Security Automation Developer Days

June 14-16, 2010

Table of Contents

Introduction

Security Automation Developer Days was held on June 14 - 16, 2010 at The MITRE Corporation in Bedford, MA. This event was the most recent chapter in an ongoing series of workshops, beginning in June of 2009 at MITRE, Bedford and February, 2010 at NIST in Gaithersburg, MD.

Eighty-six people registered for the event, and roughly 75 people were present each day of the workshop. Over the three 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

Apple	-	Shawn Geddis
Arellia Corp	-	Kevin Stephens
Belarc	-	Richard Defuria Gary Newman
BigFix	-	Eric Walker
Booz Allen Hamilton	-	Ying Qi
Cimcor	-	Timothy Rodriguez Jeff Wozniak
Cisco Systems	-	Joseph Dallatore John Stuppi Seth Hanford
Dept of State	-	Randell Adkins Vu Nguyen
DISA	-	Andy Clifford Jim Govekar Jason Mackanick Joe Mazzarella Scott Moore Alan Peltzman Pete Schmidt Jordan Shuhart Jacquie Sipes Jamaal Spearman Ricki Vanetesse
DTCC	-	Aharon Chernin
EMC	-	Dan Reddy
Federal Reserve	-	Blake Thomas
	-	Doug Taylor
G2, Inc	-	Matthew Kerr George Saylor Shane Shaffer
IBM	-	Scott Moore

McAfee	-	Kent Landfield Richard Whitehurst
MITRE	-	Jonathan Baker Sean Barnum Steve Boczenowski Andrew Buttner Maria Casipe Brant Cheikes Michael Chisholm Matthew Hansbury Daniel Haynes Jasen Jacobsen Mike Lah Daniel Mitchell Linda Morrison Lisa Nordman Mary Parmelee Emily Reid Charles Schmidt Matthew Wojcik Bryan Worrell John Wunder Margie Zuk
Modulo Security	-	Marlon Gaspar
NASA	-	Gary Gapinski
nCircle	-	Ian Turner
NIST	-	John Banghart Harold Booth David Waltermire Paul Cichonski
NSA	-	Mike Kinney Jim Ronayne Joseph Wolfkiel
RSA	-	Matthew Coles Dennis Moreau
SecPod Technologies	-	Chandrashekhar Basavanna
SRA	-	Wei Chen Deirdre Regan

Symantec	-	Jim Boyles Richard Freeman Jason Meinhart Craig Morea Dragos Prisaca Josh Turpin
TASC	-	Glenn Fournier Tom Mowbray
Telos	-	Sudhir Gandhe
Tenable	-	Mehul Revankar
ThreatGuard	-	Rob Hollis
Tripwire	-	John Wenning
Triumfant, Inc	-	Bill Goodrich
US Army	-	Gerald Crismon Vladimir Giszpenc
USAF	-	Kathleen Lynch
USDOT	-	Brendan Harris

Monday June 14th

Asset Results Format and Asset management

Background and Introduction

- SCAP content allows standardization of how to check for things
  - But how do you handle enterprise automation?
  - Results reporting, information sharing, tasking
- Need languages and interface specifications
- Building off of Assessment Results Format, Assessment Summary Results, Policy Language for Assessment Results Reporting (ARF, ASR, PLARR)
  - *Straw Poll: about ½ the audience had used or heard of ARF 0.41*
- New Specifications being introduced
  - Asset Results Format (ARF 1.0)
  - SCAP Source/Result Data Stream
  - Asset Data Model
  - Asset Identification
  - Interface Specifications
- Governance
  - Led by NIST
  - Core group of NIST, MITRE, DoD
  - Others can join
- In collaboration to transition ARF 0.41.2, ASR, PLARR to new NIST specifications
  - Also in collaboration with OVAL, OCIL, other specifications

Prioritization and Timeline

- SCAP 1.2 Timeline
  - 30 June: DRAFT specifications released
  - Close of comment period end of July
  - Specifications Final by August 15
  - Reference implementations by end of August

Which of the specifications are most important to work on for SCAP 1.2?

- Suggestions for SCAP 1.2:
  - ARF, Asset Context Model, Asset Identification, source and result data streams
- Suggestions for later work
  - Tasking and results request, summary results, interfaces
- **Jon Baker**: It would be challenging to integrate the identification specifications into OVAL/OCIL/etc and get those specs final by SCAP 1.2
- **Unidentified speaker**: What is an asset?
  - Anything you run an assessment on (things you care about): people, devices, systems, networks, organizations
- **John Banghart**: Have you considered information being an asset that has value? For example, a document has value and you may want to asset it for integrity.

    - **Dave Waltermire**: There's a lot of things we consider assets, including information. Our thinking is to start small, describe a subset of the most used models, and allow for extensibility. We have bigger problems than information.
    - **LtCol Wolfkiel**: Maybe it would be better to list what we consider assets, and show how they fit in that category.
    - **Mike Kinney**: If you can't automate it, it makes it hard to talk about and may not be valuable.
        - **Dave Waltermire**: Important to think about this effort as more than just assessments and automation.
    - **John Wunder**: Let's focus on the current state of practice.
- **Unidentified Speaker**: Do you have an example of the asset context model?
    - **John Wunder**: Not yet, model isn't done yet.
- **Same speaker**: Applications and servers are both assets yet applications reside on servers. How do you automate remediation ownership considering this?
    - **Dave Waltermire**: Relationships are very complicated.
    - **John Wunder**: We'll have another meeting to talk about the asset context model specifically, let's focus on timelines now.

Overview of Planned Specifications

- Asset Reporting Model
    - Container for an asset, what was assessed against, the results, what performed the assessment, similar to ARF 0.41.
    - How do you report on SCAP results? OVAL, OCIL, XCCDF, etc
- Asset Context Model
    - What does an asset look like?
        - IDs, IP, MAC, etc
        - **LtCol Wolfkiel**: This is an asset ontology
    - **Vladimir Giszpenc**: Isn't all that in the reporting format?
        - **John Wunder**: Maybe it can be talked about in both places. If result of an assessment, maybe it's in ARF. If it's the result of fusion and correlation, it's probably part of the context model.
- Asset Identification
    - Do you give asset a unique ID
    - Do you identify it based on other info
    - And how do you do this in the context of multiple sensors, assessment styles, timelines, etc.
    - **Unidentified speaker**: If you throw asset identification into OVAL, does that mean the other changes would also need to be adopted for SCAP
        - **Jon Baker**: Yes. It could also be optional inside system info
        - **Dave Waltermire**: We're doing the same thing with OCIL.
    - **Dennis Morrow**: Lots of work has gone into this inside CMDBs and other systems, we don't want to be orthogonal to that. Does your specification deal with the complexity of the issues (over time, different identifiers).
        - **John Wunder**: Yes, we've talked a little about this, and it should be able to reuse much pre-existing work to handle complexity.
    - **John Banghart**: Are you taking into account virtualization?
        - **John Wunder**: Yes, it's an important issue. How you treat VMs depends on what you're assessing: hardware, OS, applications…
- SCAP Source and Result Data Stream

- Bundling of SCAP results and content
- Similar to existing SCAP Data Stream ZIP files

Back to the discussion of what is viable for SCAP 1.2

- **Gary Newman**: You mentioned you've been discussing this for a few weeks, is that on the list?
  - **John Wunder**: No, this is in core working group calls. If you're interested in joining that let us know. We'll also be starting public meetings soon.
- **Gary Gapinski**: Does the timeline entail validation for products that governments must buy? This is ambitious given amount of uncertainty.
  - **Dave Waltermire**: Yes, validation against SCAP 1.2 begins in early 2012. It is ambitious, but it's not a brand new initiative…it's based off of ARF 0.41, ASR, and PLARR.
- **Dennis Morrow**: Asset identity has to be high priority, if you leverage prior art you'll be ahead of the game.
- **John Wunder**: Do people agree that the last three bullets (tasking, interfaces, summary results) should be pushed off?
  - **LtCol Wolfkiel**: ASR is heavily used within DoD
  - **Gary Gapinski**: It is difficult to use existing content. What might OMB be asking for this year on to next year in terms of reporting requirements?
    - **John Banghart**: We can't speak for OMB. Look for additional memos from them.
  - **John Wunder**: Based on LtCol Wolfkiel's comments should we include summary results?
    - *Not much comment, Dennis Morrow (?) said no.*

Asset Identification

- Description of use cases (see slides)
  - SCAP, EMAP, ERAP, OVAL, OCIL, XCCDF, context model

Asset Context Model as use for Identification

- First question: are there important differences between talking about a full asset context model vs. just an identifier. Are those things interchangeable in the specifications using asset identification? Our thought is that you should focus on asset identifier.
- **John Banghart**: Full model exists somewhere, so it's whether or not reporting is based on identifier vs. full model
- **Jon Baker**: From an OVAL perspective you need to have a choice. Tools consuming OVAL results downstream may not know the identifiers.
- **Gary Newman**(??): What's a full asset model?
  - Asset Context Model
  - **LtCol Wolfkiel**: You could build an ID model that matches arbitrary assets (IP/Mac). Full context model would include a lot more data that doesn't help uniquely identify a device.
- **Aharon Chernin**: Asset context data changes very rapidly. If you keep that context inside OVAL results it won't be valid for very long.
- **Vladimir Giszpenc**: My computer has 2 or 3 asset tag stickers on it. Find places to put identifiers to facilitate automating those things as well.
- **Dave Waltermire**: Standardize expression of existing identification information. There are other differences between the context model and identification: lots of information that isn't interesting from an identification perspective. OVAL results could also be considered a type of context.

Other Asset Identification Topics

- **Rob Hollis**: We should revisit Vald's idea of generating a GUID for a device and put it in a standardized location for SCAP tools to find.

    - **John Wunder**: Definitely valuable, more standardized and automated.
    - **Vladimir Giszpenc**: You can make it namespace and ID.
- **Gary Newman**: Uncertain about whether we can meet all the use cases. We need a scope, what's the goal?
    - **Dave Waltermire**: We're trying to support a synthesis use case.
    - **Gary Newman** (?): Is this a police work type job, where you just throw anything you find into the identifiers.
    - **John Wunder**: There are also other use cases besides synthesis, such as specifically targeting an asset.
- **Gary Newman** (?): Reminds me of the submarine that also has to fly. Maybe this should be split into two specifications.
    - **Dave Waltermire**: That difference might be the difference between identification information and context model (context model is detective work)
    - **LtCol Wolfkiel**: The fact that different types of data have different probabilities of matches doesn't mean that we need to split the specification. Why not have business rules to say which to use?
- **Gary Gapinski**: There's a false dichotomy between information that definitely uniquely identifies an asset and information that partially identifies an asset. Can't rely on information that changes.
- **Mike Kinney**: the full asset model could be huge and include many things you don't need.
- **John Banghart**: You don't necessarily need to send a fully completed asset model to send part of the asset model
- **LtCol Wolfkiel**: Need to make sure we don't pass useless info
- **Dave Waltermire**: The payload should be separate from the set of attributes used to identify it. Set of attributes works across a wider set of use cases.
- **Kevin Stevens**: If you don't know what the other tool needs to have to identify it, you need to present several things. There needs to be a specification for asking for a given ID (ie: give me your IP Address). In the absence of that you present all IDs.
- **Gary Gapinski**: Probably need some subset of information from the context model that uniquely identifies the asset.
- **Mary Parmelee**: Could you modularize the asset context model? Provide a robust set of models for identification that is used often.
- **Scott Moore**: This seems like a false question because when you consider all the ways of identifying assets you might end up with asset context as your full model anyway.
- **Dave Waltermire**: More flexibility hinders interoperability. We're trying to strike a balance.
- **John Banghart**: it should be the report requester who defines what identifiers are provided.
    - **LtCol Wolfkiel**: SCAP is very far from query interfaces.
- **Gary Newman**: Unique identifiers are constructed for a particular purpose. All have upsides and downsides. Usage of identifiers really depends on the use case.
    - **John Wunder**: You probably care about many use cases, identifiers should be used where appropriate.

Use Cases

- Asset Results Format
    - Used in source and subject identifiers
- ARF documents may be coming from many different scanners, objective of identification is to be able to correlate those.
- **Dave Waltermire**: Intent is to decouple identification of asset from context model. Multiple results from different sources can be placed in ARF using this model.

- **Gary Gapinski**: Is the subject identification valid for all results?
    - **John Wunder**: yes, even if the results don't have that data now
    - **Dave Waltermire**: this avoids duplication, ie between OVAL and XCCDF
- **LtCol Wolfkiel**: XCCDF is broken when it comes to identification: target facts do not specify enough so vendors provide different information.
- OVAL Use case…could be used to plug into the OVAL system identification model
- Basic issue is how do you associate results or context data with the asset they were collected on.
    - Given multiple sensors, collection timeframes, architectures, etc.
- Canonical identifiers identify an asset explicitly, identifying information tries to identify it based on collected information, context model just gives an entire representation.
- Core working group's thought is to allow one or more canonical identifiers OR some set of identifying information

Canonical Identifiers

- Tools have identifiers for assets
- But different tools have different IDs or no ID at all
- One option is to try to propagate a single unique ID across an organization
- Another option is to namespace IDs and allow tools to report using their own IDs
- Proposal: multiple namespaced identifiers per identification element
    - Pass around as many IDs as are necessary or desired
    - Allows federation of IDs if you want to, but doesn't require it
- **Gary Newman**: what is the canonical ID? Mac address, IP address
    - John Wunder: it's the current ARF 0.41 model, a namespace and a non-intelligent ID
    - Canonical means valid in a particular namespace
- **Eric Walker**: Seems like a similar problem to the problem being solved in CPE. In both cases you're talking about gathering disparate pieces of information to come up with a conclusion (ID).
    - Harold Booth: Both Dave (Waltermire) and I are involved in both efforts so will be taking lessons learned.
- **LtCol Wolfkiel**: This is different than CPE in that any system, sensor, or data store could give an ID to an asset and this isn't a problem. Have more than one identifier for an asset is not a problem, it's a design goal. Matching may look similar to CPE.
- **Gary Gapinski**: For the second bullet, why not use NCName in XML?
    - **LtCol Wolfkiel**: ARF 0.41 also uses xsd:anyURI
    - **Gary Gapinski**: You've cut yourself off from being able to use it as an XML namespace. You could structure a document where the namespace for an attribute was identified with the namespace for the canonical IDs namespace. Could also be useful in RDF.
    - **Dave Waltermire**: Using a URI reference as a form of identifier is well established.
    - **John Wunder**: Namespace in this context doesn't mean XML namespace, it means the namespace that the ID is valid.
    - **Gary Gapinski**: It would be worth talking about this online, as the constructs are parallel.
- Identifier could be used in any place an identifier is acceptable.
    - *Examples (see slides)*

Identifying Information

- How do you use collected information to identify assets?
- The "detective work" identification
- Probably won't be a perfect match, so you have confidence, normalization, and wildcarding concerns

- Proposal: Allow a defined list of keys that may be used to create a match vs. allowing matching on any attribute
    - Suggestion is to support a fully specified list but allow ad-hoc matching as a value-add but do not standardize on it.
    - Support wildcards on fields as appropriate
- **Gary Gapinski**: A little concerned that these items have been disassociated with the namespace that canonical IDs were found
    - **John Wunder**: That was the intent, collected data is different than an assigned ID
    - **Gary Gapinski**: it appears odd that the identifiers don't live in a namespace, but canonical IDs do
    - **LtCol Wolfkiel**: Sometimes we have a concept of a default namespace that is globally unique (ie CVE).
- **Gary Gapinski**: Same reservation as before, no association with XML namespace. Need a way of describing how items are being reported, as sensors may report information (e.g., address) in different ways.
- **Dennis Morrow**: Increasingly, collected data about devices is used to identify an asset but OVAL is not able to express this.
    - **John Wunder**: Could express this with tagged values
    - **Dave Waltermire**: It's really a term identifier
    - **LtCol Wolfkiel**: We implemented that in ARF 0.41, downside you need to express your controlled vocabulary (possible values) outside of schema and outside of the specification or give up on controlled vocabularies.
- **John Wunder**: idea of paths to keys and values is different than tagged values in that paths need to resolve to elements in the context data model, tagged values are wide open.
- **Gary Gapinski**: There are cases where you need to know where data was collected in order to tell where it is valid.
- **LtCol Wolfkiel**: In ARF 0.41 you would always be able to relate sensor data, including identifying information, to the sensor that collected it.
- **John Wunder**: Does identifying information need to be tied to the sensor that collected it?
    - *General agreement, particularly LtCol Wolfkiel*

Asset Context Model as Identification

- Some tools might not have any idea of canonical IDs, but wants a great match, so it sends an asset context model to match against.
    - **LtCol Wolfkiel**: Current DoD work uses this strategy, but without sending extra contextual information. Similar to matching on a large set of identifying information.
- Core working group's thought would be to allow this as a last resort, but disallow it in SCAP or other specifications

Overall Identification Methodology

- Proposal: Allow identifications to consist of:
    - Canonical IDs AND/OR
    - Identifying Information AND/OR
    - Context Model
- No notion of request in current methodology or specification.
- **Gary Gapinski**: No rigor to this approach, all it says is as much or as little as possible. This approaches asset identification but is not asset identification.

    - **Dave Waltermire**: You can't be comprehensive enough to "truly" identify an asset, so we need to do this.
- **Jon Baker**: It might be worth giving guidance as to what information should be provided, "as much as possible" is vague and might create large identifiers. Also, if you can't fulfill the requirements can you still participate?
    - **John Wunder**: You provide as much as you have out of the constrained list.

Confidence Levels

- How do we define confidence levels?
- Is it worth talking about confidence levels?
- **Mike Kinney**: We're talking about confidence in that specific attribute as opposed to confidence in the tool that provided it?
    - **LtCol Wolfkiel**: Sometimes, sensors are more confident in specific measures than others. E.g. NMAP. You need both overall confidence for a sensor and confidence in specific data elements.
    - **Jim Ronayne**: Confidence in sensor is a concern of the consumer, confidence in data element is concern of the sensor itself.
- **Charles Schmidt**: Do we distinguish between information we have confidence in and things that are ephemeral?
    - **John Wunder**: Is it the producer or the consumer's job to figure that out? If it's the consumer then we don't need to care because it doesn't need to be in the data format.
- **Ian Turner**: Also need to consider when a sensor has different capabilities depending on other factors (network locality, etc).
- **Gary Gapinski**: Without a unit of measure it's impossible to correlate two different measures. Also, sensors may not know their own uncertainty. Unless you can make a "mathematics of uncertainty" this is not valuable so we should leave it out.
- **Charles Schmidt**: Universal scale is percentage, since the value is either the given value or it isn't.
- **John Banghart**: What does it mean to have 100% certainty in hostname and 50% certainty in IP Address?
    - **John Wunder**: We don't want to define the math for matching these things, we just carry the values.
- **Jim Ronayne**: We're relying on the sensors to provide very specific values, maybe we need a less graded scale (sure, sort of sure, not sure vs. 0-1)
- **Dennis Morrow**: Confidence is a measure that's most valuable at 100% and at 0%, don't treat it as a metric.
- **Doug Taylor**: We should reuse prior art from business intelligence tools, data quality measures. In both cases eventual goal is action by an operator and to ensure reliability.
- **Vladimir Giszpenc**: Make it optional and move on.
    - **John Wunder**: Even if it's optional, we need to define what it means.
    - **Vladimir Giszpenc**: Let it be up to the tool…
- **Matt Coles**: Does confidence make sense with correlated or processed data?
- **Charles Schmidt**: How about high/medium/low rather than percentages? This doesn't imply mathematical values.
    - **Mike Kinney**: In order to do calculations this needs to be a number.
    - **John Wunder**: Nothing ties high/medium/low to reality, tools can misreport.
    - **Charles Schmidt**: Same problem with percentages, tools can misreport.
    - **LtCol Wolfkiel**: Need to define the number of levels of confidence. Three may be too few.
    - **Chris Johnson**: Percentages imply precision that doesn't exist.

    - **Scott Moore**: You get more fidelity with percentages. You can always determine rankings in buckets (bucket 80-100 as high).
    - **Gary Newman**: If we come up with confidences based on how it's discovered, we should just state how it's discovered.
        - **Jim Ronayne**: That isn't always how we come up with confidence
- **Eric Walker**: Confidence is handled by each organization in policy. Getting the decimal field lets you handle everything in policy
- **Mary Parmelee** and **Dave Waltermire**: If it's derived per organization, how do you cross those boundaries? Maybe you need to also send how you derived the confidence value.
    - *Straw poll: 50/50 should be included as an optional attribute.*
    - *Straw poll: decimal vs. category, decimal slightly ahead*
- Discussion of bindings
    - JSON, XML, Text string
    - Shouldn't be bound to XML
    - Bindings must be reversible
- How does ARF matching work now at DISA?
    - Primarily based on IP and Mac
    - Also, some tools can collect GUIDs (canonical IDs) from other tools

OVAL

What Has Changed Since Last Year?

The OVAL session started with a brief recap of what has changed in OVAL since last year's event and an overview of the outcomes of last year's discussion items.

Highlights

The following highlights were cited:

- Version 5.6 – Released September 11, 2009
- OVAL Adoption Program launched
- Version 5.7 – Released May 11, 2010
- Version 5.8 Planned – August 18, 2010
- OVAL Repository
    - 1525 new definitions
    - 4510 modified definitions
    - 7287 total definitions

Last Year's Topics

Last year's topics included the following items:

- Deprecation Policy Review
- Schematron Usage in OVAL
- Element Name Reconciliation
- xsd:choice Structure on Objects
- Supporting N-Tuples in OVAL

- Pattern Match on Enumerations
- Tests Reference Multiple States
- Introduce PCRE Based Pattern Matches
- Emerging Use Case: "OVAL for System Inventory?"

A brief summary of the status on each of these items was presented. Detailed minutes from the 2009 OVAL Developer Days discussion can be found on the OVAL web site at:

https://oval.mitre.org/oval/about/developer_days.html

Discussion Topics

Taming OVAL Results

This topic consisted of three closely related subtopics that together were aimed at allowing for smaller more useful results sets. This topic was discussed in response to community feedback and feature requests related to making OVAL Results more useful. The general feature requests have been:

- More granular evaluation results are needed to show why a definition evaluated as it did.
- Full OVAL Results do not scale well to enterprise.
- Include only the actionable information in OVAL Results.
- Highlight the hidden data in OVAL Results.

All of these requests come with the expectation that OVAL Results must maintain interoperability. Products must continue to be able to exchange OVAL Results.

Subtopic 1: More Granular Evaluation Results

This subtopic is based on a feature request received over the oval-developer-list. The request was summarized as "add capability to specify test result other than true or false". The requestor would specifically like OVAL to support the following examples:

1. Verifying installed version of an application:
    - `true` – when application version 7 is installed in the system
    - `false` – when the version of application is not 7
    - `not applicable` – when the application is not installed
2. Verifying file permissions:
    - `true` – when the file exists and its access rights are properly configured
    - `false` – when the file exists and its access rights are not properly configured
    - `not applicable` – when the file does not exist

In short, this can be thought of as needing to report that the system is neither compliant nor noncompliant if the application or file does not exist.

Background

As background for this discussion the current OVAL Result values were presented and discussed. As of version 5.7 the following result values are defined in the OVAL Results schema:

- `true` – the characteristics being evaluated match the information represented in the system characteristic file.
- `false` – the characteristics being evaluated do not match the information represented in the system characteristic file.
- `unknown` – the characteristics being evaluated cannot be found in the system characteristic file.
- `error` – the characteristics being evaluated exist in the system characteristic file but there was an error either collecting information or in performing analysis.
- `not evaluated` – a choice was made not to evaluate the given definition or test.
- `not applicable` – the definition or test being evaluated is not valid on the given platform.

The specific meaning and utility of the '`not applicable`' results was reviewed in some detail since it is important to note that the 'not applicable' result value already has a special meaning and it would not be appropriate to overload that meaning for another purpose. As noted in the OVAL Results schema documentation, "… a result value of '`not applicable`' means that the definition or test being evaluated is **not valid on the given platform**. For example, trying to collect Linux RPM information on a Windows system. Another example would be in trying to collect RPM information on a linux system that does not have the RPM packaging system installed." As it is defined, the '`not applicable`' result value allows content authors to combine criteria logically for multiple platforms. For example, a vulnerability definition can be written to include a `<registry_test/>` and the `<rpminfo_test/>` in order to check both Windows and Red Hat systems. In order to support this capability the 'not applicable' result value is not considered when determining aggregate results. For example, a '`false`' result AND a '`not applicable`' result will aggregate to a '`false`' result. Similarly, a '`true`' result AND a '`not applicable`' result will aggregate to a '`true`' result.

Another issue that prevents an OVAL Results consumer from processing a results document as it is and locating the desired information is the fact that some of the results information may be obscured once the result of a test is determined. Each test has a `check` attribute and a `check_existence` attribute. The `check_existence` attribute allows a content author to make an assertion about the number of Items that must exist on a system. The `check` attribute allow a content author to make an assertion about the number of Items on the system that must satisfy the referenced State conditions. When determining the result of a Test if the `check_existence` is satisfied (true) then the `check` is considered. Otherwise the Test result is the result of the `check_existence`. This evaluation process means that there is no way to differentiate a 'false' result due to failing the `check_existence` from a '`false`' result due to failing the `check`.

Proposal 1: Add a Result Value

In this proposal the suggestion was made to add a new result value like: `false_existence`. This value could be used to record the fact that the `check_existence` evaluation failed. Under this proposal the not applicable result value could continue to be used as it is.

The initial examples could be reworked as follows with the proposed `false_existence` result value:

1. Verifying installed version of an application:
    - `true` – when application version 7 is installed in the system
    - `false` – when the version of application is not 7
    - `false_existence` – when the application is not installed
2. Verifying file permissions:
    - `true` – when the file exists and its access rights are properly configured
    - `false` – when the file exists and its access rights are not properly configured
    - `false_existence` – when the file does not exist

These specific simple examples would work with this new result value.

In considering this proposal it is important to note that as defined in OVAL version 5.7 a test essentially evaluates to a Boolean with a few other possible error like results. Adding in this new result value would be a move away from this Boolean result. Accordingly all the evaluation tables that specify how to combine each of the possible result values would need to be recreated. These evaluation tables must specify how to combine a '`false`' and a '`false_existence`' result and this decision will almost certainly not meet everyone's needs all the time.

Finally, it was noted that moving away from the Boolean result and adding a new result value might eventually lead us to an unwieldy proliferation of result values.

Proposal 2: Add an Attribute

In this proposal the suggestion was made to add a new attribute to the `oval-res:TestType` to record the result of evaluating the `check_existence` attribute. This new attribute would not be considered during Definition evaluation and would simply provide high level metadata to record one step in the evaluation process for a Test.

The initial examples could be reworked as follows with the proposed `existence_result` attribute:

1. Verifying installed version of an application:
    - `true` – when application version 7 is installed in the system
    - `false` – when the version of application is not 7
    - `existence_result='false'` – when the application is not installed
2. Verifying file permissions:
    - `true` – when the file exists and its access rights are properly configured
    - `false` – when the file exists and its access rights are not properly configured
    - `existence_result='false'` – when the file does not exist

In considering this proposal it is important to note that the evaluation process is not modified in any way by this new attribute. The attribute is simply metadata about that evaluation process. As such there is no need to alter any of the existing evaluation tables. This proposal also preserves the Boolean result values for a Test and Definition. However, this proposal requires extra work on the part of the OVAL Results consumer to process the content and determine if a given result was false due to an existence failure or a state failure.

Discussion

One participant pointed out that it is desirable for the tool or the higher level context to make the decision to evaluate a given OVAL Definition on a given system or not. This is commonly done in XCCDF with a platform check in a Rule or possibly even at the Group or Benchmark level.

In considering proposal 1 the following comments were made:

- How would this proposal handle sets of items and reporting partial existence of the needed items? The suggestion was made that another result value would be needed for this situation to record partial failure of the existence check.
- It is a slippery slope with a ternary state. This could lead to a large set of possible conditions.
- Perhaps we go the other way and not allow tools to ask this sort of two part question and instead simply create two different definitions: one definition to test for applicability and another to test for the configuration setting.
- Keeping the checks (OVAL Definitions) as granular as possible allows for flexibility and ensures that this problem can be supported in OVAL as long as there is a higher level context in which a determination about which checks to run can be made.

The discussion comments led to the group questioning whether this particular issue is really something that OVAL needs to solve. One could argue that OVAL has all the needed capability today and that there is not a strong need to push this capability down into OVAL.

An additional suggestion was made to consider adding a new Test or set of Tests like the `<unknown_test/>` that would evaluate to a specified result value whenever considered.

In considering proposal 2 the following comments were made:

- When a test fails the `existence_check`, the Items that are collected as a result of processing the Test's referenced Object are not evaluated. These items are all referenced in the `oval-res:TestType` with the `<tested_item/>` element. This element holds the id of an Item in the collected System Characteristics and the result of evaluating that Item. When the `existence_check` fails, the result record for each `<tested_item/>` is '`not evaluated`'. This information should be enough to convey that the `existence_check` failed. Essentially, OVAL already has information that would allow a tool to determine that a false result was due to a false `existence_check`.
- The proposed `existence_result` attribute could be thought of as adding in yet another type of result response and therefore is changing a long standing premise of OVAL that each Definition evaluates to a single result value.
- Adding some sort of third result will simply add confusion to result consumers.

Conclusion

Throughout the discussion there was a recurring theme of whether or not OVAL should address this feature request at all. In the end there was little to no support for making any change to OVAL to

support this feature request. In response to this discussion the topic will be revisited on the oval-developer-list with the suggestion of closing out this feature request.

Subtopic 2: Lightweight OVAL Results

This subtopic is based on an open feature request to "add result directive to allow for lighter weight results with ability to track causal information". In this case the requested feature is actually a bit overloaded. There are really two distinct items to be discussed. First, adding additional directives to allow for more control of the content of an OVAL Result document. Second, highlight the causal information that is needed by result consumers to make easy use of an OVAL Results document. In this discussion the group focused on the first item. The second item was discussed in the "What is the cause?" subtopic.

Background

As background for this discussion the current OVAL Result directives were presented and discussed. OVAL Result directives can be thought of as a set of flags that describe the information included in an OVAL Results document.

As of version 5.7 the OVAL Result directives can be used to specify the level of detail included in an OVAL Results document as follows:

- Results may be included by result value. For example, an OVAL Results producer can exclude all OVAL Definitions with a result of 'not evaluated' or only include results for OVAL Definitions that had a result of 'true'.
- Either full system data and evaluation results or simply an OVAL Definition id and a result value many be included for each OVAL Result value.

OVAL Result directives are set by the result producer and included in the OVAL Result document. There is an assumption that the result producer is somehow configurable and capable of producing OVAL Result documents at varying levels of verbosity.

As OVAL Results are currently defined, there is a moderate level of flexibility in the allowed result outputs that can easily reduce the size of a result document by 20% or more in realistic usage scenarios.

Discussion

The discussion of this subtopic focused on exploring and better understanding the shortcomings of the version 5.7 OVAL Results directives. This discussion started by considering the following questions:

- Does anyone really support the directives?
    - SCAP 1.0 explicitly requires full OVAL Results.
    - Directives are not currently implemented in OVALDI**.**
- Do we need more options than thin and full?
    - Does the definition document need to be included?
        - If not, then how does a result consumer really know what was evaluated?
- Do we need to consider definition @class?

    - For example, include full results for true vulnerability definitions and thin results for all other true definitions.
- How do more options affect interoperability?
    - Are full results needed for generic content sharing?

During the discussion the following comments were made:

- Within a product I know what OVAL Definition document was used during evaluation. It would be nice not to require the source OVAL Definition document in the OVAL Results so that my clients could return OVAL Results without the source OVAL Definition document. From my server, I could then produce full OVAL Results including the source OVAL Definitions if needed.
- When the OVAL Results schema was created there was no concept of a results data stream. There did not used to be a benchmark with an id that served as a wrapper around a set of OVAL Results. At this point we may not need the full source OVAL Definition document.
- If I know an OVAL Definition id and I simply want the results back, I do not want the full definition I just requested back.
- We need to look at all the baggage that is included within the SCAP component standards. If there is simply too much baggage, then we will need to move on to other solutions.
- In XCCDF the original document is optional. A benchmark can be evaluated and then just the results can be provided without the full source XCCDF document.
- Adding more result options to OVAL simply adds complexity. We are currently guessing about future use cases of OVAL Results that we don't understand and trying to define formats that will meet those needs. Much of what is being discussed can be achieved through a simple transform of an OVAL Results document. Are we adding complexity to OVAL that is simply not needed?
- Should OVAL focus on the simpler end system assessment use case and allow others to develop solutions on top of OVAL that address exchanging result information on an enterprise scale?
- In SCAP 1.1 several different OVAL Result combinations will be included to allow for more compact OVAL Results. In SCAP 1.2 there is additional opportunity for improvement. NIST is looking into a more formal request/response format for SCAP 1.2 and additional OVAL Result granularity could be used.
- One vendor indicated that they simply make use of a transformation of a full OVAL Result document to create what they call a findings result.
- Requiring each vendor to write their own transformation will defeat tool interoperability so we really should look into how we can define a useful format in OVAL for conveying OVAL assessment results.
- When a definition is created, the author does not always know what is important to report.
- Could a simple hash or checksum of the source OVAL Definitions document be included rather than denormalizing all the result data by including a copy of the source document?
- Even when a compliance definition results in true people sometimes want to know the full details and what the relevant observed system value is.
- For result content should we extend beyond full and thin? Let's consider allowing for variations in between the two. For example, allow thin plus full definition criteria. On top of that allow full

test results. Then on top of that allow full item data. In general, more result granularity will allow for tailoring by the evaluation requestor. In order to increase granularity we need to expose a schema for declaring what the results should look like. Consider adding an OVAL Results Directives schema for this purpose. Note that other standards could reuse that schema to indicate that a given set of OVAL Definitions should be evaluated and that a given OVAL Results format must be provided.

Conclusion

During the course of the discussion the following items were noted:

- OVAL Results Directives should allow for differentiation of OVAL Results content by class.
- OVAL Results should consider allowing the source OVAL Definition document to be optional. There might be a in between option here that would include simply the ids of the Definitions that were evaluated.
- OVAL should consider creating a lightweight schema for specifying the directives that must be used in reporting OVAL Results.

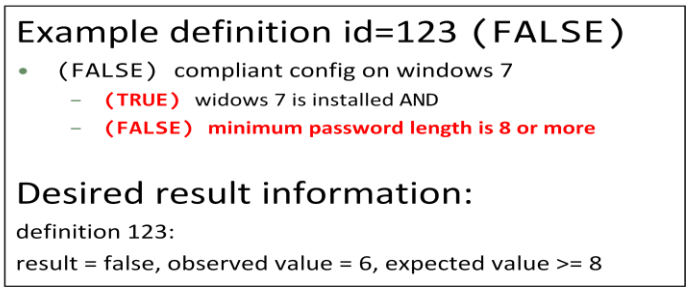
Throughout the discussion the point was made that much of what was being discussed in terms of controlling the content of a given OVAL Results document could be achieved by applying a XSL transform. The downside here is simply that the output of the transform needs to be standardized. There needs to be a standard result format of varying levels of verbosity that can represent an OVAL Definition evaluation. These topics will be further discussed on the oval-developer-list.

Subtopic 3: What is the cause?

This subtopic is based on an open feature request to "add result directive to allow for lighter weight results with ability to track causal information". In this case the requested feature is actually a bit overloaded. There are really two distinct items to be discussed. First, adding additional directives to allow for more control of the content of an OVAL Result document. Second, highlight the causal information that is needed by result consumers to more easily make use of an OVAL Results document. In this discussion the group focused on the second item.

Background

Full OVAL Results likely contain the data needed to determine the observed settings on a system that led to a compliance failure or vulnerability report. The following simple example was discussed:



In the example above, the interesting data is that the Test referenced by the minimum password length criterion evaluated to false. This seems fairly straight forward here, but as the criteria of a Definition get more complex it becomes increasingly difficult to identify the important pieces of data in an OVAL Results document.

As further background it was pointed out that generally a Definition author knows what information is of the most interest when creating a new Definition. Most OVAL Definitions include a fair amount of preconditions and other boiler plate checks that are generally not that interesting to report to an end user. With this in mind, it was noted that the most interesting data is generally examined with a State entity. Within a single OVAL Definition there can be many States and each State can have many entities. A State is used to make an assertion about an Item. Items have entities that are evaluated by State entities. Item entities may have a multiplicity of 0 – N.

Proposal: Add a `report_value` Attribute

If the assumption can be made that a definition author knows which items are most interesting to report then why not allow the author to highlight or flag those items?

This proposal suggests adding an optional `report_value` attribute to all State entities. This Boolean attribute would default to false and when true would indicate that the system value(s) should be highlighted or reported in the OVAL Results document. The current behavior would remain unchanged when the `report_value` attribute is false.

The second component to this proposals is the addition of an <oval-res:observed_value/> element in the existing <oval-res:tested_item/> element. A <oval-res:tested_item/> could have an unbounded set of child <oval-res:observed_value/> elements. Each <oval-res:observed_value/> would hold the name of the reported State entity, the data type observed on the system, and the observed value.

In considering this proposal it is important to note that this is not a complete solution. This solution would work quite well in simple cases, but more complex definition criteria would remain difficult to process and accurately report the underlying cause of a given Definition result. If accepted this proposal would also result in even larger full OVAL Result documents as it would essentially duplicate some of the data that is somewhat buried in an OVAL Results document.

Discussion

The discussion on this subtopic first focused on understanding how vendors are solving this problem today. There are products that display the desired information already. How are vendors doing this and can we develop a common solution? In response to these questions the following comments were made:

- One individual pointed out that their organization decided to create its own results format to solve this problem. This decision was motivated by the size of the typical OVAL Results file, the fact that they did not want to have to include the source OVAL Definitions that were evaluated, and the fact that they wanted an easier way to look up and report observed values. It was noted

that the current structure of OVAL System Characteristics makes it fairly challenging to look up system objects such as file or registry keys, especially when variables are used.

In discussing the "Add a `report_value` Attribute" proposal the following comments were made:

- We might want to consider essentially reporting all values and then allow a content author to deemphasize a given value. This would essentially be the opposite of the proposal. Rather than highlight data, allow an author to specifically indicate that a given value should not be reported.
- There are cases in which for a reporting purpose we might only care about a given value if some other Test is true. These dependencies will remain challenging with this proposal.
- Should there be a criteria or criterion attribute to allow for highlighting and reporting at that level? Would this help in complex cases?
- When performing an existence check a State is not used. How would we handle this if we are not using a State? Is this creating an inconsistency in where result data is held in an OVAL Result document?
  Response: When performing an existence check, collected Items are recorded with `<oval-res:tested_item/>` element in the `<oval-res:test/>`, and each `<oval-res:tested_item/>` element has its result set to '`not evaluated`'.

- Adding this new structure seems counter to the other OVAL Results discussion topics where we were considering reducing the size of an OVAL Results document.
- Hypothetically, a directive could be added to control whether or not this proposed result information is included in an OVAL Results documents or not.
- When preparing for an assessment it will become increasingly import to understand the possible directives and their meaning.
- End users are not generating content. Typically they get the content from some authority. How with this allow an end user to more easily see the reported data?
  Response: The content authorities will need to annotate their content to allow end users to benefit from this capability.

- It is often the case that the full Item data is interesting, not simply the value. For example, in the case of a Windows registry key I want to know the full hive, key, name, and value combination.
- Additionally, highlighting the observed value on a system and reporting that to an end user may not be very useful because the value might not be intended for human consumption. For example, a value might be a bit mask where each bit might have a significant meaning and the end user cannot be expected to interpret those bits. The end user needs the more human readable value that is often displayed in tool UIs.
- When States are reused adding a `report_value` attribute will mean that a value is reported all the time, even when it may not be appropriate.
- Another option would be to add an id to the items that should be reported. At the Definition level the id could be referenced in a way that would indicate that the item should be reported. This might allow a bit more granularity in determining what to reporting.

- Would it be simpler to just provide a lightweight framework and then let content authors include their own style sheet of other set of rules for what to report?

Finally, the proposed OVAL Reporting schema was reviewed to highlight how the proposals in this session differ from this new schema. More information about the OVAL Report schema can be found in the oval-developer-list archives here:

> http://making-security-measurable.1364806.n2.nabble.com/OVAL-Reports-Schema-tp4904766p4904766.html

In summary, the OVAL Reporting schema reuses existing OVAL Objects, States, and Variables to specify a set of information to collect from an end system. The formatting of this information is left entirely up to the author. The report author supplies a XSL template that allows for output as desired. It is important to note the OVAL Reporting is not about making an assertion about a machine state, it only defines a framework for collecting information and then formatting it.

Conclusion

This subtopic needs additional discussion over the oval-developer-list. There are a number of issues that need to be considered.

Optimizing Item Searches

In the OVAL Language, an object is used to specify which items on the system to collect. The collected set of items is determined by specifying operations and values for the entities that describe the item in the object. Depending on the object, and the state of the system, it is possible that a very large set of items will be collected. As a result, tools have to process this data and write it to OVAL System Characteristics and OVAL Results documents which takes both time and system resources. It is often the case that many of the collected items are irrelevant to the assertion being made and the ability to specify more specific sets of items will greatly improve the performance of tools.

This discussion focused on determining whether this is a problem that needs to be addressed, and if so, how and when should it be addressed. The discussion began with an overview of the problem and its influence on tools and the artifacts that they produce. This was followed by a review of the current capabilities in the OVAL Language for item optimization, proposals for adding optimization techniques in Version 5.8, as well as considerations regarding item optimization in Version 6.0. Lastly, the discussion ended with an opportunity to discuss any outstanding topics and summarize the thoughts of the community.

Background

The original discussion that initiated this topic is on the oval-developer-list and can be found at the following link.

http://making-security-measurable.1364806.n2.nabble.com/oval-object-criteria-tp4931198.html

The primary example that exemplifies this problem is the need to find all world-writable files on a system. In the OVAL Language, this is be accomplished by first creating a <file_test/> that will collect every file on the system.

```
<file_object                                    id="oval:sample:obj:1">
    <path              operation="pattern              match">.*</path>
    <filename          operation="pattern          match">.*</filename>
</file_object>
```

Next, a <file_test/> that represents a world-writable file will need to be specified.

```
<file_state                                     id="oval:sample:ste:1">
    <owrite                                datatype="boolean">1</owrite>
</file_state>
```

Lastly, a <file_test/> will need to be specified to compare each collected <file_item/> against the <file_state/>. Then based on the check and check_existence attributes, an assertion will be made about the state of the system.

```
<file_test                                      id="oval:sample:tst:1">
    <object                               object_ref="oval:sample:obj:1"/>
    <state                                 state_ref="oval:sample:ste:1"/>
</file_test>
```

The problem that arises out of this example is that the <file_object/> will collect a <file_item/> for **every** file on the system. Unfortunately, many of the items collected are irrelevant to the assertion of whether or not world-writable files exist on the system. For example, on the sample system, only 1,254 of the 148,745 files are world writable. The collection of irrelevant items also promotes unnecessarily large OVAL System Characteristics and OVAL Results files that are not only difficult to read, but require extra resources to process. For example, the sample system produced a 160MB OVAL System Characteristics file for every <file_item/> as opposed to a 1.24MB OVAL System Characteristics file for every world-writable <file_item/>. As a result, solutions to this problem will target the reduction of items collected.

Current Optimization Capabilities

In the OVAL Language, for each collected object in an OVAL System Characteristics file a flag attribute provides information regarding the outcome of a collected object. For example, if every item that matches the object is collected and written to the OVAL System characteristics file, the collected object will have a flag attribute of true. Or, if an attempt to collect an <rpminfo_object/> on a Windows system, the collected object will have a flag attribute of not applicable because a Windows system does not use RPMs. The OVAL Language currently supports item optimization capabilities with the flag attribute value of incomplete. The OVAL Language defines the flag attribute value of incomplete as follows.

A flag of 'incomplete' indicates that a matching item exists on the system, but only some of the matching items have been identified and are represented in the system characteristics file. It is unknown if additional matching items also exist…

The `flag` attribute value of `incomplete` allows for optimization because tools can write a subset of the collected items, which match the object, to the OVAL System Characteristics file. To highlight how the `flag` attribute value of `incomplete` can be used to optimize item searches an example is provided below.

```
<file_test      id="oval:sample:tst:1"      check_existence="none_exist">
    <object                                 object_ref="oval:sample:obj:1"/>
</file_test>

<file_object                                            id="oval:sample:obj:1">
    <oval-def:set>
        <oval-def:object_reference>oval:sample:obj:2</oval-
def:object_reference>
        <oval-def:filter       action="include">oval:sample:ste:1</oval-
def:filter>
    </oval-def:set>
</file_object>

<file_object                                            id="oval:sample:obj:2">
    <path              operation="pattern                match">.*</path>
    <filename          operation="pattern             match">.*</filename>
</file_object>

<file_state                                             id="oval:sample:ste:1">
    <owrite                                     datatype="boolean">1</owrite>
</file_state>
```

This example will evaluate to true if there are no files on the system that are world writable. The `<file_object id="oval:sample:obj:1"/>` has a `<set/>` that references `<file_object id="oval:sample:obj:2"/>` which will collect every file on the system. The set also contains a `<filter/>` that will include any `<file_item/>` that matches `<file_state id="oval:sample:ste:1"/>` which characterizes world-writable files. Lastly, the `<file_test id="oval:sample:tst:1" check_existence="none_exist"/>` references `<file_object id="oval:sample:obj:1"/>`, and with the specification of the `check_existence` attribute value `none_exist`, the test will evaluate to `true` if a matching `<file_item/>` is not found on the system.

This example can be optimized by only collecting a `<file_item/>` if it satisfies the `<filter/>` specified in the `<set/>`. Essentially, this means that only the world-writable files will be collected rather than every file that exists on the system. The second way that this example could be optimized is based off the `check_existence` attribute. In this example, the `<file_test/>` will evaluate to `true` if no world-writable files exist on the system. If at least one `<file_item/>` exists on the system, the test will evaluate to `false`. As a result, this can be optimized by short-circuiting the

collection process when the first world-writable <file_item/> is discovered. This is possible because the result of the assertion can be determined with a single <file_item/>. Both of these optimizations are valid as long as the collected objects are given a flag attribute value of incomplete.

However, it is important to note that with a flag attribute value of incomplete, on a collected object, a result of true or false cannot always be determined. According to the OVAL Language, with a flag attribute value of incomplete, it is unknown if additional matching items exist on the system and, as a result, the test must evaluate to unknown. This means that certain assertions such as check="all" are not possible because it cannot be said that all items match a specific state if it is unknown if additional items also exist.

During the discussion of the optimization capabilities using the flag attribute value of incomplete, the following questions and comments were made:

- Of what entity is the flag an attribute?
  Response: The flag is an attribute of collected objects in the OVAL System Characteristics document meaning it could be any object.

- What does one do when an object is used in more than one test with different determination characteristics?
  Response: You need to be aware of where this optimization is applied. In certain situations, this optimization could evaluate to a result of unknown.

  Response: In the case where the existence check is no matching files exist on the system, and your object is collecting every file on the system, you could optimize by stopping the data collection process once an item has been found and then setting the object's flag attribute, in the system characteristics file, to incomplete. If you try and reuse the object for an existence check, other than one where no matching items exist on the system, it would evaluate to a result of unknown.

- Has anyone attempted to apply this optimization in their tool?
  Response: One vendor indicated that they applied the optimization for one test and that the documentation provided the necessary information required to do so.

  Response: It makes more sense to apply this optimization technique in situations where the search space is large.

- A concern was raised that this optimization technique may not be as helpful in situations where the item collection is not performed at runtime such as when items are retrieved from a periodically updated database.

Optimization Capabilities for Version 5.8

With the release of Version 5.8 of the OVAL Language approaching, it is important to consider what optimization capabilities can be included if desired.

Proposal 1: Add a filter element to objects

The first proposal for optimizing item searches in Version 5.8 is to allow for 0-n `<filter/>` elements to all objects in the OVAL Language. Each `<filter/>` element will allow for the specification of an `action` attribute which specifies whether to include or exclude items that match the state specified in the `<filter/>` element. It is also important to note that with this change a `complete` object will be the set of items after each `<filter/>` has been applied to the items collected by the object's required entities. An example of this proposal can be seen below.

```
<file_object                                    id="oval:sample:obj:1">
    <path              operation="pattern              match">.*</path>
    <filename          operation="pattern          match">.*</filename>
    <filter                 action="include">oval:sample:ste:1</filter>
</file_object>

<file_state                                     id="oval:sample:ste:1">
    <owrite                                datatype="boolean">1</owrite>
</file_state>
```

The primary advantage to this solution is that the filtering capability is a well-known concept that was introduced to the `<set/>` construct in Version 5.0 of the OVAL Language. The implementation of this solution may also be very similar to applying `<filter/>` elements in a `<set/>` and could potentially allow for code reuse and abstraction. Since the solution would be applied across all objects, it would solve the problem of not being able to fine-tune item collection everywhere. In addition, the ability to specify an unbounded number of filters will provide a content author with more flexibility in determining which items should be collected as they can include or exclude as many states as needed. Lastly, the use of states provides a mechanism to specify multiple values using variables as well as specify ranges using different operations.

The most notable drawback to this solution is that the filtering of items does not make sense for every object in the OVAL Language. For example, objects that collect a single item (e.g. `<passwordpolicy_object/>`) and objects that collect small sets of items (e.g. `<interface_object/>`). In both of these examples, the ability to optimize item searches will not result in a significant reduction of the output file size or the resources required by tools to process the data. However, this drawback is mitigated because it is not required for a content author to include `<filter/>` elements in the object. Another disadvantage to this solution is that, while the `<filter/>` element allows additional flexibility, it also increases the risk of a content author specifying `<filter/>` elements that could cancel each other or even result in an object that does not collect any items (i.e. a `flag` attribute value of `does not exist`). Lastly, the introduction of an additional element to objects increases the complexity for content authors in that they need to determine when it is, and is not, appropriate to use this new capability as well as the risk, described above, that is associated with it.

During the discussion of the Version 5.8 proposal to add a filter element to all objects in the OVAL Language, the following question was raised:

- Can't you already do that with sets today?
  Response: The difference here is that when you use an object in a `<set/>` and apply a `<filter/>` to it, the object is still going to go and collect all of those items and they will be written to the OVAL System Characteristics file. This would allow you to remove them before they are written to the OVAL System Characteristics file.

Proposal 2: Add a filter and action behavior

Another option for optimizing item searches is to add `filter` and `action` attribute behaviors to the `<behaviors/>` element. Like the `<filter/>` element, the `<behaviors/>` element was introduced in Version 5.0 of the OVAL Language and is a well-known concept. This also aligns with the notion that the `<behaviors/>` element is a mechanism to provide a more granular definition of an object. The `filter` attribute will allow also for the specification of a state that will represent the items to include or exclude, as specified by the `action` attribute, from the set of items specified by the object. Again, this change means a `complete` object will be the set of items after the `<behaviors/>` element has been applied to the items collected by the object's required entities. An example of this proposal can be seen below.

```
<file_object                                          id="oval:sample:obj:1">
    <behaviors       filter="oval:sample:ste:1"       action="include"/>
    <path              operation="pattern"              match">.*</path>
    <filename          operation="pattern"            match">.*</filename>
</file_object>

<file_state                                           id="oval:sample:ste:1">
    <owrite                                    datatype="boolean">1</owrite>
</file_state>
```

The key advantage to this solution is that specific objects, those that collect large sets of items, can be targeted as opposed to applying a solution to every object even when it may not make sense. Also, since this is conceptually similar to applying the `<filter/>` element in the `<set/>` construct, it may present the possibility for code re-use and abstraction.

While this solution still uses states to specify the items to filter, it only allows for the use of one state which means there is less flexibility to fine-tune the collected set of items. At the same time, this may be beneficial as it will reduce that likelihood that content authors will create content that results in a collected object having a `flag` attribute value of `does not exist`.

During the discussion of the Version 5.8 proposal to add a filter and action attribute behavior to the behaviors element, the following question and comment were made:

- It appears to me that Option 2 is not as useful as Option 1 because it is restricted to only those objects that one thought were worthy of having a filter. Behaviors have always been an odd construct within the language and a straight forward filter that could be applied to some or even all objects would be more useful than behaviors that were only found in certain objects.

Response: An object may have a limited data set in most cases, but in some cases it may have a large data set. Thus, it makes sense to apply the filtering capability to every object such that it is available if needed.

- What Boolean logic is available for combining filters (AND, OR, NOT, etc.)?
  Response: If you have multiple filters, each filter will be applied sequentially to the set of collected items.

  Response: Is there set theory available to apply the union, intersection, etc. operations?

  Response: When you use sets in the OVAL Language, it applies all of the filters prior to the set operations (union, intersection, etc.).

Proposal 3: Add entity behaviors

A third proposal for optimizing item searches is to add entity behaviors to the `<behaviors/>` element that represent the state entities for the respective object. An example of this proposal can be seen below.

```
<file_object                                        id="oval:sample:obj:1">
    <behaviors          oread="1"          owrite="1"          oexec="1"…/>
    <path                operation="pattern                match">.*</path>
    <filename            operation="pattern           match">.*</filename>
</file_object>
```

The key advantage of this proposal is that the `<behaviors/>` element is well-known and conceptually the addition search controls, as behaviors, aligns with the notion of specifying a more granular definition of an object. In addition, this proposal will only target the objects where there is a need to actually optimize item searches (i.e. those objects that have the potential to collect large sets of items).

Out of the proposals, this is by far the most restrictive proposal in that it does not use states to specify the resulting set of collected items. As a result, the functionality of specifying ranges and multiple values will not be possible. It will only be possible to say that an item matches if its values align with those specified in the `<behaviors/>` element. In addition, since each set of behaviors will be different, the implementation will not lend itself to a clean and general solution. Lastly, this proposal will scale poorly. When specifying behaviors for an object, a decision will need to be made as to which state entities should be added. If every entity is added, the list of behaviors can become extremely large. For example, the `<file_state/>` has twenty-three entities!

During the discussion of the Version 5.8 proposal to add entity behaviors to the behaviors element, the following comment was made:

- The benefit of an approach like this or Option 2 is that you can say the things that people care about are a particular attribute, flag, etc. It is a broadening of the behavior notion whereas Option 1 is really taking states and generically applying a filter notion across everything. This approach would allow you to leverage your investment in state analysis in your interpreter

engine. If you don't have the investment or your design is different, this option, will allow you to say files are a problem so we are going to provide a targeted fix for files.

Why Can't We Just Add Additional Entities to Objects?

As outlined in the oval-developer-list discussion, there is a desire to add additional entities to objects that will allow for a more granular definition of the set of items to collect. An example of this is provided below.

```
<file_object                    id="oval:sample:obj:1"                    >
    <path               operation="pattern               match">.*</path>
    <filename           operation="pattern            match">.*</filename>
    <owrite                                    datatype="boolean">1</owrite>
</file_object>
```

The addition of entities to objects can be accomplished in two ways. The first way is to make the additional entities "required" which means they **must** be specified in the object. Unfortunately, this is not possible as it would invalidate existing content. However, this could be avoided by deprecating existing tests, objects, and states and re-creating new ones that include the additional entities. Unfortunately, this will result in the component schemas doubling in size. Lastly, this option introduces the problem of how to deal with entities where the value does not matter. For example, to specify that the value of the `<owrite/>` does not matter, it would require that the `<owrite/>` entity reference a `<constant_variable/>` that contains the values "0" and "1". This is not desirable. All of these issues can be resolved by introducing the new entities as "optional" meaning that the additional entities are **not** required to be specified in the object.

While the addition of entities, in objects, is technically possible in Version 5.8, it would represent a significant change in the OVAL Language. First, it will change the traditional notion of what an object is. An object is the minimum set of required entities necessary to identify a set of items on the system. The first option will break the notion of an object being the minimal set of entities required to identify an item and the second option will break that same notion as well as the notion that the entities in objects are to be required. Both options will also require significant changes to the schema and that tools support the additional functionality. The community was then asked to consider if this is something that should be done in a minor release?

During the discussion of the issue of why we can't just add entities to objects, the following comments were made:

- Could you just make the object entities nillable? That is the standard procedure for fields that we don't care about?
  Response: It is really a different concept. Nillable object entities are reserved for entities that form a hierarchical structure and it is necessary to ignore the lower level entities in favor of only collecting those that are higher up. For example, with the `<registry_object/>`, the name entity could be set to nil resulting in the collection of only the hive and key. You could come up with a definition for do not collect `<owrite/>`.

- Why are you suggesting that we make the object entities required?
  Response: We are just considering the different options.

- If you are identifying a set, you are still identifying the minimum set of attributes required to define the set.
  Response: By specifying the additional entities you are specifying a set.

  Response: Think of the path and filename as a composite ID for a single file on a system. Across all objects, we have been consistent in specifying how you identify a file, registry key, or RPM.

  Response: You can already use patterns in file to get a set. As soon you start dealing with sets, you need a filtering capability to identify the set accurately.

- How would we handle more complex logical filtering like world read and world write or user read and user write?
  Response: You would want a container to support that Boolean logic.

  Response: A state does that. If you look at the `<filter/>` option, a state has the child entities world read and world write or user read and user write. You would need one filter for world read and world write and another for user read and user write.

  Response: What if you wanted an OR relationship (one or the other, but not necessarily both)?
  Response: You would need an additional construct to specify the Boolean relationship between the filters.

  Response: You could still do that with sets.

- A question about Option 1, it does not seem like you need the path and filename in the object? A state contains that information. You could either declare a filepath, a path and filename, or a `<filter/>`. It is really a choice of those three.
  Response: It would be a big change in how some tools process content if an empty file object indicated that all files on the system should be collected.

  Response: It is a choice of one of those three. Either it is required to have a path and filename, a filepath, or a `<filter/>`. Any one of those three options would be sufficient and would not invalidate the current process.

  Response: That would be a very different way of using states.

  Response: I do not see why you need to restrict it that way. If you want to look for world readable files in a certain directory you need to specify the path.

  Response: You are not losing that capability, by not specifying the path, as it is available in the state.

- I think Option 1 is the most desirable of the three options. First, it is conceptually the closest to how people think about calling the potential target objects out by a `<filter/>` mechanism.

Second, it is directly symmetrical with what one can say about an object. All of the entities found, within a state, can be used to direct the object collection phase. Lastly, in the absence of a way to exercise of Boolean logic, you are restricted to a logical AND or a logical OR. One could choose both, but I strongly suggest that we surround it with something that allows the `<filter/>` elements to be combined together in a deliberate fashion as opposed to a single fashion.

- A benefit of Option 1 is that it is going to promote re-use. Rather than building multiple file objects, you could build a single `<file_object/>` and potentially filter it in different ways.
  Response: It does not require a pragma that says the optimization, which is not explicitly specified in the OVAL Language, might take place regardless. It is a way to build the capability into the OVAL Language as part of the Item collection process.

  Response: You would be able to fine-tune exactly what you are looking for.

- Would it be a big leap to have tool vendors support additional object entities?
  Response: One vendor reported that supporting additional object entities would not be difficult.
- I raised a question on the oval-developer-list regarding the fear of versioning. If a major version is required, what exactly is the problem with versioning?
  Response: I would say the biggest challenge right now is the maintenance tail we have. We have to maintain the Version 5 release which is driven by the fact that many of you in the room have products and tools that support Version 5 and are depending on it working as well as having incremental advances such that your tool can continue to process the latest Microsoft security advisories. I think it comes down to finding a way to get the Version 5 line relatively stable so that we are only doing minor tweaks. At which point, we can focus primarily on Version 6. It also means that we will stop working on Version 5.x which is hard to do.

  Response: I do not really understand the logic behind that. It seems that we have disbanded the notion of breaking changes, but breaking changes were the thing that would have precipitated a numeric major release. Outside of that, I am not sure that would. Do any vendors have thoughts on this?

  Response: Is your point that most of the changes that we have been talking about can be done in such a way that it wouldn't break?

  Response: Yes.

  Response: My counter argument is that it seems that most vendors care about the SCAP version and those are specified in 5.3, 5.4, 5.6, and 5.8. If instead of 5.8, it happened to be 6.0, it would not make that much of a difference in the sense that you would have to move from 5.6 to the next version. I do not see that as a huge difference. I agree with the backwards breaking reason for changing numbers, however, I don't understand why we are not doing it.

  Response: I would love rev more frequently and have minor versions more frequently such that we can fix issues in the language that needs to be cleaned up or fixed. We talked about this in the past and the issue is that we still have to maintain the Version 5 line for some period. I do

not see how we can start breaking all of the content that is out there. I would like to think that there is a split in the community between those who support SCAP, and are after SCAP validated, and those who are using OVAL because it solves the problem for their product and are not necessarily interested in SCAP or SCAP validation.

Response: Content is driving the concerns about doing major revs because major revs would exist in a namespace and would likely invalidate the OVAL content from a schema validation perspective. This puts us in a position where we will have to maintain two sets of content for some period; potentially two to three years. While there is not a lot of cost in doing these types of revs, there are many hidden costs when it comes to content that is broken from a backwards compatibility perspective. This is one of the reasons why there has been a historic concern regarding a major versioning change.

Response: You use the word broken, but is it broken simply because it no longer adheres to a new schema?

Response: Yes.

Response: Assuming that one were to carefully devise changes to the OVAL Language such that they did not invalidate the semantic content of the older OVAL content, providing an XSL transform would allow one to move the older content to a newer version of the language and would obviate some of the concerns that people would have with the adoption of newer easier to use schemata.

Response: The issue is if we wanted to come up with a major version, an XSL transform to auto transform the content forward would help. However, I have heard from vendors that there is a cost associated with re-queuing the content and, if something does not work, the customers will complain.

Response: That is good if you embed the XSLT in the engine.

Response: Why even transform it, the content identifies schema.

Response: To make your product support both versions.

Response: It is an ongoing maintenance problem. There may be a desire to upgrade to newest version of OVAL and having a path forward through a transform or upgrade process is desirable.

Response: We will talk about content maintenance in more detail during the next session.

Response: If we are talking about revving more frequently, there are many hidden costs that we should be thinking about such as updating documentation, writing reference implementations, and the time that it takes the community to review and implement the schemas to ensure there are no problems. All of these things have to be repeated every time we do a rev. The more frequently that we rev, the more overhead we experience.

Response: Vendors have the same issue from a QA perspective. The features we have are going to be cumulative, but if we have to do three releases to get there, it is going to cost much more. In addition, I don't know how frequently our customer base can even upgrade their products.

Response: Determining what the appropriate iteration level should be for these types of efforts is a complex problem. There is a great deal of impact one way or another.

- What is the immediacy of the need that prompted these suggested changes to the OVAL Language? If it is an immediate need, due to very large result sets, this could be added as an optional item. If people did not implement it, they would simply carry on by ignoring that particular thing. That could be a caveat of some vendors' implementations. From a technical perspective, I regard the application of a filter, at the object collection time, as almost directly symmetric with the analysis of collected objects from the standpoint of applying a test or criterion. I don't think this would be particularly disruptive to most implementations of OVAL and it would be beneficial because it would allow people to keep their result sets down as long as they were aware that such a feature existed.

Optimization Capabilities for Version 6.0

Unlike adding optimization capabilities in Version 5.8, Version 6.0 is not restricted by the requirement to maintain backwards compatibility as defined in the OVAL Language Versioning Methodology which can be found at the following link:

http://oval.mitre.org/language/about/versioning.html

As a result, there is a chance to learn from our experiences with the language over the years and improve and simplify the language at the same time.

To promote discussion, the community was asked to consider the impact of making all entities searchable in the context of Version 6.0.

Impact of making all entities searchable

- Do we really need states?
  - What about filters and ranges?
  - What would a test look like?
- Could results be based on the existence of items?
- Do we need object entities to be required?

During the discussion of the issue of looking into Version 6.0, the following questions and comments were made:

- Does anyone have any thoughts on getting rid of states?
  Response: We talked about dual role of checks against states. It does not only check that you match a state but also the existence or non-existence of a particular item. If you are not searching for a specific thing and then comparing against state, you lose that ability to

differentiate between whether or not something exists and whether or not it is the particular configuration that you care about.

Response: You would only be collecting the particular things that you cared about. If you found an item, you would know whether or not it existed and if it matched.

Response: But, I would not get the password length is 12. If I filter from the very beginning, for the length is 12, I will not get the item that says the length was 8 instead of 12.

- I am a big proponent of not making backwards breaking changes if we do not have to. I don't see any of these as required.
  Response: These are not required, just things to think about.

  Response: I tend to agree. It doesn't seem that any of these backwards breaking changes would advance the language.

Wrap-up Discussion

To conclude this discussion the group was asked the following questions:

- Are there other ways to optimize item searches?
- Is this a problem we would like to address?
  - When would we like to make a change?
  - Do we need item optimization for every object?
  - How flexible does a solution need to be?
  - Will solutions be feasible to implement?
- Other questions, concerns, comments

In response, the following comments were made:

- Does anything stick out as other ways to optimize item searches that we did not talk about today? Any different ideas?
  Response: It would be useful, if we are to do optimizations, to have an explicit or implicit ordering of things that are evaluated. The best you could do, if you come up with one or more optimization techniques, is to do a partial ordering of all of the tests or the object filtering. It would be possible to put an optimization into an OVAL interpreter if you said that you would always restrict yourself to objects that met or did not meet a particular test implicitly without saying it explicitly. For example, if the fundamental test in question was that all files on a system exhibit a certain state, there is no reason that you would have to slavishly collect all of those objects and then subject them to the test if you simply folded the test into the object collection phase.

  Response: That is essentially what the filter is doing.

Response: But, it would not, it is not necessarily that explicit if you were to just go ahead and do it implicitly. You would not get a lot of opposition from people that didn't want to see the 15 or 30 thousand files present on the system for each and every object.

Response: But, in doing that, you'll lose the ability to separate the object collection from the object evaluation.

Response: Exactly.

Response: That is an option that one tool could make.

Response: It seems you also lose the location of the specification of where the test evaluation actually occurs. You would have to put an object in the criterion.

Response: The OVAL Language is a functional specification and not a procedural specification. There is nothing that says that the objects must be collected. One could easily traverse an OVAL Definition document; perform each criterion one by one, evaluate the underlying tests, and come up with the same result. It's not necessary to postulate the existence of a large set of collected objects and the like.

Response: Right, but even in that case, you would still need to have a test that references that object or you would never get to the point where you can evaluate it.

Response: I understand that. It is not necessary to postulate the existence of an object in the collection phase followed by the evaluation of state criteria which means that there's no reason not to combine them. There is no language restriction that says one cannot combine the object collection phase with the object evaluation phase.

Response: A concern was raised that, downstream, if someone takes the results from a tool that has done that optimization and has not collected all of the items because they realized that they only needed 10% of the items. If you tried to walk that results document, you would see the test passed the check which said `check_existence = all`. This says that all files exist and even though you have not actually recorded all in the document. As a result, there would be inconsistencies which we would need to figure out how to handle.

Response: That's exactly what some people are asking for as described earlier in the afternoon.

- Short circuiting is another optimization. It does cause problems when you get to the remediation phase except in the case of the precondition. In that case, because OVAL does not have a precondition, you are evaluating the criteria and then you have to evaluate the rest and it does not have value there. In other cases, for remediation purposes, it does make sense to evaluate it. A way to short-circuit only sometimes would be beneficial, but I do not have an answer right now.
- Do we need to do something in Version 5.8 as far as adding a filter element? Is that something that's necessary to do?
  Response: The community stated that there was a need to add a filter element in Version 5.8.

Response: A concern was raised that if you do not do something, as general as the filter mechanism, you increase the idiosyncratic nature of the language. It becomes much harder for people to understand what they are supposed to be doing because on one test they may be able to use a behavior and others they may not whereas the general solution of filtering at object collection is far easier to understand and more closely approximates the real word thinking of someone crafting a particular test.

Response: It does have some potential impact on being able to report on the exceptions. If reporting on the exceptions is important, you need to make sure that the objects that are the exception aren't included in the results.

Response: One can still filter and retrieve all world writable files without necessarily producing the list of all non-world-writable files.

Conclusion

At the end of the discussion, it was concluded that the need to optimize item searches is necessary and should be addressed in Version 5.8 of the OVAL Language. Out of the proposals to optimize item searches in Version 5.8, the first proposal of adding an unbounded `<filter/>` element, to all objects, received strong support from the community. However, it was also made clear that there is a need to discuss and provide a mechanism for logically combining `<filter/>` elements in an explicit manner.

Least Version Principle for OVAL Content

Background

The OVAL Repository contains content contributed by members of the community, and provides that content for tool and community consumption. The discussion on the versioning policy started with an overview of the current policy.

Currently, the OVAL Repository only serves one version of the content, in the latest version of the language. With every new release of the OVAL Language, the content in the repository is moved forward as well. No actual changes are made to the content, and deprecated language constructs are not removed. This policy is easy to maintain, and with one version of content, it encourages tool compatibility with the latest version of the language. The downside is that there is no history of an OVAL Definition captured in the repository, and the burden is placed on the user or tool to figure out how to handle content with a later version number.

There are several key points to consider when evaluating any changes to the versioning policy:

- How difficult will it be to keep the repository up to date with accurate versions?
- Will there be a significant investment required in tools and process to handle the new policy?
- Will there be any burden placed on content creators?
  One of the primary goals of the OVAL Repository is to make content creation as easy as possible, to encourage a large content submitting community.

- How will the new policy affect tools and backwards compatibility?

- Will a change affect vendor tools, the amount of time it takes to process content submissions, or the performance of the OVAL Repository site?

The OVAL Repository also serves as a model for other content repositories, so any proposed changes should be considered from the perspective of what a general OVAL repository should support.

Proposals

The proposal presented to the group was to determine the least version for each item (definition, test, object, state) when importing to the repository. There were two options presented for how to communicate this information: either at the document or definition level. Having the least version at the document level would not change the language, and would require no changes to tools. However, in a large document, it may potentially cause a high least version because of just one definition, even when the rest of the definitions have a lower compatible version. Marking the least version on each definition would require a small change to the language, and would require tools to change the way they process OVAL Documents. This would be a more granular and accurate way to communicate version information in the document.

Discussion

A question was asked regarding how content was stored in the repository after processing. An answer was given that we store both the individual parts of the content, as well as the entire XML fragment, to allow a fast search capability, as well as fast retrieval of XML.

The suggestion was made that instead of validating against all schema versions, perhaps we should just compare the differences between versions. This would require having an algorithm for tracking changes from version to version (that is 5.6 -> 5.7, etc.)

The question was asked if this issue only affected major releases, and the answer was given that it indeed affects minor versions as well.

A comment was made that NIST has seen and dealt with this issue by schema validating content to determine least version, in a similar manner to how MITRE is proposing.

A suggestion was made to mark content only at the definition level, understanding that it would have an impact on reusable items like test, objects, and states.

A comment was made that the use of namespaces could alleviate some of the issues here by documenting the different versions at the XML fragment level.

Conclusion

The consensus was that there was a large benefit to implementing the least version principle, as it would help SCAP-validated tools use the content in the repository. There was not a clear consensus on specifically what method to use, and there were several additional features desired by the community. Also, there was a suggestion to make multiple versions of the repository available; specifically, to have an SCAP stream that is compatible with the current version of SCAP. It was noted that the benefit of

these additional features would be weighed against other priorities, and that discussion would definitely continue in the OVAL Community Forum.

Application Specific Detection

Background

The OVAL Language currently contains two application-specific schemas: SharePoint and Apache. The SharePoint schema exists to provide a method for accessing SharePoint data as previous mechanisms found within OVAL were found to be inadequate. The Apache schema exists to answer one question: is a particular version of Apache installed?

The Apache schema documentation for the httpd_object states:

> "The httpd_object element is used by an httpd test to define the different httpd binary installed on a system. There is actually only one object relating to this and it is the collection of all httpd binaries. Therefore, there are no child entities defined. Any OVAL Test written to check version will reference the same httpd_object which is basically an empty object element. A tool that implements the httpd_test and collects the httpd_object must know how to find all the httpd binaries on the system and verify that they are in fact httpd binaries."

Because the httpd_object refers to all httpd binaries on the system, a system scan would need to be performed in order to discover all binaries of the name, "httpd". The method and scope of the scan is not defined and thus left up to the implementation to define. For example, one implementation may only scan the running processes, while another implementation may only scan the local file system while omitting mounted network attached devices. The consistency of results between implementations is at risk and cannot be guaranteed because of this.

This topic has been recently discussed on the oval-developer-list. The archives of the conversation can be found here:

> http://making-security-measurable.1364806.n2.nabble.com/apache-httpd-test-tp4985454p4985454.html

The primary suggestions for addressing this issue were:

- Provide an expected path to search under via attribute or child element on httpd_object elements
- Leverage platform-specific mechanisms for verifying installed software (the Windows Registry, RPM, DPKG, etc.)
- Remove application-specific schemas that define tests which could be performed using conventional OVAL mechanisms

The problem of handling version information for specific applications extends well beyond Apache. As such, the existence and purpose of application-specific schemas needs to be re-examined.

Discussion

During the course of discussion the following comments were made:

- In general, the idea of removing or disallowing the creation of application-specific schemas is bad. SharePoint demonstrates the value of application-specific schemas by providing methods for accessing data that was once inaccessible. However, as n the case of Apache, I do not see a problem with removing application-specific schemas which define tests that can be performed using OVAL constructs and best practices.
- OVAL should encapsulate a series of trusted commands that can be invoked on a particular platform for gathering data about a particular application or platform.
  Response: There are obvious security issues with allowing OVAL to invoke scripts or binaries.

  Response: More than likely you are using OVAL for checking for the existence of a CVE or vulnerability so by invoking a particular binary, you could be exposing the system to a compromised or vulnerable binary. I am in favor of having an application-specific schema and checks because sometimes you can't leverage package management systems for discovering versions of installed software. For example, RPM doesn't detect the version of Apache that is bundled with some Oracle software.

  Response: So given the case that you cannot always trust a binary to be executed to gather information, the intelligence for determining the vulnerable state of an application and then magically pulling out the version information must be built into every OVAL-compliant tool. Is that feasible?

  Response: We cannot leverage utilities like strings, or rpm, or dpkg to pull out this information. We can't require tools to decompile binaries and analyze the code. The underlying problem is with the release and installation procedure of applications on various platforms; it's a problem that OVAL is not intended to fix.

- As we have SCAP encompass more and more applications, we will require more application-specific schemas. Relative to this issue, at NIST we have the NSRL (National Software Reference Library) which is a database of hashes for applications. They are congressionally chartered to purchase and hash software. We are trying to work with them to link up CPE information and these hashes.
  Response: Apache is a locally-developed product in essence, because most users who install Apache compile it themselves so as to install particular modules and options. As such, an organization would need to have an internal database of hashes for their internal software configurations.

- OVAL cannot go on to write application schemas for each requested application. The language would grow far too large. We need to deal with this now while we only have two schemas.
- We need to address these schemas on a case-by-case basis and under a lot of scrutiny. SharePoint demonstrates the value of application specific schemas, and others may come up in the future.

Conclusion

In general, schemas like the Apache schema were seen as lacking clarity and thus, should be removed. It was felt that if application information could be gathered using conventional methods content authors should use them instead of proposing a new schema. However, as is the case with SharePoint, if a schema adds functionality to the language and allows once inaccessible information to be gathered, a new schema should be proposed and added to OVAL.

Moving forward, the current Apache test will be deprecated and removed in a later release, pending approval. Current content that uses the http_test should be updated to leverage existing OVAL mechanisms for binary discovery.

Removing the Apache test and rewriting the test using conventional methods guarantees the consistency of OVAL Results among different implementations. By removing ambiguity and requiring authors to provide scoped information, we can place more trust in the content being consumed.

By addressing the issue of application-specific schemas early (while there are only two) we establish a mindset for proposing and adding application-specific schemas to OVAL in the future.

Adding Datatypes

This topic was not discussed and has been raised on the oval-developer-list. This discussion was planned in case there was extra time on the agenda.

Tuesday June 15th

Cross-SCAP Standardization

Dave Waltermire led this session and announced he planned on covering more than just SCAP Standardization, but also cross-protocol standardization, including ideas from the community on improving specification standardization, core schema standardization, a design pattern on request-result reporting, and identifier standardization.

Cross Protocol Standardization and Architecture

The first topic addressed was standardization using a common design pattern over multiple protocols. To standardize the way in which requests and responses are handled from a data model perspective, Dave mentioned three endpoint protocols (i.e. protocols that interact directly with assets): Security Automation Protocol (SCAP), Enterprise Remediation Protocol (ERAP), Event Management Automation Protocol (EMAP). Dave suggested that there is a common design pattern that is used to build endpoint protocol stacks, i.e. a composable architecture which starts with smaller building blocks to build up functionality as you get higher up the stack.

In designing the endpoint protocol, Dave focused on the request / response model, where the request is the action to be performed and the response is the results of performing that action. The design should be highly modularized to enable adaptability of use cases. Conceptually, the Request / Response Model would contain an Action Request and an Action Response. The Action Request would indicate the action or operation to be performed on the asset and the format of the results to be produced: in SCAP this would be the SCAP content that is to be assessed on the host as well as OVAL or OCIL or XCCDF directives to indicate the level of detail for results. For ERAP, the desired remediation, in the form of a Common Remediation Enumeration (CRE) identifier, should be included as well as meta-data that might describe additional processing or decision-making that would need to be made with regards to the remediation action. The Action Response would be the result of performing that action and it would capture the related results from that request. It would focus on identifying the actor that responded to the request and the asset that the request was targeted toward. In SCAP, this would be assessment results, the typical SCAP-validated results that currently get produced. In ERAP, it might include what the disposition of the remediation action was, whether it was applied, or failed, or a boot is required, etc.

There is currently no specification for the Action Request Format. The closest is the notion of an SCAP data stream, which is currently included as a bundle of XML files that generally get zipped up and passed around, although there is a new proposal of a super-XML instance that contains all of the pertinent XCCDF, OVAL, and OCIL content, so a single document could be used instead. Similarly for ERAP, a list of CREs could be used, or a more elaborate step-by-step listing of actions that need to be taken. Results directives could be used to negotiate the level of reporting for XCCDF, OVAL, and OCIL; the result directive could also be used for some basic negotiation of what assets identifiable data points are to be

captured. Suggestions from the audience as to what should be included in this model were provenance information with digital signatures and a run-time instruction set that could tailor a bundle interactively.

Dave then presented a notional representation of the SCAP Request payload, which included each of the individual documents in an element – any of the elements could be extracted into their own separate instance, if that was desired. Advantages of this model are that the SCAP use case can be specified (by specifying a string that states which use case is being supported); the SCAP version can be specified; it allows additional Schematron rules can be written to enforce referential integrity between documents, and it allows a hash to be calculated over the entire bundle. There were some questions about the benefits of specifying the use case in the content. There was also concern about generating a CPE dictionary for each action request to be sent, and the possibility of making it an optional component was discussed. A suggestion was made to allow content to be included in the bundle and other content be included by reference. A question was posed as to whether it was worth standardizing on such a light-weight wrapper.

The notional action response model was introduced and it has been designed to include a Report Request Information Block (for provenance and continuity with the original request), the Asset Subject Identification (to describe what the subject was for the action), and the Result Block, of which one or more are allowed. A question was asked as to whether only one device assessment in each payload, and it was responded to that there are discussions as to whether allow multiple assessments per payload. The Result vetted data will contain when the results were generated, what tools were used to collect the results, etc. – similar to what exists today in OVAL and OCIL Generator Elements. It was pointed out that OVAL Results and OCIL Results might not be available at the same point in time, in which case they would have to be contained in separate payloads. A part of the Result Block is the Asset Source Identification, which leverages the identifiable information described in yesterday's OVAL discussion. The difference between the Asset Source and the Asset Subject gives context to the source of the information relative to the results being produced. An attendee pointed out that currently Assessment Results Format (ARF) allows you to package multiple asset reports into a single document, which is a very useful feature. The Result Payload in the Result Block would be a lightweight bundle that contains the XCCDF, OVAL, and OCIL results. There was a question as to why it would be necessary to include OVAL Results along with XCCDF Results – the OVAL Results wouldn't add much in this circumstance. There was a question about the timeline for these concepts and Dave replied that these ideas are actively being worked at NIST and the hope is to include them in SCAP 1.2, which would mean validation would start in early 2012. In response to a question as to whether aspects of the model would be optional or required, Dave said it had yet to be determined and turned the question back to the attendees, LtCol Wolfkiel suggested that it should be optional for products to exchange the bundles as described in Dave's presentation. In response to a question, Dave explained that this model essentially breaks ARF up into modular components. Mike Kinney made the point that ARF is actually only a response language and Dave acknowledged that the complementary language to prompt the ARF response. LtCol Wolfkiel mentioned that that was the intention of Policy Language for Assessment Results Reporting (PLARR). In response to a questioner, Dave described that the SCAP Result Payload model he was describing can carry a variety of payloads and is not restricted to XCCDF and OVAL results.

Jon Baker started a discussion about the organization of results from OVAL and XCCDF – should there be another standard to organize results? Perhaps this new standard could be focused on use cases. Dave endorsed the concept of formulating a Results Working Group to help refine these ideas.

Specification Standardization

Dave next addressed the topic of the standardization of specifications, and stated that NIST has developed plans to create a common specification template so that all documents could have a common look and feel, thereby making the documents easier to read. Were all specifications to have a common organizational structure, then there would be a consistent presentation of content, less effort would be required to understand the information, and (potentially) there would be more complete information in each document. It's also believed that a specification template would allow authors to create useful specifications in less time. Dave then proposed a general outline for the template which included the following sections: Introduction; Terms, Definitions, and Abbreviations; Conformance; Relationship to Existing Standards; Conceptual/Data Model Overview; Implementation and Binding; and Appendix. A discussion about binding ensued and the point was made that having multiple bindings defined for a standard may be beneficial, but with the addition of more bindings, we introduce the need to define how tools that support different bindings will interoperate and negotiate a binding of choice for exchanging information. A suggestion was made to move the Use Cases section from the Appendix up into the body of the document because the complaint is frequently made that standards are trying to do too many things. However, Dave thought the Conformance section might address that need appropriately; also it can be very challenging to write good use cases. LtCol Wolfkiel warned against requiring all products from having to support all uses cases. Another attendee advocated for keeping the Use Case section in the Appendix because it will tend to grow over time. There was a question asking for clarification of whether "Purpose and Scope" in the Introduction was intended for the document or the standard. Another questioner wondered whether the document could be published similarly to the way that the W3C does.

Dave briefly touched on the topic of separating the conceptual model from the findings – it might be a "best practice" to write a separate specification for the logical model and the bindings. Dave felt this would be a situation to make a decision on a case-by-case basis. Dave then introduced a notion of a similar concept being applied to identifiers, syntax, and semantics from enumerated list authorities – i.e. define what the identifier looks like in one specification and then define what the enumeration is in another specification. There was no reaction from the attendees to this suggestion. Dave mentioned that this model is being followed by CPE.

Schema Standardization

Next Dave moved onto the topic of Schema Standardization.

Core Schema Use

Dave began by discussing the concept of a Core Schema, which would contain things like common references to be used across specifications, with a focus on core concepts that every specification uses, allowing us to consistently express those concepts. Potentially, this would allow us to have an easier time developing because common code could be written against common concepts. However, it would

introduce a lot more complexity in how the schemas are managed. Updates to the core schema would need to be coordinated with each of the individual specifications. Possible candidates for core schemas include OVAL and OCIL generator elements; XCCDF, OVAL, and OCIL data feeds; and identifier patterns. Mike Kinney asked whether this model would be just for SCAP or a full beta model for other NIST protocols too. Dave suggested that the effort would start with SCAP, but then perhaps could extend to others. Jon Baker suggested that there might be some value in breaking up the SCAP schemas into smaller, more granular pieces, which might make it easier to implement.

Identifier Conventions

Dave then turned the discussion to Identifier Conventions. Currently there exists two basic identifier schemes: URI Basic Syntax (used by OVAL, OCIL, and CPE) and a fragmented expressions (used by XCCDF, CVE, and CCE). When trying to develop XML schemas using various data types, challenges are presented by the specific formats of the various identifiers. There has been some encouragement to use the xsd:ID format, but since it is derived from non-colonized name, so the data types that are URI-based cannot be used. The same can be said for xsd:NCName. URIs have the capability to reference some fragments. Many implementations of RDF require that it follow a URI general syntax; i.e. colons after the first scheme parts are not valid, which causes many tools to reject the URI-based IDs. In an effort to try to standardize how various forms of identification are represented, Dave proposed a few different conventions for various types of identifiers.

- For CVE and CCE, use the fragment convention
- For XCCDF, OVAL, and OCIL, use the URI Basic Syntax

There was some discussion over the benefits of this scheme, which was made more confusing because of inconsistencies on the slides.

Remediation

This section is a summary of the remediation discussions held as part of the Security Automation Developer Days. The purpose of the session was to discuss in some technical detail efforts at developing emerging standards in the area of enterprise information security remediation, with a particular emphasis on progress made since the Winter Developer Days 2010 event held February 22-24, 2010 at NIST in Gaithersburg, MD.

Enterprise Remediation Automation Protocol (ERAP)

The first remediation session was a brief presentation by Chris Johnson of NIST introducing the Enterprise Remediation Automation Protocol (ERAP). ERAP is a collection of emerging open specifications intended to standardize remediation activities across the enterprise. It is envisioned as including community-developed specifications addressing such capabilities as:

- uniquely identifying remediation actions
- tracking supplemental information about remediations

- describing remediation policy
- controlling remediation tasking
- expressing precise machine-readable descriptions of remediation actions
- encoding remediation results.

ERAP is similar to, but distinct from, the Security Content Automation Protocol (SCAP).

A projected timeline was presented of the anticipated release of various publications related to ERAP and its underlying specifications. In calendar year 2010, development efforts are expected to concentrate on a standardized identifier scheme for remediation actions (referred to as the Common Remediation Enumeration or CRE), codifying useful additional metadata for remediations for various use cases (Extended Remediation Information or ERI), and a control language for communicating specific remediation tasking. For further information, refer to the presentation.

Remediation Research Project Report

Mike Kinney of the US Department of Defense gave a brief overview of a DoD research effort to build a functioning reference implementation of a remediation system following the ERAP model. This proof-of-concept will include a remediation manager tool which will consume remediation policy documents, including standardized identifiers and additional metadata for remediation actions, and allow the user to make remediation decisions in the context of that policy and standards-based assessment results (e.g., apply a specific patch on 5 particular end systems). The remediation manager will then have the capability of enacting those decisions as remediation tasking to various software remediation agents. The agents will in turn respond with the results of attempting those remediation actions.

The proof-of-concept system being developed will use the open specifications as defined in ERAP where available. However, the timeline of the research effort is more aggressive than the naturally slower pace of consensus-based community standards development, so the project expects to effectively develop prototype versions of several of the proposed ERAP specifications. Where possible, these prototype formats and other lessons learned will be shared with the community to assist in designing the consensus-based specifications.

During the discussion, several participants expressed an interest in focusing near-term efforts on developing an OVAL analog for remediation: the low-level, machine-readable language for precisely expressing what remediation steps to take on an end system. While ERAP does propose such a specification, tentatively referred to as OVRL, neither the ERAP roadmap nor the DoD reference implementation effort currently include plans to develop OVRL in 2010.

For further information regarding this DoD remediation research effort, refer to the presentation.

Common Remediation Enumeration & Extended Remediation Information: Technical Issues

Matthew Wojcik of MITRE moderated a discussion focused on technical issues arising from two of the proposed ERAP specifications, the Common Remediation Enumeration (or CRE) and Extended

Remediation Information (or ERI). While both CRE and ERI are emerging specifications still very much in development, they are the most mature of the proposed ERAP components. They have been discussed in a number of previous public forums, including the 2009 Developer Days event held at MITRE, the 2009 ITSAC conference in Baltimore, MD, the Winter Developer Days 2010, and a community teleconference devoted to CRE held April 1, 2010.

The session began with an introduction to CRE and ERI. CRE is intended to provide standardized identifiers for remediation options, a similar concept to CVE or CCE. ERI, analogous to National Vulnerability Database (NVD) entries for CVEs, defines additional data about CRE entries which may be necessary to support common remediation workflows. Definitions of terms for purposes of the discussion were given. The identified use cases for CRE and ERI were briefly described, and the basic components of CRE entries and ERI records were presented. CRE and ERI examples followed.

A participant asked whether CREs are associated with the platform where the indicator resides (CVE, mis-configured CCE, etc.), or where the remediation is applied. Examples include firewall rules or other network device configuration options that may prevent or mitigate various issues, or Windows Group Policy settings which are enacted on an Active Directory server which may be a different platform than the end system which needs to be remediated. This is an issue which had not previously been considered, and will need to be addressed in the future development of CRE and ERI.

The concept of Content Decisions, a term borrowed from CVE and CCE, was introduced. For an identifier system like CRE, Content Decisions (CDs) are the documented editorial policies that define what is in scope for the identifier system, and how to split or merge issues when creating entries and assigning identifiers. Since CRE is a the core of the ERAP model (much as CVE is at the core of standardized vulnerability management, and CCE forms the basis of SCAP configuration management), CRE's Content Decisions will have a significant impact on ERI and the other proposed ERAP specifications.

Several previously-discussed CRE Content Decisions were reviewed. These Content Decisions, already considered adopted unless compelling new arguments could be raised, were presented to ensure the participants had a shared understanding of CRE in order to move on to the consideration of new issues. While there was discussion of a number of the previous CDs to clarify their intent or effect, all are considered to stand.

The session then moved to consider issues for which there were not yet accepted Content Decisions. The goal of the discussion was not necessarily to reach consensus on any topic at this point, but to have a preliminary conversation to aid in drafting future proposed CDs.

Various aspects of CRE parameters were discussed. Consensus has been, and continued in this session, that CREs should be parameterized, but many details remain to be decided.

The first topic related to remediation parameters was a discussion of whether system objects should be parameterized, or only the characteristics of a system object included in the definition of the CRE itself. For example, when considering setting file access permissions, should the file be specified as a parameter along with the desired permissions settings, or should separate CREs created for each file of

interest, and only the desired permissions be included as a parameter? Various arguments were presented for each approach, with no clear consensus reached. Further consideration and discussion of this topic are required, with more specific examples and with additional input from those who write remediation policy and report on remediation compliance.

The question of whether CRE parameters should be expressed in literal or conceptual form was debated—whether parameters should reflect how settings are actually enacted at a low level on the end system, or instead describe the effect in more human-friendly terms. For example, "0 or 1" vs "enabled or disabled." Participants raised valid use cases for each option, but each was also argued against in favor of reduced complexity. Concerns of whether the necessary information is available to provide either type consistently were also discussed. In the end, consensus seemed to be that both literal and conceptual expressions of parameters are needed at various points in enterprise remediation workflows, and having a standard place record both, and perhaps also to map between them, might at least encourage both types to be provided when possible.

There followed a discussion of the fact that some remediation statements imply simple parameters which are straightforward to address (e.g., password lengths, screensaver timeouts), but others are more complex or are abstractions that present greater difficulty (e.g., "disable" a Windows service; would setting the startup type to Manual satisfy? Does the running state also need to be changed?). Some commonly discussed "parameters," such as installing vs uninstall a patch or application, have different methods or pre-requisites, and might be assigned different CREs through the application of other Content Decisions. Consensus seemed to be that many complex or abstract parameters, if allowed, would lead to ambiguities or potentially differing implementations, which would largely defeat the purpose of CRE. Therefore, a proposal was made to include parameters in CRE where straightforward and unambiguous, and split issues into separate CREs where problems arise. Additional specific examples are also needed to advance this discussion.

The next topic raised the question of additional options to remediation methods that are not core to the most pertinent change to the system's security model. For example, "quiet" or "no uninstall" options for patch installations, or installation directory for application installs. Should CRE or (more likely) ERI include such information, perhaps optionally? Some participants felt that this is more detail than is necessary to include. Others thought that it would be useful (and perhaps necessary) at various points in remediation workflows, and including it as an option in ERI could help standardize its presentation. Additional discussion will be necessary.

Several other topics regarding CRE and ERI were briefly introduced but not discussed in detail, due to time constraints. These will be discussed in future community forums. Refer to the presentation for details.

Future community input on CRE and ERI will be sought on the NIST emerging-specs email list. Any interested parties should be sure to subscribe to that list. See http://scap.nist.gov/community.html for how to join the list.

Wednesday June 16th

Digital Trust

Harold Booth, of NIST, led the session on Digital Trust, which was a follow-up from the session at the February Developer Days. Digital Trust in SCAP allows SCAP data to be digitally signed. The goal of this effort for the short term is to find a format to express signatures in a common way so that the tools can have an expectation of what this feature would look like and interoperate on that level. In the future, the goal is to have a way to handle the compositionality of the content – there are issues with checking the digital signatures of remote references and also the issue of tailoring.

Use Cases

Harold began a discussion about various uses cases

Content Use Case

(input) A content consumer needs to verify authenticity of a content stream. For example, if a user gets content from the National Checklist Program web site, and it is claimed to have been composed by Vendor A, a digital signature attached to the content can verify that claim.

(prior knowledge) Re-establish trust to content based upon prior knowledge. For example, if a user has already received some content and now wants to check whether content she that received at a later time or from a different source is actually identical content, a digital signature can verify that check.

Content Quality Assurance

An individual or organization signs content to assert confidence or trust in content. If content is generated by Vendor A, but then Vendor B tests the content and now wants to put its "seal of approval" on the content, it can do so by adding its digital signature.

Compositional Content

A content consumer would like to know and verify that a content stream is composed of multiple source streams. If Vendor C combines content from Vendor A and Vendor B, and perhaps decides to augment the resulting content. The content from Vendor A and Vendor B can be signed separately from the newly created block, thus providing provenance information for the two original sections of content.

Results

An organization needs results signed at the point of creation in order to verify authenticity of results.

Results (expanded)

An organization needs results signed with source content identity and/or target identity at the point of creation in order to verify authenticity of produced results. It is important to distinguish who needs to sign the results; for example, a machine would need to sign results from an OVAL check, but a person would need to sign the responses to an OCIL questionnaire.

Aggregated Results

Aggregation tools need to combine results and sign aggregated results. When combining results from different sources, it may be important to be able to differentiate between the various results components and their sources. Digital signatures make this possible.

Current Notional Digital Trust Model

Harold proposed for SCAP to use the XML Signature Syntax and Processing standard. This is a W3C Standard, which has also been adopted by IETF (IETF RFC 3275). This standard is specialized to handle XML data and is capable of canonicalization and transformation; it defers applications for verification logic; and it has hooks for X.509 certificates and PGP keys.

Example Signature

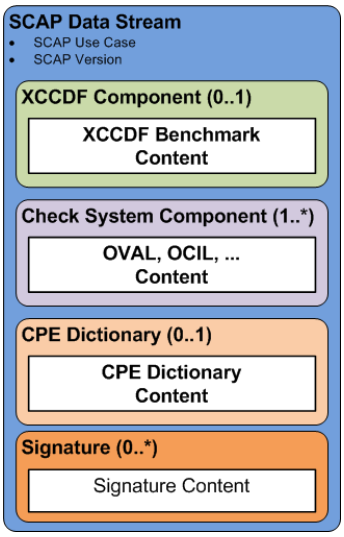
Harold then displayed two XML Signature examples. There was a question about the impact of digital signatures on content re-packaging. Harold felt that this is one of the open issues that need to be solved.

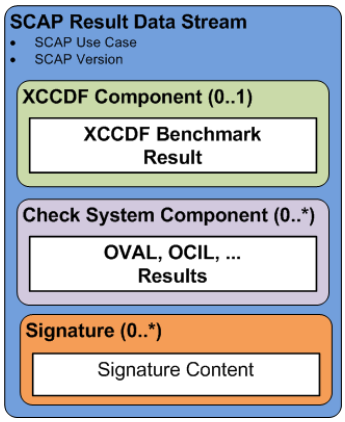
Implementation Issues

Based upon the recommendations in FIPS 186-3, Harold pointed out that the following algorithms and parameters need to be adopted:

- For RSA
  - 2048-bit key
  - SHA-256
  - PKCS #1.5 padding
- For Elliptical Curve Digital Signature Algorithm
  - 256-bit Prime Curve
  - SHA-256

Harold then presented a depiction of the signing of an SCAP Data Stream and an SCAP Result Data Stream, which would contain an *enveloped* signature.





Harold briefly described the difference between an *enveloped* digital signature, an *enveloping* digital signature, and a *detached* digital signature. An *enveloped* signature would be embedded within the XML document containing the signed content; whereas an *enveloping* signature would be over XML content that would be found within an object element of the signature itself. The consequences of an enveloped signature are that the document must have a placeholder to hold the signature, the format of the content must be same for both signed and unsigned content, and that the signature and content are coupled together. The consequences of an enveloping signature are that the processing of a document requires processing of the signature syntax, and that the signature and content are coupled together. A *detached* digital signature is not necessarily in a separate file, it is merely not contained within the document that is being signed. For example, there can be a parent document that contains both the content being signed and the digital signature. The consequences of a detached signature are that the processing of the digital signature and the content are separated; the signature format and content format can revision independently; and the digital signature and the content are separated.

There was some discussion about the dependability of PKI technology. Jon Baker pointed out that XCCDF and OVAL already support XML signatures.

Harold mentioned that the depiction of the SCAP Data Stream and the SCAP Result Data Stream that he provided would represent an *enveloping* signature, and he added that he would advise that he signature be moved to the top of the data streams.

Harold was asked which of the three methods (enveloped, enveloping, or detached) he would recommend, but he was reluctant to make a blanket recommendation, indicating that each method could work well in various use cases. A questioner asked whether the digital signature is in a separate file in the data stream depiction, but it was pointed out that the data stream actually represents several components combined in a single file.

Next Harold discussed signing results and reports. It would be possible to have different signatures provided by different sources on various parts of the reports and results. LtCol Wolfkiel then started a conversation about whether digital signatures should be required within the content, or whether vendors should be required to support such functionality. While there is merit in providing digital signatures, there could be some signature performance impacts for large results data streams. He advocated against the Validation Program requiring digital signatures. Dave Waltermire advocated for the position of vendors being required to support digital signatures, but then users deciding whether to use the capability in an operational environment.

Harold then mentioned two implementations that he has been working on. First, a Java implementation using Java 6 and has standardized on an encryption API – JSR 106. The JRE implementation mostly targets initials release. Other libraries may be available with more support as well as FIPS 140-2 validation. Next, Harold discussed a C# implementation, using .Net 3.5 on Windows XP or Vista. This version has limited native support for the necessary algorithms and parameters. This implementation also seemed to be geared toward the initial release of XML Signature and Processing. Harold queried

the audience about whether they felt that his testing of various implementations was useful to the SCAP community, and the attendees answered affirmatively.

Open Questions

Next Harold discussed how to establish identity or associating content with a key. Harold suggested to include the key within the KeyInfo element of the signature block and pointed out that this would require a separate mechanism to associate a key with an identity, i.e. how a certificate would be made available to an application is completely out of scope of this topic. However, there are ways of associating content with an identity, they are already built into the standard, e.g. using X.509Data, PGPData, or SPKIData. Harold pointed out that, from a PKI perspective, it would much easier to implement a scenario in which there were relatively few signers, whereas it would be much more complicated to support a situation with many signers. Harold is trying to encourage the acceptance of a scenario with a few signers.

The group again went back to the topic of whether digital signatures should be mandatory within SCAP and what that might mean. Concerns were expressed about adding the burden and complexity of digital signatures to vendor products. Harold took a poll of the attendees and they favored providing X.509 data within KeyInfo, and thus associating content with identity, as opposed to just providing and certificate and leaving the burden of associating content with identity to other means.

Harold then discussed some additional considerations. The first of these was transformations, which could be used essentially as a denial of service, or can confuse an application as to what was actually signed. Actually, removing transformations would simplify things. Therefore, Harold posed the question: Are transformations needed? The consensus was that transformations are not really needed.

Next, Harold discussed compositional signing, and he stated that there were two primary options that are available: using the reference element or creating a manifest which contains hashes of the references. Consequences of using the reference element is that all references are processed (hashed) as part of the core validation; a single failed hash causes the core validation to fail; but that it does not require application logic to verify references. Consequences of using the manifest are: the manifest is included as part of the signature block with hashes included; core validation can pass, even if the references failed hash validation; and application logic would be necessary to process the references.

Finally, Harold covered the topic of authorization. He started by posing the following questions: Is a common authorization model necessary? Is identity enough? What else is needed? LtCol Wolfkiel suggested that we attempt to proceed without an authorization model and see how it goes. Harold concurred with the assessment. As for whether identity is sufficient, Gary Gapinski suggested that, based upon previous discussions, it was unclear whether identity is necessary. Gary noted that the poll that Harold took earlier showed that the attendees leaned toward wanting to establish identity and thus be able to infer trust – but that is not the same as authorization. Gary also agreed with LtCol Wolfkiel's suggestion. LtCol Wolfkiel stated that authorization is definitely needed by the DoD. Harold stated that he didn't necessarily expect to come to a final decision on these issues during this workshop, but just wanted to get some opinions to help him work toward a decision in the future. Dick Whitehurst opined

that because the SCAP community uses a central repository for OVAL content, identity and authorization can become a relatively serious issue. Dave Waltermire added that it is not yet clear whether in the long run there will be a federated repository model or a single repository and suggested that these types of architectural decisions need to be made before the authorization model can be determined.

Harold ended the session by re-opening an old discussion about whether to use CMS or XML-DSig. Harold had formerly settled on XML-DSig after considering comments from the community. Dave Waltermire said that white space normalization is also a problem. One attendee stated that the CMS solution doesn't sound good based on the existing infrastructure. Gary Gapinski concurred adding that when content is being manipulated across platforms, it cannot be assured that the integrity will be preserved. From these comments, Harold surmised that he community still favored using XML-DSig.

Harold listed the following documents as being important in understanding issues around digital trust:

- XML Signature Syntax and Processing
    - http://www.w3.org/TR/xmldsig-core/
- XML Signature Best Practices
    - http://www.w3.org/TR/xmldsig-bestpractices/
- Additional XML Security URIs
    - http://www.ietf.org/rfc/rfc4051.txt

1. Cryptographic Message Syntax (RFC 5652)
    - http://tools.ietf.org/html/rfc5652

CPE

The CPE session spanned 4.5 hours and included the following sections:

1. CPE Overview: Brant Chiekes (MITRE)
2. CPE Manager Concept Development Effort: Lt Col Joseph Wolfkiel (DoD)
3. CPE Naming Specification: Brant Chiekes (MITRE)
4. CPE Matching Specification: Mary Parmelee (MITRE)
5. CPE Dictionary and Language Specifications: Paul Cichonski (NIST)
6. Representing Incomplete Product Information: Mary Parmelee (MITRE)

CPE Overview

In the CPE Overview section, Brant summarized the CPE 2.3 work since the Winter Developer Days CPE Workshop in February. Version 2.3 is scheduled to be released for public comment in July. Brant sent a pre-release to the CPE discussion list on June 9 as a read ahead for today's Developer Days session. Today's CPE session will not explain the specifications in detail. The Core Team assumes that participants

have previous knowledge of the specification. This session will be treated as a focus group for feedback on the technical approach for CPE 2.3. The CPE Core Team strongly encourages you to send feedback to the CPE discussion list. They have a very aggressive development schedule for the release of CPE 2.3. The sooner they receive feedback on CPE 2.3 the better the chance that they will be able to address your comments within the time frame of the 2.3 development schedule.

Following the Winter Developer Days workshop Brant formed a CPE Core Team to develop the CPE 2.3 specification. This team has worked intensively over the past few months to draft the CPE 2.3 suite of specifications. The CPE Core Team had a CPE Developer web conference in May to report progress and solicit feedback from the community. They released the pre-release draft specifications for informal review on June 9. Finally today is the last opportunity for real time feedback from the community before the draft specifications are released for public comment. The specifications are all now formatted as NIST Interagency Reports (IRs). Following the public comment period, a final draft will be submitted to a NIST IR review process for release sometime in September, 2010.

This is a complete rewrite even though it is a minor release. In terms of how the information is presented, the CPE specification has gone from a single thirty eight page document to a suite of specifications. They did not use most of the 2.2 verbiage. Instead they wrote new specifications while maintaining backward compatibility with CPE 2.2.

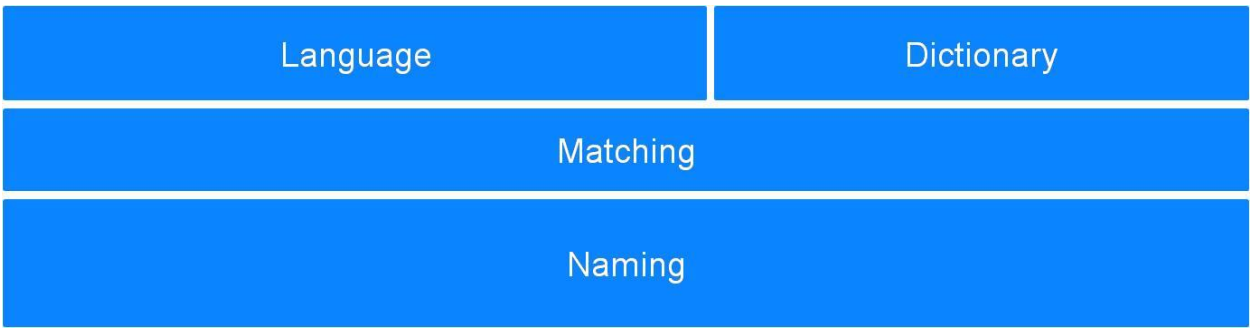
Brant thanked the members of the Core Team. The specification development approach has gone from more or less run and led by MITRE personnel with advice from the community to much more of an open community-driven standard. This process is not yet ideal, but is moving in the right direction. The Initial Core Team members included the MITRE CPE Team, the NIST SCAP Technical Team, Jim Ronayne and Shane Shaffer from the DoD. Since then vendors Seth Hanford from Cisco, Kent Landfield from McAffee, and Timothy Keanini from nCircle have joined us. The list of Core Team members can grow over time. They welcome more participation. For this round of CPE development, the Core Team has committed a great deal of time and effort. They have had multiple hours of teleconferences each week and have had a lot of commitment from this group. Their participation has resulted in a much improved product than it would have been without their participation.

Their goal for 2.3, given a limited amount of time was to address immediate needs; issues that they were aware of and that were underscored and emphasized at the February CPE Workshop. They started with this laundry list of things that the community would like to do with CPE. The goal still is to become part of SCAP 1.2. They knew from the beginning that this would need to be a minor release due to time constraints associated of the SCAP 1.2 schedule. This meant that they must maintain backward compatibility with CPE version 2.2 while satisfying as many of the items on the laundry list of requirements as possible.

CPE Version 2.2 will remain part of SCAP for multiple years. CPE maintains an official dictionary and content that needs to be maintained over time. They put a lot of thought into easing the burden of maintaining and upgrading implementations of the CPE 2.2 and 2.3 standards. They must maintain a

version 2.2 dictionary while supporting features of version 2.3. If you are happy with version 2.2 you can continue to work with it for a while.

An important new concept in 2.3 is a stack architecture. The choice of moving to this architecture is related to enabling composable capabilities without having to rewrite a monolithic specification. At the bottom of the stack is the Naming Specification. This is a minimalistic specification. All of the verbiage in CPE Version 2.2 about how to choose the values of a CPE name is gone. The CPE 2.3 Naming Specification focuses on a conceptual structure for naming and a way of binding those structures to encodings for machine transport. Naming is the foundation that specifies the conceptual model of a product or platform name called a well formed name and procedures for translating that structure into machine readable representations. The Matching specification builds on the Naming specification. The Dictionary and Language specifications are peers at the top of the stack and build on both Matching and Naming foundations.



In theory this new layered architecture will in theory give us the ability to add new layers over time that builds on the stack to extend the baseline functionality to support various use cases. Understanding the stack architecture is critical in understanding why certain things are present or absent in a given model. For example, the functionality provided in the naming specification may be loosely defined and then further constrained at higher levels of the stack such as the definition of wild card characters in the Matching specification.

Brant gave an overview of what is new in CPE 2.3.

1. They had a clear request from the community to remove the prefix property, which imposed a set relation on a CPE name that it turns out didn't work very well. The prefix property has been removed. Instead there is an abstract logical form (the WFN) that does not come with any restriction on the implementation of this form. It is a conceptual interlingua for the purpose of defining behavior without having to deal with the combinatorial explosions of ways that we would transform to different bindings or encodings.

2. There are procedures for binding to an expression that could be communicated (transforming) to machine readable representations. One could think of this like translating to a different form of natural language (e.g. from English to French).

3. For purposes of backward compatibility, they needed to retain support in 2.3 for the 2.2 URI binding. They have a way of representing products as the seven components of a URI binding. However, they could not represent some of the 2.3 features in the 2.2 URI binding form. Therefore they created a new binding as a formatted string without the constraints of a URI, which gives more flexibility in defining the grammar of the string, what certain characters mean, and so forth. There is some question as to whether we should keep the formatted string binding in the 2.3 specification.

4. They have defined a way to translate a 2.2 URI to a formatted string for up conversion of 2.2 content to 2.3 content. However, the reverse translation is lossy, because the formatted string can represent content that is new in 2.3 that did not exist in 2.2. The 2.3 has added features that have no meaning in the 2.2 model if you take advantage of the new features.

5. They have a way in which a dictionary can be maintained that provides parallel 2.2 and 2.3 names.

6. They added new attributes that were clearly requested by the CPE community including:

    a. Breaking out the software edition attribute to software edition, target software, target hardware.

    b. They added an “other” component, which is still debatable and may be removed. It is a grab bag attribute for information that is not already captured in the defined attributes.

7. There were concerns about percent encoding in the URI form, which were addressed in the formatted string. However, there are still outstanding concerns about escaping issues in the formatted string.

8. They have provided support in this new binding for wildcard characters. In 2.2 matching was at the whole component level, which was overly restrictive. The wild card characters will make matching more flexible.

9. There is a whole new approach to matching that still supports the 2.2 matching semantics.

    a. They split name level matching and language matching into two separate specifications.

    b. They added support for attribute-level matching as well as name-level matching. Users can compare the attribute level results between two names and interpret the results based on whatever their use case may be. They also provided minimal name-level matching. The idea is to provide matching tools that can be used to define different ways of comparing names in order to get the answer that satisfies you use case.

10. In the Dictionary specification:

    a. They made an effort to clarify what the rules are for accepting content into dictionaries.

    b. There is a need for a centralized dictionary along with private extended dictionaries that include proprietary or sensitive information that will not be shared. This introduces open

questions about how to address conflicts or naming collisions, but they are committed to supporting a federated dictionary approach.

c. Expanded the dictionary to provide better provenance tracking and improved deprecation logic.

d. They have provided a way for the new instance data to validate in the 2.2 schema.

11. The CPE Language specification has had no functional changes. It was just updated to align with the other specifications. No significant changes.

Generally speaking 2.3 begins down the road of decoupling CPE names as identifiers from CPE names as descriptions. From the beginning CPE names were both identifiers and descriptions or applicability statements for purposes of matching. They could not simply drop functionality that was present in 2.2 and convert to a simple enumeration like CVE. Consequently, they took the approach to map a path for more explicitly distinguishing an enumeration of product identifiers from a way of creating expressions that can be used to match against a repository to identify things that could be instances of products.

Toward the end of today we will discuss a cross cutting issue. CPE 2.2 had two logical values of ANY and NOT APPLICABLE. We introduced a new logical value for an attribute of a name called UNKNOWN. In the course of trying to work out the issues associated with UNKNOWN. They ultimately decided to remove UNKNOWN from the specification. It is present in the current specifications, but will be removed from the next draft. However, we have a better idea of what we were trying to accomplish with it and we have an idea of an alternative way to address the issue.

CPE Manager Concept Development Effort

Lt Col Wolfkiel gave an overview of a DoD effort to manage CPE names. The minutes for this presentation will be released following public release approval.

The idea that the DoD has been working through as we look at fielding multiple sensors that all have the ability to discover things that may be software on our systems; building asset databases that are not consistent across the DoD. Different people may want to keep software that they have installed on their systems private until a vulnerability is discovered on them. At the end of the day we need to know:

- What software is on our systems?
- What vulnerabilities do we have?
- What check lists do we need to be running?
- Is there old software that needs to be removed?

All of these things depend heavily on the ability to come up with standardized software names. This is what we have looked to CPE to solve for us. CPE does not gracefully address some of these issues. For example, the first day that a piece of software shows up on a system, it probably will not have a CPE name, but we want to know about it. The questions are, What is the process whereby DoD sensors find

a set of artifacts that indicate a piece of software exists on a DoD system, and how do we get from that state to where those sensors are reporting in a common CPE format?

They started working through the business logic of what it would look like to normalize software management across a large federated enterprise. Lt Col Wolfkiel walked through a conceptual diagram that depicts a target scenario for normalized software management.

In the DoD there are at least three defined tiers:

1. The DoD level, which is Cyber Command (formerly JTF-GNO (Joint Task Force Global Network Operations) wants an enterprise picture of everything on the Network.

2. The DoD component level (e.g. Army, Navy, NSA, DISA) Each of these components wants to know what is on their respective Networks.

3. Component networks can be further divided into geographical or other local installation enterprises.

Effectively it is an N-tiered structure that we want to be able to support that takes ground truth from the bottom to the top tier with some number of steps in between that we can support different levels of knowledge and detail.

At the lowest level of ground truth, are sensors; asset management scanners, vulnerability sensors, etc). Generally these are software sensors. Very few vendors are populating CPE information in their tools, but we would ultimately like them to be able to do that. The sensors need to be able to communicate to some centralized management capability and say this is all the software that I know about. If the sensor doesn't know what the CPE name is for a given piece of software then you need to be able to tell it what the CPE Name is for that software. If no CPE Name exists yet, then you need to be able to make one up until a CPE Name becomes available.

They also have the case where if this is not supportable locally, it can be passed through to the next tier and normalized there. However that happens, at the end of the day, they want normalized software names, preferably CPE Names across the DoD.

Given the first two requirements that they want to represent software found on DoD systems with CPE Names, but cannot wait until an official CPE Name is created and included in vendor tools for each piece of software that is found, they defined processes for how to manage software names in the interim. If an indicator is found that you think is a piece of software, what do you do with it? Lt Col Wolfkiel walked through an initial name assignment process diagram.

1. If the piece of identified software has a CPE Name, just use it.

2. If there is another CPE that has the same vendor and product name, but a different version, then just update the version value and report that.

3. If no known CPE Name exists for vendor and product, but the data is gracefully partitioned into vendor, product and version, then just convert that information into CPE format.

4. If it is not gracefully partitioned, then just take what you have, convert it to legal CPE form, putting all content in the product attribute.

They think that this approach will allow us to report whatever information we have immediately and map that to CPE at some later point in time.

Questions:

1. How often do you get a text descriptor of something that would go in the beginning of this diagram (the bottom box)?

Response: That box represents output from their "trickler" tool, which looks for groups of characters that indicate installed software (e.g. Apache 4.x or an OS fingerprint). It knows what are good indicators and reports on those. In a lot of ways it is a descriptive string that is a set of characteristics of installed software. Sometimes it will report things that are not indicators. An analyst looks at these strings and can reject or accept a new name based on the validity of the indicators within it.

2. How dependent is this architecture on the idea that a CPE Name would carry all of the attribute values that describe a device vice being opaque as we have talked about?

Response: Being opaque, such as being just a number is a problem if the vendor tool doesn't already know what that number designates, then there is no way to resolve it. If there were something that could map it to an opaque identifier then the original string could be deprecated at some point.

3. Where in this workflow could we insert a reconciliation process that compares what is found to a standard identifier?

Response: That is covered in a separate upcoming process.

4. You don't care about the identification piece you only care about having a common way to describe a product?

Response: It is kind of hazy where a description becomes an identifier, but at the point that I can unambiguously describe it then it is identified. There are some things that even CPE cannot describe, like products with different build numbers.

Response: You really just need a way of associating what you are finding on a system with some common identifier at some later time.

Response: If we are able to associate products with unique identifiers, then if one vendor buys another vendor, the vendor changes names, etc. That may be the point in the process where you want to associate a product with a unique identifier.

5. Are the names just unstructured strings or are you taking into consideration the record nature of a CPE Name?

Response: As much as we are capable, we are putting the right values into CPE Name attributes. But if you cannot do that, then just overload the product field with all the information.

6. So in this workflow, there is some reference to the CPE dictionary as it exists to that date.

Response: Yes, that is the next process. That assumes that a vendor has done the research and maps discovered artifacts to CPE Names.

7. If a vendor finds an artifact that they don't understand will that information ever get into the dictionary?

Response: We try not to call it a dictionary, but yes; it would be reported as a product in the asset inventory. This is not the official CPE dictionary. It is the best effort at describing a product in the reporting chain.

Response: That would require a download of the dictionary every day.

Response: Yes, and we are not sure how to fix that. The Marine Corps has approximately 40 thousand separate applications on their systems. We acquired a list of four thousand of the forty thousand names. It took two months to map these to CPE Names and submit to NIST. We are still not sure if they have made it into the CPE dictionary. Obviously we need a way to get visibility into our systems pending an approval process to get the data into the official dictionary. In the interim we want to have CPE inventory that have dramatically less rigor. If you have lists of all the information that every sensor reports, that is the starting point from which you manage your deprecation.

8. We do something similar, but we are looking to CPE as being the standard. We are hoping that our intermediary form gets replaced by a CPE Name.

Response: The Marine Corps said that there is software on their systems that will not be reported to the DoD and will have no CPE in the official dictionary. There may be DoD level software with the same issues.

CPE management workflow: This is a parallel process on normalized software maintenance. This would be part of an overall deprecation process whereby manual or automated means you would get a response that may reject and deprecate the initial name or accept it and map it to a CPE Name. Mapping relations would be tracked in an alias table of some asset management database. This is where you would keep track of what to call this set of artifacts.

Comment: So this is where heterogeneous sensors that report the same piece of software differently I reconciled.

Response: Yes, this is where we say that these sensor artifacts all mean the same thing and therefore we should report it the same way.

Response: So there is a core concept of collecting artifacts and reconciling them in an asset management database. A CPE could be considered a signature name that maps to different sets of artifacts.

At the CPE management level, the management software takes as input the lower level (raw) sensor data as candidate and map to the CPE Name if possible. Curated today by a human, if not a software name can send feedback to stop reporting this information. Send a report that says for all the names that you sent me, use these names instead.

Question: Does this delay the reporting process?

Response: This is where an automated mapping process comes into play, but we haven't solved this problem yet. We have data sets sample software and sample names, but haven't implemented anything yet. Hopefully it will scale to any Network.

Question: The things in the yellow box are artifacts not names, correct? Is CPE the best way to express this description?

Response: Typically, if you have something that you think maps to a CPE Name, you want to track its original description, and to what name it has been deprecated.

Response: These characteristics may or may not align with a CPE Name. Trying to force that into a CPE Name may not be the best approach.

Response: We think that this artifact is a CPE Name, we look at it and determine if it is a CPE Name and then map it if does designate a CPE Name. This is not a forensics tool that looks for unauthorized installation of products. We are starting with the assumption that this is a registered installation and trying to inventory what is there. We are not trying to examine every file on a machine and determine whether it is software. That is a different problem.

Comment: This identifies a gap in CPE in that we do not have a solution for the zero day problem. From a design standpoint, it seems like the front line there is some need to access a CPE dictionary.

Response: one of the things that we have discussed is a patch through, where if the CPE mapping cannot happen at one level it gets pushed to the next level for analysis. We are still working through that business logic. There is a requirement for a tiered structure, which complicates the logistics.

The sensor should be periodically packaging and reporting the information that it has to the validation and deprecation process, then getting a report back as to what to call it in the future. Periodic updates, reports back new deprecations, new names.

Comment: This implies that the management tool keeps track of every sensor that communicates with it.

Response: Yes that is a key requirement. The management tool must track which sensor report on which strings.

Response: That will not be stable if sensors move between tiers.

Response: Need to push up the raw sensor data is pushed all the way to the top tier.

Response: So I would need to go all the way to the top to see if anyone has ever mapped it.

Response: Names would be reported up to a centralized server that maps the names and pushes it down to the lower tiers so that they all report the same going forward.

Response: What if a top tier rejects a name that is important to a lower tier?

Response: That could happen. We haven't thought through this at the level of detail yet.

We will pilot this and release it as open source so that someone might pick it up and commercialize it.

Comment: Why not let the sensors report what they find and then map that to a name in a centralized repository that has that mapping for you.

Response: Issue is that there may be hierarchies in asset databases. Some tools that we think of as sensors think of themselves as asset databases. We want something that can be pushed to the edge when necessary.

Comment: If the COTs tools could report CPE natively, then it prevents redundant mapping processes.

Response: In the absence of a zero day CPE you would have the same problem anyway.

Response: Yes, these are hard problems. This may not be the right long term solution, but it is better than no solution.

CPE Naming Specification

The idea is to solicit specific advice for outstanding issues with respect to CPE 2.3. He reviewed the rationale behind defining the Well Formed Name (WFN) conceptual data structure to define CPE 2.3 behavior.

Brant then gave an overview of the WFN. The CPE 2.3 Naming specification

- Defines the WFN as a set of unordered name attribute pairs describing a product name.
- Specifies the format of a WFN
- Constrains the maximum cardinality of the attributes in a WFN to a maximum of one attribute of each type per name.
- Specifies that attributes can contain character strings, logical values (Not applicable or ANY), or a set of valid values.
- Recommend that values be chosen from a valid values list, but does not mandate specific value lists.

Brant then walked through an example of how the binding procedure works.

- The attribute pairs are unordered
- The canonical form escapes all non alphanumeric characters with a backslash
- Right now we only define the WFN are ASCII characters

For example, the CPE Name: **cpe:/a:adobe:acrobat%43%43:9.2:-::-**"unbinds" to wfn: [part="a", vendor="adobe", product="acrobat\+\+", version="9\.2", update=NA, edition=ANY, language=NA]

The formatted string looks like a URI, but does not conform to URI constraints. We have added a version specific prefix of cpe-23 for an easy way to distinguish a 2.2 form a 2.3 WFN. There is a requirement to carry across non-alphanumeric characters from 2.2. A hyphen in 2.2 has become an NA logical value and a blank in 2.2 becomes an ANY logical value in 2.3. We have also added wild card characters asterisk and question mark. Wildcards can be escaped and interpreted as characters.

Question: The asterisk means any in the bound form. There is no requirement to have a character before or after it?

Response: Yes.

Because we have no prefix property, there is no way to trim a CPE Name and so all attributes must be populated.

Question: What is the point of escaping characters that don't have a special meaning?

Response: Because they could be defined as special characters higher in the stack.

Question: why are we suing wild cards?

Answer: It is because of the unauthenticated scanner use case. The wild cards allow us to work with the ambiguity. Embedded wild cards are never in the dictionary.

Question: Why don't you just use CPRE.

Response: That is exactly what we have done.

Question: Why do we need to populate all fields?

Response: This general idea that the lack of a prefix property makes these peer attributes.

Right now there is partial escaping approach. We are still working on what should and should not be escaped.

Question: Does the asterisk match anything and does it mean the same thing logically as the ANY logical value, including a product with no edition?

Response: Yes, in the WFN it is ANY, in its formatted string bound form it is an asterisk. How it matches depends on what side of the matching algorithm it is on. This will be defined in the Matching specification.

Question: In 2.2 a dash meant null. We knew that there is no known value. This is equal to not applicable in 2.3.

Response: The whole point of having ANY is to say that you do not know if there is a value.

Response: There are different ways of interpreting how to match identifiers vice patterns. It depends on whether you are using a CPE Name as a description or an identifier. The Naming specification only provides the mechanism for interpreting them further in higher level specifications in the stack.

Brant reviewed outstanding issues:

Issue 1: URI versus formatted string: We created the formatted string as a more flexible binding to address the URI binding issues expressed by the community in 2.2. Should the formatted string actually be a URI or a formatted string? We will still have the percent encoding issues and other problems that were expressed as URI problems.

- In v2.2, CPE names are percent-encoded URIs, with colons used to delimit components
- In v2.3, we must preserve the 2.2-conformant URI as a legal binding, but it doesn't have to be the only legal binding—we can introduce new bindings, but must be able to specify a mechanical 2.2→2.3 conversion
- We have heard a variety of concerns expressed against the URI
    - Percent encoding/decoding inconsistencies
    - Issues with colons as separators
    - Impediments to implementing embedded wildcards

If we stick with a URI, we will need to redefine the way that we deal with wild card characters.

Brant asked for a show of hands if you will be hurt by the introduction of the formatted string. Two people acknowledge that it creates more work.

Comment: by demoting it from an identifier to a name, you lose the semantics of an identifier.

Response: The identifier approach was already broken. Thing that are descriptions were used as identifiers.

Response: All of the other identifiers can be easily put in URI form.

Response: CPE represents more complex semantics than some of the other standards and so should not be used as identifiers.

Brant summarized the discussion: The consensus is that we stick with the formatted string.

Issue 2: Syntax of the formatted string. Should we stick with colon separators between attributes? Brant asked Dave Waltermire to explaining why colon separators may not be a good idea. Dave replied that one problem is that languages like RDF that encode using URLs require that colons be percent encoded, which bloats the name. If the colons were replaced with a forward slash, this problem would go away.

Response: Replacing colons with backslashes makes it less human readable.

Brant polled the room for forward slashes vs. colons. Consensus was to use forward slashes.

Question: Why is there not an XML binding?

Response: There was not enough time to work out, but the WFN sets us up to be able to do that.

Response: If the slashes were colons, there would be an issue with XSID compliance, which requires non colonized names.

Question: Why not put the version in the header of the dictionary file?

Response: Because CPE Names can stand alone. It is not always part of a dictionary.

Comment: Combining forward and backslashes is confusing.

Response: this brings us to the last issue.

Issue 3: What is the right escaping mechanism?

Comment: Most vendors are representing data in XML. It would be easier to just go to an attribute value sets and use PCRE escaping.

Comment: Why not just distinguish descriptions from identifier names now and only require escaping for the description type.

Response: Because it would break backward compatibility

Comment: why not just escape the characters that need to be escaped.

Response: That was the original policy, but moved away from that to accommodate regex.

Comment: We could just pick 4-5 characters and reserve them for interpretation in the upper level specs.

Brant polled the audience: preference is to see as few escaped non alphanumeric characters as possible.

One other topic: Notion of packing the 2.3 data into 2.2 attributes (e.g. packing separate fields into edition). We will not be maintaining two dictionaries.

We came up with an algorithm for packing 2.3 values into a 2.2 edition field using a tilde separator. That would work fine as an identifier, but the blanks are potential issues in that you would get different matching answers in 2.2 vice 2.3.

Name Matching Specification

Mary gave a brief overview of the new matching approach. The major feedback that was received at the Developer Days CPE Requirements Workshop in February was:

- The all or nothing algorithm is too rigid
- Need a meaningful way to match incomplete information
- Need to match at the attribute level
- Need to be able to match attributes in any order
- Maintain backward compatibility with CPE 2.2
- Support basic tool interoperability
- Need to support a broad range of use cases

The new capabilities that were defined in CPE 2.3 Name Matching specification.

- A one-to-one comparison of a source CPE name to a target CPE Name
- CPE Language matching is “out of scope”. It is handled in the CPE Language specification.

CPE Name Matching specifies two phases of matching: attribute comparison and name matching. Comparison results are based on set theory where matching results are relative to the set of all possible match results. CPE name matching is determined based on the combined set of results (outcome) of the attribute comparison phase of the matching process. The CPE Naming Specification defines two wild card characters for matching incomplete attribute value information. The asterisk and question mark.

Outstanding issues:

How strictly to specify Name Matching: We want to provide some guidance at the name level for interoperability purposes, but want to be flexible enough to foster innovation and apply to many uses cases.

Name matching does not provide any required name level match

- An example of name matching is provided in the specification
- The method and common interpretation for comparison of attributes is specified in detail

- The combination of attribute comparison results that constitutes a “match” of one CPE name to another is largely unspecified
- There is no required common interpretation of whether two CPE names are a True or False match

Mary reviewed the trade offs. The advantage of leaving the majority of decisions about what constitutes a CPE name match to be decided by CPE implementers at design time allows the flexibility to make use case dependent precision vs. recall trade-off decisions at design time. The disadvantage is that the same set of attribute comparison results can match in some circumstances and not in others. There is no minimum baseline of interpretation for interoperability at the name level.

Question: Is the idea that you would define multiple matching methodologies and have the vendor specify which ones that they comply with? What would validation against this specification look like?

Response: This is why we decided to define a small common matching capability to base evaluation on for SCAP compliance.

Response: Less sophisticated users may need to choose from multiple algorithms for different use cases. E.g. precision vs. recall. Can we somehow build it in to the SCAP program so that when we go to buy a tool we can tell by what we are getting based on the algorithm that a tool is compliant with.

Response: What we are doing in the Name Matching specification. One piece is to provide a tool for attribute level comparison and then define one name-level function that calls the attribute functions. Given a bag of results, we can define certain outcomes. SCAP can further define constraints. We provide new ways of matching that are outside of our scope.

We have defined four name level matching requirements now.

1. Source equals target (=)
2. Source is a subset of target (**⊂)**
3. Source is a superset of target (⊃)
4. Source and target do not overlap (≠)

Question: What about intersection?

Response: So far we have decided not to define intersection. We are not saying that it doesn’t exist, just that we will not define it as a required name-level match. There is no reason that this cannot be defined outside the specification. We provide the tools to do that.

Question: The problem with using CPE in CCE entries. The configuration concepts apply to some products in the scope of Windows XP. We don’t know which ones to use.

Response: We have thought about that problem and similar problems. The approach that we are talking about in the incomplete session will help with that.

Response: We could make a valid CPE name that would express the right level of granularity. But cannot express to which products a CCE entry applies.

Mary reviewed an example of an attribute level comparison:

Source Name	part	vendor	product	version	update	edition
	a	Adobe	ANY	9.3.2	ANY	Palm OS

Target Name	part	vendor	product	version	update	edition
	a	ANY	Reader	9.*	NA	NA

Attribute Result	equal	subset	superset	subset	superset	no overlap
	=	⊂	⊃	⊂	⊃	≠

Question: If you have a wild card can you return the relationship?

Response: Yes, the set relationship. For example, in the example provided where version 9.* is matched to version 9.3.2. It would match anything after the '9.' in the version field.

Mary reviewed the four required name level matches describe the "and" joins of all attribute matches.

	Name Relationship = True	If Condition
1	EQUAL (=)	If all source = target
2	NO OVERLAP (≠)	If any source ≠ target
3	SUPERSET (⊃)	If all source and target are ⊃ **OR** a combination of ⊃ AND =
4	SUBSET (⊂)	If any source ⊂ target

	AND no source ≠ target

Question: If there is some subset and others are superset, then why is it a subset?

Response: Because we are constrained to comply with 2.2 semantics. That will likely change in the next major version.

Question: There are a lot of cases where none of those rules apply.

Response: For any two names you will get some answer. The name matching is not the only way to answer.

Comment: One way that we use CPE is that we use something similar. Users specify the platform and then once they specify the platform, we allow them to choose from all of the checks that are implied by that CPE Name. So that is a subset condition. The reverse of that is they can specify a check and get the platforms that are associated with that check.

Mary polled the audience about the name matching approach and level of specification.

Question: Is the intent to allow users to tune the matching. There are times that I want to get a broader set of results and other times I want a narrower set of results. Would this allow me to do that.

Response: there is nothing that precludes you from using it that way. That is one of many uses cases that it supports.

Comment: From XCCDF platform element perspective. We would have to define at that point what we want to specify how to leverage the matching criteria to get the outcome that they are looking for.

Comment: If you think about the 2.2 Matching algorithm it was an all or nothing match. We support that matching plus the building blocks to create other ways to extend matching to take advantage of 2.3.

Comment: change NO Overlap should be changed to NOT EQUAL

Question: What is the source and what is the target? A common application is going against a database with a description (applicability statement).

Response: This is about matching one source to one target name. The name can be any combination of descriptor to identifier type CPE Name.

Mary polled the audience for a general opinion of this approach. Consensus is that it is an improvement.

Comment: The use case we use most is the matching instances on a system.

Response: That is outside the scope of Name matching. It is more appropriate for higher level specs such as the dictionary spec.

Question: Is there any reason that source and target cannot have a wildcard? Shouldn't we restrict wildcards to one side of the equation?

Response: That is the subject of our next issue.

Comment: If you look at a SQL implementation you would never have wild cards on both sides of the equation. Why not constrain it here as well?

Response: Let's move to the wild card issue.

Wild Card matching: Right now there are no constraints on where wild cards can be used. They can be on both sides of the matching equation.

Mary gave an overview of the defined Matching Criteria for wild cards:

Source Attribute	Target Attribute	Result
i	**i + wild cards**	**SUBSET (⊂)**
i + wild cards	**i**	**SUPERSET (⊃)**
i + wild cards	**ANY**	**SUBSET (⊂)**
i + wild cards	**k**	**NO OVERLAP (≠)**
i + wild cards	**NA**	**NO OVERLAP (≠)**
i + wild cards	**k + wild cards**	**NO OVERLAP (≠)**
i + wild cards	**i + wild cards**	**ERROR**

This approach provides the greatest flexibility to the user community, but allowing wild cards in both the source and target attributes extremely complicates the comparison process.

The CPE Core Team's current position is that the specification should:

- Refer CPE implementers to a standard regular expression specification for matching.
- Only specify the matching criteria for each kind of wild card matching.
- Do not specify wild card pseudo-code.
- Constrain wildcard usage by defining an error condition when wild cards are in both the source and target attributes (no wild card to wild card matching).

An alternate suggestion is limited wild card to wild card matches:

Provide limited logic for handling wild card to wild card matching using PCRE/POSIX regular expression pattern matching behavior:

- Only match if the patterns are equal and make no determination as to containment, which allows us to return an answer in every case, and avoids the need for an extensive algorithm.
- A function hasWildcard(value) will return true if a value contains * or ?, and false otherwise
- A function isPatternMatch(pattern, value) returns true if the value will match the specified pattern where the wildcard string is converted according to: * -> .* and ? -> .?
- a ^ prepended to the pattern and a $ is appended to the pattern and the PCRE/POSIX regular expression pattern matching behavior is followed.

Comment: Suggest using PCRE. OVAL uses PCRE. If the guidance is to implementers.

Comment: suggest following SQL standard for this. Most people are using databases for this.

Comment: There is alignment from a SQL perspective.

Comment: specify that we use a PCRE style specification.

Comment: We would have to specify a substitution if we point to a specific standard.

Comment: We want to specify behavior and criteria. Not necessarily mandate a specific regex specification.

Comment: Should we use a different character besides asterisk for the ANY. An asterisk alone would mean the same as any.

Mary polled the audience: No error condition. We need a third option, but it is very complicated and will not work in SQL.

CPE Dictionary and Language Specifications

Paul gave an overview of the Dictionary specification. The CPE dictionary will limit CPE names to those who identify a single product as opposed to a set of products.

Acceptance criteria are to support the single product use case.

- No embedded wild cards are allowed.
- Names must have vendor, product and version, attribute values at a minimum. NA can be in the version attribute where no version exists.
- Names permitted in the dictionary are only allowed if one doesn't already exist at a lower level of specificity.

Comment: In our use case we need to say this represents all products for e.g. Microsoft XP.

Response: You can express that as an applicability statement and when you resolve it to the dictionary, the matching algorithm will retrieve that set of known products.

Comment: One of the use cases for CPE in XCCDF is a dictionary file that includes OVAL definition applicability statements for sets of products. If we cannot express sets of products in the dictionary, then I cannot write an OVAL definition because I can't write a CPE that represents e.g. Acrobat 3.0.

Response: The dictionary needs to represent concrete products. Then we may need something in XCCDF that is not in the dictionary.

Response: I could have rules that apply to different levels of specificity.

Response: Maybe we should relax the constraints in the dictionary. This is partially confounded by the fact that we have metadata related to the identifier. We would like to create a metadata repository so that you could define the kind of construct that you are describing. The same rules would not necessarily apply.

Response: The scope of the dictionary for something that it was not intended to do. It should be a list of concrete identifiers of real world products. Now that leaves a gap that needs to be filled. Now we are forced to come up with the right solution for this problem.

Question: Would NIST commit to supporting a metadata repository for this purpose?

Response: We cannot commit right now, but it is possible.

Question: What about container relations of packages, suites, etc.?

Response: We are thinking about that problem, but don't have time to address it in 2.3.

Response: This is a sustainable alternative to the current unsustainable combinatorial explosion of all combinations of all products.

Comment: It is a common misconception that all names must be in the CPE Dictionary. There is no requirement that everything that is expressed as a valid CPE Name. We are taking the step but 2.2 continues to exist. NIST does not plan to strip out the abstractions. Now the use of wild cards, you can use an abstract names to match actual product names. However, we will lose the ability to associate an OVAL definition that has any flavor that is in the dictionary.

Comment: Biggest problem is the deprecated process. If you match to a deprecated name, then it would be nice to redirect to the set of current names. Can't tell if it is an identifier or a match.

Response: Deprecation of abstract terms could be handled in the metadata repository.

Comment: deprecation does not mean that same thing anymore.

Response: There are two kinds of deprecation. When you learn more information, which you will deprecate to the set of CPE names that match that, and the other because incorrect. We plan to capture that in metadata.

Comment: We will not be able to use deprecated names to manage our product list because deprecation means something different to us. It means no longer valid and points to a current valid CPE if one exists.

Paul showed an example of deprecation when new information is discovered. It could be a set of more specific product names or a single product name that it is deprecated to.

Comment: Won't there be the same problem in the tiered DoD model? One sensor retrieves more information about a product than another. Ultimately they get resolved to a single name.

Response: We don't tell the sensor to use a more detailed name than it is capable of collecting. If I have a sensor that gives me more info, then I collect both. Somebody is going to have to build an app to determine which of the 20 names apply to the installed product.

Comment: What happens if the sensor has a bug that brings back the wrong version?

Response: Then that gets deprecated to 3.*, but we never allow multiple names to identify a single product.

Comment: If we are excluding the more general term then searching becomes more complex. If you search on 3.0* versus 3.*.0 would not match.

Response: That is currently an error condition. You are talking about a search use case rather than a acceptance criteria.

Comment: First if I was searching for adobe acrobat 3.0 English. As the entries in the dictionary get more detailed then it becomes harder to match.

Response: There may be a use case that gets every attribute match so that you could write your own algorithm to match what you want.

Comment: We may not want to deprecate at the attribute level. If the vendor changes the product doesn't change.

Response: Maybe it makes sense to make real time deprecation be outside of the dictionary.

Comment: What is the intent of the dictionary?

Response: To represent a canonical reference of the most complete and current information that we know.

Paul explained the CPE Dictionary data model:

They support a one-to-many relationship between a deprecated name and its replacement names.

Comment: There could be the reverse relationship.

Response: This is true. We will look into that.

Paul discusses backward compatibility issues.

- They are removing the NVD ID name. It is not useful for dictionary maintenance. Paul asked if anyone cared about the ID and there was no response.
- They are also removing the old deprecation logic since it is one-to-one versus one-to-many.

Comment: Does the one to many deprecation accommodate the more detailed name that is submitted? What happens when I add another one later?

Response: because it is not future proof and the same pattern doesn't always hold.

Question: Is that kind of information useful for this data. Is there a base of users who would use this?

Response: For example, if a vulnerability applies to windows XP, then when you retrieve all service packs only some of them will apply to the vulnerability.

Comment: If I go to more specific names. I am not going to look for an oval definition that goes to service pack 1, etc. because it was deprecated when an English version.

There is an expectation of what the name means that is not resolved by deprecation. Could we use an attribute that indicates the nature of the change.

Response: We may want to add a deprecation element, the reason for the change.

Brant requested that the audience read the specification and respond to the discussion list.

Matching Incomplete Information

Mary gave an overview of how the Core Team attempted to address the handling of sparse information with the UNKNOWN logical value. The Core Team concluded that the problem was too complex to handle within the construct of the CPE Name. For non-credentialed scanners, it seems like a good idea, but it introduced too much complexity into the specification.

The proposed solution is something that they have only begun to work out. It is outside the scope of 2.3, but they intend attack it quickly following the 2.3 release. We want to develop a community data model that is developed by the community with a technical working group. The data model would be implemented in a metadata catalog that is designed to capture the information around a CPE Name that helps identify the incomplete name.

Comment: Right now there is a well defined set of attributes for the name. Will this be included in the metadata catalog?

Response: Yes, this won't change the CPE Name, rather it will add metadata that helps define and manage the CPE Name. This would allow us to capture that ancillary information.

Question: Would the friendly name be in the dictionary?

Response: We don't know yet what metadata belongs in the metadata catalog versus what belongs in the dictionary.

Brant provided closing remarks for CPE. He summarized that Specification next steps and requested comment.