Security Automation Developer Days

June 8-12, 2009

Table of Contents

Introduction

Security Automation Developer Days was held on June 8 - 12, 2009 at The MITRE Corporation in Bedford, MA. This event was unprecedented in the breadth of coverage across a wide variety of existing and emerging standards and the large number of attendees.

Eighty-nine people registered for the event, and 60-65 individuals were present each day of the conference. Over the five days, fourteen sessions were held and this document contains a comprehensive summary of each of those sessions.

As you prepare to review these minutes, the authors would once again remind you that the standards cannot continue to advance without ongoing discussion of the issues throughout the year. This is accomplished through dialogue in the email discussion lists. A complete list of these email discussion lists can be found here: http://measurablesecurity.mitre.org/participation/index.html . Please sign up for those lists that interest you.

What follows is a detailed summary of the discussions from the event.

Attendee List

Belarc	-	Richard Defuria
		Gary Newman
		Peter Tracy
BigFix Inc	-	Ben Hobbs
Booz Allen Hamilton	-	Michele Ellison
		Conrad Fernandes
		Jason Smith
		Michael Sor
		Colin Troha
CA	-	Michael Petersen
Calwater	-	Ramesh Dhullipaua
Center for Internet Security	-	Blake Frantz
		Steven Piliero
Concurrent Technologies Corporation (CTC)	-	Kirk Johnson
DISA FSO	-	Jim Govekar
		David Hoon
		Paul Inverso
		Terry Sherald
		Ricki Vanetesse
DOE	-	Peter Wettergreen
DSCI	-	Vladimir Giszpenc
		Leon Sung
EWA-Canada	-	Dawn Adams
Fortinet	-	Raymond Chan
G2, Inc	-	Melanie Cook
		Matthew Kerr
		George Saylor
		Shane Shaffer
		Greg Witte
Gideon Technologies	-	Kyle Key
		Dragos Prisaca
Guidsoft	-	Bill Ren
Hewlett-Packard	-	Pai Peng
		Peter Schmidt
HQDA	-	Lilliam Cruz

Juniper Networks	-	Stephen Hanna
Lockheed Martin	-	Jim Hartig
		Wesley Snell Jr
McAfee, Inc.	-	Kent Landfield
MIT Lincoln Laboratory	-	Stephen Boyer
MITRE	-	Jon Baker
		Steve Boczenowski
		Steve Boyle
		Andrew Buttner
		Maria Casipe
		Michael Chisholm
		Daniel Haynes
		Robert Martin
		Brendan Miles
		Linda Morrison
		Lisa Nordman
		Charles Schmidt
		Larry Shields
		Glenda Turner
		Matthew Wojcik
		Bryan Worrell
		John Wunder
		Margie Zuk
Modulo Security Solutions	-	Marlon Gaspar
nCircle	-	Timothy Keanini
	-	Natalia Smishko
Netiq/Attachmate	-	William Graves
NIST	-	John Banghart
		Paul Cichonski
		Timothy Harrison
		Christopher Johnson
		David Waltermire
NSA	-	Mike Buckley
		Jason Hage
		Joseph Wolfkiel
OpenPages Inc	-	Gary Zakon
Prism Networks Pvt. Ltd.	-	Maneesh Jolly
ProSync Technology Group, LLC	-	Joe Wulf
Qualys, Inc.	-	Parminder Singh

Secure Acuiity, LLC	-	Andrew Bove
Sparta	-	Jim Ronayne
SPAWAR | SAIC | EMA	-	Jeff Cockerill
Symantec	-	Jim Boyles
		Chris Coffin
		Jason Meinhart
Telos	-	Sudhir Gandhe
		Peter Smith
ThreatGuard, Inc.	-	Robert Hollis
Tripwire, Inc	-	Rob Etzel
		Robert Huffman
		Adam Montville
		Claudia McNellis
Unified Compliance Framework	-	Dorian Cougias
Other	-	Karen Dixon-Brugh

Monday June 8th

SCAP Lifecycle and Document Overview

Briefer John Banghart started with a slide that described NIST's view of the current state and longer term vision of Security Automation within the federal government. SCAP is one piece of the broader picture that they are bringing forward, working with other government agencies as well as commercial interests. The slide depicts four columns, each representing an ontology: SCAP Ontology, Software Assurance Ontology, Network Event Content Automation Protocol (NECAP) Ontology, and the Framework Ontology. The Software Assurance Ontology and the Framework Ontology are largely conceptual at this point. The slide reflects three categories for security standards within these ontologies: 1) current SCAP standards, 2) emerging standards, and 3) conceptual standards.

The four columns feed up into ontologies, i.e. the ontologies build off of the information in the columns in order to encapsulate them in additional data. Ultimately, the information that gets generated within the four columns gets rolled up into three areas: Situational Awareness, Remediation, and Compliance Reporting. Specifically, NECAP rolls up into Situational Awareness and Remediation, while the Framework Ontology rolls up into Compliance Reporting. SCAP rolls up into all three: Situational Awareness, Remediation, and Compliance Reporting; but the Software Assurance does not roll up into any of them. The slide also reflects 1st Order and 2nd Order views of the data. E.g. the 2nd Order view would be appropriate for a high level organizational view, whereas the 1st Order view would be appropriate for a CIO or the White House. The basic point of the slide is that there are multiple levels of abstraction for rolling up these technologies.

A point was made that relationships that exist between standards in different columns should be depicted on the slide, e.g. between CWE and CVE, between CCI and CCE, etc. A question was asked about whether organizational structures existed so that results could be filtered back to government agencies, and it was agreed that this was an issue that was yet to be determined. Another question was asked why the Software Assurance column was not depicted to roll up into the Compliance Reporting area. Another attendee questioned whether there was a problem in that output from different tools do not aggregate data together, and the point was made that that is exactly the point of these standards, i.e. allowing tools to share data.

Discussion ensued about various interpretations of the slide with the general points being that the slide is probably overloaded and puts too much information in one picture; the vision should be separated from the status; and there should be more detail provided about the relationships between the columns and it was agreed that more detail and more clarity needs to be added to the slide.

John then moved onto the SCAP Lifecycle slide, which describes the NIST SCAP Lifecycle for a particular version. The individual community standards have their own lifecycle, but this NIST Lifecycle process is independent from them. Phase 1 in the slide refers to the ongoing, independent review process of the individual community standards and specifications, such as CVE and OVAL. Each of these standards

operate independently of SCAP and may have release and review cycles that do not match with the SCAP lifecycle phases. The first phase which maps specifically to SCAP is Phase 2. Phase 2 occurs in June and involves NIST considering whether to add new specifications into SCAP or the Validation Program. These evolving specifications must meet NIST's set of criteria to be considered. In Phase 3, which occurs in September, an updated draft of NIST SP 800-126 and DTRs gets released (see below). The updates described in NIST SP 800-126 include all of the new specifications that NIST has considered for adding to SCAP.

The chart below depicts the SCAP Lifecycle phases.

Phase	Description	Prod Dev	Date
Phase 1	Individual Standards Review		Continuous
Phase 2	Review Candidate SCAP Specifications		September, Year 0
Phase 3	Deadline for Publication of Draft SCAP SP 800-126 and DTRs (IR 7511)	X	January Year 1
Phase 4	SCAP Beta Content Available	X	April, Year 1
Phase 5	Deadline for Publication of Final SCAP SP 800-126 and DTRs (IR 7511)	X	January Year 2
Phase 6	SCAP Content Final	X	March, Year 2
Phase 7	Laboratory Tool Validation Period Begins (DTR Effective Date)	X	April, Year 2
Phase 8	Laboratory Tool Validation Period Ends (DTR Expiration Date)		April, Year 3
Phase 9	Tool Validations Expire and Mandatory Content Maintenance Period Ends		April, Year 4

In the middle of the slide is a yellow bar, which represents the 15 month SCAP Product Development Period (denoted in the above chart by the 3rd column), which is the time period that vendors will have to develop their products to meet the requirements of this SCAP revision. This period begins at Phase 3 when the draft NIST SP 800-126 is released and encompasses Phase 4 (3 months later, i.e. December, when SCAP Beta Content is made available), Phase 5 (another 9 months later, i.e. September, when NIST SP 800-126 is made final), Phase 6 (another 2 months later, i.e. November, when SCAP Content is made final), and Phase 7 (one month later, i.e. December, when Tool Validation Testing begins). Phase 8 covers the next 12 months when Laboratory Tool Validation Testing will be available for this version. Essentially, testing for each version will extend for one calendar year; when the new year begins, testing for the next version will begin. Phase 9 extends for another 12 months and represents the deadline for when validations for that version will expire and the point at which NIST will no longer support content associated with that version.

Questions were posed about the NIST Validation program and it was made clear that the program is directed toward SCAP products, but not the content itself – content will be validated with a separate program. Also, some products not yet a part of SCAP may be validated by the Validation Program. It

was also made clear that, although individual community standards may create new versions, the version of these standards is fixed within SCAP as it is defined by NIST SP 800-126.

John then started a discussion on document and data relationships. For example, while SCAP will select a specific version of OVAL to be included in its program and this will be documented in NIST SP 800-126, OVAL has its own program to determine compatibility with the standard. But the product validation requirements for SCAP will be defined in IR 7511, which is pointed to by NIST SP 800-126. It is important for vendors to understand that the requirements for all of the specifications within SCAP are described in NIST SP 800-126, so SCAP content should be written against these requirements. Content authors should also keep in mind the various data sources, or enumerations, when writing content. These data sources include such things as the OVAL Repository, NVD/CVE, the CCE List, and the CPE Dictionary. Finally, content authors should reference existing policy, e.g. a vendor hardening guide, when writing SCAP content.

John also mentioned that NIST SP 800-117 is a high-level document which describes how SCAP can and should be used. An important consideration is also the National Checklist Program as described in NIST SP 800-70 Rev 1.

John then described the criteria that NIST will use to determine whether to consider an emerging specification for inclusion into SCAP:

- Be in final specification format.
- Address use cases that are consistent with the SCAP vision and provide value and utility to the SCAP community.
- Demonstrate a high degree of maturity.
- Demonstrate uptake and adoption by product vendors, agencies, and organizations.
- Demonstrate interoperability with existing SCAP component specifications.
- Have a specification submission that is accompanied by a functioning reference implementation.
- Have a specification submission that includes evidence of public vetting (mailing lists, workshops, etc.)
- Be operationally used in organization(s) as a pilot or full operations.
- Be free from proprietary claims of intellectual property.

SP 800-126 Review

NIST SP 800-126 is the technical specification for the SCAP protocol, which describes what the content looks like and the validation program for both products and content and will ultimately provide a feedback cycle for what is include in NIST SP 800-117.

The purpose of NIST SP 800-126 is to describe the current version of SCAP 1.0 and to document the history of what SCAP is today. The document also describes the interrelationships of the SCAP component specifications. The document doesn't repeat information from the component

specifications, but rather adds value by describing things that may be loosely defined or unknown when trying to integrate multiple specifications together. The document describes what content should look like, so that organizations that are authoring content will have a clear picture of what they need to produce and so that organizations that are developing tools which consume that content will be able to develop their tools to produce the desired results.

The document provides an Overview, SCAP Component Basics (which points to information on individual specifications and data feeds), General Requirements and Conventions (i.e. interrelationships), and Use-Case Requirements.

Section 3 of NIST SP 800-126 describes the basics of the SCAP components and separates these components into three categories: languages (XCCDF and OVAL), enumerations (CPE, CCE, and CVE), and metrics (CVSS). Each of the components is described with a brief overview, links are provided to related resources, and some examples of content are provided. The audience seemed generally satisfied with the level of detail provided in section 3. One attendee, however, thought the examples were hard to follow and thought visuals would be helpful.

An enumeration is a standardized nomenclature for defining identifiers and a collection of semantic definitions. The semantic definitions are often represented in an XML data feed, but this can cross the line between what's an expression language and what's an identifier. An opinion was offered that an enumeration should be merely a list of identifiers, the format of that identifier, and some references to that element that allow it to be differentiated from other entries in the list..

What litmus test should be used to determine which reference is acceptable and which is not? Can this litmus test be standardized? The kind of references should be to documents that discuss the core area that the enumeration covers at approximately the same level of abstraction as in the enumeration list. Enumerations should not reference other enumerations, only expression languages should reference enumerations.

Does NIST SP 800-126 really concern itself with the relationship between the components? Doesn't NIST SP 800-126 refer to standards that already have been developed? NIST has found that it easier to break things down and show the rules that exist between the specifications if they are as granular as possible. By following these "best practices" it makes the specification easier to define. So will NIST be separately defining the data feeds that would be commonly used by SCAP? Yes.

Should NIST merely specify the definitions that they're moving toward and encourage other to follow along? Yes, that is what NIST is trying to do with the SCAP site. Much of the content in the emerging specifications section defines the types of specifications or reference implementations that are needed for each new component.

Section 4 of NIST SP 800-126 discusses broad requirements and conventions for SCAP with respect to use cases. A convention is best practice guidance that applies to content and products without regard to use case. A requirement is a rule that must be followed for all use cases.

Should there be requirements or conventions for the use of groups within SCAP content? FDCC makes use of two types of groups: control groups used to define references to NIST SP 800-53, etc. and real groups which categorize various types of configuration guidance. Therefore, within FDCC documents there are two types of abstractions for groups, allowing multiple levels of nesting to categorize the settings groups. Other SCAP content that has been generated that take different approaches: some don't contain control groups and are flat, and other don't control groups and have a hierarchy. Should a rigid structure be adopted for specifying groups? One opinion was that following the XCCDF specification, which is a loose interpretation of using groups, is sufficient. Another attendee cautioned over the use of groups because information can be included in a group that has nothing to do with the rule that is referring to that group. Using groups affects the way that scores are rolled up in an XCCDF document. Another use case for XCCDF is to produce documents, and using groups is a way to separate the document into chapters, which is very beneficial. Restricting authors from the use of groups would be an impediment for this use case. Having style guides for various types of XCCDF documents, to act as guidance but not requirements, could be very helpful. NIST clarified that they weren't asking whether requirements should be specified for groups; rather whether guidance should be provided or conventions recommended. An attendee pointed out that since the use of groups can affect pass or fail, therefore uninformed use of groups can lead to false positives or false negatives and this could cause dire consequences. There was some disagreement whether CCI could help this problem. The CCI proponents suggested that by providing a data stream at the rule level, the confusion could be avoided. However, the counter-point was made that in previous experience a large amounts of exceptions were caused making the results difficult to interpret. A suggestion was made that combining groups with profiles could alleviate these problems. Groups are too powerful to just throw away.

Next the discussion turned to the use of rule identifiers. Should there be SCAP requirements or conventions for rule identifier naming? This topic had been addressed in an earlier discussion on the XCCDF forum and some points made included that strict rule naming requirements could cause a problem in the following ways: internationalization, tailoring (if the title ends up being overly specific, tailoring could change that), and editing (you could end up with a rule doing what you want, but with a title that was misleading). Therefore, the broad consensus was that a simpler, less descriptive title for rules was the way to go. A vendor suggested that because of scope, i.e. that rules may need to be localized, the ID must be strictly a unique ID. Then attributes can be added and given a title, and then that title can be stated in multiple languages. For these reasons, identifiers should be unique and addressable, but no more requirements should be placed on them. Another attendee added that identifiers must be persistent – all identifiers must be unique and persistent. Another added that perhaps, in order to ensure uniqueness, there needs to be an identifier management framework. It was then pointed out that without meaningful names, content management becomes very difficult with the tools that are currently available. It was then pointed out that the Benchmark Editor and the Recommendation Tracker, both tools developed by MITRE, allow you to have abstract IDs and still view content by descriptive information. A question was posed that if there was a requirement to use a namespace with some sort of serial identifier, how difficult would it be for organizations that have already generated content. It would be very difficult. A compromise solution was proposed that would assign a default namespace for content generated up to the present, and then have a special

discriminator character that would flag a new way of using namespaces in the future. A vendor responded that they had created their own namespace already and wondered whether everyone who generated content had done the same. Another attendee suggested that organizational namespaces must be instituted to promote sharing of content. It was pointed out that OVAL had adopted a standardized global identifier. Another point was made that this was because it was always a requirement of OVAL to have a global repository, but that requirement was never assumed to be needed for XCCDF. It was then explained that for CVE a high coordination cost is paid for a unified identifier. But OVAL content is managed more loosely, a contribution model that is lightly moderated. This allows a namespace that can define the organization. This scenario is similar to that which is required currently by XCCDF. It was pointed out that XCCDF has a separate field that is intended to be a globally unique ID field. This field, the <Ident> field, is optional and rarely used. A namespace could be created for this field that could be used as a globally unique XCCDF identifier and this solution would be backwards compatible. The concept of trust for identifiers was mentioned. The discussion was summed up by suggesting that strong consideration should be given to using the OVAL ID naming strategy, but that some consideration should be given to using the <Ident> element as a means of marking up content. A suggestion was made that any convention adopted should be added to the XCCDF specification, rather than just developing a convention for SCAP. A further point was made that an effort should be made to formalize conventions that exist between the standards; i.e. there needs to be coordination between the six SCAP standards.

The discussion then moved to the use of CPE names within XCCDF documents. Should CPE names be allowed on rules, groups, and profiles? Although this practice is completely legal according to XCCDF, this practice is not allowed in the current version of 800-126 so that products that had been validated against SCAP 1.0 would not be invalidated. Is this geared just to SCAP 1.0? This restriction will likely be required for SCAP 1.0 and then relaxed at some point in the future. A suggestion was made that if the reason for using this restriction was because of validation, then it should be documented in the validation documentation, but keep the XCCDF specification as is. The response was that this could cause a problem for the end user who might write content using this feature, when the validated product didn't support it. A suggestion was made that, since we're suggesting that to redefine XCCDF, perhaps it would be more appropriate to poll the vendors to see what they implemented and then adjust the specification so that it matches existing implementations. DISA stated that they do have plans to generate content that uses complex, multi-CPE types of documents, specifically for some enterprise tools that they deploy with specific benchmarks or baselines that they test to and they deploy to the field and they must test against those periodically. They will be against multiple components and possibly different settings compared to standardized guidance. McAfee has a concern about usability: it doesn't make sense to have users know the target benchmark. In these cases, it makes it a lot easier to attach rules to the CPE. The decision was made to keep it as is for now, and work quickly to relax the restriction in the future. In response to a question, it was pointed out that this was an SCAP restriction, not an XCCDF changes and that this was needed to be consistent with what's validated today. This solution is needed because of an imperfect situation that currently exists. However, this does not preclude a customer from requesting this functionality in a vendor's product. A vendor suggested this is a scary path to go down, i.e. to disallow something in SCAP that is allowed in the specification. A

suggestion was made to differentiate between FDCC scanners and authenticated configuration scanners, which would allow scanners to support the more complex rules. Eventually that will be the case, but not right away. The comment was made that if vendors are doing something different with their content than what is documented in 800-126, they should send those comments to NIST as part of their review of 800-126. When the point was made that some products might be able to do more than what is required by 800-126, it was pointed out that content validation will be more restrictive than product validation.

The next conversion centered on whether the <identity> element be required in <TestResult) elements. There is some ambiguity around this issue because it is not addressed in 800-126 – this issue was mentioned in a comment received in feedback on the current draft of 800-126. The <identity> element was designed to identify the security principles that are used for the evaluation of the checklists. This could be made optional, which is the way it is defined in XCCDF, or it can be made required. A clarification was asked whether this issue referenced content or a tool, and the response was that it refers to the validated output content from a tool. From the perspective of not including it, one wouldn't have the artifact of knowing what security principle was used for evaluating a specific set of values, and that could potentially invalidate the results that one gets. A question was asked why it was optional in the current XCCDF specification, but no one seemed to know. A suggestion was made to raise this issue to the XCCDF forum to consider changing the standard. It was suggested that this must be reported for FDCC reporting. This is required for use cases of FDCC reporting, but is not required for authenticated configuration scanners who don't have the additional FDCC requirements. The decision was made to bring this issue to the XCCDF community and Charles Schmidt took the responsibility for this action item.

The next item for discussion was what content should be required for the <target> element. The <target> identifies the host that the assessment is performed on. But there are other attributes that are associated with the <target>: the <target address> which there can be zero or more of and refer to the IP addresses; there is also a long list of <target_facts>. Of this set of information, what should be required? There are a set of requirements for ARF, which includes IP address, mac address, default gateway, IP address of the DNS server, but not sure if all of this information needs to be listed a the XCCDF level. LtCol Joe Wolfkiel took the action to provide the complete list of ARF requirements to NIST. It was mentioned that the definition of the <target> definition should be clear, direct, and unambiguous, so perhaps the question should be what information is needed to make the definition of the <target> unambiguous. But ambiguity is measured on a continuum; so generally, more information makes something less ambiguous. Another comment was that while this information is useful, it is questionable whether it belongs in XCCDF, and more, sufficient attributes to describe an element vary depending on what kind of device it is. The consensus was that this is a non-requirement or a very small set of items should be required. There was a suggestion to add a feature to obfuscate the identity of an object.

As the XCCDF discussion neared conclusion, the question was asked whether there were any other spots in 800-126 specifically prohibits existing XCCDF functionality that is currently in the specification, but none were mentioned. A question was asked, with regards to the uniqueness of rule names, is there

any discussion of versioning of rule names? So as changes are made to benchmarks, is there some way of identifying changes? This is an optional feature. The suggestion was made to look at lessons learned from OVAL and Charles Schmidt took this action item.

Next the discussion turned to OVAL and the first item for discussion was should compliance and inventory definitions be segregated to provide granularity in result reporting. The point of this is to use extended definitions to reference the inventory definitions from compliance definitions so that in results we could turn off whether we wanted result reporting for extended definitions or not. Essentially, this would allow the ability to include inventory checks in the results or not. This would provide the capability to generate different levels of results for different items. There is no standardized way to pass a control path between XCCDF and OVAL. Then the proposal is to establish the convention to have inventory definitions to be referenced via extended definitions. The question was asked: what is the alternative? Technically, you could use a test instead of an inventory definition. But this could be enforced only by a human review, therefore, this should at most be a style suggestion and not a requirement. It was pointed out that 800-126 contains a section on style guidance. Perhaps there should guidance to extend whenever possible. But that is not exactly what is trying to be accomplished here – we want to limit this to inventory definitions. But a strong caution was made to limit this to only inventory definitions, due to experiences from OVAL.

The next subject discussed was whether a vulnerability definition be allowed to extend a compliance definition. Why limit it was asked. Why allow it was responded. The point was made that you shouldn't write out functionality just because you can't anticipate a potential use case.. But we are not trying to satisfy all potential use cases, either. Admittedly, there is a difference between SCAP Validation use of OVAL and OVAL validation, but we should be careful not to cut out functionality in case an important use case emerges in the future. So only cut something out now if it's hurting us. The consensus was that no restriction should be adopted in this case.

An attendee asked how long comments on 800-126 will be accepted. The response was that this comment period was allowed for the Security Automation Developer Days event. Comments from this event will be collected along with some internal comments that have been made, and a new draft will be produced. The document will then be released and a formal public comment period will be announced.

A question was asked relating to inventory definitions, or checking for the presence of a CPE, whether that's a good thing or a bad thing. At the XCCDF level, the definition of a return of an OVAL inventory definition and the result of an XCCDF rule, was there any determination? No specific comments have been received by NIST. The question was clarified: If you have an OVAL definition, whose primary reference is a CPE, and it's an inventory class definition, and that returns the value true it means that piece of software is installed on the machine. If you have an XCCDF rule that wants to reference that OVAL definition, there are multiple ways to interpret that: 1) this a good application and I want to know if you have it, or 2) this is a bad application and I want to know if you don't have it, or 3) I'm curious if you have it. Pass or fail in an XCCDF rule does not always correspond the same way to true or false in an OVAL inventory definition. Let's keep 800-126 as it is now, but in the future we should not have a hard

mapping from definition result to rule result, there are situations where you want differing mappings depending on the situation. Let the rule result determine what the check result should be.

Another question referenced the fact that throughout 800-126, it referenced both OVAL 5.3 and Oval 5.4. NIST wanted to support content from both 5.3 and 5.4. But since OVAL is backwards compatible, if you supported OVAL 5.4 then you'd support both versions. From a validation perspective, NIST can only claim what it tests; the 5.3 / 5.4 issue only affects FDCC scanners; and that there are 5.3 products validated and 5.4 products validated. Until this situation is remedied, 800-126 will need to refer to both OVAL versions. Some of the wording needs to be clarified.

The discussion moved to Section 5: SCAP Use-Case Requirements. This is about assigning requirements that are specific to individual use cases and three are three major use cases within 800-126. The first use case is configuration verification, which is using XCCDF and OVAL for configuration assessment. This is a well-explored use case with all the work having been done in FDCC, etc. The 2nd use case is vulnerability assessment, of which there are two different flavors: one which combines XCCDF and OVAL, where policy is defined in XCCDF and tests and check provided in OVAL; and the second is an OVAL definition which returns a result and would allow you to get real-time situational awareness. The 3rd use case is for inventory collection. An attendee pointed out that currently there are no SCAP validated scanners which can handle stand-alone OVAL. Also, it is not necessary to restrict the stand-alone OVAL use case to just vulnerability assessment. A question was asked where patch assessment fits. That would be part of configuration verification. With regard to inventory assessment, it was pointed out that OVAL can support "do you have this piece of software?" but not "which piece of software do you have?"

A question was asked of LtCol Wolfkiel, for use cases for configuration verification that use stand-alone OVAL, do you see that as an OVAL-only validation? LtCol Wolfkiel replied that he lumps them all together – he wants software that does it all, XCCDF combined with OVAL as well as stand-alone OVAL. A question for the entire group was should anything stand-alone OVAL show up as an SCAP use case? If it's stand-alone OVAL it's not SCAP, it's OVAL. Currently, SCAP validated scanners handle both XCCDF and OVAL, but not just OVAL. It depends whether you want to have validated stand-alone OVAL as part of SCAP, which could contain references to CPE and CCE. One attendee suggested that a FDCC scanner might not need to support stand-alone OVAL, but an authenticated configuration scanner should be able to support stand-alone OVAL. Another attendee felt that the stand-alone OVAL use case should not be part of SCAP because SCAP is an integration effort. But there certainly is value to the stand-alone OVAL validation process because there are elements of the language that vendors may want to take advantage of. There was the concern expressed of the confusion that could be caused by an SCAP stand-alone OVAL validation program when a separate OVAL validation program will exist outside of SCAP. A suggestion was made that there is a need for the ability to cross reference stand-alone OVAL with OVAL content that is wrapped in XCCDF. There was a question whether there was a use case for stand-alone XCCDF with manual checks and non-technical checks. If stand-alone OVAL is allowed by 800-126, then shouldn't stand-alone XCCDF be allowed too? Someone pointed out that the conversation may be getting off track of their intent, which is to codify history. Still another attendee questioned what is meant by a stand-alone OVAL validation. It should not mean validating OVAL that is

intended to run by itself, but it should mean a validation of OVAL without considering other SCAP standards. In response to questions about the validation program, NIST stated that it's not currently clear what versions of OVAL will be covered by the OVAL validation program. Being able to process OVAL content only within SCAP validation, will only be 5.3 content. Which means in the use case for NSA, a separate OVAL validation is what they need if they want to push content out that is later than 5.3, because SCAP is going to define OVAL as 5.3. A concern was expressed that if NIST was not going to have the SCAP have a stand-alone case, then the independent OVAL validation should be at version 5.3.

Tuesday June 9th

SP 800-126 Review - continued

When the conference re-convened on Tuesday morning, the session opened up for questions on NIST SP 800-126. A question was asked on Section 4.7 Data Sources. This section contains a table which contains a column for Stream Locator, which appears to be filenames, but they don't look familiar. Was this a typo or a look ahead? These names were updated to give them a more consistent appearance, but some of them are in error.

In the CPE section, Section 4.3, a notion is made of CPE names that are not in the official CPE Dictionary. That is not intentional. Even though there is no specific prohibition from using non-sanctioned CPE names, NIST would like to encourage SCAP content creators to use official CPE names whenever possible; and if a CPE names doesn't yet exist, they should try to contribute it. This could be a problem at times, for instance if a content generator is working on a secret network and there is no direct connection to a central repository to submit a name. So the specification should not explicitly prohibit this user from using the name in this type of situation. One attendee described the rules that he follows: the private use of SCAP-compatible was not discouraged, especially with CPEs; where an official CPE name existed, then you have to use that one and you could not create your own; if you create your own CPE name, then you are encouraged to submit it to the repository, but are not required to do so. To clarify, for enumerations in general, a name is either in the enumeration or it's not part of the enumeration. There is no concept of a private name. For CPE, if you're using it more as an expression language, you will find yourself in unsupported areas because that is not an intended use of the standard. A follow-up question: by "in the dictionary" do you also mean abstractions of things that are in the dictionary too? Does the abstraction of something that's in the dictionary count as something that's "in the dictionary?" The original intention was to list all the different combinations in the dictionary. But that's not what has happened, so the question is is it more important that you are using specific entries or legal components and relationships from within the dictionary? Technically, this would not be a legal use of CPE. A vendor made the point that there are cases in which vendors' tools must reference names that will never be common or "official" and will never be in the dictionary. How does a vendor implement that? This vendor merely created its own namespace, that may look "official", but is not. Another attendee cite cases with financial institutions, health care, energy who all have applications that have such special cases. The best solution for CPE might be to compute the platform

identifier so that its consistent between implementations, but also have the flexibility to allow for a custom trading application that may be proprietary. Another attendee offered that things have to be able to be represented that will never be common. Some vendors have many, many custom names. Another attendee suggested that because there are so many custom applications, secret and classified environments, it will not be possible to build a CPE dictionary to serve all. Therefore, the best solution is probably to build an expression language and an uncommon enumeration. From another: We can still satisfy both sides as long as we take into account the user requirements. We may not be able to enumerate everything, but we can make a laudable effort on enumerating the most important things and do it in a way that enables automation. One key element is to be able to compute that we're talking about the same platform. But at the same time, enable the use of a consistent naming structure for interchanging information that is not in the dictionary or will never make it into the dictionary. Another attendee suggested having different namespaces: the standard CPE dictionary in most cases, but in the secret environment use a localized namespace. NIST thought this suggestion had merit and stated that one problem with CPE is that if there is a requirement to use it only with the official CPE dictionary, then it precludes the use of CPE in some specific cases, such as classified environments, situations with restricted network connectivity, and where organizations have products that they don't want to disclose. It's in the vendors' best interests if we allow a uniform approach to do product naming so they build a common framework across their products that use that same basic capability. There's no reason it can't use names from the official dictionary as well as internal names as long as the semantics and the syntax of that usage is consistent. NIST described that individual vendors should be generating their own CPE identifiers for custom applications. Why can't a CPE name be produced by each vendor as a way to identify their own products? NIST is trying to map NVD entries to CPE names and it's important that they have some mechanism that they can match up the CPE name that they're associating with that vulnerability with a CPE name that's being discovered on the network. This problem becomes easily manageable if we look at it as a federated library system where you have to have namespaces.

Battle Stories from Vendors

The way patches were implanted in XCCDF is very painful, only one check for all patches – all or nothing. A solution was offered: within the patch enumeration, you can separately list each one of the individual patches, you can score them separately as well. It's really up to the implementation. The problem described was not a limitation to the standard, but rather a convention used in FDCC. Jon Baker asked if any of the attendees had started to use or look into the new Windows Update Searcher Test. This was added in the Fall of 2008. This test was added specifically to meet that patch policy use case where, instead of doing a patch-by-patch check, you were making sure the system was compliant with the organization's patching policy. A vendor responded that this test is very useful for desk tops but not very useful for servers.

Certification and validation was discussed by one vendor. We need to get to a point where it is very lightweight or not even there. Other standards don't require such testing. The end goal should not be certification, it should be interoperability. I want to feel good about validation, but that's not how it is today. If I pass validation, I want some assurance that I'm going to be interoperable with other vendors'

products, but that's not how it works today. Another vendor advocated for an interoperability event. Others chimed in with support of operability testing over validation testing. LtCol Wolfkiel mentioned the recent event at the DOE Cyber Security Conference where a few vendors with validated products had difficulty interoperating.

Ht_current_user … with much sympathetic laughter. NIST responded that a proposed solution under consideration is that for certain vendor product implementation types, for instance if it can't access ht_current_user because of the way the product is implemented, the settings will stay as part of the FDCC content, but the validation record will indicate that the product cannot assess those settings. The vendor replied that the user needs to degrade their security so that the tools can read that key. NIST stated that they claim what they can test and verify and that's it. It should be recognized that there are real implementation challenges with the policy as OMB has defined it. NIST allows product validations to go through, but if a product cannot pass 100% of the tests, they are obligated to indicate it as part of the validation record.

An attendee stated that he had a problem that SCAP is Windows only. NIST is doing its best to engender content for other platforms.

The Semantic Web

The semantic web has been a frequent topic of conversation within many of the different individual communities. Some of the benefits of this technology are very intriguing and might offer solutions to some of the problems that have recently challenged us. The goal of this session was to introduce this technology to our development group and to start exploring how individual initiatives might leverage it to their benefit.

The discussion started with an overview of critical business problem in our industry that is driving us toward these emerging W3C technologies. From the supply side, the problem is centered on tool to tool communication. Looking around the show floor of RSA, we all see countless vendors each pitching their own products. Yet very few of these tools can talk to one another other. Our market is incredibly fragmented and therefore it can only trend toward consolidation. But the cost of consolidation is very high (vendor acquisitions and mergers, time to market problems) and so it just doesn't happen.

On the demand side, we all have a multi vendor environments and nothing is talking to each other. Even with shared syntax (common names for the things we are talking about) users are struggling to understand the relationship between what different tools are saying and how the data being present by a tool relates to a given user's problem. Turns out that security is very observer centric and each user can have a different viewpoint, all of which may be valid. We require a semantic way of looking at things.

Looking at SCAP, and specifically SP800-117 and SP800-126, we see key phrases like "organizing and expressing security-related information" and "content interoperability across automated tools". These all point to multi vendor solutions and therefore the need to look at things in a semantic way.

At the core of things is the notion of syntactic interoperability versus semantic interoperability. Syntactic deals with mapping of information. Unique identifiers allow different pieces of data to be related to each other. We are currently very strong in this area. But semantic thinking brings inference into the equation. In short, a tool working on the semantic level might see some piece of data and then infer something else, even though that connection has not been explicitly defined.

W3C Semantic Technologies

RDF/RDFS/OWL/SPARQL are a collection of technologies that could enable the type of semantic level of communication that would help solve some of the problems we all face. This part of the discussion introduced these technologies and attempted to show where each is appropriate.

First, a few myths about ontologies and the semantic web were presented. Probably the most important point made here was that these technologies are not a silver bullet that will solve all of SCAP problems. Please see the slide deck for the complete list of myths and absurdities presented.

Even though these semantic technologies will not solve all our problems, there is a good chance that they can help solve some of the harder ones that we are currently dealing with. To understand the possibilities, we need to understand full W3C Semantic Technology Stack.

At the lowest level of the stack are the identifiers that form the basis of tool to tool communication. These identifiers are both the URI/IDs that make up our enumerations, and the character sets (like UNICODE) that help us create higher level languages. Just above these identifiers are the syntactical structures that get defined. In the W3C Semantic Technology Stack, XML is used for this purpose. This syntactic structure is enforced through validation mechanisms expressed in XMLS schema and the data held in these structures can be accessed via XML Query.

Notice that up to this point the stack is very familiar to most SCAP users. It is important to point out that the SCAP community already is heavily invested in the W3C Semantic Technology Stack. Expanding ourselves to a semantic level of data interoperability will not require us to throw out what we have been doing, but rather to build upon what we already have.

Built on top of this XML layer, we start to get into the technologies that enable inference. First is the need to represent data in a way that can support inference. RDF has been developed for this purpose. In actuality, RDF is itself XML, but it is special XML that enables data to be represented as a graph instead of as a classical table. This graph becomes the foundation for the top levels of the stack.

The discussion then turned to these top levels and introduced three core categories of technologies: vocabularies, ontologies, and querying. These areas are what is referred to as the Semantic Web. This is where inference starts to happen. RDFS (RDF Schema) is used to define the vocabulary of data relationships that inference will be based off of. OWL, and the different versions of OWL, take these relationships a step further and enable the definition of full blown ontologies. SPARQL is a query language similar to SQL that enables a user to express their own point of view and retrieve results based on the inference power of RDF/OWL.

A fair amount of time was then spent presenting a tutorial of RDF, RDFS, and OWL. These notes will not try to reproduce this tutorial. If one is interested in learning more about these technologies then it is recommended that an external resource focused on introducing these technologies is leveraged. An extensive list of resources is presented at the end of the briefing.

The Implications of SCAP Content

Presenter Andy Bove's background is with middle-ware and meta-data and back in the 1990s XML was seen as the silver bullet, but it didn't work because end-points couldn't agree on basic definitions. Point of the briefing is to understand the issues, shortcomings, and desires as they relate to SCAP content, and then initiate a consensus process for prioritization. A basic premise is that the problems currently being faced by SCAP are not unique to us.

Content validation and content understanding are two topics that the briefing will initially focus on. Content is influenced by our context. In 1986, ISO 8879 (SGML) coined the term "*heteroglossia*" which means speaking with multiple voices. People who develop code, etc. are really expecting more of a "*homoglossia"* environment. The e-commerce people from the mid-90s experienced similar problems that we are currently dealing with, e.g. they didn't have common vocabulary or a method of defining an ontology so they could share the meaning. The reason that W3C is putting the impetus on RDF, is because the integration, the thing they tried to do with XML alone simply did not work. If we think all we need is XML, we are ignoring the past, and therefore we're destined to repeat it.

The mission: 1) to explore methods that will ensure that published content will be processed "correctly" by all SCAP validated tools; 2) to explore methods that will ensure that consumers and producers of content are in a trust relationship such that parties are who they say they are and that content is not compromised in any manner during a transaction.

The conventional approach is to validate an XML instance document using a grammar-based language (e.g. DTD, Relax NG, XML Schema); and validate using a rule-based language (e.g. Schematron). What's the problem? Context and re-use, which is really about meaning. That is, we have inter and intra file/document relationships; style impacts understanding; derived vs. declared meaning/intention; normative vs. informative.

There is no current definition for a vulnerability benchmark. Nor is there a definition for a patch benchmark, nor for an inventory benchmark. The only example we have is FDCC content.

An attendee proposed that some of the problems that we've experienced has to do with the fact that no vendor has implemented the entire specifications. In theory, if the vendors had implemented the complete specifications, we'd be having far fewer problems. Another attendee stated that it's not true that if all vendors implemented the complete specifications that 800-126 would b a blank document, because there are a lot of rules that needed to be defined to provide structure. Andy Bove suggested that 800-126 describes the business rules of SCAP. There are relationships between OVAL and XCCDF that are not specified, but they are inferred by looking at the FDCC content.

The "Context" of Content

The real value of SCAP is what you can do between the specifications. This does not mean that the specifications have no value, but rather it is what you can do by applying OVAL and XCCDF in the context of CVE and CCE with CPE. But it's a double edged sword: since there was no model, developers made up their own interpretations. It wasn't until the creation of the chart in 800-126 that memorializes how to turn "Trues" and "Falses" into "Passes" and "Fails" as a function of OVAL classes did we have the model we needed. Now everyone can look at that chart and derive the same interpretation. Another problem is that we "mix metaphors" relative to the management of the content. Software developers use software control system to manage the software that they're developing. Why? To protect the value of the software. Andy then advocated for using re-use of content programmatically, that than cloning (i.e. cutting and pasting code from one area to another, which does not memorialize the code path).

What does "valid" SCAP content mean? One guess: It accurately assesses the condition you're looking for. That's correctness. There are degrees of validation.

Is SCAP really a special case? XBRL, HL7 – these are formats with similarity to SCAP. These standards help to transact business. SCAP helps us to transact the business of security.

Transacting the Business of Security

- Can I create a standard simple open format to describe my message structures and data content rules?
- Can my partners validate their transactions in test BEFORE they send them?
- How do people know what I will send them?
- I want something that's simple and standards based – leverages existing XML components
- Can I generate HTML documentation that is readable by business analysts?

Lessons learned from transacting business on the Internet:

- the ability to design transactions consistently,
- the ability to document their usage in a clear way
- the ability to drive software that can apply rules and test information content to ensure correct compliance.

The most important factor is context because context imparts meaning. If we want to share something we have to have a shared mental model about meaning.

BPEL is an Interesting Comparison

"The current semantical validation performs about 400 checks. These are checks to ensure references within the BPEL, to WSDL documents or to XSD Schemas are valid and there are checks to ensure the rules defined by the BPEL specification are not violated." What SCAP needs is a specification, and then it needs rules to make sure the content that claims to meet that specification are not violated.

Andy suggested that these are the things we want:

- The ability to have multiple structure instances selected by context
- The ability to include structure from sub-assembly of components
- The ability to have enhanced element semantics
- The ability to leverage semantics at attribute level consistently with elements
- The ability to version content model components

An attendee suggested that we add "unique and persistent" identifiers to this list.

In the SCAP Content Validation Program we are taking what's written in 800-126, which has normative and informative guidance relative to the model being described, and we are creating an SCAP Content Derived Test Requirements document (will define test cases), and from that will be created an Implementation Guide. These processes and tools will run in the SCAP Labs and will be augmented by reference images, artifacts to support testing, policy settings, etc. to perform testing for form, syntax, semantics, and correctness. These are the current goal of the SCAP Content Validation Program, but it is a work in progress.

The Products and By-Products of the SCAP Content Validation Program are:

1. Feed Back:
    a. Errors
    b. Warnings
    c. Recommendations
2. Feed Forward:
    a. Augmented Metadata
    b. Cross Reference
3. "Digitally Signed Package"

The Management of Content.

There should be a difference between the way that content is managed in development than the way it is managed after it has been released. An attendee pointed out that this could be difficult when content is signed. As a part of the content management process, dependencies need to be articulated, reported, and managed. Also version control needs to be managed. An example of a dependency would be that sometimes XCCDF documents have the same filename and this is difficult to resolve. Another is if an OVAL definition gets updated, how do you know which version you get? A vendor pointed out that our reality is that our formalism goes general to more specific, e.g. given an OVAL definition, there's no way to know how many XCCDF documents reference it. Knowing this would be helpful for content maintenance. This type of ability could be considered a management tool, or a decision support tool.

One attendee use the DNS system as an example: it is very concerned about the freshness of data, it's a distributed system, and the originators of a DNS content don't really case how many people are using it. For SCAP, we need to levy the appropriate requirements. Do we care about content as long as it is in the freshness date? Or do we need to memorialize content and go back and retrieve earlier versions of

the content – which would make it very different from DNS. A vendor pointed out an interesting point about DNS. DNS has two zones to make it effective: get_host_by_addr and then get_host the opposite way (using an IP address to resolve to a hostname) – this is an excellent example of the need for bi-directional.

A vendor pointed out that if you if you're going to validate content then you're going to have to reproduce those results. For instance, if the decision is contested, you're going to have to be able to independently verify the results. So you'll need a methodology for showing the results as consistently reproducible. You'll also need a rich semantic model that will allow for the reproduction of results when the products are no longer supported. Andy pointed out that you can't program for relationships that you don't know exist, but the ontological approach allows you to derive these relationships at a later time which is very powerful. A vendor added that you should avoid over-engineering to prevent taking advantage of such things.

Adaptation

Andy then discussed his thoughts on the compliance use case for adaptation. Given the assumption that the universe of rules is relatively fixed, then we can "dial" how to use these rules: which rules apply, to what degree they apply, what policy to they relate to, and what platform is applicable. The consensus of the attendees was that this was a good idea. DISA commented that this is the general idea behind CCI.

Andy then compared MPEG7 to TK's Semantic Web session. MPEG7 is a way of wrapping meta-data around audio and visual searches. It faces the same validation problem that SCAP does. Andy then displayed a flow chart for a validation process for MPEG7 and suggested that, this process isn't necessarily appropriate for us, nevertheless our community could benefit by looking at how other communities have solved similar problems.

What Is Validated Content?

Andy posed the question: what is SCAP Validated Content? Content that is tested by the labs and found to be consistent with the requirements established in 800-126. The question was posed whether SCAP results can be referred to as SCAP Validated Content. An attendee suggested that the SCAP Validation Program refers only to SCAP Content authors. But sometimes the lines are blurry between input content and output content. NIST stated that one school of thought suggests that there will be two phases of SCAP Validation: Phase 1 is syntactic checking is relatively easy, Phase 2 is correctness checking and is much more difficult. One vendor suggested that content that is output from a product should be validated as part of the product validation which should be a distinct process from content validation. NIST replied that there is a connection between content validation and content validation – the two are not mutually exclusive, i.e. validated content of *type X* is guaranteed to run on validated product of *type X* and this validated content is guaranteed to produce correct results. NIST has a specific use case of needing to have valid content in the National Checklist Program (NCP). LtCol Wolfkiel pointed out that these statements do not apply to CPE.

One attendee pointed out that in the Content Validation Program, Semantics and Correctness is more important that Form and Syntax. A vendor asked for a definition of "correctness" and Andy replied that

it is still being derived, and it's very similar to the correctness testing done on FDCC content. This means for a known image, the evaluation of the rules return an expected result either for a negative or a positive. How does this process scale out to applications that are more complex than Windows operating systems? Answer: Still to be determined. Where will the motivation be for vendors to produce SCAP content for their products? Answer: To be on the NCP site. Also, for the business case where an organization generates SCAP content for a fee, it would be important to such a service provider to have that content validated. Maybe we can strive to make public the rules that we develop for form and syntax and semantics and correctness so that, even if a content author doesn't go through the formal validation process, they can still benefit from the program to generate better quality content. The point was made that SCAP Validation is not SCAP, it is just a part of SCAP. Don't get too hung up on validation – you can use SCAP without using SCAP-validated tools.

John Banghart made the point that Andy is working of a document to describe the SCAP Content Validation Program and they are hoping for a lot of community feedback when the first draft of the document is made available.

BOF - SCAP Validation

At the end of Tuesday's scheduled sessions, a discussion focused on SCAP Validation was led by John Banghart of NIST. The goal of the discussion was to summarize where we as a community want to take the validation program and to explore ideas about expanding the current validation program to cover things outside of SCAP. It was pointed out that the topic of validation came up in many of the previous discussions and this just goes to show how central this topic is to the work going on in the community.

Starting with a review of the current SCAP Validation program, it was stated that SP800-126 has its validation program (known as SCAP Validation) and that the requirements for this are driven by the contents of SP800-126. Individual initiatives like OVAL would have their own program with different sets of requirements driven by the initiative's specification. We need to start thinking about this in terms of a security automation validation program. We are no longer just validating SCAP. We need to think about what validation means to things outside of SCAP.

Being a birds of a feather session, the floor was opened up for concerns and issues regarding the current SCAP Validation Program. The hope was to talk about pain points or concerns that the community is experiencing related to the current program or where the program is headed.

The SCAP Validation Program was written around use cases, specifically compliance. But we have found that there are a lot more use cases (other than FDCC compliance) that products can be used for. Products can't be validated for these other use cases and unfortunately people don't want to use the products until they are validated. This is leading to stagnant growth of certain use cases. DoD has the same issue. A use outside of compliance (but within SCAP) but SCAP validated tools don't work. They are being forced to stand up their own validation program to determine if tools can meet their needs.

It is clear that there are use cases that fall outside of the current validation program. The current program is focused around the compliance use case and FDCC. SP800-126 should help outline these

other use cases and should help drive the expansion of the validation program. NIST wants to continue to expand the program to cover additional use cases.

One of the ways that NIST hopes to expand the current validation program (outside of SCAP) is to bring in additional programs that may be separate from what SCAP defines. For example, there has been work to establish an OVAL Validation program that could be used to test tools function related to SCAP and how they can be used in some other use case. The current validation program is looking to restructure to pull in these additional programs.

It was noted that we as a community can't lose sight of the end goal which is that tools have to work together. The validation program needs to have this as their ultimate mission and always aligns itself with this goal. It main focus should be in providing assurance for this interoperability.

The discussion then turned to the need for a defined interface (e.g. web services) between tools enabling the communication that is necessary to perform the things being called out in our multi-vendor use cases. The validation program would be a great place to coordinate this shared communication. The question was asked if this is something that the different vendors need/want, and should this be part of validation? It was noted that there are two parts to this. There is the data plane and the control plane. Most of SCAP deals with the data plan by standardizing on schema. For the control plane, there are number of technologies available to help with this. The hard part in on the data plane and this is where our efforts have been focused.

Part of the challenge of the validation program is defining what interoperability means within SCAP. The validation program actually removed a few capabilities because those capabilities did not offer any interoperability components. For example, the way vulnerability database was defined there was no interoperability with other SCAP validated tools. This capability ended up being removed from the validation program.

One point that was made was that interoperability in terms of SCAP is just the exchange of XML documents. It was noted that this is the data level of interoperability. We also can't forget access to the Official CPE Dictionary, NVD, and other data sources that are not part of the XCCDF and OVAL XML streams. The question we need to ask is what type of communication exists between these entities? For example, NVD stood up a web service last year to serve up vulnerability data to the CND Pilot program within the DoD. This web service was designed with the possibility of being used by other vulnerability sources. Any organization could stand up a web service using the same WSDL file and provide vulnerability data using the same data model. It is these types of interfaces that we may want to standardize on.

Standardizing on these interfaces assumes that we have decided on what information it is that we want to share. Especially as we look outside of SCAP and at use cases beyond XCCDF and OVAL.

The 80-20 rule does not work for SCAP Validation. Unlike web browsers that are ok if they get most of the HTML standard correct, SCAP Validation needs to determine 100% compliance. If a user is sharing data between two tools, then they need to know that the tools are in-step with each other.

All the content that is produced should be readable and usable by whatever product has been validated by whatever validation methods have been used. Having this enables people to say "This is what you need to do."

The discussion then shifted to specific questions that vendors may have about the SCAP Validation Program.

Wednesday June 10

Current OVAL Issues

Deprecation Policy Review

A review of the new deprecation policy started the OVAL discussions since having a solid understanding of the policy will help in discussing many of the topics at this year's event. Prior to April 2009, the deprecation policy used by the OVAL Language was not adequately defined, and there was no documentation that described how language constructs could become deprecated, what happened to a construct once deprecated, or how a deprecated language construct could be identified. The new "OVAL Language Deprecation Policy" was developed to address these issues and more.

The creation of the deprecation policy was motivated by the needs of the OVAL Community for a defined process. As OVAL continues to mature, and become more widely adopted, it is important to ensure that the process by which the language may change is well documented and that reasonable conventions are in place for the community to support.

Several key points of the new policy are outlined below:

- All OVAL Language constructs must be deprecated before being removed.
- The duration of deprecation will be in terms of releases.
- Language constructs will remain deprecated for at least one release.
- Deprecated constructs will be marked with machine readable indicators.
- Prior to any release, candidates for deprecation will be announced and discussed
- Prior to any release, deprecated constructs will be discussed on a case by case basis for removal.
- There is no requirement that a deprecated construct be removed from the language after any number of releases.

Following the overview of the deprecation policy, several items were discussed and clarified. These discussion topics are summarized below.

Nominating Language Constructs for Deprecation and Removal

The question of how a given language construct is nominated for deprecation, and how a deprecated item is later nominated for removal, was raised. Under this policy, the OVAL Moderator will work towards a consensus within the community before any construct is deprecated, or removed, from the

language. It was clarified that there is no requirement to remove a construct that has been deprecated. However, it is undesirable to have the language become completely bloated with deprecated constructs. A balance between the desire to support deprecated constructs for long periods of time, and the desire to allow the language to evolve free from the burden of carrying year's worth of deprecated constructs, needs to be achieved.

Requiring Vendor Support for Deprecated Constructs

Next, the issue of requiring support for deprecated constructs was raised. Currently, there are numerous tests and other language constructs that have been deprecated. Should the vendors be required to support all of these deprecated constructs?

Historically, deprecated constructs have been considered part of the OVAL Language and therefore considered to be required for a vendor to implement. Products still need to support deprecated constructs to ensure that content, which utilizes deprecated constructs, will be properly supported. Content authors need to be aware of deprecated constructs and avoid using them. Vendors that are implementing support for OVAL must also support deprecated constructs.

With the introduction of machine readable deprecation flags, it is now much easier for users of OVAL to make informed decisions about deprecated constructs.

Deprecations Occurs with New Releases

The question of when a language construct can be deprecated was brought up. Basically, the group wanted to better understand when an item could be deprecated?

A language construct may be deprecated only with a release of the OVAL Language. Constructs will not be deprecated without a new release of the OVAL Language. Similarly, deprecated constructs may only be removed with a new release and they cannot be removed at any other time.

Deprecation Information for Existing Deprecated Items

Given that this deprecation policy was published in April, and that there has not been a release of the OVAL Language since the fall, will this new deprecation information be added to the existing deprecated constructs?

The most significant change of Draft 1 of Version 5.6 was the addition of deprecation information to all of the existing language constructs that have been deprecated.

Schematron Usage in OVAL

Schematron has been used in OVAL since Version 5.0. Compliance with the Schematron rules has been considered optional since its introduction. Schematron is used in conjunction with XML Schema to express data validation constraints that XML Schema alone cannot express. Schematron leverages XPath expressions to define constraints and relationships within the OVAL Language. Schematron can report both warnings and errors during validation.

At this point, Schematron validation can be prohibitively slow. With large XML documents, Schematron validation often takes several minutes. This has been a large factor in considering the Schematron rules to be optional.

Currently, Schematron is optional for all use cases. It is left up to organizations to determine how, or if, they will support Schematron. The OVAL Repository currently requires all submissions to comply with the Schematron rules because it will help ensure that the best content is available in the repository.

This discussion is intended to serve as an opportunity to consider how OVAL should use Schematron moving forward.

Requiring Schematron

The discussion was started by considering the implications of requiring Schematron validation. It was acknowledged that Schematron validation may be more important when producing content than when consuming content. For vendors that consume content, it is probably less important that all imported content is validated against the Schematron rules with the assumption that the content author has already done this validation, and that the tool follows the spirit of the Schematron rules.

It was agreed that moving forward it is important for all published content to comply with the Schematron Rules. This will ensure that high-quality content is available as well as place the burden of content validation largely on the content producer rather than each and every downstream consumer of that content.

Changing Schematron Rules May Break Backwards Compatibility

Currently, Schematron rules are created and modified with each version of the OVAL Language. These changes are typically done to add additional restrictions to the valid structure of OVAL definition documents. This flexibility with Schematron rules helps to ensure that increasingly higher-quality content is produced. However, this maybe concerning to some vendors as it will result in documents that were considered valid in one minor version of the OVAL Language to become invalid in a subsequent minor version of the language.

In general, it was agreed that the ability to add new data validation rules through Schematron, as the OVAL Language matures, is desirable. As long as the changes coincide with OVAL Language releases, the rules should continue to be matured over time.

Validate the Rules or Follow Their Spirit

The discussion shifted to validation and whether or not tools should be required to actually evaluate the Schematron rules, or if the requirement should be that the tools must follow the intent of the rules. Given that there is a strong desire not to force implementations on the vendors, requiring all the vendors to specifically support Schematron may not be the right approach.

The consensus was that it is important to follow the spirit of the Schematron rules. It does not matter if content producers and consumers actually utilize the Schematron rules. The Schematron rules are simply an expression of requirements for OVAL content. A content producer or consumer should be free

to use the rules through Schematron or to review them, understand them, and ensure that the rules are followed.

Schematron Validation Requirements

From a product validation standpoint, it is important that a content consumer can report errors in any content that is being imported. However, it is not required for tools to do this all the time, and it is not required that an XML Schema or Schematron rules are used. The intent of the rules must simply be verified.

Conclusion

The consensus was that for OVAL, compliant content should be considered to be any content that complies with the XML Schema and the requirements expressed in Schematron. With this agreement, the discussion was then summarized as follows:

- Official content must be compliant with the XML Schema and Schematron rules.
- Any output from a tool must be compliant with the XML Schema and Schematron rules.
- Any tool must be able to detect noncompliance with the XML Schema and Schematron rules.

Moving forward these items will be captured in the OVAL and SCAP Validation programs. The OVAL documentation will be updated to reflect the requirement that states compliant content must satisfy the XML Schema and Schematron statements.

In order to ensure that the highest quality OVAL content is available, Schematron rules will continue to be allowed to become increasingly stringent with each release of the OVAL Language.

Element Name Reconciliation

In naming and managing OVAL Language constructs, the following guidelines have been used:

- Make element names as intuitive as possible.
- Reduce schema bloat when possible.
- Utilize consistent naming patterns.

As the OVAL Language evolves, these guidelines begin to contradict each other. For example, if a typo is found in an element name that makes the element slightly less intuitive, but, fixing the typo in a new release will add to schema bloat since the misspelled element cannot be removed. As another example, when creating a new test, it may be possible to utilize an existing state. In this case, reusing an existing state will reduce schema bloat, but it will also reduce readability. Due to these inherent contradictions, the naming pattern of a test aligning with object, state, and item names has broken down and element names have diverged.

This discussion focused around a proposal to bring all test, object, state, and item names into alignment. Along with the effort to realign the element names, a convention would be established that all names will align. This convention could be automated to ensure that element names do not diverge again.

Impact of the Proposal

If accepted, this proposal would introduce new tests, objects, states, and items where ever there was a misalignment of elements names. In the process of renaming elements, any incorrectly-named elements would be deprecated. This change would not invalidate any existing content, but it would add schema bloat.

Benefit of the Proposal

If accepted, the proposal would ensure that a constant naming pattern is followed for all future changes. Establishing a constant naming pattern will simplify some implementations since the relationship between an item in the system characteristics schema will always align in name prefix with the corresponding test, object, and state in the OVAL definitions schema. The proposal would also begin the process of removing all misnamed elements by deprecating them according to the deprecation policy.

Proposal Discussion

When considering a change to OVAL, the development burden for making a change should be minimized when possible. People, when given proper documentation, can learn to overcome changes and inconsistencies. There has been concern about the large size of XML documents and thought should be given to making changes to the OVAL Language to reduce instance document bloat.

As the discussion progressed, two different conventions were considered. The first was the proposed convention of ensuring that all element names align regardless of potential schema bloat. The second convention was to lean in favor of reducing schema bloat and reusing elements whenever possible. This second convention would suggest that element names do not need to be maintained to always align, but, when originally created, the names should align. Then, after being created, the names should be allowed to diverge if the result of the divergence will be to reduce schema bloat.

During the course of the discussion, it was pointed out that there is no documented or machine-readable mapping between tests, objects, states, and the corresponding item in the system characteristics schema. It was agreed that this is something that should be documented, and clarified, with the next release to ensure that there is no ambiguity in this relationship. Currently, the mapping is assumed to be based on an element name prefix. Encoding the mapping in element name prefixes may lead to trouble down the road.

Concerns about the way in which OVAL Language schemas use namespaces to differentiate versions were raised. It was suggested that some of the challenges of managing the language over time might be easier to address if the schema namespace included both the major and minor version of the language.

Conclusion

It was concluded that leaving typos in the language is not a good idea and that any of the naming inconsistencies that are due to typos should be addressed.

Keeping the mapping between items, tests, objects, and states as implicitly defined, is not desirable. This is something that should be corrected, and an explicit mapping should be created.

At this time, aligning the names of tests, objects, states, and items is not worth the implementation effort. In the future, perhaps with a major revision, element names should be brought into alignment.

As long as an explicit mapping between tests, objects, states, and items is provided, an effort should be made to reduce schema bloat and reuse existing elements when possible in the future.

Choice Structure on Objects

At OVAL Developer Days 2008, the notion of introducing a choice structure into objects in the OVAL Language was discussed and agreed to. Since then, a proposal has been made for how to actually implement this choice structure. What remains to be discussed is whether or not this new structure can be added into OVAL Version 5.6, or if this structure introduces too large of a change for a minor revision.

Issue and Proposal Review

As background for discussion, a review of the issue and the standing implementation proposal was held. That proposal can be found here: http://oval.mitre.org/community/archives.html#nabble-td1485589.

In the course of reviewing the issue, the following example was used:

```
<file_object id="oval:sample:obj:1" version="1" xmlns="...">
    <path>c:\windows</path>
    <filename>foo.exe</filename>
</file_object>
```

Here the current file_object is presented. Currently, a file can only be identified by a combination of the path and filename element as seen above. It is not possible to have a single string representing a complete file path today. This becomes an issue when the complete file path is not available in any other form. On Windows systems, the registry frequently holds complete paths to files that need to be examined. It is also common to store complete paths to binaries in configuration files on other platforms as well. In OVAL, these other sources of information are often queried to determine the location of a file that is then examined with a file test or file permission test. In the current version of OVAL, there is no way to support these subsequent tests because files can only be identified by a separate path and filename.

The proposed solution to this issue would allow the following two methods for representing a file_object in OVAL:

```
<file_object id="oval:sample:obj:1" version="1" xmlns="...">
    <path>c:\windows</path>
    <filename>foo.exe</filename>
</file_object>
```

OR

```
<file_object id="oval:sample:obj:2" version="1" xmlns="…">
    <filepath>c:\windows\foo.exe</filepath>
</file_object>
```

Basically, a file_object could be expressed as it is today with the path and separate filename, or with a combined filepath element.

The XML Schema for a file_object currently looks like this:

```
<xsd:sequence>
    <xsd:element name="behaviors" type="win-def:FileBehaviors" minOccurs="0"/>
    <xsd:element name="path" type="oval-def:EntityObjectStringType“/>
    <xsd:element name="filename" type="oval-def:EntityObjectStringType" nillable="true"/>
</xsd:sequence>
```

The proposed XML Schema would look like this:

```
<xsd:sequence>
    <xsd:element name="behaviors" type="win-def:FileBehaviors" minOccurs="0"/>
    <xsd:choice>
        <xsd:sequence>
            <xsd:element name="path" type="oval-def:EntityObjectStringType"/>
            <xsd:element name="filename" type="oval-def:EntityObjectStringType" nillable="true"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="filepath" type="oval-def:EntityObjectStringType"/>
       </xsd:sequence>
    </xsd:choice>
</xsd:sequence>
```

The important change to notice is that an xsd:choice has been introduced. The xsd:choice would allow for two different sequences of elements. Introducing this xsd:choice structure would be a distinct change from the very consistent pattern in OVAL of only having one possible sequence of child elements for any given object or state.

It is also important to note that the example above is based on a file_object and that there are several other places in OVAL that could benefit from this same xsd:choice structure. If it is agreed that this choice structure on objects should be introduced, a similar pattern will be implemented to support choice structures on the other objects too.

As previously stated, the notion of introducing a choice structure was agreed to at OVAL Developer Days 2008. This is now being considered for inclusion in Version 5.6 because it does not break backwards compatibility, and addresses a known deficiency that is preventing certain types of tests from being developed.

Proposal Discussion

Looking at how tests have evolved in OVAL to date, the lack of support for a choice structure has been worked around by simply creating new tests. This can be seen in the user_test and the user_sid_test. Ideally, a choice structure could have been used there to allow the original user_test to support identifying a user by username or by SID. This workaround caused the OVAL Community to learn these two different methods for looking up a user. It would have been cleaner, and resulted in less schema bloat, if a choice structure had been introduced instead of a new test.

In surveying the vendors in attendance, it was agreed that this addition would not be a significant burden to support. In fact, the benefits of adding this choice structure far outweigh the implementation cost of supporting the structure.

Conclusion

A proposal will be created and distributed to the oval-developer-list for adding the proposed choice structure into the current file_object. A list of other objects that would also benefit from this structure

will be included in the proposal, and assuming agreement over the oval-developer-list, those changes can also be made in Version 5.6.

Supporting N-Tuples in OVAL

The OVAL Language currently supports several data repositories that may return query results as n-tuples. However, it does not currently provide a mechanism for representing these n-tuples, or for checking the state of a result set that contains n-tuples. OVAL has solid support for result sets with single value results and arrays of single values, but leaves something to be desired when working with WMI, SQL, and XML where n-tuples common.

This discussion focused on reviewing the issue and considering three different proposals for addressing the issue. While the examples and the discussion was in the context of WMI, the understanding was that the issue is common to several data stores that the OVAL Language currently supports and that a common solution based on the proposals will be developed. Given the increasing demand for adding this capability to OVAL Language, the group considered whether or not this feature could be added to the OVAL Language in Version 5.6.

Background

The two examples below highlight the current OVAL capability, and the desired capability. The first example is a WQL used to query WMI that selects the 'ScreenSaverTimeOut' field from each 'Win32_Desktop' class in WMI.

```
SELECT ScreenSaverTimeOut FROM Win32_Desktop;
```

The second example selects the 'Name' and 'ScreenSaverTimeOut' fields from each 'Win32_Desktop' class in WMI. This example will select 0 – n pairs of results where it is important to maintain the relationship between the 'Name' and 'ScreenSaverTimeOut' fields.

```
SELECT Name, ScreenSaverTimeOut FROM Win32_Desktop;
```

Using the second example, it is possible to know what a specific user's 'ScreenSaverTimeOut' is set to. With the first example, it is only possible to examine all the 'ScreenSaverTimeOut'values on the system, and not individual user's values.

It is also important to note that this capability has not been supported in the OVAL Language because there has never been a solution that did not introduce an entirely new structure for developers to implement and content authors to learn. The group was reminded that a primary goal in the development of Version 5.0 of the OVAL Language was consistency in order to ensure that once a single test in OVAL was understood all other tests would be easily understood. This was also done to reduce the implementation burden for developers.

Current WMI State

As background, the current win-def:wmi_state and win-sc:wmi_item were reviewed before any proposals were presented. Below is an example of a win-def:wmi_state that highlights the result element as it is in Version 5.5.

```
<wmi_state id="oval:sample:ste:1" version="1" xmlns="...">
    <result datatype="string" operation="equals" >user2</result>
</wmi_state>
```

Below is an example of the win-sc:wmi_item as it is in Version 5.5.

```
<wmi_item id="1" xmlns="...">
    <namespace>root\CIMV2</namespace>
    <wql>SELECT Name FROM Win32_Desktop</wql>
    <result>user2</result>
    <result>user1</result>
</wmi_item>
```

Proposal One – 'record' Datatype

The first proposal considered introduces a new 'record' datatype that would allow an entity to have child 'field' elements. The proposed changes to the win-def:wmi_state are shown below:

```
SELECT Name, ScreenSaverTimeOut FROM Win32_Desktop;

<wmi_state id="oval:sample:ste:2" operator="AND" version="1" xmlns="...">
  <result datatype="record" operation="equals" entity_check="all">
    <field name= "Name" datatype="string" operation="equals">user</field>
    <field name= "ScreenSaverTimeOut" datatype="int"
           operation="less than">600</field>
  </result>
</wmi_state>
```

Under this proposal, the current 'result' element would remain unchanged and a new 'record' datatype would be introduced. This datatype would allow mixed content and define a 'field' element. Each 'field' element would have a unique 'name' attribute that would distinguish one 'field' from another. Field elements would have their own datatype and operation to allow for different datatypes and operations to be specified for each 'field' as seen in the example above. Field elements would also support 'var_ref', 'var_check', and 'entity_check' attributes in the same way that other standard entities in the OVAL Language support these attributes.

When considering this proposal there are a few issues that should be weighed:

1. This proposal will keep the result entity closely aligned with other entities.
2. However, close alignment will leave several unneeded and unused attributes on the 'result' element. For example, the 'result' element would allow for a 'var_ref' attribute which would not really have any meaning.
3. The 'datatype' attribute, rather than an element name, is being used to indicate that the element will have child elements. This is different than most of the other constructs in the OVAL Language and is not considered good XML Schema practice.
4. Adding in the 'record' datatype will change the nature of the result element such that it does not align with any of the other entities in the OVAL Language.

Proposal Two – 'resultset' Entity

The second proposal considered introduces a new 'resultset' element that would allow an entity to have child 'field' elements. The proposed changes to the win-def:wmi_state are shown below:

```
SELECT Name, ScreenSaverTimeOut FROM Win32_Desktop;

<wmi_state id="oval:sample:ste:2" operator="AND" version="1" xmlns="...">
  <result datatype="string" operation="equals" >user2</result>
  <resultset entity_check="all">
    <field name= "Name" datatype="string" operation="equals">user2</field>
    <field name= "ScreenSaverTimeOut" datatype="int"
           operation="less than">600</field>
  </resultset>
</wmi_state>
```

Under this proposal, the current result element would remain unchanged and a new 'resultset' element would be introduced. This element defines a child 'field' element and supports the 'entity_check' attribute. The 'entity_check' attribute would allow for assertions to be made about multiple 'resultset' elements. Similar to the first proposal, each 'field' element would have a unique 'name' attribute that would distinguish one 'field' from another. Field elements would have their own datatype and operation to allow for different datatypes and operations to be specified for each 'field' as seen in the example above. Field elements would also support 'var_ref', 'var_check', and 'entity_check' attributes in the same way that other standard entities in the OVAL Language support these attributes.

When considering this proposal there are a few issues that should be weighed:

1. The 'resultset' entity is unlike any other entity in the OVAL Language and diverges from previous efforts to ensure that all entities are similar in nature.
2. The proposal would result in the creation of an oval-def: ResultSetType that would be reused in other logical locations in the OVAL Language.
3. This new element would have only the needed attributes which would simplify it, but also make it even more different than other elements in the OVAL Language.

Proposal Three – Sequential Result Entities

The third proposal considered introduces several new 'result' elements where each new element is sequentially named. The proposed changes to the win-def:wmi_state are shown below:

```
SELECT Name, ScreenSaverTimeOut FROM Win32_Desktop;

<wmi_state id="oval:sample:ste:2" operator="AND" version="1" xmlns="...">
  <result datatype="string" operation="equals" >user2</result>
  <result_1 datatype="string" operation="equals" >user2</result>
  <result_2 datatype="int" operation="equals" >333</result>
</wmi_state>
```

Under this proposal, the current 'result' element would remain unchanged and several new 'result' elements would be introduced. Each of these new 'result' elements would be just like the existing element except that they would allow for tuples with a few more elements to be represented.

When considering this proposal, there are a few issues that should be weighed:

1. This solution will support tuples that are limited in size by the number of sequentially named 'result' elements. This will address some cases, but there will always be a request for one more 'result' element.

2. This proposal does not allow the complete set of elements in the tuple to be treated as a unit. The other proposals consider the complete tuple to be the unit for comparisons.
3. This is not a complete solution, but merely a workaround that remains consistent with the other structures in the OVAL Language.

Discussion

In the follow-up discussion for proposal two, the suggestion of using the choice structure that was discussed in the previous session was made. A choice structure here would allow a state to have either a traditional 'result' element or the new proposed element. This would ensure that this new element would not be used with the existing 'result' element.

It was pointed out that the 'field' elements from proposal one and two need to also support unnamed fields. For example, a user might want to use 'SELECT * FROM Some_Table'. This would select all of the fields in that table without naming them. The first two proposals must support this to be effective. The suggestion was made allow the children to be either 'named_field' elements or 'anonymous_field' elements where the anonymous version would have a sequence attribute to uniquely identify one field and the named version would have a 'name' attribute to uniquely identify one field.

In proposal two, it would not make sense to use both the proposed 'resultset' element and the existing 'result' element in the same state. This might be justification for using a choice structure here or at least using Schematron rules to prevent this.

In proposal one and two, it is important to note that the child 'field' elements would all be logically and'ed together during evaluation. So in the examples for proposals one and two, the evaluation results from each field are being and'ed together to determine the overall evaluation result for the entity.

It was also pointed out that when possible, Schematron rules should be avoided. The first preference should be to define language constructs through XML Schema. Then, if all else fails, use Schematron. This will make the language simpler to follow and avoid some of the challenges of Schematron.

These proposals do not address what the corresponding system characteristics items would look like. The proposal assumes that the items will have a parallel structure. When the final proposal is made to the oval-developer-list, an example showing the item in the system characteristics file is needed.

Another option would be to simply overload the existing 'result' element by inserting comma separated values. This would allow for some improvement over the existing capability, but would not allow per field comparisons with different datatypes and operations.

RDF may offer some assistance in supporting n-tuples. Perhaps utilizing RDF in this particular structure is worth exploring. The issue is that this seems like a very large change for OVAL. In the near-term, RDF is not likely a good solution, but should be considered for a major revision or as a long-term solution.

This discussion is assuming that all of the data returned is tabular. In the future, OVAL may need to support data that is returned as a graph. In fact, because XML is hierarchical, there may already be a need to support non-tabular data. How would this proposal support querying XML where node sets are

returned? At the moment, it is not clear how OVAL would support doing further comparisons on XML data that can be stored in any number of different locations. The solution is not just simple XPath statements to retrieve single values or n-tuples.

This proposal, and the notion of adding RDF to OVAL, led to the suggestion to allow for a 'development' branch, or similar, to run in parallel to the current official branch. This discussion led to a general consensus in the room that an experimental branch is really needed for this and many other reasons.

Given that this need to support n-tuples is really a new area for the OVAL Language to support, it might be better to define this as an entirely new construct and not attempt to fit the solution into the existing entity structure. It may actually be harder for users to learn the new structure if it is not clearly broken out as a new structure. Given the benefit of this capability, it is well worth the cost to teach new users how to use it as a new unique construct.

Regarding the third proposal, we should ensure that we develop a good generally-applicable solution. Simply developing a workaround will not solve the entire problem.

The discussion shifted to whether or not this capability could be introduced in a minor version. One perspective was that as long as backwards compatibility is maintained it is acceptable to introduce capabilities like this in a minor release. To others, this seems like a major release type of capability since there is a fairly high impact on introducing it. It will introduce an entirely new construct to several tests in the OVAL definitions schema and items in the system characteristics schema.

Conclusion

When the discussion concluded, it was agreed that a case could be made for including this capability in OVAL Version 5.6, or deciding that this capability could only be introduced with a major revision. The consensus was that this capability should be further discussed and explored before adding it to Version 5.6. A discussion will be started on the oval-developer-list to continue the dialogue on this topic.

Regarding the merits of the specific proposals, there was strong support for proposal two. Since the structure is really representing an entirely new concept in the OVAL Language, it is at least okay to be inconsistent, and perhaps should be inconsistent, with the other entities. Also, there was no desire to develop a partial solution to this issue.

During the course of the discussion, it was generally agreed to that OVAL does not currently allow the community to easily explore new solutions. There is a strong desire to setup a 'development' branch, or similar, to allow new ideas to be explored and vetted before they are potentially integrated into the official OVAL Language. Most participants agreed that this 'development' branch would be a great place to explore solutions to this issue.

Pattern Match on Enumerations

The ability to use the pattern match operation on enumerations has been an open issue for the OVAL Language since Version 5.0 was released. Enumerations were added to the OVAL Language in order to restrict constructs to specific values, ensure tool interoperability, and increase content readability. As a

result, the pattern match operation could not be used on constructs that used enumerations. The deficiency caused by this restriction can be easily demonstrated with the following example.

```
<auditeventpolicy_state id="oval:sample:ste:1" version="1" xmlns="...">
  <account_logon datatype="string" operation="pattern match">
    AUDIT_(SUCCESS|SUCCESS_FAILURE)
  </account_logon>
</auditeventpolicy_state>
```

Currently, this is not a valid auditeventpolicy_state because the string 'AUDIT_(SUCCESS|SUCCESS_FAILURE)' is not included in the enumeration that restricts that allowed values of the account_logon entity. In order to correctly perform this check, two tests would have to be created that check for the value AUDIT_SUCCESS and AUDIT_SUCCESS_FAILURE, and the two tests would have to be embedded in a criteria construct which uses the OR operator. This is much more verbose, and most likely doesn't align with the content developer's thought process.

In Version 5.3 of the OVAL Language, there were Schematron rules that restricted the operations permitted on enumerations to just 'equals' or 'not equal' as the 'pattern match' operation on a finite set of strings did not make sense. Additionally, allowing the 'pattern match' operation undermines the reasoning for having enumerations to begin with. However, in Version 5.4 of the OVAL Language, these rules were accidentally dropped making a workaround possible.

Pattern Match on Enumerations Workaround

The workaround can be implemented by using a variable reference for an entity's value and then specifying the regular expression in that variable. This work around is demonstrated below.

```
<auditeventpolicy_state id="oval:sample:ste:1" version="1" xmlns="...">
  <account_logon datatype="string" operation="pattern match"var_ref="oval:sample:var:1"/>
</auditeventpolicy_state>


<constant_variable id="oval:sample:var:1" version="1" datatype="string" ...>
  <value>AUDIT_(SUCCESS|SUCCESS_FAILURE)</value>
</constant_variable>
```

In this discussion, the decision to be made is should this workaround be embraced and accepted in the OVAL Language, or should the rules that prohibit this workaround be put back in the OVAL Language for Version 5.6.

If the rules are dropped, and not put back in the language, it would allow the opportunity to close a long outstanding issue. However, this could also be very dangerous because it would allow content developers to place whatever values that they wanted in making it much more difficult to ensure that content is valid, and as a result could cause issues with vendor tools.

Conclusion

After some discussion, a consensus was reached that in Version 5.6 of the OVAL Language the rules should be left out and that the workaround be accepted as part of the language. It was made clear that the documentation should specify that any regular expression used in the workaround should matched against the enumerated values already defined in the OVAL Language. It was also recommended that content developers be encouraged to anchor their regular expressions. Otherwise, unexpected results with vendor tools could occur.

Tests Reference Multiple States

The capability to have a test reference multiple states has been a topic of discussion for many years and has been requested by many members of the OVAL Community. The major motivation behind this capability is that content developers would be given the ability to specify an acceptable range of values as well as produce content that is much more readable. Additionally, during the 2008 OVAL Developer Days Conference (Pg. 21), there were discussions regarding whether or not states should be put back inside tests for Version 6.0 of the OVAL Language. During these discussions, the OVAL Community expressed that a change of this magnitude which would require all of the existing content to be re-written was not in the best interest of the OVAL Community, and should not be pursued any further. However, if members want to reconsider this issue for Version 6.0 of the OVAL Language, the topic can be discussed in more detail.

Proposal

This proposal would grant content developers the ability to reference multiple states in a single test allowing for the use or ranges in a clear and succinct manner. A simple example that can easily demonstrate the value of this capability can be seen below.

```
<min_passwd_len datatype="int" operation="greater than or equal">8</min_passwd_len>


<min_passwd_len datatype="int" operation="less than or equal">16</min_passwd_len>
```

With this new capability, a content developer would be able to make a single test that references one state that evaluates to true if the min_password_len is greater than or equal to 8 and another state that evaluates to true if the min_password_len is less than or equal to 16. This would allow an author to easily write a single test to ensure that the minimum password length was between 8 and 16. Currently, the method of performing this same check would require the author to create two separate tests, one for each allowed state. This is cumbersome, often counter intuitive, and it diminishes the readability of the OVAL content.

Impact of Change

This change would require the addition of a new attribute 'state_operation' on the oval-def:TestType as well as changing the multiplicity of each test's state element to unbounded. An example of this change can be seen below.

```
<xsd:element name="state" type="StateRefType" minOccurs="0" maxOccurs="unbounded"/>
```

The new attribute 'state_operation' would be based on the oval-def:OperatorEnumertion datatype which would allow the operations AND, OR, XOR, and ONE to be performed on the states in order to logically combine them.

Lastly, these changes would not invalidate any existing content because it allows content developers to reference multiple states in a single test object instead of just one state object as currently defined. The new attribute 'state_operation' would not break backwards compatibility because it would have a default value of AND, and if a content developer is referencing only one state, it would impact the evaluation of the test. It is also important to note that historically test-level attributes have been added to the OVAL Language in minor revisions. However, the same cannot be said for changing the multiplicity of an object.

Benefits of Change

The introduction of this proposal into the OVAL Language would allow ranges of values to be specified for the values of states in OVAL definitions. As a result, it would simplify content authoring because it would remove the extra criteria, tests, and states necessary to perform this same functionality.

Conclusion

As a result of the discussion, it was determined that extending the multiplicity of states to unbounded as well as adding a 'state_operation' attribute would be beneficial to the OVAL Language because it would simplify content authoring, increase the readability of OVAL definitions, and allow for ranges of values to be specified in state objects. It was also decided that this change would be acceptable for Version 5.6, and that a proposal would be sent out to the oval-developer-list for further discussion by the OVAL Community.

Introduce PCRE Based Pattern Matches

This topic was originally discussed on the oval-developer-list and at the 2008 OVAL Developer Days Conference (Pgs. 21-23) which was focused on driving Version 6.0 of the OVAL Language. It came to the attention of the OVAL Community that the majority of the existing OVAL content was utilizing the PCRE regular expression syntax even though the POSIX regular expression syntax was defined in the OVAL Language. As a result of these discussions, it was decided that the change from POSIX to PCRE in the Version 6.0 release of the OVAL Language would be beneficial. However, more discussion was needed to make the final decision.

Proposal

As a result of the discussions mentioned above, a proposal was introduced that would require the addition of a new operation called 'pcre pattern match' in the OperationEnumeration datatype as well as the deprecation of the operation 'pattern match' in the OperationEnumeration datatype. This proposal favors the addition of a new value in the OperationEnumeration datatype rather than adding an additional attribute that specifies the regular expression syntax because it follows how things were previously done with other OVAL Language structures. The major question regarding this proposal was whether or not the change fit in the OVAL Version 5.6 release.

Impact of Change

The major impact of implementing this proposal is that it would introduce the 'pcre pattern match' operation and would deprecate the 'pattern match' operation which was specified to use the POSIX regular expression syntax. Due to the conditions of the OVAL Deprecation Policy, these changes would not invalidate existing OVAL content in the next release. Additionally, this change would suggest that the OVAL Language supports both POSIX and PCRE until the POSIX-based 'pattern match' operation is officially removed from the OVAL Language.

Benefits of Change

The primary benefit of making this change is that it would help ensure a standard meaning for all OVAL content. Currently, content authors are using PCRE syntax instead of POSIX out of habit. Users are accustomed to PCRE syntax and implementers are supporting PCRE syntax. For the most part, POSIX is not being used. Creating a 'pcre pattern match' operation would allow vendors and users of OVAL to declare that a given string is a PCRE-based regex and not a POSIX-based regex. This would ensure that all regular expressions are evaluated in their intended syntax. The change would allow much of the existing content to be corrected and brought into alignment with the regular expression syntax of OVAL.

Major Arguments for the Proposal

The first major argument that arose during the discussion was that it would be very unrealistic for the OVAL tool vendors to implement multiple regular expression syntaxes and it would potentially hinder the adoption of the OVAL Language by new vendors. Another argument for making this change is that most OVAL compatible tools, and OVAL content, are already using the PCRE regular expression syntax. Along the same lines, the majority of tool and content developers are already comfortable with the PCRE regular expression syntax and it is counterproductive to have to train developers to use a different regular expression syntax. Additionally, most regular expression syntaxes, Python, Java, Visual Basic, and Perl to name a few model the same syntactical behavior as PCRE. Lastly, OVAL has made it a priority to promote interoperability and reduce vendor burden in order to further the adoption of the OVAL Language. The addition of many regular expression syntaxes would increase the burden on the developers and reduce the potential for interoperability.

Major Arguments Against the Proposal

The overwhelming argument against the proposal is that by specifying a particular regular expression syntax it would be effectively limiting the capabilities of the OVAL tool and content developers, and would not necessarily solve every vendor's needs. Essentially, in the end, it should be up to the developers to make the decision about which syntax is best for them as this flexibility would help expand the community. Also, instead of choosing a single regular expression syntax, a subset of all of the regular expression syntaxes could be used to ensure that every syntax is compatible. The final major argument against making the switch to the PCRE regular expression syntax is that it does not support internationalization and it would alienate many communities that may be interested in becoming involved in the OVAL Community.

Conclusion

The two most notable options that developed out of this discussion are listed below.

1) Add a new value 'pcre pattern match' to the OperationEnumeration datatype and deprecate the value 'pattern match' from the OperationEnumeration datatype. The value 'pattern match' would then be removed at a later time as specified by the OVAL Language Deprecation Policy.

2) Change the documentation in the OVAL Language to specify the PCRE regular expression syntax as the syntax of choice rather than the POSIX regular expression syntax.

As a result of the discussion, the group came to a consensus that, regardless of the option selected, the change should be made in the Version 5.6 release of the OVAL Language, and that these options would be presented to the OVAL Community on the oval-developer-list for further discussion.

OVAL for System Querying

An emerging use case for the OVAL Language, which has been requested by multiple members of the OVAL Community, is the ability to obtain data from the system without making an assertion about its state. Currently, the OVAL Language provides a framework for performing a system inventory with respect to some pre-defined state whether it be in a compliance, inventory, patch, vulnerability, or miscellaneous definition. This new use case would allow OVAL content authors to request all of the items on a particular system using existing OVAL objects. The major questions considered during this discussion include:

1) Should we support this capability in the OVAL Language?

2) Is there enough support to do the work to implement this capability?

Major Discussion Points

First the issue of whether or not the OVAL Community should embrace the use case for system querying in the OVAL Language was raised. As a result, the Open Checklist Reporting Language, also known as OCRL was mentioned. OCRL is an emerging language specification for collecting a system's state and generating human-readable reports. This then raised the question of whether or not this capability was needed in the OVAL Language, and if so, is a new specification even necessary? The group decided that, before attempting to create an entirely new specification, they would like to see if the capability could be built into the OVAL Language. However, before this use case could be built into OVAL, it was proposed that a model should be built to define what it means to collect the data from the system, and how the data should be represented such that it can be understandable by a human. At that point, it could be determined whether or not this capability is outside the scope of the OVAL Language, and if so, would be best left to a new specification like OCRL. Next, a concern was raised about the privacy and security implications of being able to collect information about a particular system. It was then clarified that system querying would only pertain to authenticated systems meaning that unauthorized users would be unable to collect information about any particular system. Another concern about implementing this capability is that you would now be dealing with large sets of data and it would be advantageous if you could keep a local cache and only retrieve the differences between the cache and the system. It also might be beneficial to have higher level constructs that allow you to filter the data sets. This use case could also be useful in assisting interviewers that need to answer OCIL questions.

Lastly, it was mentioned that the OVAL system characteristics file was not the answer to implementing this use case because it cannot convert the system characteristics data into CCEs, CPEs, or CVEs.

Conclusion

It was the general consensus of the group that they liked the idea of having support for system querying in the OVAL Language**.** However, it seems that efforts might be better suited improving OVAL's ability to make assertions about machine state, and that it would be great if the OVAL Community could present some proposals on the oval-developer-list to facilitate additional discussion to decide if it makes sense to put this capability in OVAL (e.g. does it undermine existing goals of the OVAL Language?), or if it would make more sense to add an additional specification such as OCRL.

OVAL Repository Considerations

The key considerations for this topic focus on answering the following two questions:

1) Should inventory definitions be required to have a CPE name?

2) Should compliance definitions be required to have CCE IDs?

These questions suggest that if an inventory definition cannot obtain a CPE name or if a compliance definition cannot obtain a CCE ID their class would be changed to 'miscellaneous'. That is not to say that if an inventory definition or compliance definition are candidates for CPEs and CCEs respectively, but they are unable to immediately obtain these names and identifiers, that they would automatically become members of the 'miscellaneous' class. This notion of putting definitions in a 'miscellaneous' class is reserved for definitions that cannot obtain their respective names or identifiers because they do not actually qualify for them as defined in the corresponding specifications, not because they simply don't have the identifiers or names assigned yet. Two examples of definitions that would go into the 'miscellaneous' class under this proposal can be seen below.

1) A definition which determines if Microsoft Windows XP SP2 or later is installed is not an inventory definition

2) If a product XYZ reaches the end of life it is not a vulnerability definition because it is not a candidate for a CVE ID.

A major discussion point regarding this topic was how would the introduction of a 'miscellaneous' class affect the production and consumption of content in the OVAL Repository. The general consensus was that it would be useful to make the change and clarify the distinction of what can really be an inventory or compliance definition. It was also mentioned that it would not drastically affect the production and consumption of the OVAL Repository content. Another key discussion point was that it would be beneficial to establish a convention on how to handle compliance definitions and formalize what it means to be a compliance definition. Lastly, a question was raised as to whether or not the mapping of definitions belong in OVAL or CPE and CCE where the majority of the work is being done, and whether or not it makes sense to extend CPE and CCE to define where the definitions should exist. The problem with this is that all of these specifications CCE, CVE, CPE, OVAL, etc. have grown independently and it would need to be determined who would be responsible for taking on this task.

Conclusion

By the conclusion of this discussion, there was agreement that it would be beneficial to require inventory definitions to have CPEs and for compliance definitions to have CCEs as it would make the OVAL Repository consistent with vulnerability definitions that are required to have CVEs. As a result, it was decided that this discussion would be continued on the oval-developer-list for further consideration.

XCCDF Current Issues

This goal of this session of Security Automation Developer Days was to review the existing use cases that the language is attempting to support and get opinions on if certain use cases should be added or removed. With the desired use cases in hand the hope is that discussion of a few proposed features can be had. This part of the discussion was aimed at addressing the issues that have been raised on the XCCDF mailing list over the past couple of years.

Before getting the discussion started there was one announcement regarding XCCDF. An XCCDF Board will be established to help the moderator make decisions on proposed changes and to help lead the standard through future revisions. This board will hopefully represent a wide range of XCCDF community members and is intended to provide some transparency to the process of evolving the XCCDF language. More information about the board and how to become a member will be made available to the community over the discussion list.

Use Cases

The first topic for this session was about the use cases supported by XCCDF. First, a review of existing use cases was presented along with a list of current requirements that these use cases generated. Please see the slide deck for more details.

The question was presented to the group about whether XCCDF is trying to do too much. Should we trim down some of these use cases. The response was that the list of use cases is correct, we just have to be careful that we don't keep trying to add everything into this. We may have reached the limit as to how much stuff we can fit into XCCDF and still have a language that is manageable.

The feeling is that there is a lot in the standard that is not being used today. Having some of this is great for future growth, but this might be a sign that we are trying to do too much. The problem is mostly for tool implementers that must try and built support for all of this stuff even if it isn't used. We may need to realign our use cases and requirements. It was pointed out that the SCAP Validation program is not requiring tools to do everything. Rather the expectation is that tools will implement those features that are necessary for the job at hand.

There was some follow-up noting that XCCDF as it currently stands does a great job for what it is trying to do, and the expectations for content and tools is well balanced. The previous points were more a warning to keep in mind about growing too big.

It was also noted that many of the things we have or will talk about are things that we need and although adding more to XCCDF presents the danger of making it too bloated, the alternative is to create a separate standard which would have its own disadvantages including determining how it interfaces with XCCDF

External Tailoring

Currently, tailoring of an XCCDF document is done through profiles but these must be embedded directly into the XCCDF document itself. The proposal is to support profiles outside the XCCDF document and have this external profile reference back to an XCCDF document.

One advantage of this is that it allows the original document to stand alone and not be affected by future profile changes. The authority can be confident that their guidance is not being affected. Today, we have to modify the original document. If the authority makes a change, everyone that has performed individual tailoring must redo their work. Also, since the original documents have been modified, it is hard to determine if the original guidance has not been changed. One can't use hashes, etc to verify the file's status.

It was noted that this is more than just an external profile, but we also need to be able to add new rules and groups externally.

There are a lot of technical challenges that must be worked out as we work though this. We must figure out how to trust external content.

The consensus in the group was that this is a feature that should be supported and that a proposal should be made for the next release of XCCDF. Users are currently doing this in a proprietary way due to the need. It is time to standardize this.

Automatic Tailoring

There currently is some automatic tailoring in XCCDF through the platform element. One can use a CPE Name to state that a given rule/group/profile should be skipped if the platform under review does not match the name. Do we want to do more? Do we want to have the evaluation path of an XCCDF doc be determinable by returns of certain checks?

We need more than just CPE Names when making these types of automated tailoring. For example, there may be a need to use a certain profile if a given patch is not installed. In essence this is a question about creating an if-then-else condition within XCCDF.

The existing <question> element is targeted at this type of functionality, it just isn't geared toward automatic handling of these conditions.

We have to be careful not to try and encode the entire world into a single checklist. At some point it is worth having separate benchmarks instead of combining them into one with conditional statements to determine which part to use.

In general the conversation was focused around the group trying to understand the needs surrounding automatic tailoring. The needs to do the things enabled by this feature are there, but including this inside XCCDF is not so clear. The group was trying to find the line about what should be inside XCCDF and what should be pushed out to an external technology. Automatic tailoring could very well be on the other side of the line. More exploration of this topic is needed before consensus about its conclusion can be reached.

Remediation

The discussion then shifted to remediation. Currently XCCDF has a <fixtext> element that accepts any unformatted text. There has been talk within the community about developing a remediation language and we should to consider how this might fit into XCCDF. The question was posed to the group about adding more standardization around remediation information in XCCDF.

One idea would be to model something similar to how assessment languages are leveraged using the <check> element. It was mentioned that we have to be careful not to bloat XCCDF by adding too much into the language, and that we should try to externalize remediation information as much as possible. This would be in-line with the <check> element idea in that XCCDF would call out to a remediation language instead of formatting the information inside the <fixtext> element.

There wasn't much discussion on this topic and it was mentioned that the group needs to know more about the work going on around remediation before specific decisions can be made. This topic should be revisited when the remediation work is more mature.

Checker Control

The next topic discussed was about encoding within an XCCDF document more control over the operations taken by a checking tool. There are many instances currently within the XCCDF Specification where decisions are left up to an individual checking tool. For example, a checking tool can choose to evaluate <Rules> in any order that it sees fit. Do we want to close some of these areas and provide more structure over a checking tool's operations?

Regarding the deterministic ordering of rules and groups, should we consider enforcing an order that checking tools should evaluate? For example, assume there are 6 rules in a given XCCDF document, should a checking tool be expected to evaluate those 6 rules in order of appearance? Currently, a checking tool can evaluate in any order.

The group initially struggled with finding a reason why this would be necessary. It was then pointed out that while the ordering of doesn't seem to matter when dealing with rule leveraging OVAL, the addition of OCIL may change that. It may be important to control the order that questions are presented to the user.

One user did mention that they have seen different results from tools when rules were presented in a different order. If a tool remediates during the assessment, then evaluating a certain rule first might trigger a remediation that changes the results of a rule run later. In other words, a problem can arise if there is a change is system state during the evaluation of the rules.

These two reasons did not seem to warrant an immediate change to XCCDF as both OCIL and remediation are things under development and something that XCCDF is only considering at this time. There seemed to be consensus in the room that the topic of deterministic ordering of rules and groups should be investigated further as new things are added into the language.

The next topic regarding checker control was support for chained checks within a single rule. There are times when some sort of logical structure is needed, beyond that what is provided by complex checks. Based on the results of one check, then some other check should be performed. The question was posed to the group about if this is something we want added to XCCDF and how much work would be required by the vendors to implement.

The first vendor to speak up was initially in favor of the feature. Others were also initially in favor as this type of feature is mandatory in the world of compliance checking. We must be able to say things like "if this is a database server then ...". It was pointed out that OVAL does already provide this type of functionality so we already can make those types of statements. The response to this was that having this feature only in OVAL makes it impossible to map from existing standards like PCI into SCAP. There was a little bit of confusion about why this was the case. However, it does sound like the vendors need this capability. The question is does it belong in XCCDF or does OVAL already provide it?

It was again mentioned that as we bring in new languages like OCIL we may find that we need functionality like this. Yes, OVAL provides this feature, but when we start mixing different checking systems we lose the ability to leverage OVAL for this. For example, content may want to first ask the user a question, and based on the response to that question, run some type of OVAL check.

A question was asked regarding if we add this feature, then what is the impact to scoring? It was noted that the idea of chained checks was limited to just a single rule, so the rule will still fail or pass just like before. So scoring should work in the same way that it does currently.

As with the previous topic, the general feeling of the group was this is another feature that XCCDF needs to continue to investigate as we start to bring in other potential checking systems like OCIL.

The final topic in this section was related to periodicity of rules. Do we have a need to mark certain rules with the notion that their results become invalid after a certain amount of time? An initial response was that this is something that could be useful as the notion of perishable objects is very real.

However, from a policy perspective the type of periodicity that is currently used is at the benchmark level and not the rule level. We say things like "only trust the results of this benchmark if it has been run within the past 2 days". One suggestion was to extend this proposal to the benchmark level as well as the rule level.

On the compliance side, the user may want to allow an override, also known as a waiver. They may want to say that a certain configuration can be allowed for some specified period of time, but after that the configuration needs to be reported as a failure. In other words, a given rule might be skipped for 30 days but then must be evaluated after that time period has expired.

We may have to address this at a variety of different levels. Users may want to rescan systems 15 day after a certain patch was released. This is would require the notion of tagging the entire benchmark with a periodicity value. Maybe this feature should be pushed up to a higher level control language.

It was mentioned that periodicity also means that we need tools to maintain state between different evaluation runs. Tools will need to keep information about the time and dates run. This statement is not entirely true as XCCDF does already have timestamps throughout the results format. We could use a timestamp with this feature and say skip a rule for 15 days after the given timestamp.

One view was that there is no doubt that this is needed for some use cases. But this it should not get added to XCCDF. This capability should be provided by something else.

It was noted that we kind of got off track with the discussion. The original proposal was around periodicity of evaluating certain rules. Put another way, suggestions that a given rule needs to be evaluated every 2 days, etc. The thinking was not to force a tool down a path of requiring temporal control, but rather to provide a way for the content creator to make a suggestion about these things and mark this up in a defined field. Tool could choose to use this information or not.

An individual noted that the problem with the above is that it will eventually pull you down the more complicated path we were just on. If you put this into XCCDF, vendors are going to do something with it. But if we don't define it more than each tool will produce different answers. For example, if a rule expires, should it result to 'false' or 'not checked'?

One suggestion was to change the goal of this discussion to "periodicity guidance". We had been talking about "periodicity instructions" and this has the notion of forcing a tool to do things, as opposed to just making a suggestion. It might be useful to have a control language read periodicity guidance within an XCCDF checklist and then take action if deemed appropriate. XCCDF could add something similar to the current <fixtext> element. The notion of actual instructions was considered by many in the room to be too strict for XCCDF's needs and should instead be pushed up to a control language.

There are operational limitations on when a benchmark can be evaluated (i.e. off business hours). In addition, there are desires to reevaluate after specific events like a new patch being released.

The general consensus from the group was that XCCDF can put in this type of guidance into a text field if it wants, but that this should just be guidance and should not be something that is enforced and trying to drive the tool. Most felt that we should not try to push this into XCCDF at all.

Result Mapping

Currently, the mapping between the results of a check and the results of a rule are not codified anywhere. Is this something we should allow an individual rule to specific how to interpret results from a checking system? Or should XCCDF define an explicit map that always holds?

One view was that this mapping should be part of SP800-126 and not part of XCCDF. Of course this would assume that we were just providing a single map. The real question we need to answer is if XCCDF needs to allow each rule to define the mapping or if we can just state the mapping once.

An example of the dynamic rule mapping would be when a rule calls a checking system and wants to say that if the checking system returns TRUE, then the rule should pass, otherwise the rule should fail. A different rule may call the same checking system but state that the rule should pass only if the checking system returns FALSE.

If we want support a proprietary checking system, then we need to allow result mapping in the rule. Users of a proprietary checking system can't edit SP800-126 to add in the mapping.

Maybe we need both options? SP800-126 could define the default behavior and then allow the individual rules to override the default.

An example was brought up using OVAL and the different classes of definitions. An XCCDF content writer may want a rule to pass if the OVAL inventory definition returns TRUE saying that a given product is installed. But they may want to create a different rule that passes if some other pieces of software is missing. This means they would want to pass if the inventory definition returns FALSE. In response to this, it was noted that maybe what is needed in this case is not a result mapping, but a new OVAL class of definitions. This would allow us to always write an OVAL Definition that returns TRUE for the desired XCCDF pass condition.

Another suggestion was brought up about pushing this type of mapping down to the individual checking system. For example, OVAL could define that a true result for a given definition is a pass condition, etc. The problem with this approach is that it forces each individual checking system to know what higher level languages are leveraging it. This is unrealistic as there may be some language in some foreign country that uses the checking system that we don't know about. Any mapping needs to live with the language that is doing the leveraging, not with the language being leveraged.

It was pointed out that this past example points out the need for an integration document that defines the requirements of an acceptable checking system and how a checking system can be integrated into XCCDF.

Another thought that was presented was to add a negate attribute to the <check> element. This would give the XCCDF content author the ability to control the mapping on an individual rule element. But is a negate sufficient? We might need to map a NOT_APPLICABLE result from a checking system to a fail result for the rule.

The consensus in the room was that XCCDF should experiment with the idea of allowing individual rules to explicitly state the result mapping that they require. It is unknown however at this time if this type of proposal is the best way to go.

Lists in Values

Currently XCCDF Values can only contain single data points. But checking systems, specifically OVAL in today's world, allow for the import of a list of values. This capability is necessary for rules that want to check a condition on a list of files. The capability to pass a list of values to a checking system has been a long standing request for XCCDF. Is this something we want to go ahead with?

It was noted that some of the FDCC check require this functionality. There need becomes apparent during profile creation. Instead of hard coding the values directly into the checking system, we would like to set the profile to contain different lists that could be passed.

A question was posed about if we need to worry about homogeneous types within this list? It was agreed that this is a great question and something that would need to be figured out during the implementation of this proposal.

Maybe we need to pass a tuple of information, especially when dealing with ACL's. Having a more robust solution like this would be a good thing and something that we need to work on with the implementation.

The consensus in the room was that we need to move ahead and work on this proposal for the next version of XCCDF.

Value Population

Right now values are unidirectional. The values are set via defaults or via a profile within the XCCDF document. This value is then passed to the checking system. There have been requests to enable this value to then change based on results from the checking system. Taking this proposal further, it might be nice to enable an XCCDF Value to be set based on values returned by a specific check.

It was mentioned that we are probably going to have to go down this path if we want to go down the if-then-else path. However, the response pointed out that the initial if-then-else proposal was based on a TRUE return from the checking system, not an actual value.

The discussion moved on to point out that we would need to specify how XCCDF can accept values from a checking system, and how the checking system can present values back to XCCDF. We need to define the interface between XCCDF and the checking system.

Maybe what we are talking about is not a checking system (a TRUE/FALSE mechanism) but rather a data collection system. We want to be careful not to pigeon-hole ourselves into the checking system paradigm.

It was noted that there already is a <check-import> element within XCCDF that was partially designed for this purpose. We might be able to improve this existing feature.

The consensus in the room was that this type of functionality is needed. We should review the current functionality and propose some improvements to it to enable the current needs as discussed.

Versioning

The final discussion topic was about versioning individual Rules/Groups/Profiles within a benchmark. There currently is an optional version element on every item. Do we need to make this mandatory? Do we need to add additional metadata to this? This discussion was based on a question that had been posed earlier in the week.

One member pointed out that we definitely need a version at the benchmark level. But as far as individual items, he questioned the need there. Every time a rule or group changes, the benchmark version should change. So we should be able to use the high level benchmark version for all of our needs.

If we start talking about external profiles, then the need to version the individual rules becomes more important. Granted we could always link an external profile to a specific benchmark version.

We only need to start thinking about versioning individual rules is if we get away from the idea of an XCCDF document as a single bundle. If we start breaking XCCDF into its individual pieces and allowing these pieces to be shared independently, then we will need to start versioning these individual pieces.

Consensus in the room was that we should leave versioning as it is, but that if we change how we package XCCDF information or we start dealing with external profiles, then we should revisit this topic.

The discussion moved on to a different versioning topic. It was mentioned that the discussion earlier in the week might have been about a slightly different idea. We were talking about the need to deal with which version of certain check we are calling out to. In other words, do we need to include both the id of a specific check and a version number? Currently we just include the id. Since checks constantly change, the XCCDF content writer has no way of stabilizing the check that they want to include.

As much as we all hate the idea of trying to carry around a lot of version information, it really is something that is important as we reference external checks. We need a way to verify that we are evaluating the exact content that was intended.

We need some amount of trust that the content we are evaluating has not been swapped out from underneath us. Versioning is a start, but we may need to go further.

It was mentioned that if we start requiring the XCCDF <check> element to include both the id and the version of the desired check, then if the check ever changes to fix a bug or something similar then we will need to go back and rev the XCCDF doc that includes this check. There is precedence for not being so strict with versioning. OVAL does not require the version of an individual test when including it in a definition criteria for this exact reason. Also, programming language do not force you to include the version of an API call when you use it. There reason for all of this is to allow fixes to be leveraged without having to go back and explicitly change everything that calls it.

At the same time, we can always encourage specific version encoding in the content we write and then allow tools to make decisions about what paths to follow. If we specify the intended version, then at least a tool knows what the intension was and can make a better decision about whether to leverage a newer version or not.

Consensus in the room was that we need to start being very specific about any external content we are referencing. We should work on a proposal to include both the id and version inside the <check> element.

Thursday June 11

Remediation

This session presented the initial work of an ongoing collaboration between NIST, DoD, and others to consider standardized remediation. The presentation was prepared by Matt Kerr of G2, working with Matthew Wojcik of MITRE and David Waltermire of NIST. The discussion focused on the basic approach taken to date and various issues which have been identified. Ongoing participation and feedback was also requested.

Note: names or acronyms used here for various remediation functions are preliminary and for discussion purposes only. They are likely to change.

Definition of Terms

For the purposes of the discussion, a remediation was defined as a change to a computer's configuration or installed software in response to organizational policy, a software flaw, mis-configuration, or non-compliant setting. It was noted that standardized remediation approaches may generalize to any configuration change, though that is not identified as a core requirement for this effort. Actions such as installing or uninstalling applications or libraries should be considered as in scope, however, because they are so frequently motivated by security concerns.

Use Cases

The following use cases have been identified:

- Remediate one or more computing assets for all problems found
- Remediate one or more computing assets for a subset of problems found
- Apply one or more remediations to one or more computing assets regardless of their current state (i.e. without requiring a prior assessment)

Derived Requirements

Based on the identified use cases, a set of derived requirements were presented. These include the need to:

- Uniquely identify a remediation (i.e. a Common Remediation Enumeration)
- Express additional data about a remediation, including mappings to applicable platforms, related vulnerabilities or configuration issues, etc.
- Define a format for exchanging that additional data (a Markup Language)
- Express precisely what steps to take to apply a remediation, in a machine readable format (provisionally called OVRL or Open Vulnerability Remediation Language)
- Develop a method of specifying which remediations are acceptable for certain types of assets as defined by function or role (a Remediation Policy Specification)
- Develop a method of specifying which remediations should be applied to which specific assets in an operational environment (a Control Language)

    - The Control Language further must be able to describe sets of computing assets, and specify values for any remediation parameters (e.g. minimum password length)
- Express the results of remediation actions (e.g. success, failure, error)

CRE

CRE or Common Remediation Enumeration is the tentative name for the idea of a standard list of identifiers for remediations. An individual CRE would represent a set of actions to remediate a specific vulnerability or mis-configuration. If a particular remediation requires multiple "atomic" actions to actually accomplish the intent, those would all be considered the same CRE. For example, if mitigating a particular vulnerability requires running an executable with certain arguments and changing multiple configuration values in files, registry, or other configuration database, those would all be considered part of a single CRE.

The CRE concept and workflow should leverage lessons learned from the Common Vulnerabilities and Exposures (CVE) and Common Configuration Enumeration (CCE) efforts. CRE IDs should facilitate communication between different groups and coordination of various tools in much the same way as CVE and CCE IDs.

With those considerations, a CRE entry has been preliminarily defined as:

- A unique identifier
- A textual, human-oriented description of the remediation
- Supporting references

There was broad consensus that CRE is needed to support vendor interoperability and compliance reporting. Adoption of CRE would require an ongoing mapping activity by vendors to their internal remediation identifiers. CRE entries would also have to be updated over time as new information about each remediation is discovered and released; users of CRE would have to track those updates.

Certain risks were discussed if CRE is used as the sole coordination point for remediation. If a CRE ID has not been published and incorporated by all parties involved, no remediation automation for that issue would be possible. Timeliness in creating and mapping CREs will be critical. Also, a mis-match could lead to unintended changes to target systems.

There was a discussion of possible formats for the identifiers themselves, especially relating to the question of whether local remediation identifiers should be supported as well as global ids. Several anticipated there would be need for local CREs, though it is unclear how local CREs could be used to direct remediation tools. Presumably the necessary mapping would need to be coordinated directly between the issuer of the local CRE and their tool vendor(s). Local CREs could be valuable for reporting purposes however, particularly in conjunction with machine-readable directives for remediation actions (see OVRL, below).

CREs, both global and local, may allow for integration with a tool which does not directly support the standard, if an organization is willing to perform the mapping itself. That is much less likely in the case of OVRL. This may actually facilitate remediation automation in environments that have managed

systems using general-purpose configuration management tools, which tools may be slow to adopt security-specific standards.

Models for the structure of CRE IDs were discussed, including CPE, CVE, CCE, and OVAL. Consensus seemed to be to keep semantic content in the ID as low as possible, partly based on lessons learned from CPE. Assuming a requirement for local CREs, options discussed included using an organizational namespace component in the ID structure (similar to OVAL), or having a reserved block of IDs (if a structure more similar to CVE or CCE is used). Several present expressed a preference for incorporating an organizational namespace; it was viewed both as more elegant than reserved blocks, and as removing most risk of ID collision.

Remediation Markup Language

Additional data, beyond the minimum required to differentiate between different CREs, will also be required to support the identified use cases. Applicability of a remediation to platforms, vulnerabilities, or configuration issues could be provided by a mapping to CPE, CVE, and CCE names. Prerequisites for performing the remediation, e.g. disk space requirements, and whether a reboot is required, will be necessary. Links to machine-interpretable descriptions of the remediation (see OVRL below) may also be included eventually. Certain metadata about CRE entries themselves should also be provided, such as creation and modification dates, deprecation status, and provenance information.

A standardized format for exchanging CREs, their metadata, and the additional data about the remediations will be required. A strawman example was presented and discussed.

OVRL

Open Vulnerability Remediation Language or OVRL (a preliminary name) would provide a standardized means of creating machine-readable, functional descriptions of remediations. This would be analogous to OVAL, where CRE is analogous to CVE. An OVRL statement would describe exactly what steps a compatible remediation tool must take to perform a remediation action.

OVRL could potentially reduce the need for certain mapping activities around CRE. Compatible tools could be directed precisely what steps to take, rather than coordinating by ID.

OVRL should leverage lessons learned from OVAL (and XCCDF if applicable). It should reuse OVAL constructs where possible, since there is significant overlap in the functionality of identifying a machine artifact for checking its state (OVAL) and for modifying its state (OVRL).

An OVRL statement would include:

- Prerequisites for successful remediation, such as disk space or existing software state requirements
- Order of operations
- The changes to make to the system
- Follow-up actions, such as restarting a service or rebooting
- Steps to take in case of error

Discussion topic: Should OVRL use the OVAL schema, extend OVAL schema, or create a separate OVRL schema?

There was broad agreement that the OVAL object structure should be reused somehow in OVRL, because of the clear overlap of function. This could lower cost of adoption for those who have already implemented OVAL, and present a lower learning curve to new adopters. However this reuse is accomplished, any changes to the OVAL object structure should be tracked in OVRL and vice versa. Simply copying the pertinent OVAL schema elements into an OVRL namespace would require maintenance and revision of the same items in two places, duplicating work and possibly leading to inconsistencies.

Directly extending OVAL to support the additional semantics required for remediation was seen by several as problematic and adding too much complexity to OVAL. Also, OVRL is expected to go through revision rapidly during early development, and that should not cause rapid OVAL version changes. It was mentioned that the possibility of an OVAL component schema for remediation was discussed at OVAL Developer Days 2008. There could potentially be ways to address OVAL adoption and validation requirements that would allow frequent version changes for such a component schema without adversely affecting OVAL assessment adoption and validation.

A strawman example OVRL remediation statement was presented, which directly used items from the OVAL namespace where appropriate, and used new schema elements from a notional OVRL namespace for new, remediation-specific functionality.

It was pointed out that the OVAL object structure is only suitable for a limited set of necessary remediations—Windows registry key changes, possibly certain file actions, and perhaps some others. Patch installations and application installation or removal are only a few examples of required functions that likely could not leverage current OVAL objects. In such cases, if OVRL was directly using OVAL objects, would we propose creating new OVAL schema, or instead create an OVRL object structure? The former could again lead to OVAL versioning problems; the latter could be inelegant and confusing, as some system artifacts would be referenced using OVAL objects and some OVRL objects.

It was suggested that OVRL should be able to "call" OVAL definitions for prerequisite checks etc. It is still to be determined whether the references to such definitions would be directly in the OVRL remediation statements, or in some wrapper language (perhaps XCCDF). The XCCDF spec does include the <fix> element which is intended for expressing how to remediate a rule, but no known implementations exist.

The dangers of accepting and acting on arbitrary OVRL content were discussed. Since OVRL as imagined is intended to describe system state changes, malicious OVRL content could easily subvert any machine it was run on. Digital signatures, content validation, and proper establishment of trust will be critical in any use of OVRL. This is not unique to OVRL, but important to note.

Discussion topic: "Undo" functionality—is it possible to reverse a remediation? Should OVRL support describing this? If so, how?

Several attendees agreed that true undo is essentially impossible for remediation, except possibly in certain specific circumstances. One participant experienced in automated remediation estimated that even for patches, rollback will be unsuccessful in 75% of cases. Trying to support undo often implies keeping full history of all system state changes made during a remediation—and how long must those be kept? A supporting comment was made pointing out the additional complexity of the interactions of multiple changes over time, and the fact that the remediation tool will not be the only thing changing the system. However, it may be worth considering supporting "compensating controls" intended to return the system to some known state in the context of the remediation. This may be more feasible than a true rollback.

Remediation Policy Specification

This concept involves allowing an organization to identify the allowed or preferred remediations for a vulnerability or mis-configuration. E.g., stating that for a given CVE, a specific CRE is the only allowed remediation for a particular CPE. This should also support specification of remediations to apply to certain machine types without requiring a prior check. These machine types might be defined by OS type, server vs domain controller vs desktop—in other words, functional or organizational categories, rather than e.g. IP address ranges. The latter would be addressed by the Control Language (see below).

There is some conceptual overlap between Remediation Policy and the Control Language. The scenario presented is that system types as described in remediation policies would somehow be translated to identifiers of specific machines in the enterprise. These asset identifiers would then be used in the creation of control language statements. For example, a Remediation Policy document might specify certain CREs should be applied to "Internet-facing web servers." In an operational environment, that type description would have to be mapped to actual IP addresses or hostnames in the enterprise, for use in Control Language documents. The specifics of how this mapping would occur is currently considered out of scope for this effort.

Discussion topic: Should XCCDF be explored as the language for expressing remediation policies?

Many of the scenarios for describing which remediations are allowed for various types of machine are similar to the use cases for XCCDF profiles. XCCDF may therefore be a logical candidate for expressing Remediation Policy. This would allow Remediation Policy considerations to be presented as standalone documents, or integrated into the types of checklists currently expressed in XCCDF.

Concern was expressed over adding additional complexity or bloat to XCCDF. A counter point was raised that if the <fix> element currently in XCCDF is usable for this purpose, there may be little added complexity to the specification itself. For the use case of specifying remediation options to be used if a system is found out of compliance with a portion of a benchmark, this seems a natural fit: an XCCDF rule specifies what it means to be in compliance; it links to an OVAL definition for how to check; it could link to an OVRL statement for how to remediate. Additional complexity and cost to implement XCCDF are definitely concerns, but the question was also raised that if XCCDF is not to be used, what would be the alternative? If XCCDF is suitable, introducing a whole new format could be avoided.

It was mentioned that more information may be necessary than is available at the XCCDF rule level. An XCCDF rule might fail for various reasons, which might need to factor into which (if any) remediation action is taken.

Regardless of the final approach, the point was made that a requirement should be to support lightweight statements of remediation policy as well as more complex "compliance requirement, assessment, remediation, follow-up assessment" documents.

Remediation Control Language

This concept involves specifying which remediations to enact on what specific machines in an enterprise. The term "control" is used in the sense of directing functional components in the enterprise to take specific actions—e.g. invoking a remediation tool. The Control Language is seen as the standardized way for some centralized decision-making process to coordinate between policy, (optionally) assessment results, and remediation actions on specific hosts.

In a Control Language document, those machines might be described by attributes such as IP addresses or ranges, hostnames, or Active Directory or other domain membership. This contrasts with the functional or role categories which would be defined and used by Remediation Policy documents. The Control Language would also specify values for variables used by remediations (e.g. number of characters to set as the minimum password length). It might also support whether the user or admin can delay remediation implementation, and if so for how long or how many times. There may be support for other operational parameters as well.

It is acknowledged that similar use cases exist for coordinating assessment and other functions already in the scope of SCAP. Any work on a Control Language for remediation should take that into account. There may be some conceptual overlap with the emerging work on the Policy Language for Assessment Result Reporting (PLARR).

Returning Remediation Results

This discussion focused on the topic of remediation results. Is it necessary or desirable to return results of a remediation attempt, or is it sufficient to be able to specify a follow-up machine state assessment? If results of the remediation attempt are desired, what type and level of detail should be supported? Several participants agreed that results from remediation efforts are necessary, supporting at a minimum result codes such as success, fail, error, with timestamps and indications of what machines were involved. Also, if remediations are deferrable, indications of whether they are deferred will be important. These results will be needed to knit together tools from different vendors, e.g. assessment, remediation and reporting.

The point was also raised that a post-remediation machine state assessment may be able to discover that a remediation wasn't successful, but not why the remediation failed. Without that, the possibility exists for an infinite "assess, failed remediation, assess, failed remediation" loop. There was also speculation that failing to support the return of remediation results might prevent future multi-component use cases not yet anticipated.

Other Comments

- Certain aspects of remediation decision-making and policy definition were noted to overlap with the problem domain of Business Process Execution Language (BPEL). It may be worth exploring BPEL's applicability to remediation, or even broader concerns in security content automation.
- It was mentioned that there are certain existing patents applying to remediation, some held by participating organizations. The possibility exists for remediation standardization efforts to enter areas protected by one or more patents.

Conclusion of Remediation Discussion

These remediation standardization efforts are just beginning. Further feedback and participation is definitely desired. The clear consensus from those present was that focusing first on developing CRE would provide the quickest path for initial remediation capabilities to be integrated into the standards-based security content automation workflow. The Markup Language and Control Language may also be necessary in order to define inputs and outputs adequately to allow for possible validation testing. There was general agreement that OVRL and Remediation Policy Specification are important, but of lower priority.

The NIST Emerging Specifications Discussion List will initially be used for discussion. See http://scap.nist.gov/community.html to subscribe.

CCI Use Cases for DoD

Dave Hoon, who supports Field Security Operations (FSO) at DISA presented Common Control Identifier (CCI). FSO's mission is:

> Responsible for enhancing availability and security of the Global Information Grid by ensuring adherence to Information Assurance and NETOPS Policies including **development of guides and procedures**; training of Combatant Commands, subordinate and service components; **implementation of standard IA solutions; formal certification reviews and tracking compliance metrics**.

FSO's functions include the following:

> **Develop, Implement and Maintain IA Security Guidance and Processes**. **Conduct Full Scope Security Reviews** and Provide Assistance. Provide **Certification and Accreditation Support** and Perform as The **Single Certifying Authority for DISA**. Develop and Implement a NETOPS Evaluation and Certification Program. Perform **Computer Network Defense Service Provider assessments** and **make Certification recommendations. Implement Security Architecture and Information Assurance Tools**. Develop and distribute IA Training Products and Provide IA Training. Develop, Implement and Maintain Vulnerability Management Systems.

FSO's perspective comes from all sides of the problem. They act security content authors, they write tools to assess using that content, they write tools to remediate using that content, they perform assessments using that content, they collect data from the assessments, they develop POA&Ms, etc. This provides FSA with a full view.

Documents that they consider to be source policy include DoD 8500 Series, IAVMs, CTO's, SP 800-53, CNSS 1253, CJCSM, etc. Getting these documents down to meaningful terms is a struggle. Today they have 39 STIGs, which translates into roughly 125 security checklists, that they maintain internally. They

produce 13 -15 new development guides each year. They also have a regular maintenance program: every two months they do an update of their security guidance and publish that.

Everything they develop is maintained in the database, is in proprietary format, and in their vulnerability management system. When they enumerate all of the requirements they come up with over 45,000 or more vulnerabilities and requirements that they track. They perform more 10,000 asset reviews annually. They see the struggles people have in implementing security.

As good as SCAP has been, there are still problems. Determining what to do, what is security guidance? When are you done? When are you secure? Have you covered all the requirements that you need? There are also problems with compliance reporting. Starting with so many documents, how do you make the relationship to the stand-alone requirements? Sometimes these map back, sometimes they don't. There are also problems with security guide development. There's a high demand for new and updated security guidance, they need to go through various approval authorities – it's difficult for FSO to keep up with the demand.

FSO sees CCI as helping out with these problems. CCI is a de-composition of an IA Control or an IA industry best practice into single, actionable statements. Take the IA Control, de-compose it into what the requirements, and that's what a CCI is. A CCI is not necessarily a requirement, just an enumeration of several requirements. A CCI List is a collection of CCI Items, which express common IA practices or controls. The CCI data specification is currently out in draft; it has been released to NIST and NSA and they've received several comments back and the document has been updated appropriately and re-released.

CCI Use Cases

The following are some use cases for CCI as envisioned by FSO.

Secure Product Development - Vendors can use CCI to incorporate security requirements into their products as part of the development cycle and then they *'will know when they are done.'*

IA Compliance Reporting - CCI allows detailed reporting of compliance to IA Controls and will include the ability to report partial compliance.

Security Guide Development - CCI data model in VMS will supports dynamic STIG generation based on asset characteristics and this supports consistent guide development from external sources.

FSO believes that CCI helps to bring clarity to some of the vagueness in the 800-53 IA controls. Also, it will help with compliance reporting, to act as "glue" to bring the results together that are received from many different sources.

Dave then illustrated these points by showing an example IA Control (IA-5) and showed how many requirements are derived from that single control. He then showed several CCIs that can generated from that single control.

FSO sees CCI to be a link between various policies to individual SCAP standards. Within the data model, a CCI item doesn't apply values, it doesn't prescribe what needs to be done. Rather, it just states what the capability is, what the action is that needs to be performed. A CCI item contains a name, a description, a note for additional clarification (if needed), a parameter, and also an ability to cite the source references.

An attendee pointed out that SCAP content frequently refers to DISA STIGs. However, the DISA STIGs can change. How does one know what version of the STIG that a document refers. DISA's content will reference CCIs. Another attendee pointed out, however, that DISA STIGs are becoming reference-able resources, and recommended that they be placed in some sort of wrapper. The DISA roadmap calls for STIGs and checklists, but they will be replaced with benchmarks expressed in XCCDF and making references to SCAP data (CCE, CVE, etc.). Again, the point was made that noting version information when referencing documents is critically important and this will become obvious when we enter the audit space. Dave responded that the current CCI specification could handle the referencing of version information.

Matt Wojcik stated that CCE is attempting to reference document versions. Deprecation information is also important. An attendee asked whether CCI would support nested mappings, e.g. with regards to 800-53 and 1253. Yes.

We also need to consider the propagation of mapping. We need to consider exercising transitive relationships. By building a layer approach to handle the identifiers, we reduce the amount of effort required to maintain it. Agreed.

An attendee said he was anticipating an explosion of relationships. E.g. CVE is related to CPE – several relationships can apply: has been recorded, has been verified, has been acknowledged by the vendor, can reasonably be assumed in an operational setting to apply, etc. The nuanced differences in these relationships is important. The authority who sets these relationships is going to be critical.

CCI Business Rules

A CCI must meet certain criteria to be considered a valid CCI.

- **Single requirement** – The CCI represents a single capability that was decomposed from the source policy document.
- **Actionable –** The CCI represents an action that can be taken against the system or an organizational policy.
- **Measurable** – The action that the CCI is describing will be something that can be determined or measured.

A DoD Use Case

DISA has the notion of a Security Requirements Guide. Not a product level, but more abstract, it specifies a technology, e.g. an operating system. The you can drill down to which operating system, then drill down further for which version, etc. CCI is the glue which then maps these Security

Requirements Guides into the policy document, e.g. 800-53v3. DISA plans to begin development with this technology very shortly and the plan is to fully utilize SCAP.

CCI Way Ahead

Current:

- DISA FSO will establish the initial CCI List based on NIST SP 800-53 v3.
- The list will then be distributed to the Consensus Group for review and comment.
- Utilize a comment tracking matrix to capture all reported actions to the consensus group that will track the status of the actions through completion.
- An updated matrix will be distributed on a monthly basis in an effort to provide a status of outstanding issues.
- Additions to the CCI List can be submitted to DISA FSO for inclusion to the CCI List. Submissions should be sent to cci@disa.mil.
- It is anticipated that the CCI List will initially be published on a semi-annual basis.
- DISA FSO will work with source policy owners and authoritative sources to validate CCI item and reference mappings. When CCI reference mappings are validated, DISA FSO will update the CCI list with the *CCIRefMapValidatedBy* and *CCIRefMapValidatedByDate* data elements.

Future

- As the CCI process and specification matures, DISA FSO will consider alternative options for the management and moderation of the CCI List.

The Windows XP FDCC Content was analyzed:

- To document the mappings of the Checks to the Source Policy Document, NIST SP 800-53
- To provide a sample compliance report for compliance to the following IA Controls:
  - AC-1 thru AC-21
  - AT-1 thru AT-5
  - AU-1 thru AU-12
  - **IA-5** thru IA-6
- To document the mapping of the checks to sample CCIs developed from the same controls mentioned above
- To provide a sample compliance report for compliance to the draft CCIs
- Compare the compliance results with and without CCI

After performing a Windows XP FDCC CCI analysis, DISA found that CCI enables factual representation of compliance results and that CCI helps scope content development and completeness.

A questioner asked that, given the lack of automation support for CCI harvesting out of policy, is that also a feedback loop to the practicality of policy from a supportable collection and appliance checking …

if it's all manual then you'll never be able to get it in a timely manner. Does this really help in risk reduction? Yes, but DoD would still be interested in the mapping CCI provides.

Where do you see the CCE to CCI mapping going? DISA stated that if vendors create CCEs, then they will reference them in their CCIs. Is the vision to have CCIs in XCCDF or not? DISA will be putting CCI references in the XCCDF they generate. NIST suggested that XCCDF reference CCEs, and then the CCEs can reference other identifiers, such as CCIs.

SCAP Enterprise and Asset Reporting Gap Analysis

Enterprise Reporting Concepts

The concept is that vendors make tools that do assessments. What the vendors do within the tools to collect the assessments is their problem, but that data belongs to the enterprise and so should be available to the enterprise in the format that it wants and with the fields that it wants. Presentation uses DoD as an example use case for network assessments, but DoD is not very unique as an enterprise with regards to networks and IT.

A typical U-shaped area is defined. Network assessment policies are pushed from a high-level policy server down to the asset level. At that level you have vendor tools to perform the actual assessments (both SCAP and non-SCAP). Results of running that content should be fused across all tools and propagated back up. That information can be used to track compliance posture, inventory, license state, to help make remediation decisions, and in general to help systems administrators and managers at all levels of oversight to manage their infrastructure. Since not all tools are SCAP, SCAP content alone does not solve the problem. A standard format, however, makes data actionable and useful (since it can be correlated).

Simulation concept is about characterizing a network using C*E and describing the potential attack patterns. Defense in depth evaluation can provide probabilities for how successful attacks will be (Lincoln Labs is working on this). Any simulations depend on having good data.

Enterprise reporting is about packaging data to give appropriate levels of data to each level in the management chain. The lowest level system administrator needs all the data, down to individual patch settings. As you go up the chain however, less and less information is required to make effective decisions. Information is tailored to the level of situational awareness that you actually need. Information is also tailored based on the characteristics of the device itself (you don't want web server logs from a firewall) and the role of the person looking at it. Devices need to be placed in both operational and technical contexts.

DoD architecture is used as an example. Data interfaces have to be defined, taking into account network infrastructure, policy, authorization, bandwidth, etc. Having defined interfaces allowed DoD to contract out specific pieces of their enterprise to different groups since defined interfaces ensure interoperability. DoD's goal is to build out an interoperable infrastructure based on SCAP. This architecture looks very much like other enterprises, however, so interfaces should be standardized allowing for policy differences.

There is a concern about reliance on assessment content. Who is creating the content, and who is paying for it? It's pointed out that that content exists right now in a variety of proprietary formats; standardization will reduce the content burden, not increase it. Analogy is made with the web: there is paid content and free content, but as long as the format is standard everyone can access it.

Question about standardization ONLY on schema, or on APIs as well? There is a general agreement that it's both, but before you can standardize API or interface you need to standardize the aggregation model. 4 million XCCDF results files is not realistic for enterprise reporting.

Question about using SNMP: operational sets from SNMP could be reused rather than reinventing the wheel. This would enable integration down to the lowest level implementations, but at the same time you probably want a gateway to a less lightweight format (XML) that's easier to manage. SNMP is network ops, so good for collection but maybe not for reporting.

Assessment Results Format

ARF is intended to be a single format for sharing network assessments. Includes:

- OVAL/XCCDF results, since SCAP tools are producing them
- Other information generic assessments usually collect: device lists, operating systems, patch levels, software inventory, ports and protocols

ARF is organized at the top level by a device model, then each device by a set of CPE inventory records (software installs, basically). Settings, patches, vulnerabilities, services, and exposed ports/protocols are attached to that inventory record as opposed to the device. This is different than CRF, which is just a list of findings.

Entity identifiers, which are made up of a namespace and unique ID, are used heavily to aid in replication. When creating a new entity, a new ID should be chosen. Then, when that entity is subsequently referred to (potentially a delete or modify) that same ID can be used to point to it.

Discussion moves to the network model, which is made up of commonly known network information. This does not include router ACLs or firewall rules, since there is a lot of logic to describe them and wasn't very useful. It's intended for a future version, however, and since ARF is extensible it's possible to add to it. The future concept is that ARF will be ARRL: the assessment results reporting language. It's very focused on devices now, but policies, facilities, etc also need to be assessed.

Current version of ARF does not include hardware information, but since implementations discover it they're considering adding it back. ARF includes unrestricted tagged value pairs (name and value), and tools were populating those with hardware information.

ARF vision is to be able to buy enterprise-reporting tools commercially (network assessment devices, asset managers, SIMs, etc) and have them be able to speak ARF to exchange information. Asset manager gives local sysadmin the ability to manage all of his devices by organization, geographic area, etc.

Correlation is discussed. Information coming from several different sources but talking about the same asset needs to be deconflicted. Information about open ports is probably more accurate coming from an

external network scanner, but information about software inventory is probably more accurate from a host-based tool. ARF/SCAP allow three different pictures of the network to be combined into a single (hopefully more accurate) view. ARF facilitates this by including confidence levels and timestamps.

CPE Record is made up of 1+ instances, ports and protocols, running services, vulnerabilities (CVEs), configurations (CCEs), patches, and scan data (content, scanner, etc). Generic function, role, and family are also included to help describe higher-level constructs. OVAL and XCCDF results/system characteristics can also be included, and the idea is that if you're passed XCCDF/OVAL you should return the results.

Operational attributes include device descriptors that can't be assessed. ARF includes both generic and DoD-specific fields. For example, fields include DoD Network and Network, with DoD Network being an enumeration of possible DoD networks. This is how most OA fields were treated. ARF could also be pluggable, with a namespace for DoD and commercial operational attributes.

Comments on ARF should be sent to LTC Wolfkiel, version 1.0 will be released in 6-8 months (from June 11, 2009). The version being demonstrated is 0.41. A reference implementation of an ARF consumer is currently being built.

Results from ARF could easily be fed into a SIM (security information manager) tool, especially given correlation functions between vulnerabilities and asset descriptions. If ARF succeeds, however, it should be able to inform tools outside of SIMs or network tools, such as business intelligence tools. XBRL (Extensible Business Reporting Language) could be used on a large scale to aggregate it.

Is ARF lightweight enough? Names and structure are very built-out in order to be unambiguous, but it adds both runtime and development time costs. The idea is to create something that's usable for everyone, not just for DoD. Increasing modularity, especially in operational attributes, would mitigate some of this. Based on converting existing tools within DoD to use ARF, ARF was never completely filled out but it wasn't too bad. Since in ARF almost everything is optional even though the tool couldn't populate it, it didn't hurt.

Leaving things optional to start off, in order to keep implementation cost down, is ideal. Things can be closed down later.

Summary Results

ARF Summary Results should provide information about findings on multiple assets by count or list rather than detailed results by asset. For example, "for each CPE that meets this pattern, I'll give you the count". Or, "for each CCE parameter value, here's the list of assets that have that value set". Allows you to get tailored results for a single finding across multiple assets, while ARF allows you to get tailored results for many findings across a single asset. General consensus is that ARF summary results, as a first shot and on first glance, is on the right track.

Summary Results is reusing many of the ARF schemas, and is really just another way of looking at the same data. So although ARF is released and Summary Results is not, they need to refer to each other. They can still evolve separately (fast-track one but slow-play the other) but need to be related.

Brought up that there are some common constructs in ARF, Summary Results, and PLARR. Back in the day there was a notion of an "SCAP Commons" schema that all of the protocols and schemas could leverage. There needs to be a graceful way to do it, because if they're tied too closely together it will stifle innovation. Asset device model is used and defined separately by ARF, XCCDF, and OVAL. The more tightly things are knit together the worse the changes in one thing affect another, but having a "lego blocks" SCAP model would make development and integration easier.

Use case is discussed: new vulnerability of flash comes out, query is "what computers have Windows XP and Flash installed". Another use cases is separating out results by region (example of an operational attribute).

Question is asked about whether ARF reports are classified, since they're bringing vulnerability off a host. Answer is that other than having it in a standard format, there should be no additional risk. No one in the DoD has said it should be either classified or not, so the assumption is not. Guidance is inconsistent.

Policy Language for Assessment Results Reporting

Idea is a way of telling network assessors what information they should put in an ARF through an automated format. PLARR should allow a system administrator tailor his system results/ARF results.

Set of use cases range from reporting and aggregation (am I vulnerable to this CVE?), SIM aggregation (pulling in different feeds from different tools), and compliance reporting.

First section of PLARR is defining what you're looking for. You can request check content (XCCDF and OVAL directly) or enumeration content (a list of C*Es) and include options like due dates and freshness dates. Second section is defining which assets you want to scan. Uses entity identifier and other asset fields from ARF to select assets. Last section is how you want the results returned.

One example is to scan a subnet for a vulnerability (CVE). A filter is used to only return assets that have the vulnerability. A second example shows how to send SCAP (XCCDF) content and get back SCAP results.

One question is what is the scope of PLARR? Does it include transport specifications or is it just an XML schema. One option is to accept an XML document at a given endpoint, but that doesn't handle asynchronous results very well. Other option is to include transport in the specification but not in the XML schema. Use existing facilities at the web service or transport level, such as WS-Eventing.

Question about architecture: is PLARR asking a data source for information or asking targeted systems? Probably a data source, some possible PLARR implementers are an ARF data repository or a network scanner. Another issue is how to handle errors. Is there an ARF error, PLARR error, or do you rely on

transport to send errors back (e.g. 500 error from HTTP). Relying on transport is easier but can cause inconsistencies, including an error response type would add complexity but remove ambiguity.

Question is asked about why PLARR is specifying both application layer and session layer, and what would you do in the case of an air-gap network. The idea is to specify 3 or 4 common ways but don't limit PLARR to only those options. If the standard doesn't specify session information tools will have no way of talking. It's pointed out that we're seeing more and more command and control requests in general, so while PLARR might need to specify it now to get things moving but in the future the problem is much larger than PLARR.

At a more conceptual level, PLARR supports a wide variety of requests. Suggestion is made the PLARR should allow you to cherry pick SCAP results: thin vs. full results, include extended definitions, only include "vulnerability" result types. ARF and PLARR aggregations also work on SCAP results, so that might cover the same use case.

Question about whether PLARR is simply a request format or is a command and control format. Does it just ask for results, or does it tell scanners to run this OVAL content? PLARR is explicitly avoiding the command and control question by saying "I need these results" as opposed to "run this scan and give me the results". It may end up coming down to scanning in order to meet the requirement, however. More risk in building a control language vs. a request for information.

Question is asked about whether PLARR supports specifying the type or extent of results. Answer is that aggregations and ARF summary results support it, but there may be other ways. Another question is whether assessment tools can report all their data back on the fly to a central repository and run queries on that local cache rather than remote tools. That architecture decision might not require PLARR, depending on whether it's internal to a tool. Periodicity in PLARR, however, could allow for this type of pre-staging information. One downside of aggregations is that the collecting tool needs to know how to aggregate ahead of times, which requires extra communication. Just returning vanilla results doesn't require aggregation instructions so can be one-way.

Another question is whether PLARR and ARF need to be so closely tied. Do you want to send XCCDF/OVAL with PLARR and have vanilla XCCDF/OVAL results returned external to ARF? External XCCDF, OVAL, CND/SCAP Core, and ARF schemas are also reused in PLARR. NIST suggests not limiting PLARR to ARF or even to assessment results reporting. Give the community time to look at PLARR external to ARF.

Another point to consider is currency of information: especially in aggregations you can combine results that were assessed a week ago with things that were assessed yesterday. ARF return could include a time period, but you probably can't break it out per-device.

In the case of a freshness stamp, you need to define what happens if the data doesn't meet the criteria. Does it rescan immediately, does it wait until fresh results come in, or does it respond with an error? Point is brought up that query languages vs. pushing out information as it arrives or forcing a rescan: it's

much easier to just implement a request service vs. tying into operational requirement about when to transmit things.

BOF - NECAP

The session began with a general inquiry of those who have heard of the NECAP effort. George Saylor introduced the agenda of the BoF session. NECAP is about the network configuration and event management space. The purpose is to figure out what do we need that doesn't already exist to capture events, what an event is, and what to do with the events. Looking for feedback for what works and doesn't work for what is being planned. This is not a new effort yet, this will be a feasibility study – not something vendors need to worry about implementing yet.

This is not intended to be a security focused standard, but many of the use cases are security related.

There is work already going on within MITRE on the Common Event Expression (CEE) effort that may be usable within NECAP, dealing with logging.

The criteria for this feasibility study is several-fold. One is to determine the achievability of the goals, of the project. What will the value be to the community? What interaction can there be with existing protocols (such as SCAP) and with components such as CAPEC (Common Attack Pattern Enumeration and Classification) for attack patterns? Can the events that are captured allow an end-to-end traceability to chain together the events across multiple systems to get a picture of what happened during an attack? A goal will be to apply lessons learned from SCAP, so as not to not repeat any mistakes that were discovered in the original CVE/OVAL/etc. projects.

A question was asked about the word Metrology, which is the science of measurement. NIST even has an entire Metrology department.

Bill Heinbockel will be releasing a draft of the CEE documentation for Blackhat. Rather than a 'kitchen sink' dump of information in syslog, providing guidance about how to put better quality information along with name/value pair information to assist in getting better log data.

A goal of being able to extend CAPEC such that it would include information about what an attack would look like, such that a red team could describe "this is what we did" and a blue team could verify, "yes, that is what it looked like". This can help take the attack information from being 'magic' to more of a 'science'. A reach goal of CAPEC would be to describe 'what does this part of this attack look like to some kind of sensor'. This would allow SIMS and such to recognize an entire attack.

No real threat enumeration type of information available at this time. It may provide value for situational awareness, but not a lot of information sharing in this area at this time. There are schemas, but not a lot of data to go into it. A question was asked about familiarity with "iSight"(sp?), they've been doing a lot of work to create an enumeration, expression, and measurement of threats. They're commercial, but an offer was made to make an introduction for George to the company.

Someone else also mentioned an effort to track threat patterns. What CPEs do they target, what CVE's or CWEs do they leverage, etc. In a neutral non-methods and means way, in English. George agreed it would be good to know more about who is out to get you in general threat terms.

George outlined potential components such as reporting and a method to query. Here is a way to correlate data and create a standard report and get information back in a standard way.

Query languages have an inherent difficulty in dealing with a distributed environment. George referred to an earlier discussion around difficulty of getting information from nmap to a higher reporting tool. If two different organizations have completely different processes and procedures, there's no real way for anyone else to repeat their results reliably, making it non-scientific.

Policy would be a very difficult area to tackle. What kind of decisions can be made about what you log, how long you keep it, what filters you can or can't apply, etc? The problem is that storing all of the network event information is huge, and will be a storage issue. Some places are only able to keep a day of information or less, limiting the time available to make decisions based on the data. The ability to make decisions fast based on good information is critical.

It was mentioned that if NECAP follows the SCAP pattern, this kind of policy would fall outside of scope. Policy seems to be distinct from these specifications. George agrees that he felt it was outside the scope for NECAP.

An example use case: if you see an alert for an Apache web server and know you're patched for it or are not running an Apache server, allows a consumer to risk rate the event. Allows for prioritization of resources to higher risk attacks. By bridging between NECAP events and SCAP data, would allow for determination in tools. Allows for elimination of some potential false positives.

By the end of this fiscal year, a feasibility study will be released for NECAP. Many industry experts and stakeholders are working with DoD/NIST/MITRE people on aspects of the feasibility study. Looking at ARF for help in what is an asset, how do you define that asset if it reports an event. Do we need to understand this information as it was (considering DHCP and DNS changes) or only as it is today?

SNMP information like CPU utilization and disk utilization are examples of information and network management event information that are not really security related, but will fall under the general scope of NECAP. Again, initial work will more focus on security event issues.

Something not raised in the slides: on the software weaknesses, there are weaknesses that don't have CVEs. These weaknesses may be against a product or other, but may still have event signatures. Some events being dealt with may need to have a 'reliability' or confidence level based on things such as distances from the source for the information. If we recognize an attack pattern that is not related to a CVE, there may be a CWE or CAPEC signature that will describe this type attack. Not all applications (such as small or proprietary pieces of software) will ever have CVEs associated with them.

Another issue brought up the difficulty of trying to lock down a schema first and add data later. The benefit of "triple stores" was mentioned as being potentially useful for this use case. George confirmed

that semantic methods are indeed being looked at. It was also raised that some thought should be given to importing data from synthetic models. Thought will be given to being able to run simulations and see if data could lead to potential attack signatures.

The larger challenge will be on the large log producer side. There can be issues where we cannot have something like a firewall needing to do a lookup for every event being tracked. The consumer side may need to process and bring in additional data that would be prohibitive for the producer. Potentially technology may catch up with this issue and solve the problem in the future.

George stressed this is not yet a program. This is just a research project at this time. The 'NECAP' name is being used for this effort just to look at the feasibility study. The study should be available at the end of this (2009) fiscal year. The subject is very complex and large because of just how much data and how many events can exist.

Question was raised that for NECAP (or whatever it eventually becomes), what is it that NECAP will automate? Speculation on answers said attack response, remediation, threat capture, increase and decrease in logging verbosity, ability to provide information for network access decisions. It was raised that there are techniques used in financial areas and others that do complex event processing to make determinations. However, these are generally proprietary and specific to an industry. George commented that there are some open source efforts in this area, and there may be some key ideas (Open GIS (sp?)) that may be able to leverage some of this data.

It was raised that at the core, it's really about logging standard events in a commonly agreed upon and standard way, with standard information.

In some cases, security operations are often looking at the same events over and over from day to day, for the same kind of attacks. This will help lend to automated responses to this sort of event. It was raised that we could automate assessment of events related to specific incidents more easily, than from spreadsheets and other locations by hand.

It was questioned about what the vision for NECAP on the original NIST 'future state' slide that was presented at the beginning of the conference. A lot of this is about situational awareness and interoperability. There is a lot of room for innovation in this space, more than is currently available today.

Question was raised what tools are currently doing or allowing in the market today for events. Someone mentioned a company called Agent Logic (?) that deals with just this type of complex event analysis from disparate sources. Another vendor product was also named that performs in this space, and it is being attempted to bring them into the conversations around NECAP. George raised that a lot of companies have spent a lot of money on SIEMs and still don't fully trust them, and do a lot of work by hand outside the SIEM. This will be an issue to help build awareness around the need to build confidence in the tools and what they're reporting.

It was mentioned that this effort is just at its beginning, similar to CVE and such were many years ago. Important to grasp where we want to go and start moving things in the right direction in how we deal with events and the key issues, and allow for the ongoing improvements to situational awareness and more as we go. We should hopefully be able to learn from the existing processes, and it shouldn't take us 10 years to make the same improvements we've seen in that space.

George provided himself as a contact for any questions, issues, or discussions on NECAP. He can be contacted at george.saylor@g2-inc.com.

Friday June 12

SCAP Assessment / Collection Gap Analysis

Assessment/Collection Gap Analysis

This session is about guidance authoring, exploring requirements for gathering and evaluating information, and determining unmet needs or gaps. Two new efforts towards meeting some of the gaps are introduced – Open Checklist Interactive Language (OCIL) and Open Checklist Reporting Language (OCRL). Discussion starts with OCIL and then, OCRL.

Open Checklist Interactive Language (OCIL)

Introduction

OCIL is a language for conveying checks that can only be done manually, and requires user feedback. It has constructs for expressing a check with a set of questions, possible user responses, and evaluating them. User responses can be Boolean (yes/no or true/false), text, numeric, or a choice from a list. Similar to OVAL, results can be pass/fail, or any of the exceptional cases such as error, not applicable, unknown, or not evaluated.

A team at MITRE has been working on writing up security benchmarks. They find OCIL useful in cases where they know where and how to collect the information to accomplish a check, but cannot programmatically evaluate it. OCIL has constructs for representing instructions or a series of steps in guiding the user on how to go about answering a particular question.

A chaining structure is built into OCIL. Given an answer to a particular question, it can either evaluate to pass/fail, any of the exceptional cases, or a reference to a follow-up question. With follow-up questions, one can create an ordered or sequence of questions that can branch out into multiple paths. This allows authors to build sophisticated questionnaire structures.

Finally, OCIL contains a structure to store and record results.

Structure of OCIL

The top-level structure in OCIL is the Questionnaire. It contains metadata (e.g. title and short description), and an actions object. Actions contain a set of references to test actions. A test action

object includes a reference to a question, and a handler. The handler determines what happens when a question is answered. For instance, does it return a result of pass/fail, an exceptional case (i.e. error, unknown, etc.), or move on to a follow-up question?

A short example was presented. Questionnaire is to "Ensure that the latest version of the CIS Windows XP Guidance has been applied." It contains a single question – "has the CIS Windows XP Guidance been applied?" The handler contains the following: ON_YES return PASS, ON_NO return FAIL, ELSE return an ERROR. Alternatively, a handler can return a reference to a follow-up question on ELSE.

OCIL Use Case

There have been a significant number of recommendations that cannot be automatically tested. OCIL came about in hopes to fill in this gap. For instance, those information that needs to be physically checked (e.g. is the server room locked?). In some cases, it is important to know what the user knows or to assess the user himself (e.g. do you lock your safe at the end of every day?). Or perhaps, the guidance is poorly worded, too abstract, or subject to interpretation (e.g. are all unnecessary services disabled?).

OCIL supports simple responses and returns results compatible with XCCDF scoring. The Questionnaire object is the atomic unit within OCIL that can be referenced from XCCDF Rules.

Discussion on OCIL Use Cases

Question #1: Is there a community need for OCIL? There had been a lot of positive responses on OCIL such as 'this is something we need in order to do what we need to do'. But there are also some arguments expressing that 'this is no longer automation since it requires human interaction for feedback'.

The first set of responses is about different use cases for OCIL –

Response #1.1: OCIL is introduced to the chemical facility for some agency within DHS. The response was very positive ('they absolutely loved it'). It is because 90% of their new guidelines cannot be done on a computer system. A lot of the guidelines are on computer operations, but a significant part of it deals with environmental, physical, chemical, etc. It includes questions such as – have you checked the door? Is the computer in a locked room? How do you know it's in a locked room? For all you know, they could just be propping the door with a chair. These types of questions go right next to the top-level question – is the data center secured? In addition, one could get into policies and procedures asking questions such as did you read this policy? Does the policy contain X? There is no way to get to that information automatically.

Response #1.2: In some cases, the user may need to be interviewed to gather some information. Collected information is used as input to a technical trek. Example questions given – what are the appropriate shares that should be in the system? Who is the backup user? What should the user rights be? These are pretty common across the board.

Response #1.3: A frequent question is about comparing user lists. Do these users have need to know capabilities? There's no way that the computer would know the answer to that question.

Discussion continued on to the need for integration between interactive and technical checking systems. At this point, there is no support for this within OCIL because the issue spans across checking systems. The authors deemed that this feature is more appropriate for XCCDF than an individual checking system. It was proposed that an if-then-else structure be added into XCCDF where results from one checking system may be used to populate variable inputs to other checking systems.

Response #1.4: If there are many checking systems called by XCCDF, then it would be much more useful to be able to pass information between checking systems.

Response #1.5: Representing policy content is difficult because it contains a combination of administrative and technical checking. Unless there's a way to integrate them, it will continue to be a challenge. If the goal is to show full compliance, then integration of these two types of checking is a must.

Response #1.6: As we move along, we would need something to 'orchestrate' all the checking systems. We basically use XCCDF for this purpose, as much as it allows us to do so. But it definitely has some gaps that could be filled with some kind of an if-then-else structure.

The discussion continued with the concern about scalability of automating the checking process within an enterprise, as interactive checks mix in with automated checks. One argued that it definitely changes the workflow but it is necessary to do so. The example use case is determining shares. Without being able to harvest that information, the automated checks are useless.

There was a feeling that throwing interactive checks with automated checks takes away the power of automation because one needs to do something manually. But it is deemed to be used as a last resort when there's no other way to do a check programmatically. One argues that this is automation, only geared towards the user.

Interactive checks may have a different scope. Perhaps, some interactive checks go before, automated ones. It is suggested that there needs to be some scoping to allow that to happen. There could also be a difference in frequency in checks. For instance, interactive checks may be done on a quarterly basis, while automated checks may be done on a weekly basis.

A question was raised asking for more information about the proposed if-then-else clause for XCCDF Rules. An example was given to answer the question. A single XCCDF Rule will have an if-then-else clause. The first check will contain an OCIL questionnaire checking whether a certain policy on passwords is met. If it passes, then it's done. Otherwise, moves on to another check.

There was a concern raised on whether and how this would appear to the right person.

Another question was raised about expressing that one system is less secure than others. The reply was XCCDF scoring. If the scope of an OVAL check is an instance of an application or a machine, what is the

scope of an OCIL check? Is it still limited to a machine? Those types of questions typically relate to abstractions such as people, organizations, and processes. An OVAL check is able to present results to the user as evidence.

A proposal was made about defining scope as the frequency of performing OCIL checks. Package OCIL checks by scope and somehow tie them back up to an asset after.

There was a comment about not using OCIL in asset benchmark. It is good for performing technical checks that cannot be automatically done for some reason. But it shouldn't be used for non-technical checks. It should be set at a much higher level separate from OCIL.

In response to this, one mentioned that in DISA, almost 30% of the checks of any benchmark includes interview, observe, or examine checks. A third per asset benchmark is non-technical check.

There was also a concern about certifying systems without a human ever touching them. So, this is something that is desirable especially for auditors.

No matter who or what answers the question, the data still needs to be structured and used in an interoperable way.

It was pointed out that XCCDF benchmark was not just a technical benchmark. The vision was to provide guidance authors to include everything into one place – prose, call outs for whatever need to be called out, whether it is automatic or interactive check. The focus here is an architectural decision that has to be made. We still have a major portion of any policy in a commercial standpoint, to bring all these things together.

Another comment was made about focusing on where we need to go and using what we already have at our disposal. There's nothing that says we can't use XCCDF for writing up organizational policies and non-technical checks. It doesn't mean that XCCDF is only for a single machine. One can write a benchmark in XCCDF that spans through multiple systems.

As an example, at MITRE, we have a tool that updates our asset database by asking users what is the bar code of your machine, where are you located, etc. This information cannot be harvested from the machine itself.

Another point was made that not all assets are material. People are assets too. There needs to be a model of distributing questions down to systems.

A participant tried to summarize what has been discussed. Basically, there are all sorts of questions we would like to ask to certain people. Some of them will span across multiple assets and others, on a single one. This is probably something that needs to be fixed or made clear in XCCDF. But, you probably need something in OCIL to indicate the target users of the questionnaires whether it may be a machine or an actual person. Is this something that XCCDF needs to be able to do? But this sounds like a lot of work for XCCDF.

There was a proposal to gather up and sit down to figure out the architecture on how all these emerging standards fit together. We need to describe exactly what our model, if it exists, is on how to make sense of all the benchmarks.

Artifacts

The ability to include artifacts was one of the first features requested in OCIL, i.e., being able to attach a file supporting a particular answer. By nature, OVAL has artifacts built into it. OCIL doesn't currently have this ability. There was some consensus that this feature is needed.

An issue was raised about the changing nature of artifacts. How does one keep track of changes or preserve them? Is it going to be by value or by reference?

Are there timestamps? If we're asking users about a certain question, then it is desirable to know when it was answered. The response – Yes, timestamps are supported in OCIL.

There was a request to add the provider (i.e. author or custodian) of the artifact. Also, add the ability to add some prose.

What is the behavior of the artifact? The artifacts are either physically embedded or have links. It is associated with an OCIL results file. The question is the roll up. How does one interpret the results back to the caller? Up to what layer does it get included with the results?

Can you just include it as a question? Fill in who you are. Who's answering this question? Who did you get the information from?

Can you have the answers in that questionnaire impact other questionnaires from another OCIL document? The response – there is no cross-referencing between multiple files. But within a file, yes, questionnaires can impact other questionnaires.

It was also brought up that there is a demand for artifact gathering. Having questionnaires bring back common artifacts may also be useful.

Another point was made about making sure we look into XCCDF as well. There is some dependency here where artifacts get returned as results to XCCDF. We need to make sure that this is actually possible. A participant mentioned that it is possible to return informational results back into XCCDF.

One of the participants commented that people in organizations have "roles". There are times where you might want to interview more than one person. Are we going to have the ability to do that? Will we be able to return multiple results? And aggregate these results in some way? So, the result becomes a set of PASS/FAIL, instead of a single value.

Variables

It was proposed that variables be supported in OCIL. The questions are – what exactly do we want to do with variables? What would be the behavior? Where will it be used? An example would be a choice question. Variable inputs can specify which set of choices is applicable to a particular environment. Another set of choices may be applicable in other environments.

One of the participants suggested that we consider OVAL as an analog in answering these questions. A use case would be for tailoring policies as described earlier. Are there other use cases?

There was also a proposal that variable information be passed to certain people or targeted at some set of users.

Another use case was presented. What's the role of a server? Is it really performing the role of a server? This is something that can be asked to a person to check.

There was also a request for variable import and export.

It was brought up that we need to find a way to make sense of all these benchmarks. Perhaps, it would be helpful if we have a component model in an abstract level.

The concept of iteration was brought up. Notionally, we are already doing this in XCCDF.

Unevaluated Collection

There was a proposal to include support for collecting artifacts without evaluating them in OCIL. No objections with this idea.

OCIL Profiles

It was proposed that OCIL provides support for tuning questions and handler structures similar to an XCCDF Profile. There are two use cases. First use case is allowing OCIL to be used in a standalone manner without XCCDF. Second use case is allowing results from other questionnaires to affect question tree structures.

Response: It is a horrible idea. Obviously, we want a layered approach in dealing with various checking languages. XCCDF already has a tailoring capability. We don't want to have every checking language re-implement the same capabilities in other languages. We can take lessons learned from OVAL. It has not found any use for including profiles so far. It is unclear what drives these use cases.

It was proposed that external variable passing be supported as an alternative way. We don't want to add any unnecessary additional complexity to the language. If you have variables, then you can accomplish the same feature.

Where to use variables in OCIL is still unclear. It was proposed that this issue be discussed over a mailing list.

Another point was made about composability of existing specifications. It is important to allow variable passing between benchmarks. This issue is not necessarily for OCIL but in a general sense.

The current intent for OCIL is to get it into a position so that it can be included in the next revision of SCAP. It needs to meet all the requirements such as a draft of the specification, content, and reference implementation. There was a plea to the community to take a look at OCIL. Is it OK to run with OCIL in its light-weight version today? Is it going to address current issues? Is it OK to include OCIL in SCAP even

without variables, artifacts, and other features discussed today? Is OCIL going to get us two or three steps further in security automation?

There was a proposal about talking about OCIL outside of MITRE, perhaps, through a mailing list of some sort. Today, OCIL doesn't have its dedicated mailing list. It would be useful to have one as we move further with OCIL and discuss the issues. We also need to start thinking about the big picture, i.e. the architecture, and how OCIL fit into this model?

It was also asked, for those who have experience working with OCIL, to talk about their experience and whether the current version of OCIL is up for consideration into SCAP. In addition, it would be great if they could talk about the features that they cannot live without. One of the participants described that their experience with OCIL is positive. However, they cannot live without variables. They can hack their way through without the artifacts but definitely, need variable passing. In addition, the need for persistent ids for questionnaires was also brought up.

Open Checklist Reporting Language (OCRL)

Introduction

OCRL is a proof-of-concept for a very specific need that MITRE found. The question is whether the community finds any use cases for it. It is a language similar to OVAL and OCIL as a checking system. However, it does not include support for performing checks. It only has structures for collecting data and formatting them.

Suppose there are two data sets with some sort of a relationship. OVAL can evaluate the sets but cannot compute any cross correlations between them. OCRL supports the ability to cross correlate sets of information. Unlike OCIL, data for OCRL exists within the system, but its evaluation is not possible. Thus, a report is only a state of the system, and not a final evaluation.

The idea is a check would create an OCRL Report and followed it up with an OCIL Questionnaire. Originally, OCRL includes constructs for representing questions. Since OCIL already supports structures for questions, it was taken out.

OCIL Report Definition Structure

A single OCRL document can have multiple report definitions. A report definition is composed of two parts – data sources and reporting blocks. The data sources section contains a set of data sources. It deals with the extraction of information. A data source can be any of the following types – WMI, File, Executable (scripts), and OVAL. The reporting section deals with organizing data collected. It iterates over all the data sources and allows for expressing cross correlations between them.

OCRL Example Document

Consider the following recommendation: *limit the number of groups to which an individual account belongs*. An OCRL Report is created by extracting two data sets – A = {usernames}, B = {group names}. Each set is taken from a different WMI data source. For each username in A, a list of group names in B is printed out to produce an OCRL Report. The user is instructed to review the report and decide whether changes need to be made in order to be compliant.

In this example, it shows that OCRL is able to perform cross correlation between data sources. It uses an internal data model agnostic of the nature of data sources.

Q & A Session 1

Q: Does it normalize the fields of data sources by type?

A: No. OCRL treats everything as strings. This approach is probably something that needs to be examined and improved in the future.

Q: What is the reasoning behind choosing the four different types of sources (WMI, File, etc.) as opposed to the OVAL equivalence?

A: There is no OVAL equivalence. It was decided to include these four types of sources because OVAL can partly perform extractions from them. OVAL can handle very specific field extractions but falls short in more complex type of extractions. In addition, it cannot do any cross correlations.

Proposal: If existing specifications provide capabilities that can be reused, then leverage them as much as and whenever it is possible.

OCRL Moderator: Agreed. If OVAL has certain capabilities that can be reused, then it should definitely be utilized.

Q: What is the level of effort to include these capabilities into OVAL rather than creating a brand new specification or language?

A: The current state of OCRL definitely needs more work. It is not as mature as OCIL. This is more appropriately answered by OVAL moderators or project leads.

Comment: This is a feature that has been requested in OVAL for years. We are now talking about building a separate language.

OVAL moderator: At the OVAL session, this was brought up in the last slide as an emerging use case. Two questions – should it be supported within OVAL? or is this a distraction? It was pointed out that the requested capabilities are already supported in OCRL.

OCRL moderator: Two part question – is this a use case? Where do we put OCRL (i.e. with OVAL or separately)?

Response: This is definitely a use case that has been asked by our customers repeatedly. We have a separate proprietary tool to meet this need. It is a requirement for us (both extraction with cross referencing and presentation of data).

Response: It is not a foreign concept to OVAL. First phase is collecting system characteristics. Second phase is performing evaluation. Tools such as ovaldi already implement this capability. It hasn't been officially specified under OVAL that it should be done and may be able to do this process. It is a good analog to what OCRL is trying to do here.

OVAL moderator: There are people already doing this. Basically, they are generating system characteristics files similar to a report, and later, performing evaluation. There is a break between the collection and evaluation mechanisms.

OCRL moderator: Basically, this is what OCRL is doing for an OVAL data source. The ovaldi is used to create systems characteristics report, ingest it the internal data model, organize the collected data, and present it on a report. OCRL has additional capabilities to support other data sources that OVAL does not handle.

Use Cases

A short read out of the use case slide. Use cases include collecting information useful only within the context that requires cross referencing between data sources; need to evaluate human knowledge. OCRL is set up so that it can be called out from XCCDF. It returns informational results because it doesn't actually evaluate.

This use cases are taken from the SharePoint and Tivoli guides that MITRE has been working on: (a) use an OCIL question, (b) run the OVAL interpreter to produce system characteristics file, (c) apply stylesheet to present a report on a browser, and (d) answer the OCIL question.

Q & A Session 2

Q: Would this be equivalent to doing SQL queries (or queries made to a relational model) across tables? Gather the information and bring them back to the user. Basically, you have data collection and querying mechanisms to support the decision process.

A: Yes.

There was a consensus about the need for the capabilities that OCRL supports – data extraction with cross referencing and presentation of results. One had already built a proprietary solution to support this capability. The main issue within the discussion was whether or not to include OCRL capabilities into OVAL.

One voted that it should be a component to OVAL – both extraction and formatting. There was a disagreement about including formatting into OVAL. However, this was due to the fact that formatting is a poor choice of word. A better word for the capability is cross referencing. With that clarification, there was an agreement that cross referencing needs to be included in OVAL as well.

It was pointed out that there is a difference between what OVAL and OCRL can do. OCRL can actually send a query that spans machines and across domains, which OVAL cannot do. In addition, one can build a proprietary solution that performs querying and cross referencing but no standard exists for it.

A comment was made about the challenge of making a decision on where to place OCRL. It is a tough balance. We don't want a proliferation of specifications. But, we don't want a single specification be the 'swiss army knife' of security automation, either.

The OVAL moderator pointed out that if this capability is added into OVAL, then objects with no states or tests associated with them may exist. It doesn't really make sense to check the state of something we're trying to do a report on. In working with the SharePoint guide, there are cases where data are only collected and not tested. They end up having objects with states and tests they never use. An implication of this is we end up with free standing objects in OVAL. It is unclear about whether this is a negative or positive implication. But definitely, this is a change in OVAL system pattern.

The OCRL moderator pointed out that these things do not necessarily go into the OVAL repository. The OVAL moderator agreed. He pointed out that new structures will be added into the language. But, the language was not built around writing a framework for doing assertions about a machine's state.

Q: During the XCCDF session, there was a concern about building a huge monster of memory for an XCCDF interpreter. Is this a concern for OVAL as well? Would this be a higher hurdle for implementers to leap? Are there use cases where this capability is not required? Why would we add this to OVAL if this would be a huge hurdle for implementers and there are a lot of things we can do in OVAL that does not require this capability?

One voted that this should be a requirement in OVAL validation. In line with this, another commented that OVAL validation does not require everything to be implemented. To be validated, one needs to implement most but not all. One should be able to do everything they are being tested on. From an SCAP perspective, it's different.

The OVAL moderator commented that this capability would probably be an option similar to SCAP DTR. Probably, a set of test requirements will be built. But, leave it as an optional capability and not require it.

SCAP moderator: We're still in the decision process. Until we get the official word out, don't take these statements as the official word yet.

Another comment pointed out that the correlation capability should be there. Just being able to collect data is not at all that useful. Offload the correlation to post-processing. Build these capabilities in an incremental fashion.

It was also pointed out that this is really an architecture issue. It is definitely a need. How to put it in the spec? What would be required to that? Actually, make it fit to the architecture we're trying to do.

Another comment – this is definitely one of the areas where an OVAL spec would be useful to define these types of higher level use cases that aren't very easily defined in schema documentation.

Wrap-Up

Did we miss anything?

Are there topics that we should have talked about that we didn't? Architecture. CPE, we didn't talk about it because it's such a train wreck right now, but we probably need to at some point.

Should we have talked about CVSS? Yes, from an integration into other existing standards, and being able to use it, call it, and correlate it – yes, we should have talked about it.

Should we have talked about CCSS, what a CCE metric might look like? There is a draft spec available for CCSS. NIST would be glad to collect any comments relating to CVSS or CCSS. Should we have talked about CCSS? Perhaps, yes, but right now people are more focused on CCE.

One important action item for the entire community is to try to put some structure around the email lists – having regular telecoms on each of the standards. MITRE will certainly be working toward setting up such telecoms.

Does anything else need to be discussed?

More integration discussions need to be held, e.g. mini-sessions with demos to display the capabilities of the products.

DOE would like to talk about adoption issues in the future – implement, adopt, and promote it in his organization. NIST mentioned that another event that might help in these areas is the annual Security Automation Conference at NIST. DOE would be a good source for use cases in the future because of their confederated environment.

Another attendee would have liked to have heard more about over-arching use cases. Where will the larger use cases be posted, and where might lessons learned be found? A vendor stated that he would like to hear more about use cases form users. NIST sated that use cases should be sent to the SCAP discussion forum.

Steve Boczenowski pointed out MITRE's Making Security Measurable web site – measurablesecurity.mitre.org

The DOE rep mentioned that some vendors couldn't support some of the use cases that they had prepared at the recent DOE Cyber Security Conference. A vendor stated that this type of thing should be put on the discussion forums so that it will be publicized when vendors succeed and when they fail.

A question was posed: Was the event too SCAP-focused? There are several other use cases that can make use of these standards. NIST reminded the attendees of the vision slide which illustrated that SCAP is just one of four columns. Another attendee countered with the suggestion that this audience may not be the right group of people for some of the other columns.

When should we get together again? One proposal was to meet more frequently for shorter periods of time (e.g. 3 times a year for not more than 3 days each). Also, perhaps combine meetings with other events, like the RSA Conference.

An attendee asked about an action item list from the conference. Steve Boczenowski explained that MITRE would be providing comprehensive meeting minutes and that action items would be pulled out

of there. NIST encouraged attendees to check the scap.nist.gov web site. Telecons and web-x's could help us make progress with the individual standards.

With regards to architecture development, it was suggested that a good first start is a concept of operations document, which would describe various use cases. This would help to lay out the vision. (Bob Martin pointed out that MITRE has written some of this type of thing. Go to the MSM page - measurablesecurity.mitre.org – and click on "About")

Is there a demand for entry-level education in the SCAP standards? Yes. NIST added that NIST SP 800-117 is a good entry level document for SCAP.

A huge THANK YOU goes out to all the community members that attended Security Automation Developer Days and shared their opinions about the topics being discussed. It is this community participation that drives the development of our work and enables us to progress down the standards path.