Test Strategy & Planning Document for X.509 Certificate Validation

---

1. Testing Scope & Objectives

Key Attributes to Validate:

- **Certificate Structure:** The validity of attributes such as the version, serial number, signature algorithm, subject, issuer, public key, and validity period (Not Before, Not After).
- **Certificate Trust Chain:** Ensure that the certificate can be validated by following the chain of trust from the leaf certificate to a trusted root certificate.
- **Key Usage & Extended Key Usage:** Ensure the certificate's usage complies with its intended purpose (e.g., digital signatures, key encipherment).
- **Revocation Status:** Verify that the certificate is not revoked by checking CRLs (Certificate Revocation Lists) or using OCSP (Online Certificate Status Protocol).
- **Signature Algorithm & Key Strength:** Ensure that the signature algorithm is secure (e.g., SHA-256 with RSA 2048+) and that the public key is sufficiently strong (e.g., RSA key size ≥ 2048 bits).

Importance of Verifying Certificate Integrity:

- **Trust Establishment:** The integrity of a certificate is essential to establish trust between parties in secure communications. An invalid or compromised certificate undermines security.
- **Prevent Attacks:** Verifying the certificate's integrity helps prevent man-in-the-middle attacks, certificate spoofing, and other malicious activities.
- **Compliance:** Ensuring certificates comply with industry standards and regulations (e.g., GDPR, PCI-DSS) ensures a secure environment.

---

2. Manual vs. Automated Testing

Manual Testing:

- **Certificate Structure Inspection:** For validation of complex certificate attributes such as the presence of extended key usage, SAN (Subject Alternative Names), and proper certificate field values, manual testing is essential to catch edge cases.
- **Edge Case Analysis:** Some edge cases, such as certificates with custom extensions or certificates from obscure certificate authorities, may require manual review to ensure proper handling.
- **Chain of Trust Inspection:** While the process of chaining certificates can be automated, manual inspection might be needed for non-standard trust chains or scenarios involving custom or self-signed certificates.

Automated Testing:

- **Standard Certificate Validation:** Automation tools can efficiently check if a certificate's fields (e.g., validity period, key size, signature algorithm) follow the correct format and meet security requirements.
- **Revocation Check (CRL/OCSP):** Automated checks for revocation status via CRLs or OCSP responders can be done regularly to ensure certificates are still valid.
- **Performance Testing:** Automated scripts can be used to measure the performance of certificate validation, ensuring that validation processes scale efficiently under load.

---

3. Test Scenarios

Positive Test Cases:

1. **Valid Certificate:** Ensure the certificate is correctly issued, within the validity period, and chains correctly to a trusted root certificate.
2. **Correct Structure:** Verify that the certificate contains all required fields such as version, serial number, public key, signature algorithm, subject, issuer, and valid extensions (Key Usage, SAN, etc.).
3. **Proper Key Usage and Extended Key Usage:** Validate that the certificate's intended usages align with its defined key usage and extended key usage fields.

Negative Test Cases:

1. **Expired Certificate:** Check that the certificate's validity period has expired (Not After date).
2. **Malformed Certificate:** Provide certificates with invalid encoding or incorrect field values to ensure they are rejected.
3. **Revoked Certificate:** Test certificates that are revoked (either listed in a CRL or flagged via OCSP).
4. **Invalid Chain of Trust:** Test certificates that fail to chain to a valid root certificate, either due to a missing intermediate certificate or an untrusted root.
5. **Unsupported Algorithms:** Test certificates that use deprecated or unsupported algorithms (e.g., MD5 or RSA key size less than 2048 bits).

Edge Cases:

1. **Certificates with Non-Standard Extensions:** Test certificates containing custom or non-standard extensions and verify they are properly handled.
2. **Certificate with Multiple SAN Entries:** Test certificates with multiple SAN values, ensuring that each value is validated correctly.
3. **Large Certificate Size:** Test certificates with a very large public key or long validity periods to ensure the validation process handles these edge cases.

---

4. Automation Strategy

Tools and Frameworks:

- **OpenSSL:** Used for validating the structure of certificates, checking revocation status (via CRL or OCSP), and ensuring compliance with cryptographic standards.
- **Java KeyStore API (JKS):** For managing and validating certificates in a Java environment, ensuring compatibility with Java-based applications.
- **TestNG:** For running and organizing unit tests for certificate validation, especially in scenarios involving batch processing of certificates.
- **Custom Scripts (Python/Java):** For automating revocation checks, trust chain validation, and performance measurement under various loads.

Test Data and Variations:

- **Data Generation:** Use a variety of certificates generated with different key sizes, algorithms, expiration dates, and extensions. Include both valid and invalid certificates to thoroughly test all validation scenarios.
- **External Data Sources:** Utilize publicly available certificate databases (e.g., root certificate stores, CRL servers, OCSP responders) for testing trust chains and revocation status.
- **Fuzzing:** Use certificate fuzzing tools to generate malformed certificates to test error handling and robustness.

---

5. Security & Performance Considerations

Cryptographic Standards:

- **RSA Key Length:** Ensure that RSA certificates use a key size of at least 2048 bits, as smaller keys (e.g., 1024 bits) are no longer considered secure.
- **SHA-256 or Stronger:** Ensure that certificates use SHA-256 or stronger signature algorithms (e.g., SHA-384, SHA-512) to prevent vulnerabilities like collision attacks associated with weaker algorithms (e.g., MD5, SHA-1).
- **Cipher Suites:** Test that the certificates are compatible with modern, secure cipher suites (e.g., TLS 1.2 or TLS 1.3) that support strong encryption methods.

Performance Measurement:

- **Latency Testing:** Measure the time taken to validate a single certificate, as well as the time to validate multiple certificates in batch scenarios.

- **Scalability Testing:** Implement automated performance tests to measure the impact of certificate validation on system performance under load (e.g., validating 1000 certificates per second).
- **Revocation Status Checks:** Ensure that the time to check certificate revocation (via CRL or OCSP) is minimal and does not impact overall performance.

Conclusion

This test strategy outlines a comprehensive approach to validating X.509 certificates, focusing on ensuring correctness, integrity, and security. By incorporating both manual and automated testing, as well as leveraging appropriate tools and frameworks, the testing process ensures that certificates are robust, trusted, and compliant with cryptographic standards.