Discussion Topic for ITSAC

Open issues in XCCDF to be covered at the IT Security Automation Conference XCCDF Workshop: 10:30 – 12:15, September 29, 2010, Baltimore Convention Center

This document serves to provide background material for the upcoming XCCDF workshop to be held at the IT Security Automation Conference (ITSAC). Participants are encouraged to consider the issues, questions, and proposals raised in this document ahead of the workshop. The workshop will consider the topics listed below in order, although time considerations may require skipping some topics. Questions, concerns, and additional topics can be raised via the XCCDF mailing list. Minutes of the workshop discussions will be published following the workshop.

Table of Contents

Unique Benchmark IDs

Currently, the id attribute of a Benchmark is of type NCName with no further restrictions on its form. The id field is intended to uniquely identify a Benchmark document. Within XCCDF, the Benchmark id attribute is used within a TestResult entity to link a given result to a given source Benchmark, specifically in cases when the two reside in different files.

Unfortunately, the requirement that id attributes be unique has proven an elusive goal. Content producers have not always been diligent in giving different ids to different versions of a given Benchmark, and different content producers have, on occasion, assigned the same id to their respective content. This not only causes a problem for linking TestResults reliably back to their source Benchmark, but many tools have been written with the assumption that the id field is, in fact, unique and can become confused when this turns out not to be the case.

It has been suggested that conventions be adopted that will better ensure the global uniqueness of Benchmark id attributes. This corresponds to issue XCCDF-58 in NIST's issue tracker.

Issues

1. Should conventions on the id attribute be enforced by the schema, or should they simply be noted in the XCCDF specification? (Note that modifying the schema to enforce constraints that would reduce the chance of id collision will, by definition, deprecate at least some content given that we have examples of id collisions today. Such a change would need to take place in a major release.)
2. What information included within the id field would best ensure id uniqueness? (E.g. author namespaces, document version, product and product version, etc.)
3. If Benchmark ids are to be made to obey structural conventions, what about other id fields? (E.g. Profile ids, Group ids, Rule ids, etc.) Should similar conventions be adopted for these? Would the goal be to create globally unique components (such as exist in OVAL) or should globally unique references be made by first identifying the unique Benchmark id and then further referencing the document-unique (but not necessarily globally unique) id within that Benchmark?

Sample Proposal

One solution would be to adopt a format with designated fields for the author namespace, the document version, and an additional field for a "title". I.e., id="xccdf:usgcb.nist.gov:1.0.0.0: WindowsXP". This would eliminate collisions between different content authors because they would be using different namespaces. The explicit inclusion of a document version field would also encourage correct behavior with regard to providing different Benchmark ids for different versions of a Benchmark. The use of an string title (as opposed to a simple numeric designator such as used by other standards) not only allows for some degree of descriptive information to appear within the id, but would also allow for an automated, mechanical process by which existing XCCDF content could be updated to conform with the new id scheme.

Modifications to the XCCDF schema to require such a format could be included in XCCDF 2.0. In the meantime, the specification could be updated to encourage new content to adopt the described convention.

Local vs. Remote Imports

Currently the import statements in the XCCDF schema assume the six imported schemas are local and in the same directory as the XCCDF schema. This corresponds to issue XCCDF-46 in NIST's issue tracker. Some have suggested that this be changed so that these imports are from canonical, remote locations. There are pros and cons to each approach that have been noted in previous discussions:

Advantages of local references:

- Tools can validate content without network connectivity. According to the XCCDF specification, content validation "should" occur during the first phase of loading an XCCDF document so this is an important capability.
    - It was countered that tools could intercept remote references and decide, due to lack of network connectivity or for other design reasons, to use local copies known to the tool instead. Others argued that allowing individual tools to define their behaviors would nullify most of the advantages of switching to using a remote reference in the first place.
- Users and tools can have direct control over which imported schemas are used to validate content without modifying the XCCDF schema file.

Advantages of remote reference:

- Pointing to a canonical location means that tools can always be directed to the latest version of each imported schema.
- Using remote references prevents "branching" of the standards. For example, if XCCDF 1.2 is distributed bundled with a local copy CPE 2.3, then users will likely continue to use this local copy after CPE 2.4 and beyond are released. This would mean that either users would have to manually update the schema bundles residing with XCCDF or find themselves needing to support CPE 2.3 for use with XCCDF while supporting CPE 2.4 for other applications (such as OVAL). If one considers that multiple SCAP standards have references to supporting schemas, this problem quickly becomes significant as each implicitly has its own "suite" of different versions of supporting schemas. Using remote references allows all standards to automatically update to the most current version of supporting schemas together.
    - It was countered that the XCCDF 1.2 specification explicitly associates XCCDF 1.2 with CPE 2.3 and the community deliberately chose not to support forward compatibility. (That is, the community decided against XCCDF using "CPE 2.3 or later".) As such, the "branching" is already explicitly written into the specification anyway. Others argued that the linking of standards to each other should be controlled above the level of XCCDF (i.e. by SCAP) and that using remote, canonical references would make such control easy.
- It was noted that local files can become lost or corrupted (in addition to becoming obsolete). If only local files are available, then tools cannot recover without deliberate human assistance. However, having canonical remote references provides a location that is always to a present and uncorrupted source.

Prior discussions on this topic resulted in deadlock within the community. It is clear that either solution can be made to work in a technical sense. It is also clear that either solution leads to its own set of challenges for tool users and developers. As such, "consensus" on this topic will need

to involve a solution that everyone considers workable, if not necessarily ideal. Towards this end, it may be useful to consider the following questions:

1) Who has the greatest need to dictate the specific copy of referenced schemas used when using XCCDF content:
    a. The XCCDF specification and schema?
    b. Content producers?
    c. Tool users?
    d. Tools?
    e. The authorities responsible for the referenced schemas?
2) Who should be responsible for ensuring that, by default, the correct referenced schemas are used?
    a. The XCCDF specification and schema?
    b. Content producers?
    c. Tool users?
    d. Tools?
    e. The authorities responsible for the referenced schemas?
3) Of the advantages listed above, which one provides the greatest benefit to users? To tool developers? To content producers?

Given the challenges posed by this topic in the past, it would be very useful if participants could provide hard examples, either at the discussion or ahead of time via the mailing list, of scenarios where one solution or the other is particularly advantageous.

Allow XCCDF check-content-ref statements to refer to other XCCDF documents

Sometimes a recommendation may be to follow the policy listed in another document. Currently, this can only be handled by checking instructions that explicitly ask the user if this has been accomplished. However, if the second policy was also expressed in XCCDF, it is not unreasonable to expect that a checking tool could actually run an assessment based on that document's policy. Proposals were made to allow checks in XCCDF Rules to reference content (in part or whole) in other XCCDF documents. This corresponds to issue XCCDF-53 in NIST's issue tracker.

This topic was discussed in February of 2010. At the time there was some interest in the capability, but community members felt that this modification was not of enough immediate use to include in XCCDF 1.2. However, the consensus was to review that decision at a subsequent developer meeting following the release of XCCDF 1.2.

Issues

1. Referencing an entire XCCDF document is more complicated than referencing other existing checking system documents because of XCCDF tailoring. Check references would need to pass additional information in order to identify a Profile and possibly to perform other tailoring activities. This not only requires the existence of additional fields, but standardization of a format to express this information.
2. Referencing a single, external XCCDF Rule is also complicated: the href attribute identifies the XCCDF document, the name attribute identifies the individual Rule, and <check-export> elements handle population of relevant Values in the external Rule. However, this usage not only adds requirements to XCCDF interpreters (they must now be able to handle invocations identifying a single Rule rather than a document and Profile) but it also changes the requirements on document processing and context:
    a. The entire remote document will need to go through some of the Loading steps so that extended Items can be resolved. Does the entire Loading process need to be executed, however? What about other processing steps?
    b. Should the requires, conflicts, and selected properties be considered when a specific Rule is called out?
    c. The raw value of the target Value could be controlled by a simple <check-export> element, but what about other Rule and Value properties that are controlled by selectors, such as selection of individual Rule check and Value operator fields?
3. In either of the above scenarios, it is not unreasonable to consider cases where the same target XCCDF document is referenced using different tailoring options. This means interpreters would either need to process a single XCCDF document multiple times (Load and resolve extensions and possibly other steps) or would need to track the variations passed along with different requests. This significantly increases overhead and/or complexity.
4. Having XCCDF documents reference XCCDF documents adds another level of complexity to document processing. Specifically, XCCDF interpreters must now be written so that they could be invokable by other XCCDF interpreters rather than existing as top-level structures.

5. If additional information is to be passed to a check to support control manipulations, should the control instructions be manipulable by through XCCDF tailoring actions? For example, selecting an Enterprise Profile on a source XCCDF document, one might wish to invoke a target XCCDF document using an equivalent level of Profile.

Sample Proposal

The check element would be given a new optional element called "check-control" consisting of one or more "any" elements. This element would hold a XML structure that would be passed directly to the interpreter. Each language that could be called from XCCDF would be responsible for defining a schema for the transmission of additional information. Strict processing would ensure that the checking language interpreter would be able to understand the instructions. If the referenced language had no use for additional information, then there would be no need for it to define such a schema. This field could be used to convey such information as interpreter command-line instructions, details on what metadata the checking engine should return, and pre-processing instructions.

There is precedent for such a construct - OVALDI reference implementation already defines an evaluation-id schema that allows it to be passed an XML file that enumerates the list of definitions it should evaluate within a given file. This allows the OVALDI to be given an OVAL file but only evaluate a subset of the contained definitions. Currently, XCCDF references to OVAL are singletons (a definition name is provided) or the entire file (no definition name is provided). If the evaluation-id schema were codified as part of OVAL (rather than a part of the OVALDI as it is now) XCCDF could include a reference to this file in its check-control in an OVAL check to cause the OVAL interpreter to process a subset of the definitions in the indicated file.

```
<xsd:complexType name="checkType">
    <xsd:annotation>
      ...
    </xsd:annotation>
    <xsd:sequence>
      ...
      <xsd:element name="check-export" type="cdf:checkExportType"
              minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="check-control" minOccurs="0" maxOccurs="1">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:any minOccurs="1" maxOccurs="unbounded"
                  processContents="strict"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="check-content-ref"
              minOccurs="0" maxOccurs="unbounded"
              type="cdf:checkContentRefType"/>
      ...
```

```
    </xsd:sequence>
      ...
  </xsd:complexType>
```

In the case of XCCDF, a Profile element could be employed. This document will refer to this as the "control Profile". If the id attribute of the control Profile matched one of the Profiles in the target XCCDF document, then that target Profile would be employed to tailor the target source and all other content in the control Profile would be ignored. If the id of the control Profile did not match any Profile ID in the target XCCDF document, then the control Profile itself would be treated as explicit tailoring instructions of the target, explicitly providing selectors to identify content and possibly extending an existing target Profile. The current XCCDF schema allows documents that just consist of a Profile element, so it would be possible to create control Profiles as their own files using the current XCCDF schema and use a check-control-ref structure to reference them.

It should be noted that the above is effective an "external Profile" as discussed in meetings in early 2010. At the time, external Profiles were felt to be unnecessary, but multiple parties have requested that this decision be reviewed. If the above proposal is felt to be worthwhile, it is not a large step to allow a control Profile to be used directly on a target XCCDF document without the need of a source XCCDF document. (And vice-versa.) The community should consider whether external Profiles of the described format would be a useful addition to XCCDF.

XCCDF 2.0 Preliminary Discussion

It is anticipated that in the next year or two there will be a new major release of XCCDF. A major release is one that is not required to preserve backward compatibility with existing content. This provides an opportunity to implement significant improvements to XCCDF's structures and capabilities that would not be possible in a minor release, which must preserve backward compatibility.

This discussion will look into some initial ideas that might be considered in a major release. It is hoped that community members will continue to think about ways in which their use of XCCDF could be made easier and more effective and share these ideas via the mailing list (xccdf-dev@nist.gov) after this workshop. The nature of XCCDF 2.0, including determining whether there is any real need to create a major revision of the standard, will be guided by the feedback received from the community.

Sample Topics of Consideration

Below are some topics that could be addressed in a major release.

Use Cases

The XCCDF specification identifies 7 (partially overlapping) use cases. These are:

- Express sophisticated guidance from security experts
- Support tailoring by auditors and system administrators
- Support generation of human-readable guidance
- Support generation of HTML guidance
- Support conversion to other XML formats
- Enable tools to perform automated assessments of systems and report on findings
- Support remediation of non-compliant configuration
- Support expression and automated testing for vulnerability alerts

In some of these use cases, XCCDF has enjoyed strong adoption (e.g. expressing guidance from security experts and supporting automated assessments, at least for individual systems). In others, it has not seen significant use (e.g. support of remediation and use in vulnerability alerts). A major release is a good time to review the use cases associated with XCCDF and better align them with how people currently use XCCDF and how people wish to use XCCDF in the future.

Some possible new or extended use cases to consider might include:

- Assessment orchestration – Currently XCCDF's structures work best when a benchmark is targeted towards a single endpoint. Some users have been asking for ways to use XCCDF to support guidance that would cover multiple endpoints in different roles. For example, users have proposed guidance that includes both machine and user assessment, or guidance that governs the configuration of a network domain (e.g. with specific roles for entities such as domain controllers, externally facing servers, user workstations, etc.)
- Result orchestration – Users have noted that XCCDF results tend to be a relatively large data dump of all findings. Suggestions have been made for the inclusion of logic that would support simple reasoning over results (possibly akin to OVAL's <directive> elements) to allow Benchmarks to perform complete assessments, but only return targeted, relevant details for a particular scenario.

XCCDF Structure Revisions

A major change is an excellent opportunity to clean up the structures of the XCCDF XML. This can go beyond the simple removal of deprecated items and include structural enhancements that make it easier to maintain and create content within the standard. Some possible areas where this could be accomplished include:

- Values and Complex-Values – The new ability to include complex-values (lists and external data structures) in XCCDF 1.2 was forced to create parallel structures to maintain backward compatibility. It would be significantly cleaner and less confusing if these structures could be unified.
- Platform identifiers – The evolution of XCCDF has seen the use of no fewer than 4 distinct mechanisms to identify the platform targeted by a Benchmark. Each of these required the schema to include a new piece of XML content so the new structures could be included while preserving backward compatibility. While a major release of XCCDF would allow us to remove the obsolete platform identification structures, it might be worth considering a platform reference structure that is generalized, similar to a Rule's check structure, which would allow new platform structures to be adopted without the need to continually add on new fields. The generalization of other areas of the schema might also be possible.

Specification Organization

One of the common complaints about XCCDF is that it tries to do too much. At the same time, no capabilities of XCCDF have been identified as generally unnecessary. One possibly way to resolve these issues would be to split the XCCDF specification into sub-components, each focused on a specific use. (E.g. an automation control piece, a guidance encapsulation piece, a remediation control piece, a result processing piece, a tailoring piece that could underlie all the preceding components, etc.). This is similar to the approach adopted by CPE in the recent CPE 2.3. Such a reorganization could not only (in theory) simplify the XCCDF specification by breaking it into modular components, but it could allow implementers to focus on specific capabilities and be clearer in their descriptions of what XCCDF capabilities they support. It is not clear how such a split could be made and where best to draw the lines to do so, but the community should consider if such a modular structure for the specification might be beneficial. In some cases, subdividing by use cases might be made more effective if the schema could be restructured slightly as well – hence this topic is being raised in the context of a major revision to XCCDF, even though a simple reorganization of the specification alone would not require a major release.

An alternative and less drastic possibility would be to split the XCCDF specification along lines of usage. For example, create one document that focuses on the structure and content of an XCCDF document and another document that focuses on how XCCDF documents should be processed. Currently these topics are interleaved in the XCCDF specification resulting in a fairly long document. More than one experienced community member has claimed that aspects of the specification are underspecified when, in fact, they are set out, albeit in a single sentence of a single table, buried within the document. Breaking the specification along these lines could help remove some of the confusion brought about by trying to cover everything in a single place. This change could likely be done without resorting to a major revision of XCCDF.

Other Topics

The preceding examples are by no means exhaustive. Community members should consider their own use of XCCDF (both current and idealized) and provide feedback at the workshop or via the public mailing list.