**Network Security Project 2 Design Documentation**

Group 10

We have relocated the server functionality from the customers’ local networks to the cloud using strongSwan, and changed the cloud network to use the private IPv4 address space 10.1.0.0/16.

Diagram

Description automatically generated

**Routing**

**Firewall**

1. We simply set the iptables rules as our firewall.
2. We unconditionally accept the internal and the vagrant ssh network (virtual machine) traffic.
3. For each gateway, we accept the IKE session traffic that come from the other sides from port 8080.
4. Drop anything else.

**PKI**

We generate the certificates based on the Elliptic Curve Digital Signature Algorithm (ECDSA). The certificate fingerprint is generated using SHA512 hash function. The Root CA is valid for 10 years. The intermediate CA is valid for 5 years. The end-entity certificates are valid for 2.5 years. Except for the root CA certificate, each certificate is generated by first generating a ECDSA private key, then generating the Certificate signing request, finally issuing the certificate from the private key and with CA. The details for certificates generation are as follows:

1. We generate the root CA. The subject of it is "C=FI, O=CSE4300, CN=CSE4300 Root CA". Put the root CA certificate into “/etc/ipsec.d/cacerts/caCert.pem”.
2. We generate the intermediate CA signed by the root CA. The intermediate CA is assigned with pathlen parameter, which is equal to 0. The subject of it is "C=FI, O=CSE4300, CN=CSE4300 INT CA". Put the intermediate CA certificate into “/etc/ipsec.d/cacerts/intCaCert.pem”.
3. Generate the end endity certificate signed by the intermediate CA. For site A, the subject is "C=FI, O=CSE4300, CN=CSE4300 Site A 172.16.16.16", site B is "C=FI, O=CSE4300, CN=CSE4300 Site B 172.18.18.18", cloud is "C=FI, O=CSE4300, CN=CSE4300 Cloud 172.30.30.30". We set the serverAuth flag for the cloud, and the clientAuth flag for site A and site B. Store the site A certificate in “/etc/ipsec.d/certs/siteACert.pem”, site B certificate in “/etc/ipsec.d/certs/siteBCert.pem”, cloud certificate in “/etc/ipsec.d/certs/cloudCert.pem”.
4. Generate the Certificate Revocation List (CRL) for both the root CA and the intermediate CA, store them into “/etc/ipsec.d/crls/”
5. Finally, we store the secret keys of root CA and intermediate CA into vault. Each gateway has a copy of root CA certificate and intermediate CA certificate, the secret key that belongs to them, as well as the end entity certificates for both sides that will be used when communicate.

In conclusion, our PKI architecture has one root CA and another intermediate CAs, where the private key of the intermediate CA is used to sign the end entity certificates and the private key of the root CA can be kept on a smartcard stored in a safe or at lease on a system disconnected from the Internet. The private root CA key is never stored on an insecure or online system. Securing the root CA enables the PKI administrator to revoke any certificates and recreate the PKI from scratch, if any intermediate CAs are compromised.

**IPSec**

1. We set the IKEv2 as our key exchange protocol.
2. We select our ciphers according to the Commercial National Security Algorithm (CNSA) Suite where the strongSwam mentioned in their security recommendations: <https://docs.strongswan.org/docs/5.9/howtos/securityRecommendations.html#_cipher_selection>, which is, we use aes256gcm16-prfsha384-ecp384 for the IKE and aes256gcm16-ecp384 for the ESP.
3. We use static ip addresses, store the certificates of both sides in the gateway in advance, and do identity and CA check when authentitcate in case of the man in the middle attack.
4. We set the VPN connection to load a connection and brings it up immediately when started up.
5. We set dpdaction to hold, so that empty INFORMATIONAL messages (IKEv2) are periodically sent in order to check the liveliness of the IPsec peer. A trap policy is installed which will catch the traffic when the IPsec peer is dead, and tries to re-negotiate the connection on demand.
6. We make the Perfect Forward Secrecy (PFS) enabled just as the default. It makes IPsec peers negotiate an independent session key for each IPsec or CHILD SA. This protects the long-term confidentiality of the IPsec traffic if the IKE shared secret is leaked. Note that the session keys of the first CHILD\_SA of a new IKEv2 connection are derived from the IKE shared secret. However, subsequent CHILD\_SAs will use independent keys if PFS is used.