19:29:64:ab:f9:09:05:b8:e0:
88:03:58:3f:50:f6:86:af:2d:5a:0b:62:1f:2b:ca:61:14:39:
5f:51:a8:8e:2e:0b:1f:28:26:f0:6c:d3:9b:fb:65:ad:ca:ae:
c7:ec:eb:15:11:7b:0b:22:5e:b0:eb:79:44:21:fc:be:88:11:
9c:be:71:50:ab:1e:ba:e5:6b:d5:25:60:7b:3a:db:93:de:98:
e4:5f:58:27:b9:1b:f2:de:66:ab:c3:df:61:9b:99:41:3c:16:

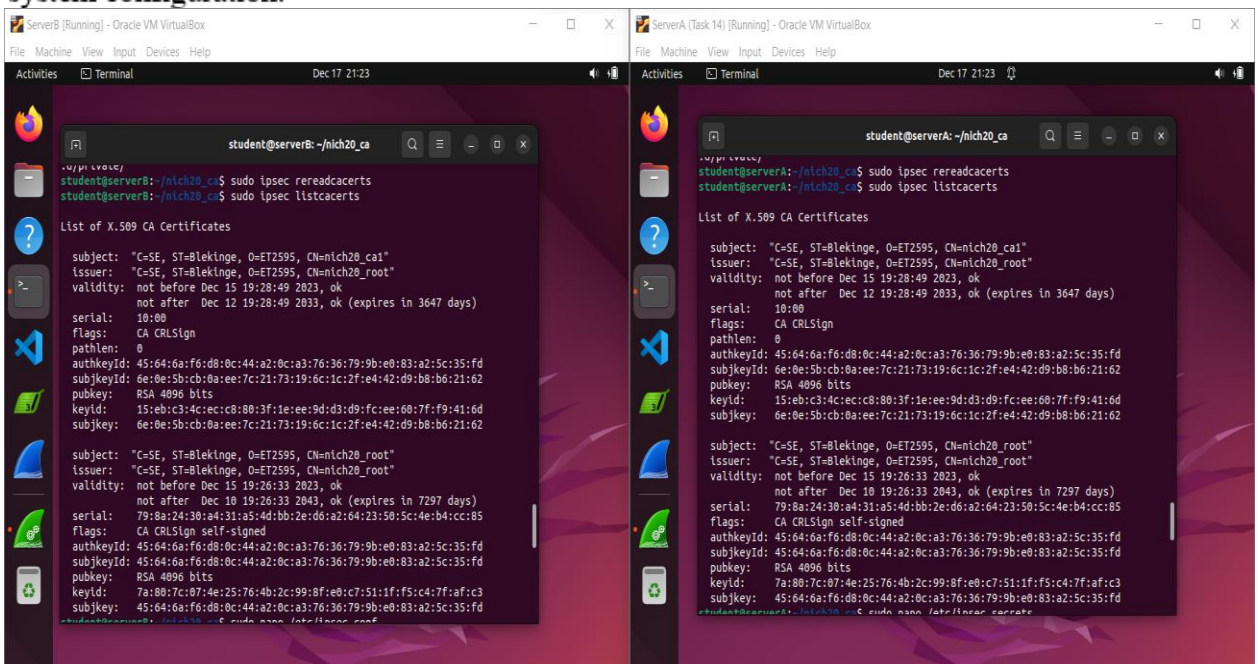

student@serverB:~/nich20_ca$ openssl x509 -text -noout -in ca1/certs/192.168.70.6.cert.pem
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 8194 (0x2002)
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = SE, ST = Blekinge, O = ET2595, CN = nich20_ca1
Validity
Not Before: Dec 17 19:45:22 2023 GMT
Not After : Dec 14 19:45:22 2033 GMT
Subject: C = SE, ST = Blekinge, O = ET2595, CN = 192.168.70.6
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
00:ab:ef:aa:66:10:c0:eb:55:de:cd:1f:46:e4:03:
d3:d5:45:2d:c7:71:79:ef:91:a8:5c:2c:b2:63:9b:
cf:08:4b:90:e7:6b:2d:1a:34:8b:96:6e:65:da:73:
2d:ed:1f:59:92:3f:2c:3a:6d:1e:96:95:a7:39:bf:
0a:0c:d2:6c:f9:a2:30:aa:b9:5b:91:0f:95:35:91:
77:9d:f5:d7:93:5b:ac:53:d5:eb:3b:2b:b9:21:f6:
e2:24:be:60:81:fe:ab:5a:5a:05:47:47:b0:bf:12:
96:77:9b:be:9a:4c:1c:1e:b7:bb:39:ac:67:37:24:
9f:59:de:d9:b1:f9:0c:b4:c0:2f:a8:4e:c5:99:69:
35:df:e8:21:61:b8:89:a9:0e:0d:3f:3a:85:17:d2:
ac:ae:aa:50:ce:08:08:0c:16:3f:2a:ab:95:6f:db:
0b:5e:e5:f0:5f:52:dd:fe:f3:00:65:53:48:2e:c4:
7b:af:03:d6:51:6f:89:87:dc:c5:ee:76:10:e5:5f:
01:8c:76:62:89:57:2d:5c:23:4b:0d:5a:d1:77:62:
3a:69:7a:ee:bb:24:b6:a5:ad:93:82:c3:76:d5:3f:
c2:fb:19:56:c1:08:aa:e3:11:e8:78:00:1d:1b:b9:
67:af:69:2b:b3:59:25:51:20:64:3f:17:8a:dc:5d:
36:09
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
X509v3 Subject Key Identifier:
D0:D3:E7:DF:17:1D:F6:90:10:4B:DB:37:A1:4B:37:9A:D9:33:7F:96
X509v3 Authority Key Identifier:


student@serverA:~/nich20_ca$ openssl x509 -text -noout -in ca1/certs/192.168.70.5.cert.pem
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 8194 (0x2002)
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = SE, ST = Blekinge, O = ET2595, CN = nich20_ca1
Validity
Not Before: Dec 17 19:36:40 2023 GMT
Not After : Dec 14 19:36:40 2033 GMT
Subject: C = SE, ST = Blekinge, O = ET2595, CN = 192.168.70.5
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
00:be:19:d8:cd:86:bf:ed:3a:d3:7d:e9:92:81:ef:
a8:3c:f2:d2:84:5a:8a:df:b3:cf:d3:c5:93:cf:ca:
28:b1:83:f6:70:10:20:9b:a2:6a:25:db:ea:a0:3c:
b9:03:cb:cb:d2:f8:5d:47:29:8c:8f:f7:be:63:a5:
58:7c:bb:90:46:a1:bd:04:e8:d3:19:ee:92:d9:5e:
bf:a4:69:df:43:3a:a5:04:c8:bf:8a:c7:19:36:f8:
ff:8d:e7:6c:cc:b5:f0:fb:4f:66:47:06:9e:6a:a9:
62:64:f9:44:a3:ff:a5:71:7e:4a:9d:5a:74:c5:26:
60:07:06:bc:55:d5:5c:14:a7:d8:ed:3a:07:10:e2:
21:e7:08:fe:08:ec:d8:70:72:ea:1d:88:21:b2:53:
d6:25:d4:9c:2d:ff:62:7d:94:97:ee:55:40:af:d2:
f9:e5:37:03:c6:f1:c1:76:b4:2e:db:f2:5e:31:d2:
28:66:1c:e2:9e:7c:c6:53:87:3e:1d:a8:af:d9:07:
4a:ea:20:0e:bd:3a:14:20:cb:13:41:e1:a6:44:80:
91:88:b1:b2:7d:c4:de:74:fa:9b:bd:f5:be:3d:ee:
74:f9:af:e8:1c:27:13:27:80:50:94:c7:c7:98:fe:
bf:3c:b1:f4:39:94:07:65:b3:a1:c8:6c:3b:b7:68:
c3:db
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
X509v3 Subject Key Identifier:
D5:1F:73:DD:09:B3:2D:1B:5C:C6:43:6B:6D:71:44:2A:1B:92:7F:F3
X509v3 Authority Key Identifier:

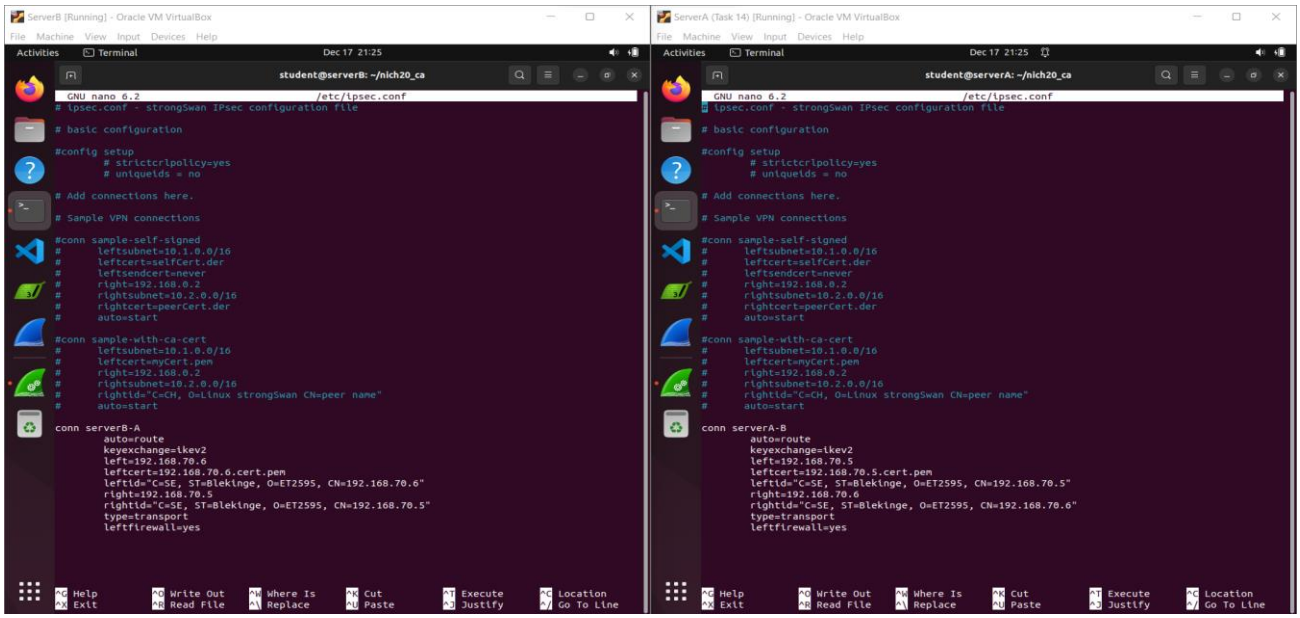


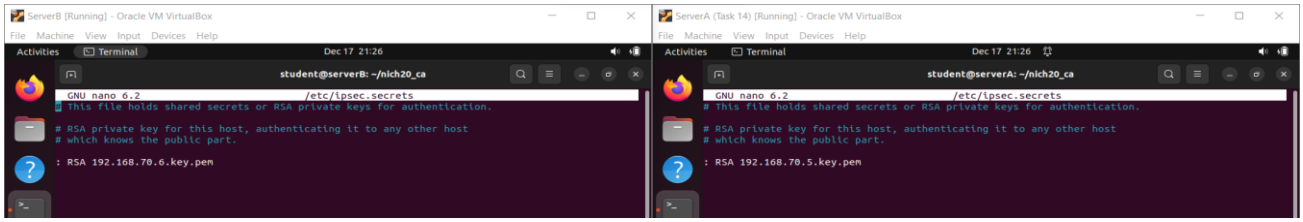
Subsequently, we proceeded to place the Root CA and CA1 certificates into the "/etc/ipsec.d/cacerts/" directory, while situating the newly generated certificates in the "/etc/ipsec.d/certs/" directory for both Server A and Server B. Furthermore, the private keys pertinent to these servers were securely stored in the "/etc/ipsec.d/private/" directory, ensuring their accessibility when needed for secure communication.

Before this, we took the necessary steps to validate and incorporate the updated CA certificates by executing the command "sudo ipsec rereadcacerts." Following this, we confirmed the successful recognition and integration of these certificates into the system by listing the available CA certificates through the command "sudo ipsec listcacerts." These meticulous steps were instrumental in verifying the seamless integration and recognition of the certificates within the system configuration.

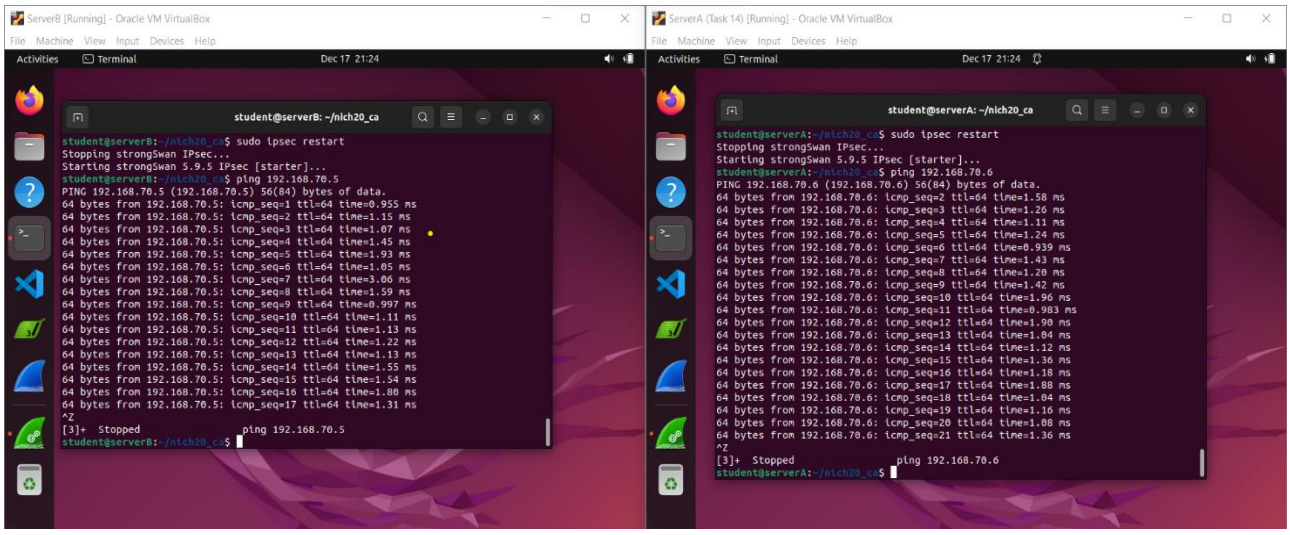


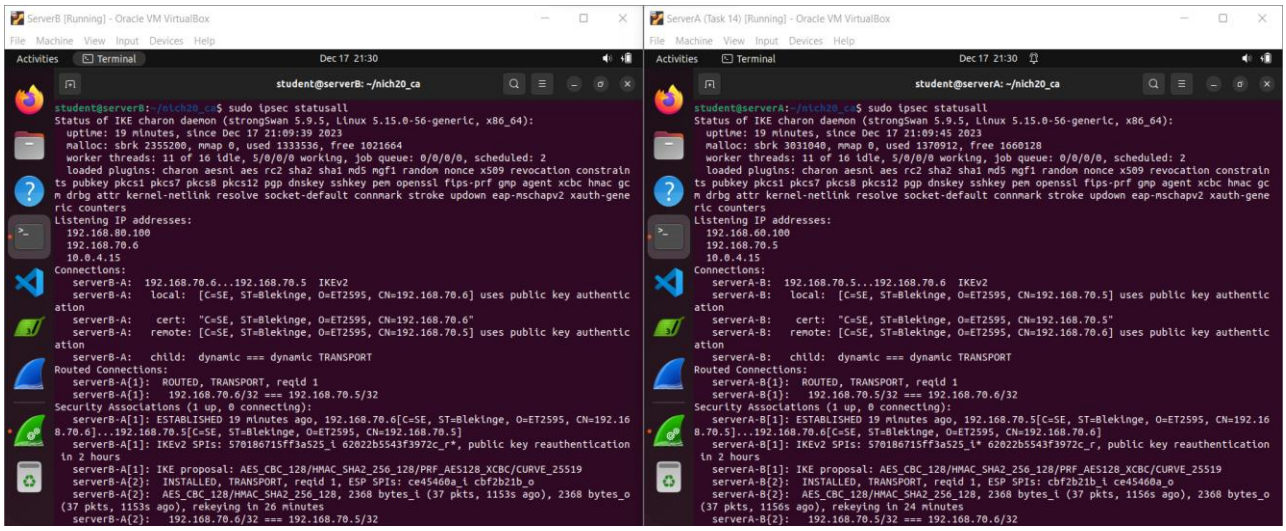
Below, the customization and configuration of the ipsec.secrets and ipsec.conf files are demonstrated, showcasing the adjustments made to suit specific requirements:



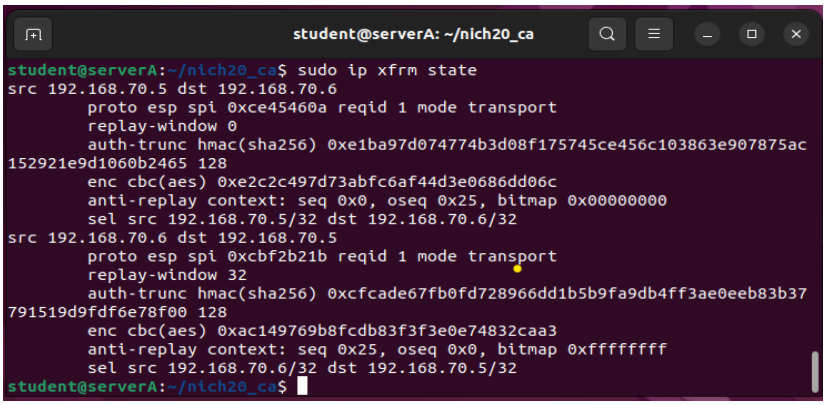


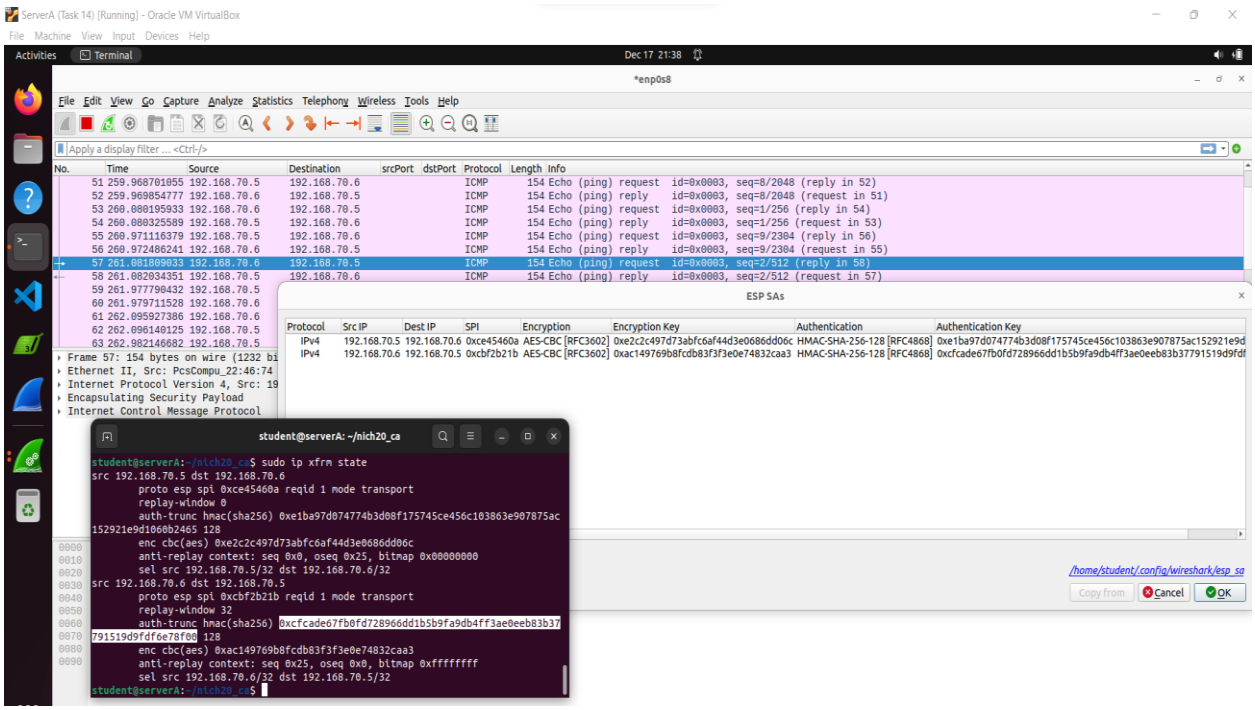
The connectivity between Server A and Server B was established by executing ping commands on both servers. Server A initiated the connection by pinging the IP address 192.168.70.6, while on Server B, the connection was initiated by pinging the IP address 192.168.70.5. These commands were instrumental in verifying the connectivity and successful establishment of communication between the two servers. This is where we witness the successful pinging formed between Server A and Server B.



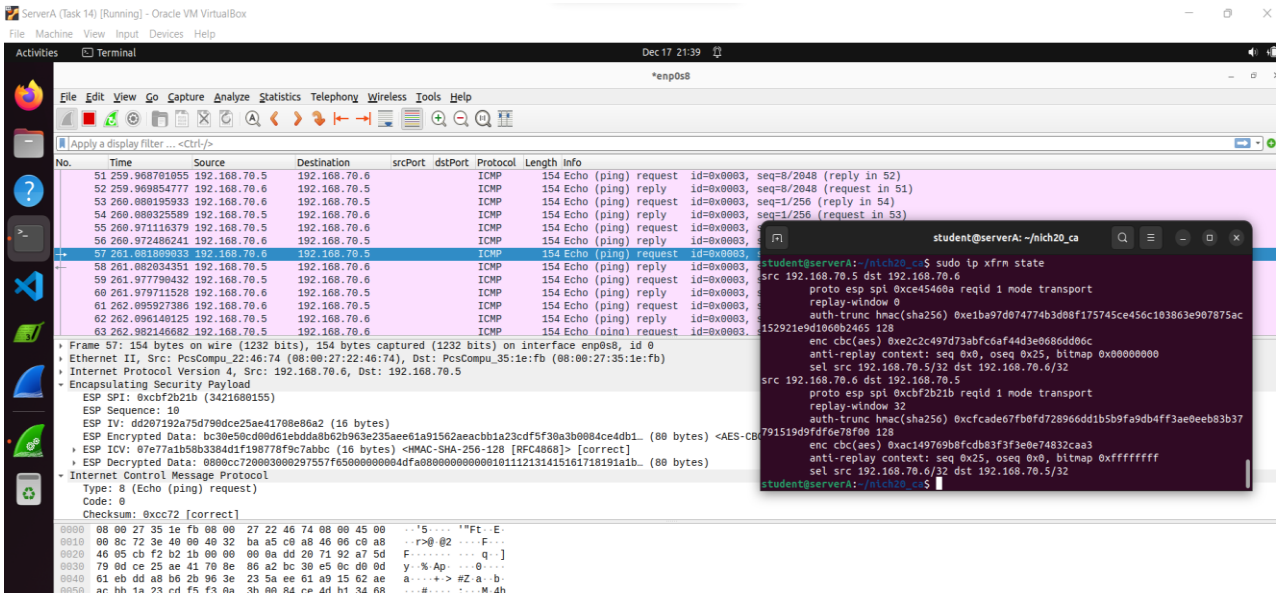


The presence of encryption is clear in ensuring the security of the traffic at this location. Measures have been taken to safeguard the transmitted data through encryption.



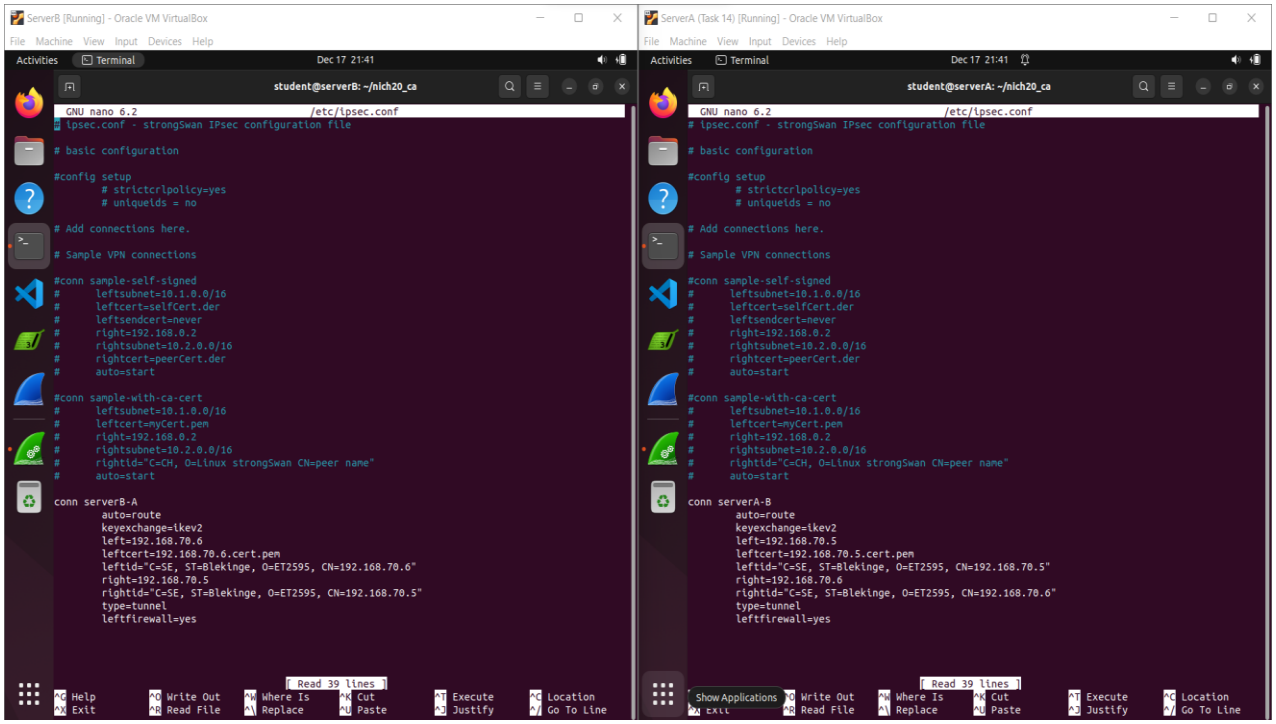


The method used to decrypt the traffic appears to align with the process executed in Task 16. It seems the decryption of the traffic follows a similar approach as the one employed in our previous Task 16.

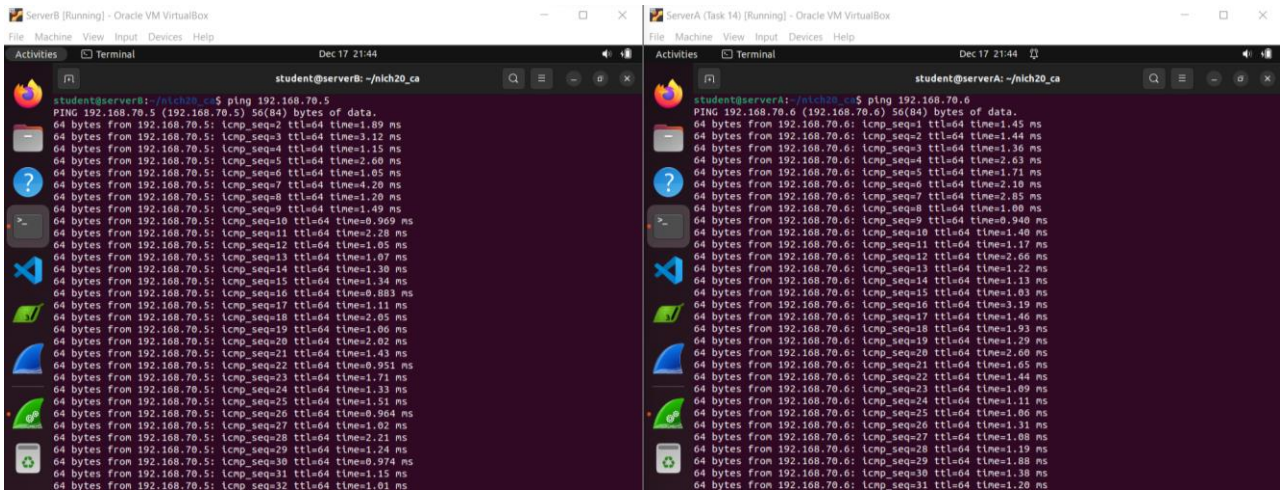


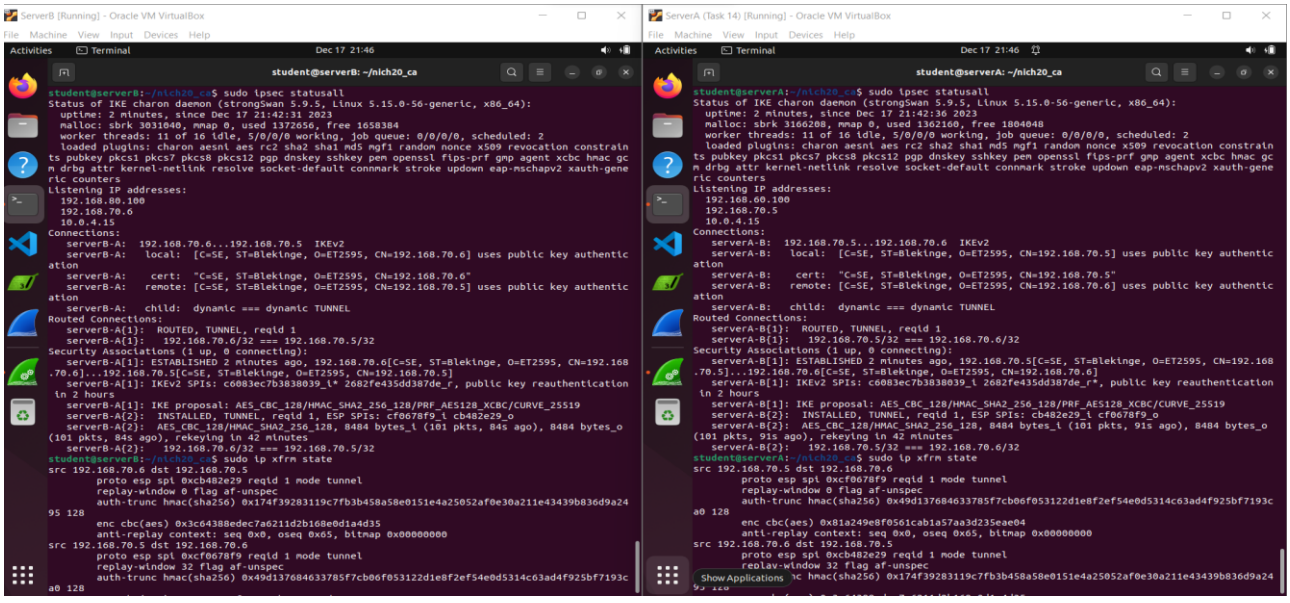
Performing of Task 19:

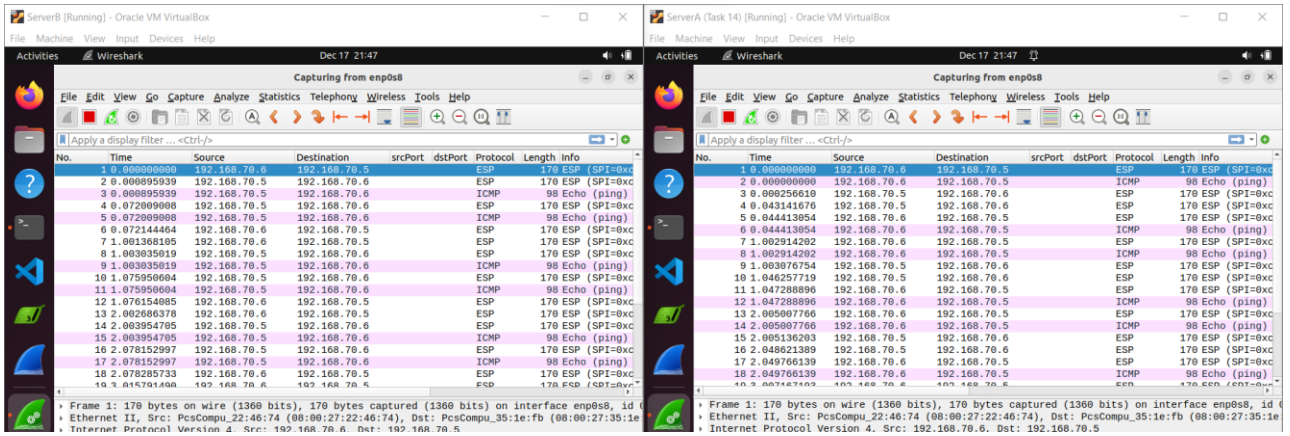
Tunnel mode maintains consistency with the majority of the settings established in task 18. With a notable alteration being made to the "type" parameter within the ipsec.conf files. Here for both Server A and Server B. the setting of the "type" to "tunnel" is pivotal as it enables the IPsec connection to function in a tunneling configuration, primarily utilized to secure communication between distinct networks or subnetworks.

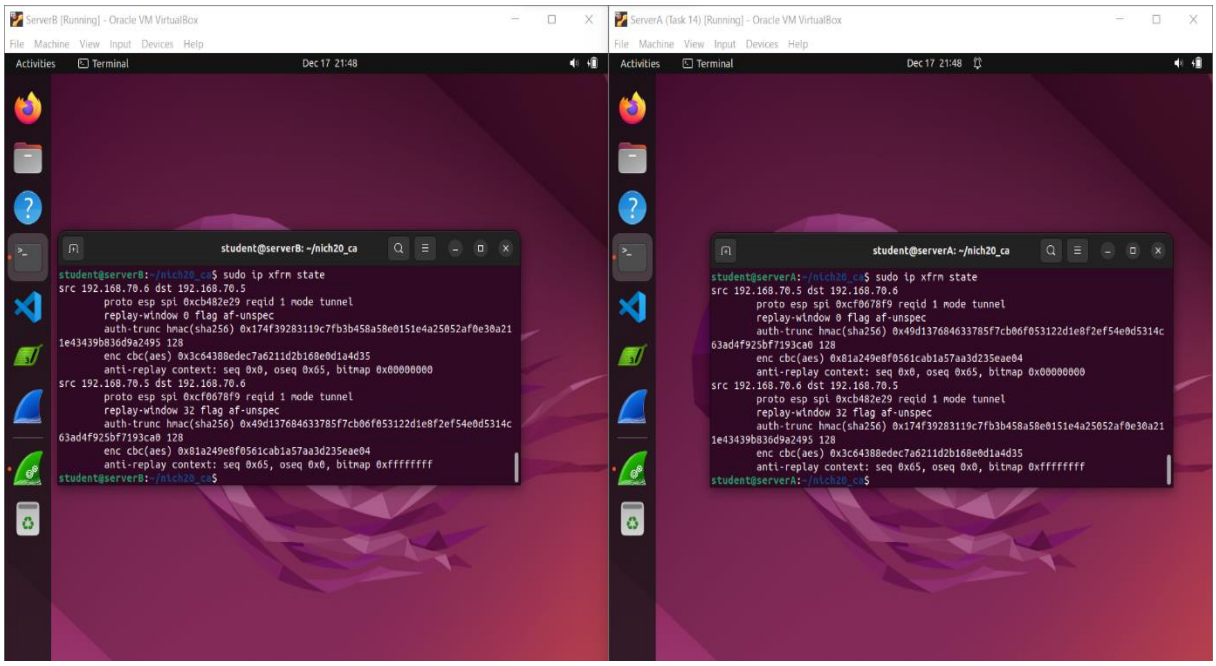


Sure, the communication link between Server A and Server B was established by using the 'ping' command. On Server A, the connection was started by pinging the IP address 192.168.70.6, while on Server B, the initiation of the connection was made by pinging the IP address 192.168.70.5. This process allowed both servers to check for connectivity and response time between them by sending test packets back and forth.



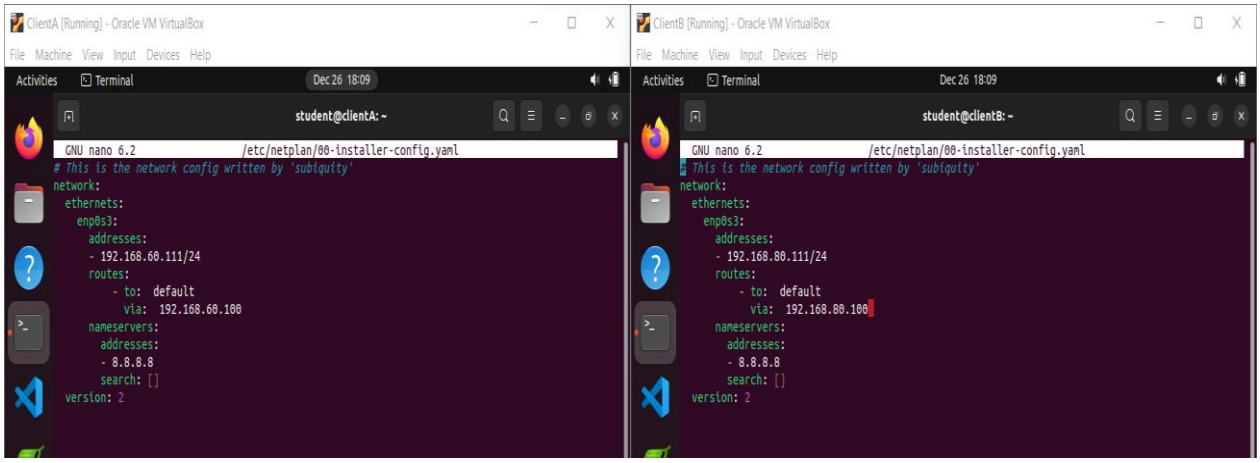




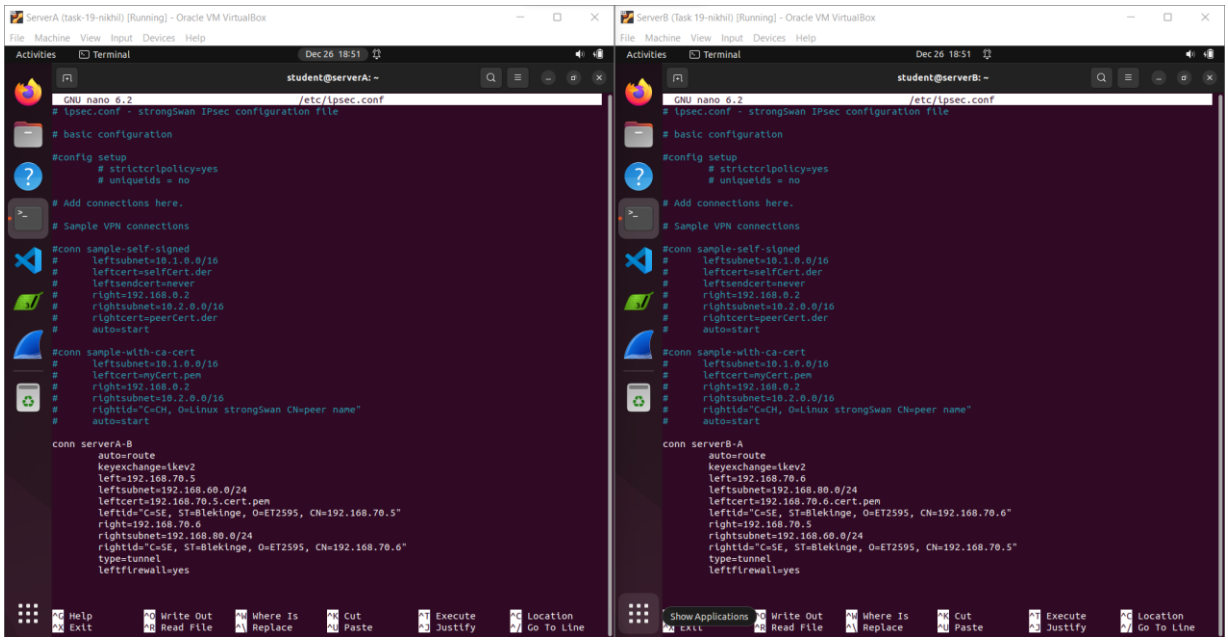


Performing of Task 20:

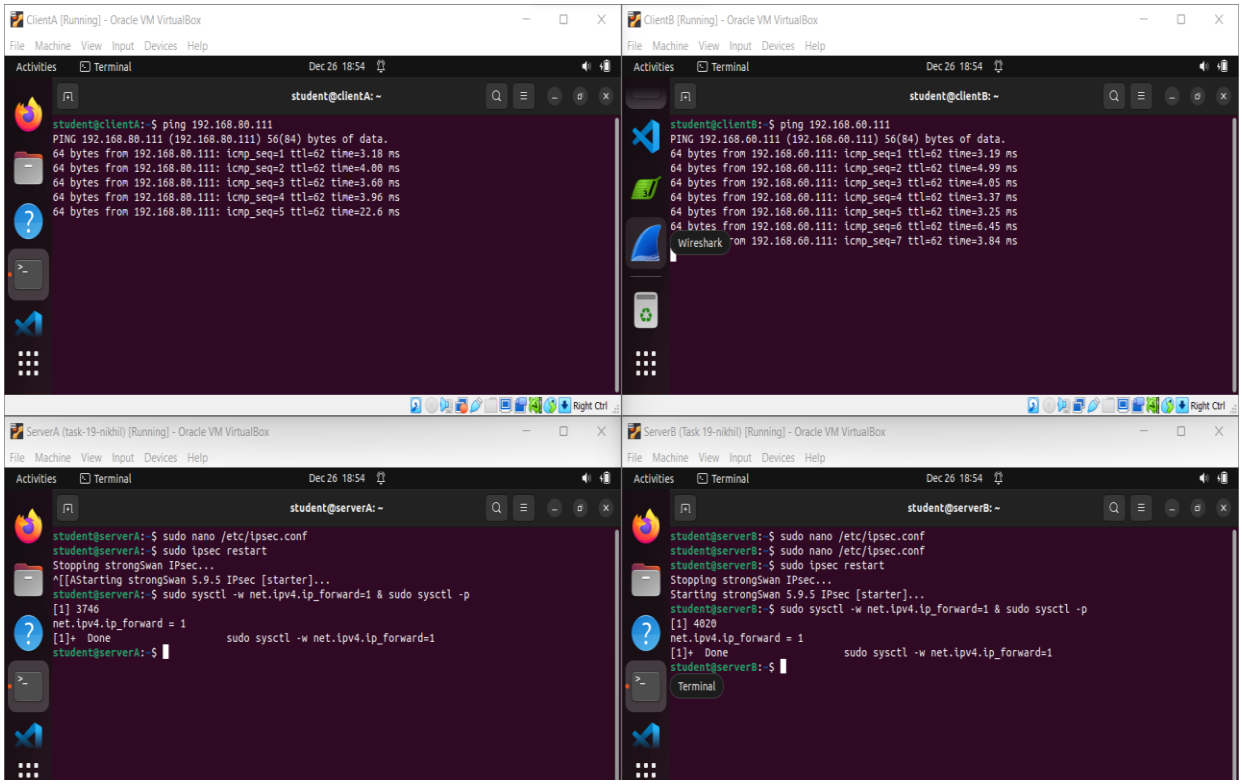
Certainly! A ping test was carried out to confirm the tunneling link between Client A and Client B. This connection was set up by configuring specific gateways for each client. In the "/etc/netplan/00-installer-config.yaml" file, Client A's gateway was set as 192.168.60.100, while Client B's gateway was configured as 192.168.80.100. Through this configuration, both clients were able to communicate with each other using the designated gateways, ensuring a functional and connected tunnel between them.

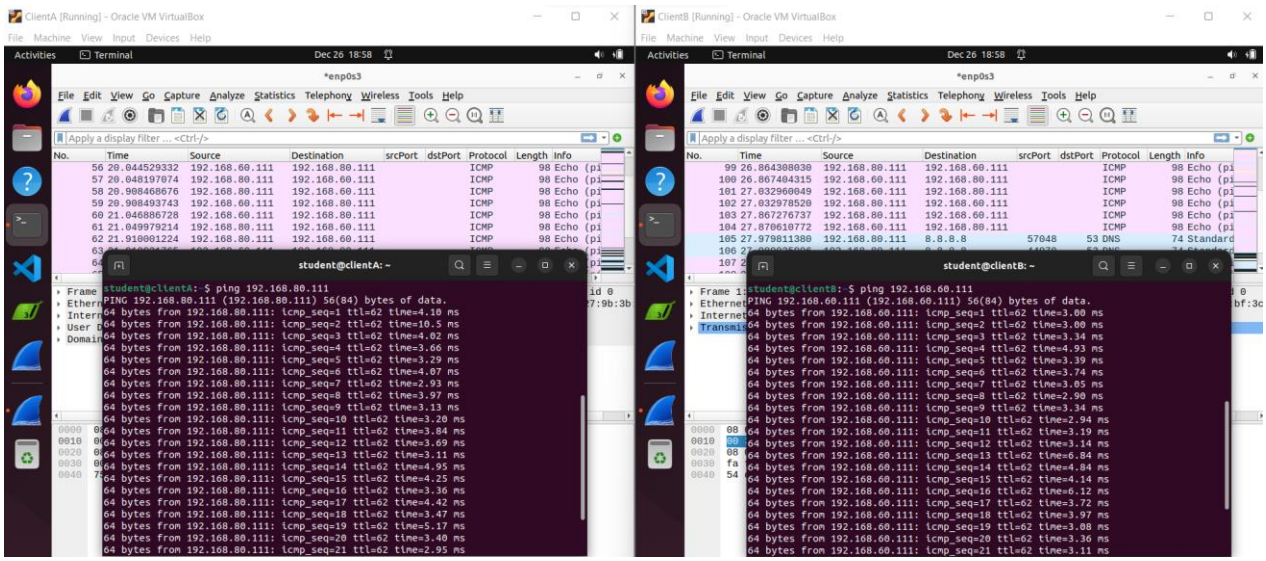


In the ipsec.conf files of Server A and Server B, specific subnet values were defined for the IPsec connection. Server A's configuration sets the left-subnet value as 192.168.60.0 and the right-subnet value as 192.168.80.0. Conversely, Server B's configuration reverses these values, setting the left subnet to 192.168.80.0 and the right subnet to 192.168.60.0. This setup ensures that each server accurately recognizes and directs traffic to the corresponding subnets at the opposite end of the established tunnel. Meanwhile, the ipsec.secrets file retains its previous settings, maintaining the necessary security credentials for the IPsec connection without any alterations.



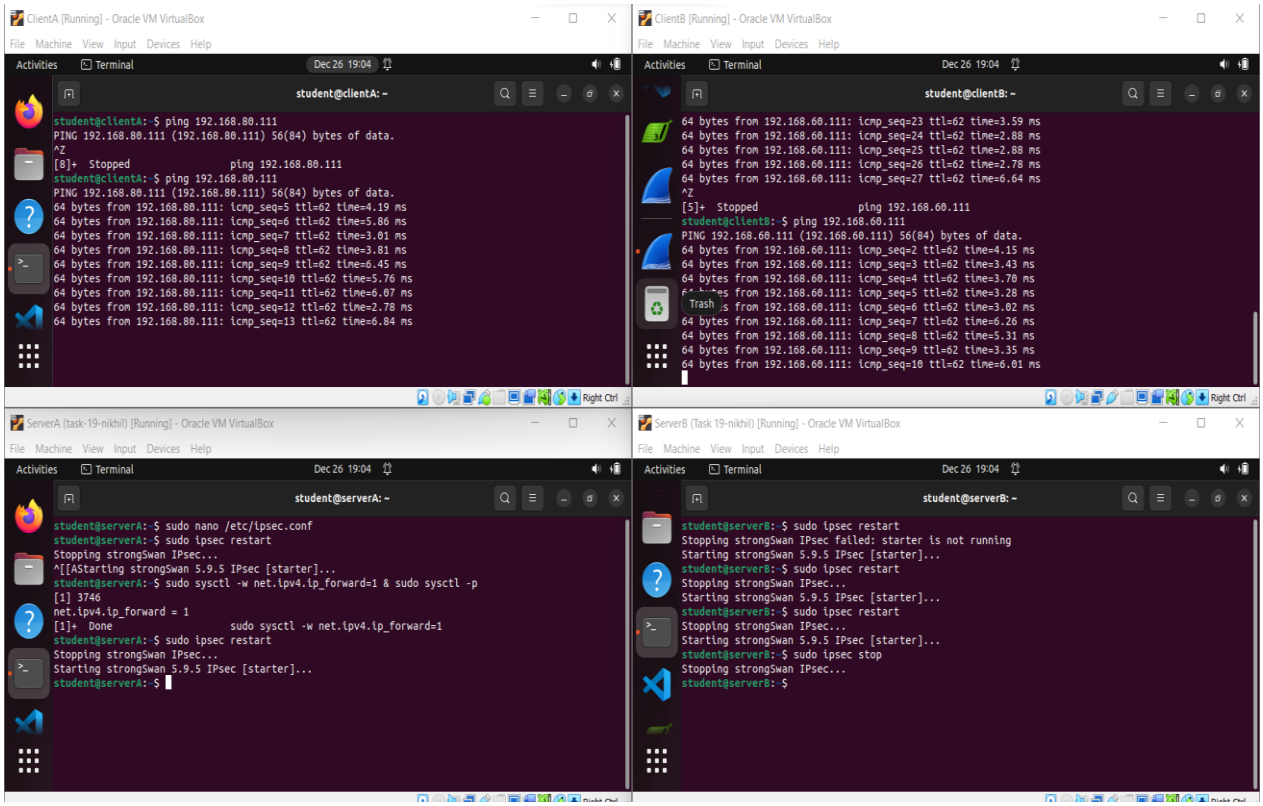
to enable IP forwarding on both servers, two commands were utilized. The first command, "*sudo sysctl -w net.ipv4.ip_forward=1*", modifies a system parameter to allow the forwarding of IP packets between network interfaces. Following that, the second command, "sudo sysctl -p", is executed to apply and activate these system changes effectively. These commands work together to enable dynamic adjustments of system settings, ensuring improved network routing capabilities on the servers.



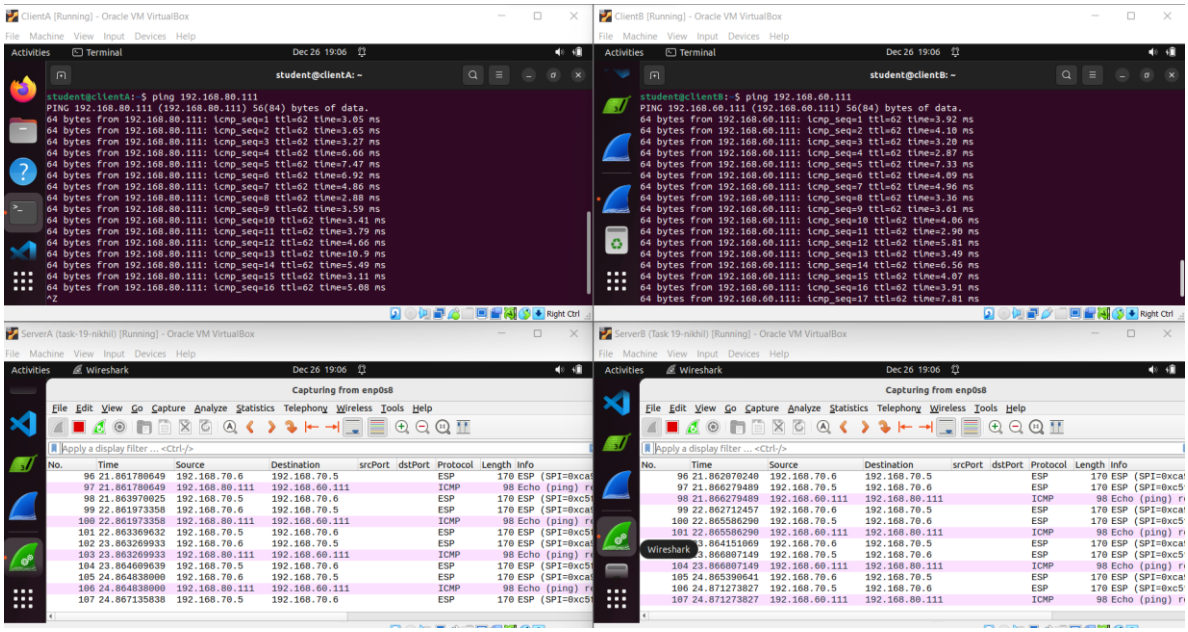


To make sure that the established tunnel was being used for communication, initially trying to spot ESP (Encapsulating Security Payload) packets didn't give us a clear answer and was quite tricky, especially when dealing with complex technical documentation. However, seeking help from the Stack Overflow led us to a simpler and more reliable method.

While we had IP forwarding facilitated, we operated a test. Whenever we deliberately stopped the IPsec service on either of the servers, the ongoing ping communication also stopped. This clear observation, shown in the provided figure, strongly suggested that the communication between Server A and Server B was actually relying on the set-up tunnel. This test effectively showed us that the established tunnel was indeed crucial for the communication between the two servers.

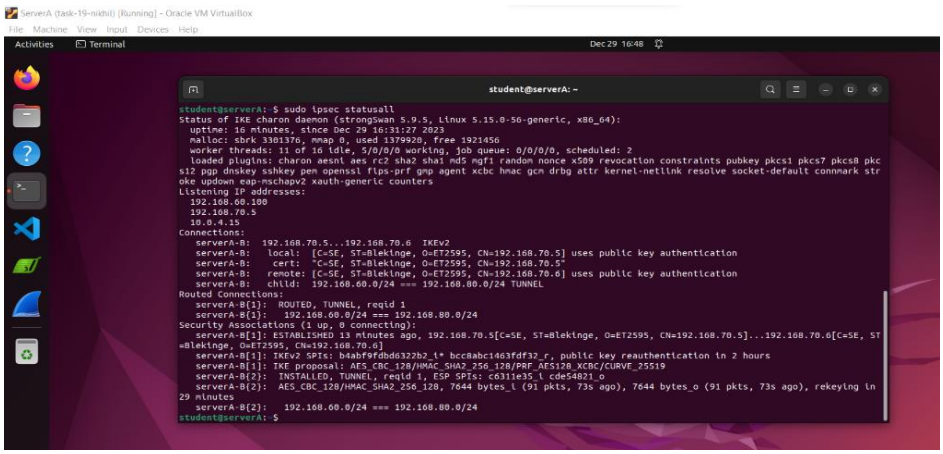


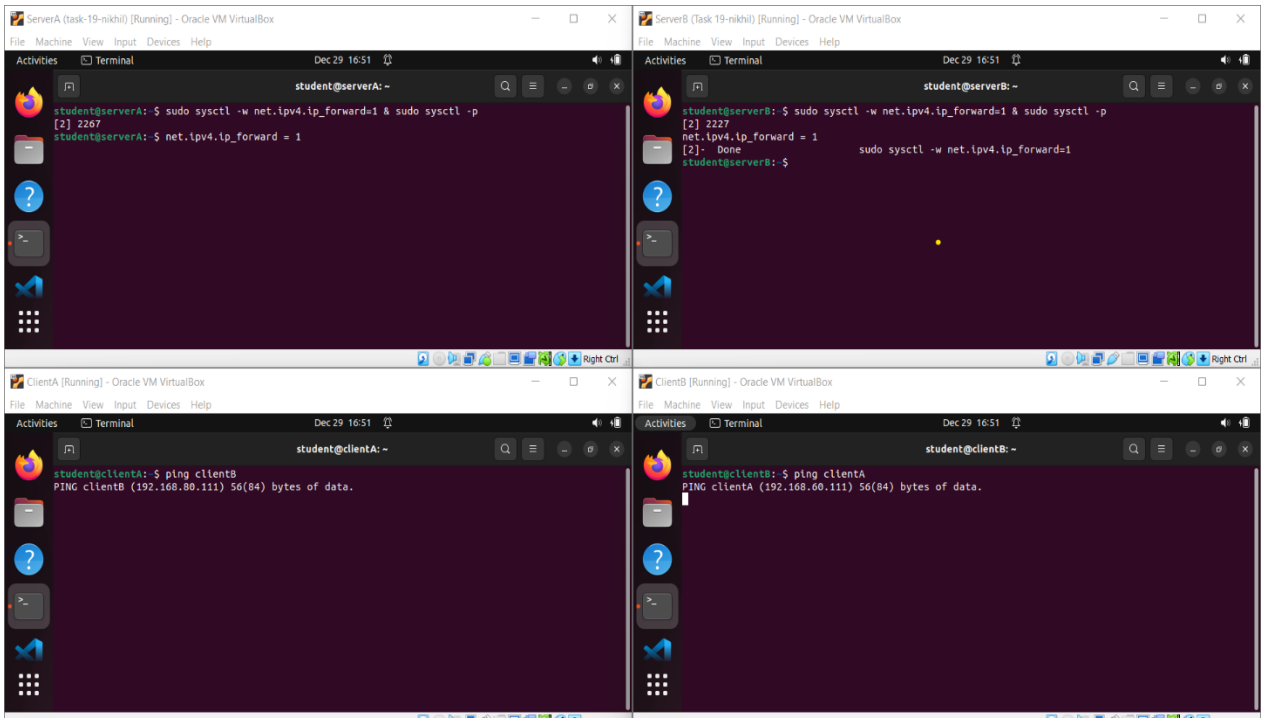
The ESP (Encapsulating Security Payload) traffic observed originating from both servers is a strong indicator of the active involvement of the IPsec tunnel in securing the communication flow between them. This ESP traffic presence signifies that data being transmitted between the servers is undergoing encryption within the established tunnel. This encryption ensures that the exchanged information remains confidential and maintains its integrity throughout the communication process.



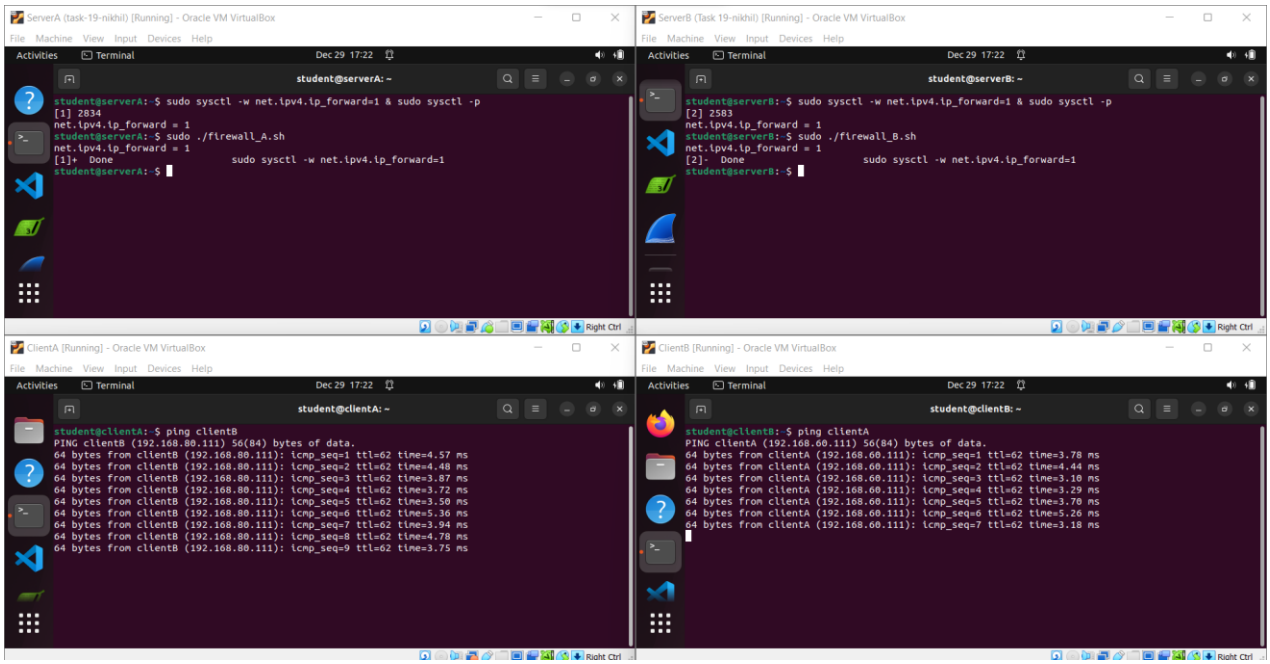
Performing of Task21:

In this particular task, while keeping the configurations in 'ipsec.conf' and 'ipsec.secrets' unchanged, the focus was on adjusting the firewall rules on both servers. These modifications aimed to enable site-to-site communication and ensure that both clients could access the internet through their respective servers. These adjustments were intended to facilitate communication between the sites in a secure manner while allowing Client A and Client B to access the internet via their designated servers. Each screenshot showcased the required alterations made to the firewall settings, ensuring seamless communication and internet access for both clients through their respective server setups.

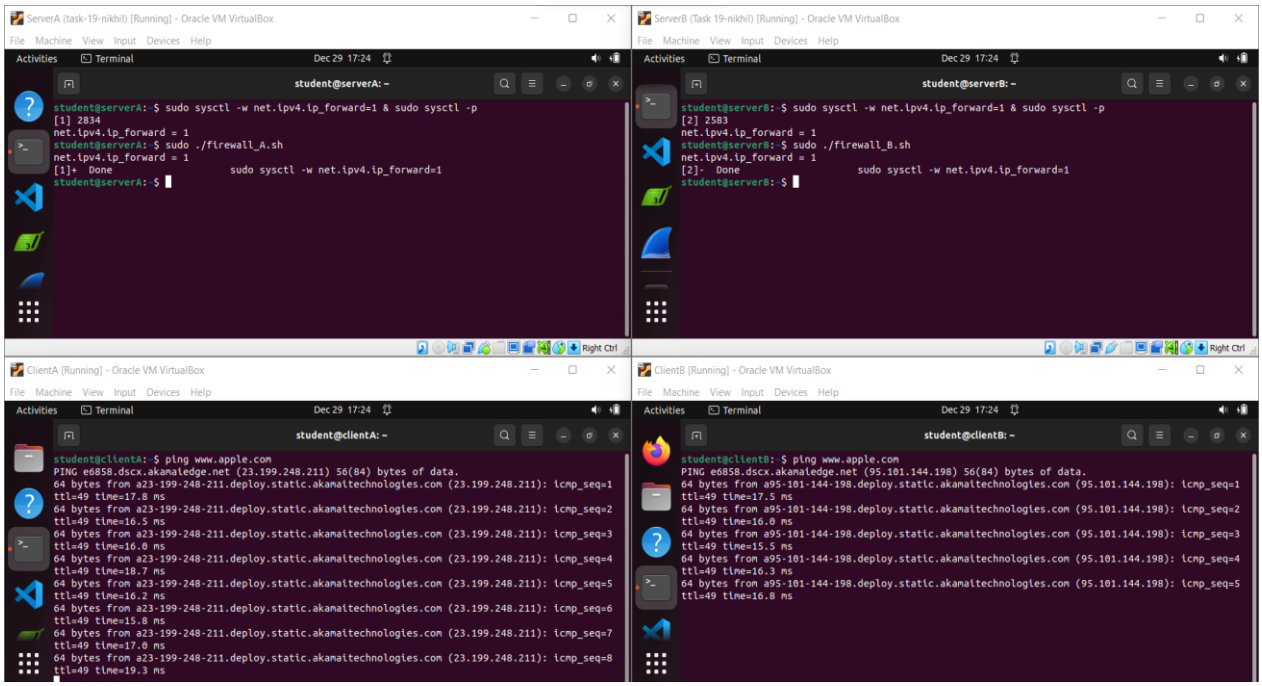




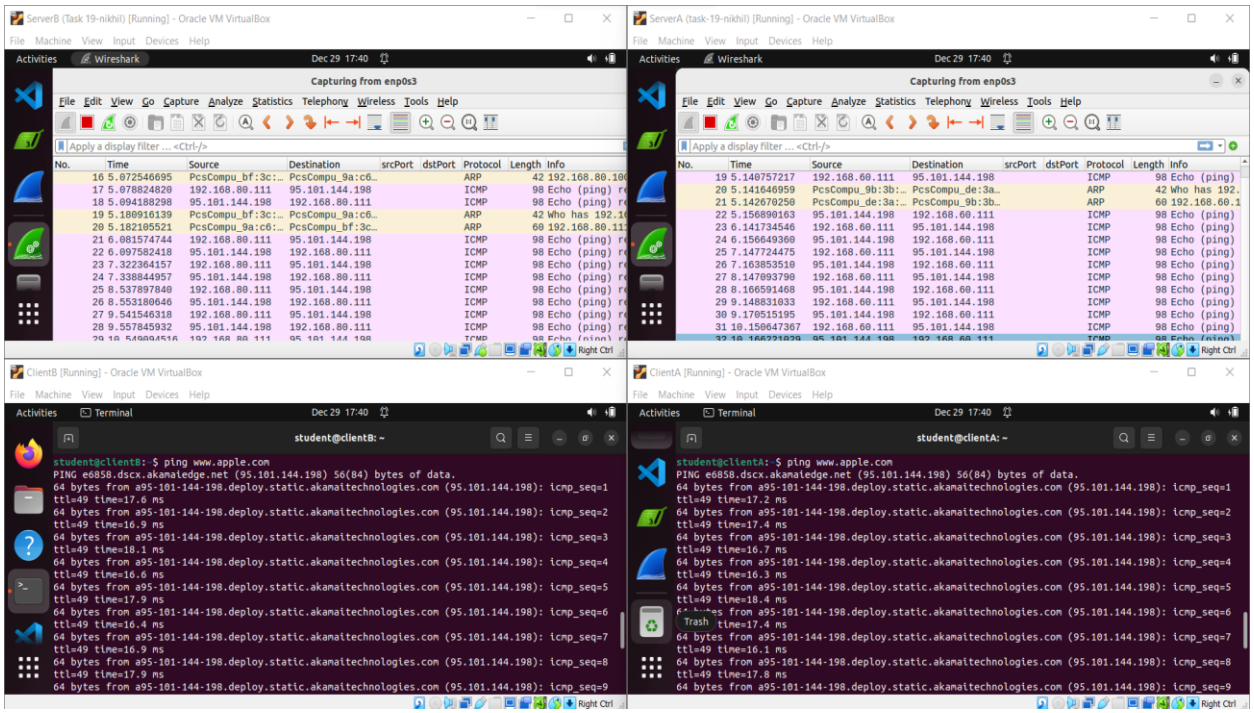
At this point, despite setting up appropriate rules in IPsec and enabling IPv4 forwarding on the servers, the clients were unable to ping each other. To resolve this, specific firewall rules were crafted and implemented on both Server A and Server B. Once these firewall rules were applied, an immediate change was observed: the clients regained the ability to ping each other successfully. These adjustments in the firewall settings directly facilitated the establishment of communication between the clients, resolving the issue of ping connectivity that existed previously.



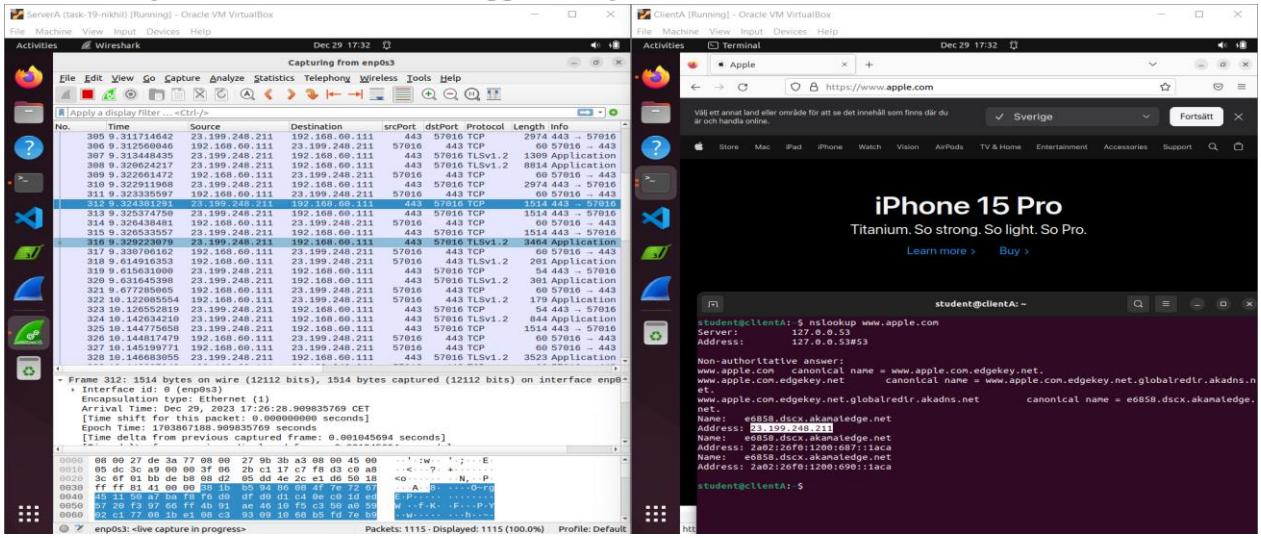
After setting up firewall rules on both Server A and Server B, the clients regained their capability to ping each other effectively. This positive outcome clearly indicates that modifying and implementing the firewall rules played a crucial role in resolving the communication problems that existed earlier. These adjustments directly facilitated the restoration of connectivity between the clients, addressing the previous issues they encountered while attempting to communicate.



Demonstrate the network connection between Client A and Client B through Server B to Server A using Wireshark captures on the respective machines.

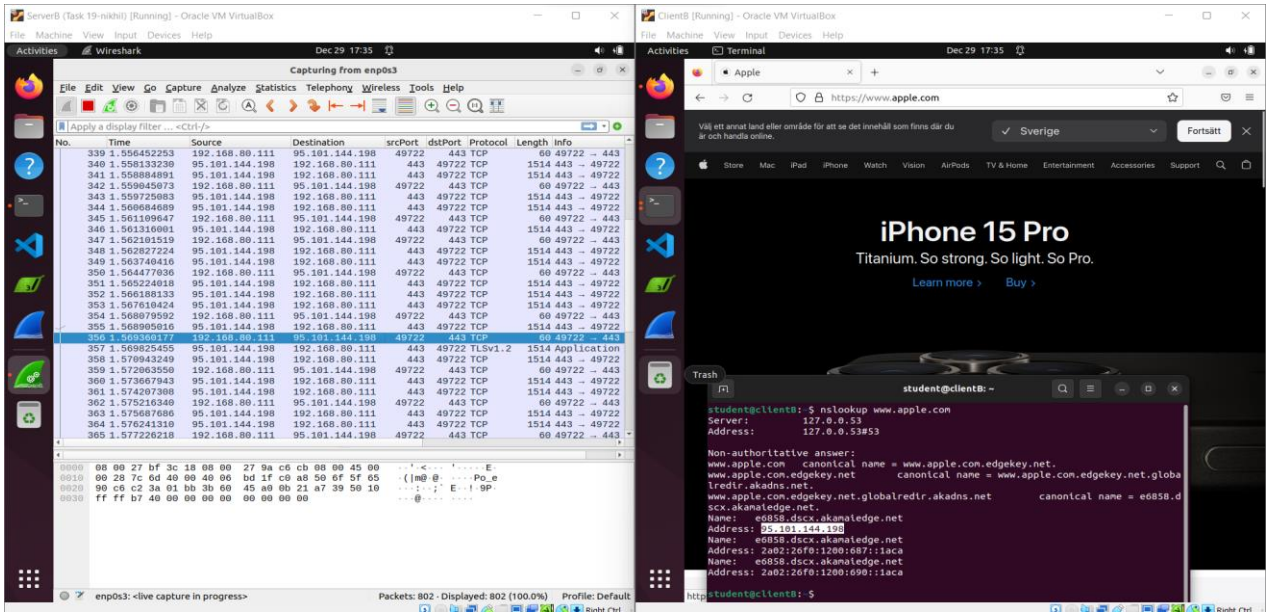


After confirming the ping connectivity, it was ensured that both Client A and Client B had internet access through their respective servers. This was validated by successfully accessing facebook.com from both clients, affirming their ability to utilize their servers for internet connectivity. This verification was supported by visual evidence.

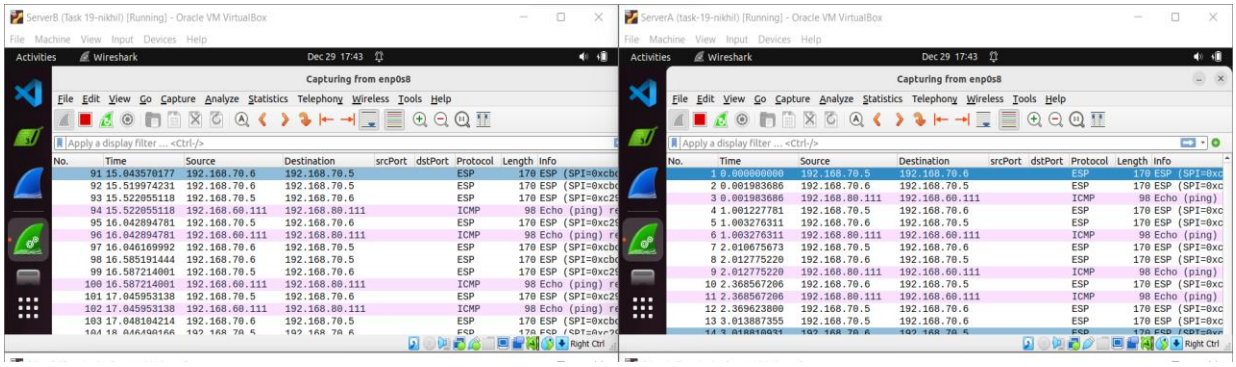


To ensure that both clients could access the internet via their assigned servers, we utilized Apple's IP address, 23.199.248.211, to confirm connectivity. The validation was confirmed by examining Wireshark session logs, which indicated successful internet connections for both clients routed through their respective servers.

We compared the recorded web page reload on Server A when Client A accessed the internet with the *nslookup* displayed in Client A, revealing the identical IP address, 23.199.248.211 Similarly, we conducted the same comparison for Client B's internet access, analyzing the data captured on Server B. This comparison highlighted the consistency between the web page reloads and the corresponding ping requests, reaffirming that both clients were effectively accessing the internet through their designated servers.



Certainly! Encapsulating Security Payload (ESP) traffic was captured separately on both Server A and Server B. This examination involved observing and analyzing the ESP traffic present on each server's network interface. The goal was to comprehensively study the encrypted data flow within the communication channels of both servers.



On Server B, a detailed capture of ESP (Encapsulating Security Payload) traffic was conducted. This detailed analysis aimed to examine the ESP traffic more closely, allowing for a comprehensive understanding of the encrypted data flow within the network. This deeper examination on Server B provided a more in-depth insight into how the Encapsulating Security Payload operates within the communication framework.