HPE In-Semester Project

Certificate Chain Validation

By -
Pranav Kulkarni
Teja Juluru

Mentors -
Prof Rajesh Gopakumar
Vicramaraja ARV

Features

- Written in C++
- Uses OpenSSL libraries
- Supports both CRL and OCSP validation
- Cmake used to generate makefile
- Can handle both PEM and DER encoded CRL files
- Ability to display more information using ‘*-v*’ flag.

Program Flow

1. Convert the chain file encoded in PEM format into a format usable by openSSL.
2. Extract the serial numbers from the chain file.
3. Convert the PEM or DER encoded CRL file into a format usable by openSSL.
4. Extract the revoked serial numbers from the CRL file.
5. Check for the serial numbers of the chain file in the CRL file.
6. Additionally, validate each certificate in the chain, from leaf to root by sending an OCSP request.

Opening the chain file

1. Input the path of the chain file from the user.
2. Create an *SSL_CTX* object using the function **SSL_CTX_new()**.
3. Set the *SSL_CTX* object to use the chain file from the path provided by the user.
4. *SSL_CTX* now contains all the certificates except the leaf in a *STACK_OF(X509)*. Extract the stack using the function **SSL_CTX_get0_chain_certs()**.
5. Extract the leaf certificate from the *SSL_CTX* object using **SSL_CTX_get0_certificate()** function.
6. Insert all the certificates into a new *STACK_OF(X509)* object.

Opening the CRL file

1. The program can handle both PEM and DER encoded files.
2. This is achieved by checking the ***first byte*** of the file. If the first byte of the file is ‘0x30’, then it is a DER encoded file and can be handled accordingly.
3. If it is a DER encoded file, use **d2i_X509_CRL_bio()** function to create an *X509_CRL* object.
4. If it is a PEM encoded file, use **PEM_read_bio_X509_CRL()** function to create an *X509_CRL* object.

CRL Validation steps

1. Get a *STACK_OF(X509_REVOKED)* from the *X509_CRL* object. There is one *X509_REVOKED* object for each entry of serial number in the CRL file.
2. Iterate through the stack and extract the serial number from each *X509_REVOKED* object. Store each serial number in a hash map.
3. Iterate through each certificate and check if its serial number is present in the hash map.
4. If the serial number is found in the hash map, then the certificate is revoked.

OCSP Validation

1. The link to OCSP server can be found in the Authority Information Access (AIA) extension of the certificate. There can be multiple links.
2. Iterate through the certificate chain and extract the URL to the OCSP server.
3. All the OCSP URLs can be extracted from the certificate using the function **X509_get1_ocsp()**. The extracted URLs are stored in a *STACK_OF(OPENSSL_STRING*).
4. Create an *OCSP_CERTID* object for the current certificate and its issuer.
5. Create an *OCSP_REQUEST* object using the *OCSP_CERTID* object

OCSP Validation (Continued)

6. Create an *OCSP_REQ_CTX* object using the connection *BIO* and the host and port of the URL.

7. Set the *OCSP_REQ_CTX* object using the *OCSP_REQUEST* object we created earlier.

8. Connect to the OCSP responder using the connection BIO and send the *OCSP_REQ_CTX* object we created. The response sent by the server is stored in an *OCSP_RESPONSE* object.

9. Extract the status of the certificate from the response.