Test Strategy Document

X509 Certificate Validation

By : B S SAGAR

Date : 22-03-2025

Table of Contents

1. Test Scope & Objectives
    - 1.1 Key attributes of X509 to be validated
    - 1.2 Importance of Verifying certificate integrity
2. Manual vs. Automated Testing
    - 2.1 Aspects of Manual testing
    - 2.2 Consideration for Automation
3. Test Scenarios
    - 3.1 Positive test cases
    - 3.2 Negative test cases
    - 3.3 Edge cases
4. Automation Strategy
    - 4.1 Tools and Framework
    - 4.2 Test Data handling
5. Security & Performance Considerations
    - 5.1 Security Considerations
    - 5.2 Performance Considerations

Overview on X509:

X509 certificates are a fundamental part of Public Key Infrastructure (PKI) and are used to verify the authenticity of digital identities over secure networks. These certificates enable **SSL/TLS encryption**, ensuring secure communication between clients and servers.

Each X509 certificate contains details such as:

- **Subject Name**: Identifies the entity (e.g., website, organization)
- **Issuer**: Certificate Authority (CA) that issued the certificate
- **Validity Period**: Start and expiry date of the certificate
- **Public Key**: Used for encryption and authentication
- **Digital Signature**: Verifies the authenticity of the certificate

1. Test Scope & Objectives:

1.1 Key attributes of X509 to be validated

i. Validate whether the certificate complies with X509 standards which includes subject, issuer, certificate validity, signature, serial number.
ii. As certificate will be digitally signed, verify the digital signature algorithm.
iii. Verify the start date and end date of the certificate.
iv. Verify the certificate against CRL and OSCP, to check if it is revoked.
v. Validate the certificate authority.
vi. Verify certificate's Key usage and extended key usage fields.

1.2 Importance of Verifying certificate integrity:

Certificate integrity verification is most important as we check below,

i. The certificate has not been tampered with, protecting against man-in-the-middle (MITM) attacks and fraudulent certificates.
ii. With authentication we can verify that the certificate is from an identified and verified source which can be trusted.
iii. Validate the certificate against security standards and industry regulations (ex: RFC 5280 (X.509 v3 certificates) and relevant RFCs for TLS/SSL (e.g., RFC 5246, RFC 6066)).

2. Manual vs. Automated Testing

X509 Certificate validation testing can be done with a combination of manual and automated testing

2.1 Aspects of Manual testing:

When it comes to manual testing of X509 certificate validation, we need to mainly consider the visual inspection and real-time validations which might frequently

change and need manual intervention. Below are few cases that required manual intervention:

i. Visual inspection of certificate structure
ii. Serial number – which is generated uniquely every-time.
iii. Testing and validation of a certificate from a rare or untrusted CA, invalid extensions, missing information, mismatched hostname to find any unique behavior of the certificate.
iv. Validation of revoke status of the certificate, where CRL and OCSP responses are directly accessed and analyzed.

2.2 Consideration for Automation testing:

Automation will be crucial for consistent, repeatable tests, data validation, reducing manual errors and is essential for handling more number of certificate validations required.

i. Signature Validation against Issuer public key,
ii. Validity and expiration date or status of the certificate.
iii. To check revocation status of the certificates by connecting to CRL and OCSP servers.
iv. Validation of certificate chains ensures that the entire path to a trusted root CA is correct and intact.

3 Test Scenarios:

3.1 Positive test cases

i. Compliance to x509 standards which includes required fields such as subject, issuer, validity, signature, and the extensions (such as Key Usage, Extended Key Usage, etc.)
ii. Common Name (CN) validation of the certificate.
iii. Valid certificate with correct structure and signature
iv. Issuer validation by checking the CA's root certificate, included in the trust store.
v. Signature validation against the Issuer public key.
vi. Signature Algorithm (commonly accepted algorithms like RSA or ECDSA.)
vii. Validity of the certificate that is within its Not Before and Not After validity period.
viii. Strength of the public key
ix. Validation of a self-signed certificate by manually trusting the issuer (self-signed root CA).
x. Validation of key usage extensions (e.g., digital signature) for its intended purpose (eg. SSL/TLS, email signing) along with key usage (eg. server authentication, client authentication) based on its intended usage.
xi. Verify that a certificate is not listed in the CRL or OCSP as revoked.

3.2 Negative test cases:

i. Validation of expired certificates where certificates will be rejected.
ii. Validation on Malformed certificates where incorrect details are present or certificate that does not comply x509 standards.
iii. Test the system for rejection of revoked certificates.
iv. Test the system to identify Invalid signature where it might be tampered with.
v. Attempt to validate a certificate where the current date is earlier than the certificate's Not Before date.
vi. Test a certificate signed or self-signed by an untrusted or unknown Certificate Authority (CA) that is not present in the trust store.
vii. Validate a certificate with improper or incorrect Key Usage extensions (e.g., a certificate intended for server authentication being used for code signing).
viii. Test a certificate that cannot be validated because its certificate chain is incomplete, or an intermediate certificate is missing.

3.3 Edge cases:

i. Test a certificate that uses an outdated or unsupported signature algorithm (e.g., MD5 or SHA-1).
ii. Test certificate that uses weak key size, such as RSA keys smaller than 2048 bits.
iii. Test the system to reject certificate which is expired but not revoked.
iv. Test the system to accept the certificate that is signed by multiple CAs using different signatures where both traces back to truster CAs.
v. Test the system to accept a very large certificate with a huge RSA key size like 8192 bits or a certificate chain with many intermediate certificates, without any performance or memory issues.
vi. Certificates with unusual or max-length fields

4 Automation Strategy:

4.1 Tools and Framework:

i. Java – Core language used
ii. Java keystore API – To Parse the certificate
iii. OpenSSL – script command to generate certificate
iv. TestNG - for structured framework and testcases management.
v. Data driven – To drive test data using data provider
vi. Maven – Project and dependency management
vii. GitHub & GitHub Actions – Remote repository and Continuous Integration

4.2 Test Data handling:

i. Generate a diverse set of test certificates using OpenSSL
ii. Store test certificates in a version-controlled repository
iii. Use parameterized tests to run scenarios with different certificate variations

5 Security & Performance Considerations:

5.1 Security Considerations:

i. Restrict to allow only RSA keys of 2048 bits or greater for security.
ii. Always ensure that SHA-256 (or stronger) is used for certificate signing and hashing. Reject certificates that use outdated or weak algorithms like MD5.
iii. Ensure OCSP responses are valid and timely, and CRL is periodically checked and up to date to avoid relying on expired or revoked certificates.
iv. Validate against weak keys and CA's which were compromised.

5.2 Performance Considerations:

i. Use performance testing tools to measure the time taken to validate certificates, particularly large chains or high-traffic environments.
ii. Test how well the system handles large numbers of certificates and how efficiently it can perform revocation checks.
iii. Implement parallel validation testing (where applicable) to simulate real-world performance under load.