Path Discovery and Validation Working Group

Path Discovery and Validation Client Questionnaire

This questionnaire is intended for vendors to provide information about applications that include certificate validation capabilities or plug-ins that add certificate validation capabilities to other applications. The applications or plug-ins may either perform path discovery and validation on their own or may delegate path discovery and/or validation to another system (a delegated path discovery or delegated path validation server).

1. Organization name:

2. Point(s) of contact (name, address, phone number and e-mail address)

3. Product Name and version number:

4. On what platforms (e.g., Linux, Mac OS X, Windows XP, etc.) does the product run? If the product is a plug-in that provides path discovery and validation capabilities to other applications, list the applications for which plug-ins are available (e.g., Apache 2.0 on Linux).

5. Does the client:

   □Perform path discovery and validation (please answer questions 7 through 14).

   □Perform path validation while relying on a Delegated Path Discovery (DPD) server to perform path discovery (please answer questions 6 through 9).

   □Rely on a Delegated Path Validation (DPV) server to perform both path discovery and validation (please answer question 6).

   Comments:

6. If the client relies on a DPD or DPV server, what protocol is used to communicate with the server?

   □SCVP If SCVP, which draft version of SCVP was implemented?

   □XKMS If XKMS, which version of XKMS was implemented?

   □Other

   Comments (if the client relies on a DPD or DPV server, please list any servers with which the client is known to be interoperable):

7. Using the naming conventions from section 6 of the *NIST Recommendation for X.509 Path Validation*, what functionality does the path validation module implement (i.e., is it a Bridge-enabled PVM, a Bridge-enabled PVM with Advanced CRLs, etc.)?

   Comments:

8. Has the path validation module been successfully tested using the *Public Key Interoperability Test Suite* (PKITS) as specified in appendix A of the *NIST Recommendation on X.509 Path Validation*? (NOTE: successfully tested means that the module produced the expected results for every test that Appendix A of the *NIST Recommendation on X.509 Path Validation* indicates should be run for the module)

   □Yes

   □No

   Comments (if the path validation module has been tested, but did not produce the expected results for every test, please provide information about the test results):

9. Can the path validation module use OCSP to determine certificate status?

   □Yes

   □No

   Comments:

10. The path validation module can obtain the CRL needed to validate a certificate when (check all that apply):

    □The certificate does not include a cRLDistributionPoints extension, but the relevant CRL is in the authorityRevocationList or certificateRevocationList attribute of the certificate issuer's directory entry (where the CRL is obtained by using LDAP to query a directory whose DNS name or IP address is known from local configuration information)?

    □The certificate includes a cRLDistributionPoints extension with a distribution point name that is of the directoryName name form, and the relevant CRL is in the authorityRevocationList or certificateRevocationList attribute of the directory entry specified by the directoryName (where the CRL is obtained by using LDAP to query a directory whose DNS name or IP address is known from local configuration information)?

    □The certificate includes a cRLDistributionPoints extension with a distribution point name that is an HTTP URI that points to a file containing the CRL.

    □The certificate includes a cRLDistributionPoints extension with a distribution point name that is an LDAP URI where the LDAP URI specifies the LDAP server's name (IP address or DNS name), the directory entry in which the CRL is located, and the attribute (authorityRevocationList or certificateRevocationList) that holds the CRL.

Comments:

11. Does the path validation module construct certification paths by:

□Always starting with the end certificate and building towards the trust anchor (please skip question 13, but answer questions 12 and 14.)

□Building from both the end certificate and the trust anchor depending on the PKI architecture that is encountered (please answer questions 12, 13, and 14)

(NOTE: Always building from the trust anchor towards the end entity is not an option since the information required to build in this direction may not be available. See RFC 2587 for more information.)

Comments:

12. When building certification paths from the end certificate towards the trust anchor, the path validation module can find the preceding certificate in the certification path by (check all that apply):

□Obtaining the CA certificates located in a certs-only CMS message that is pointed to by an HTTP URI in an authorityInfoAccess extension (if the implementation examines the file extension, it must accept and process both .p7c and .p7b files)?

□Obtaining CA certificates located in an LDAP accessible directory that is pointed to by an LDAP URI in an authorityInfoAccess extension that specifies the LDAP server's name (IP address or DNS name), the directory entry in which the certificates are located, and the attributes (cACertificate and/or crossCertificatePair) within which the certificates may be found?

□Obtaining certificates from the cACertificate and crossCertificatePair attributes of the certificate issuer's directory entry by querying a locally configured directory (when a certificate does not include an authorityInfoAccess extension)?

Comments:

13. If the path validation module sometimes builds certification paths from the trust anchor towards the end entity, the path validation module can find the following certificate in the certification path by (check all that apply):

□Obtaining CA certificates located in a certs-only CMS message that is pointed to by an HTTP URI in a subjectInfoAccess extension (if the implementation examines the file extension, it must accept and process both .p7c and .p7b files)?

□Obtaining CA certificates located in an LDAP accessible directory that is pointed to by an LDAP URI in a subjectInfoAccess extension that specifies the

LDAP server's name (IP address or DNS name), the directory entry in which the certificates are located, and the attributes (cACertificate and/or crossCertificatePair) within which the certificates may be found?

□Obtaining certificates from the cACertificate and crossCertificatePair attributes of the certificate subject's directory entry by querying a locally configured directory (when a certificate does not include a subjectInfoAccess extension)?

Comments:

14. In what types of PKI architectures can the path validation module construct certification paths (check one):

□*Hierarchical* – A PKI in which all end certificates that need to be validated are issued by certification authorities that are hierarchically subordinate to the trust anchor, but in which the trust anchor CA may have cross-certified with non-subordinate CAs (i.e., the hierarchy may be part of a larger mesh architecture). The certificates in the certification paths that need to be validated do not include any constraint extensions, but the certification paths may include self-issued certificates and CRLs may have been signed with different keys than the keys used to sign the certificates which are covered by the CRLs.

□*Simple Mesh* – In addition to the hierarchy described above, the PKI consists of a simple mesh architecture in which there is only one path from any CA to any other CA. Certification paths that need to be validated are not limited to the hierarchy and may include certificates with nameConstraints and policyMappings extensions.

□*Complex Mesh* – In addition to the hierarchy and simple mesh described above, the PKI consists of a mesh architecture in which there may be more than one path from one CA to another CA, although constraint extensions may result in one or more of the paths being invalid. Certificates in certification paths may include nameConstraints, policyMappings, policyConstraints, and path length constraints.

Comments:

Additional Comments: