SCAPv2 Fall Workshop

OVAL/SCAP Authoring Application

High-level Overview

Agenda

- Caveats!
- Design Goals
- 5 Layer Application
- Feedback & Next Steps

Caveats

- A high-level "sketch" of a notional application
  - This application does not exist
  - Loosely based on limited, use-case-specific applications we have built and are in use by customers
  - Conversation-starter, not intended as a specific design or technology recommendation
- We aren't proposing a specific standard
  - Perhaps, formats that emerge from this project could lead towards a standards development initiative
- Software development initiative, TBD
  - A commercial product
  - A community-built open-source initiative (coalition of the willing!)
  - An open-source product funded by the community and/or a public sector initiative

Design Goals

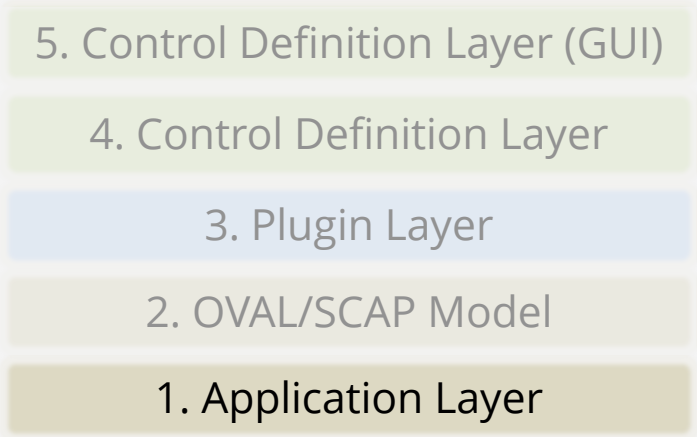
- Simplify Content Authoring
  - Make it easy for users with little-to-no OVAL experience to write checks to address common operational requirements.
  - This is achieved by enabling efficient collaboration between Content Authors and SCAP experts
  - SCAP experts can solve challenging authoring problems in a way that is easily reused by those with little-to-no SCAP-specific knowledge.
- Improve Support for Content Development
  - Make it easier for users with deep OVAL expertise to create and maintain non-trivial bodies of content.
  - Address the challenges of creating and maintaining non-trivial bodies of content that can require sophisticated collaboration efforts, dependency management, package management, versioning, deduplication, and other functions common to software development.
  - This is achieved by providing a programmatic authoring mode thereby allowing content developers to leverage tools and practices developed by the software development community to address these challenges.

Application Overview

Application Layers	Purpose and Role
5. Control Definition Layer (GUI)	Quick and easy authoring capabilities for Content Authors
4. Control Definition Layer	
3. Plugin Layer	Powerful authoring capabilities for Content Developers
2. OVAL/SCAP Model	Responsibility of application developers
1. Application Layer	

Layer 1: Application Layer

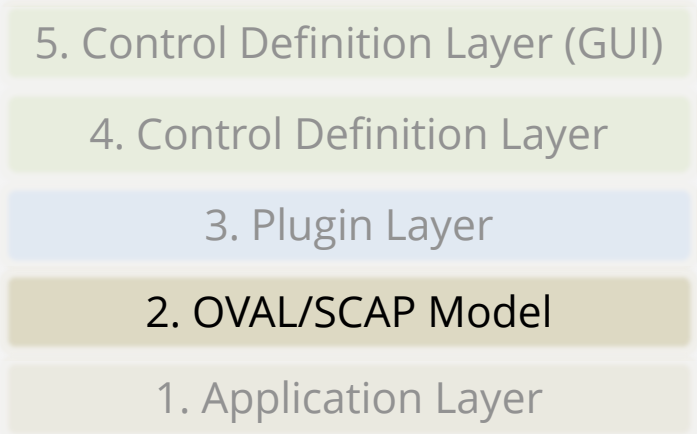
- Includes application logic
- A variety of capabilities to designed to simplify common authoring tasks such as
  - Creating OVAL IDs
  - Canonicalizing and deduplicating elements
  - Resolving dependencies between elements, etc.
- Generation of Layer 2: OVAL/SCAP Layer
  - E.g. generating packages and classes from schema documentation



Layer 2: OVAL/SCAP Model

joval CONTINUOUS MONITORING

- A programmatic model of OVAL, XCCDF, ARF, etc.
  - E.g., programmatic and implementations for an OVAL definition and each specific OVAL test, object, state and variable
  - Windows.FileTest(), Windows.FileObject(), Windows.FileState(), etc.
- Limited Simplification
  - Developers can author content using the OVAL/SCAP Model directly
  - Because model reflects OVAL/SCAP elements very closely (same organization, attributes, etc.), using the model requires the same expertise as writing OVAL XML without an authoring tool.
- It does offer advantages
  - Use of software development tools and practices
    - E.g. encapsulation, package management, dependency management, etc.
  - Application layer can manage deduplication, ID creation/management, etc.



Layer 3: The Plugin Layer

- This layer is composed of "plugins" written by OVAL/SCAP experts that simplify specific authoring use cases.

- Most plugins are functions with simple signatures that use the OVAL/SCAP Model Layer to generate OVAL.
  - Windows.DirectoryExists(<path>) function would use the Model to create the required Windows file_test and file_object.
  - Windows.AcrobatReader.IsInstalled()
  - Windows.AcrobatReader.Version.Equals('x.y.z')
  - Cisco.SnmpMibsIncluded(<mibs>)

5. Control Definition Layer (GUI)
4. Control Definition Layer
3. Plugin Layer
2. OVAL/SCAP Model
1. Application Layer

Layer 3: The Plugin Layer: Sample Plugin

```
import OVAL.model.windows as Windows

def DirectoryExists(path):
     file_object = Windows.file_object(
          'path': path,
          'filename': None
     )

     file_test = Windows.file_test(
          'title': 'The path exists `{0}`'.format(path),
          'object': file_object,
          'check': 'at_least_one_exists'
     )

     return file_test
```

5. Control Definition Layer (GUI)

4. Control Definition Layer

3. Plugin Layer

2. OVAL/SCAP Model

1. Application Layer

Layer 3: The Plugin Layer: Sample Plugin


```
import OVAL.model.windows as Windows
import org.cisecurity.oval.repo.windows.known_folder_variables as KnownFolders

def DirectoryExists(path, known_folder=None):
    if known_folder:
        path_variable_id = KnownFolders.getVariableId(known_folder, path)

        file_object = Windows.file_object(
            'path_variable_id': path_variable_id,
            'filename': None
        )
    else:
        file_object = Windows.file_object(
            'path': path,
            'filename': None
        )

    file_test = Windows.file_test(
        'title': 'The path exists `{0}`'.format(path),
        'object': file_object,
        'check': 'at_least_one_exists'
    )

    return file_test
```

5. Control Definition Layer (GUI)
4. Control Definition Layer
3. Plugin Layer
2. OVAL/SCAP Model
1. Application Layer

Layer 4: The Control Definition Layer

- Content Authors use Plugins to write content
  - Little-to-no SCAP-specific knowledge required
  - Write text files, one file per rule/definition, grouped into folder hierarchies that reflect benchmark organization or groupings of OVAL defintiions
- Supports use of various formats
  - Authors can write scripts combine Plugin calls with some simple syntax provided by the Application Core (AND/OR, etc.).
  - Authors can use formats like YAML, JSON, etc.

5. Control Definition Layer (GUI)

4. Control Definition Layer

3. Plugin Layer

2. OVAL/SCAP Model

1. Application Layer

Layer 4: The Control Definition Layer (example)

```
title: "Foo application is using the current folder”
id: 1.2
version: 1
tests:
  - AND:
    - windows_directory_exists:
        title: Foo application folder exists
        path: application_foo
        known_folder: PROGRAM_FILES
    - windows_directory_does_not_exist:
        title: Old foo application folder does not exist
        path: application_old_foo
        known_folder: PROGRAM_FILES
```

5. Control Definition Layer (GUI)

4. Control Definition Layer

3. Plugin Layer

2. OVAL/SCAP Model

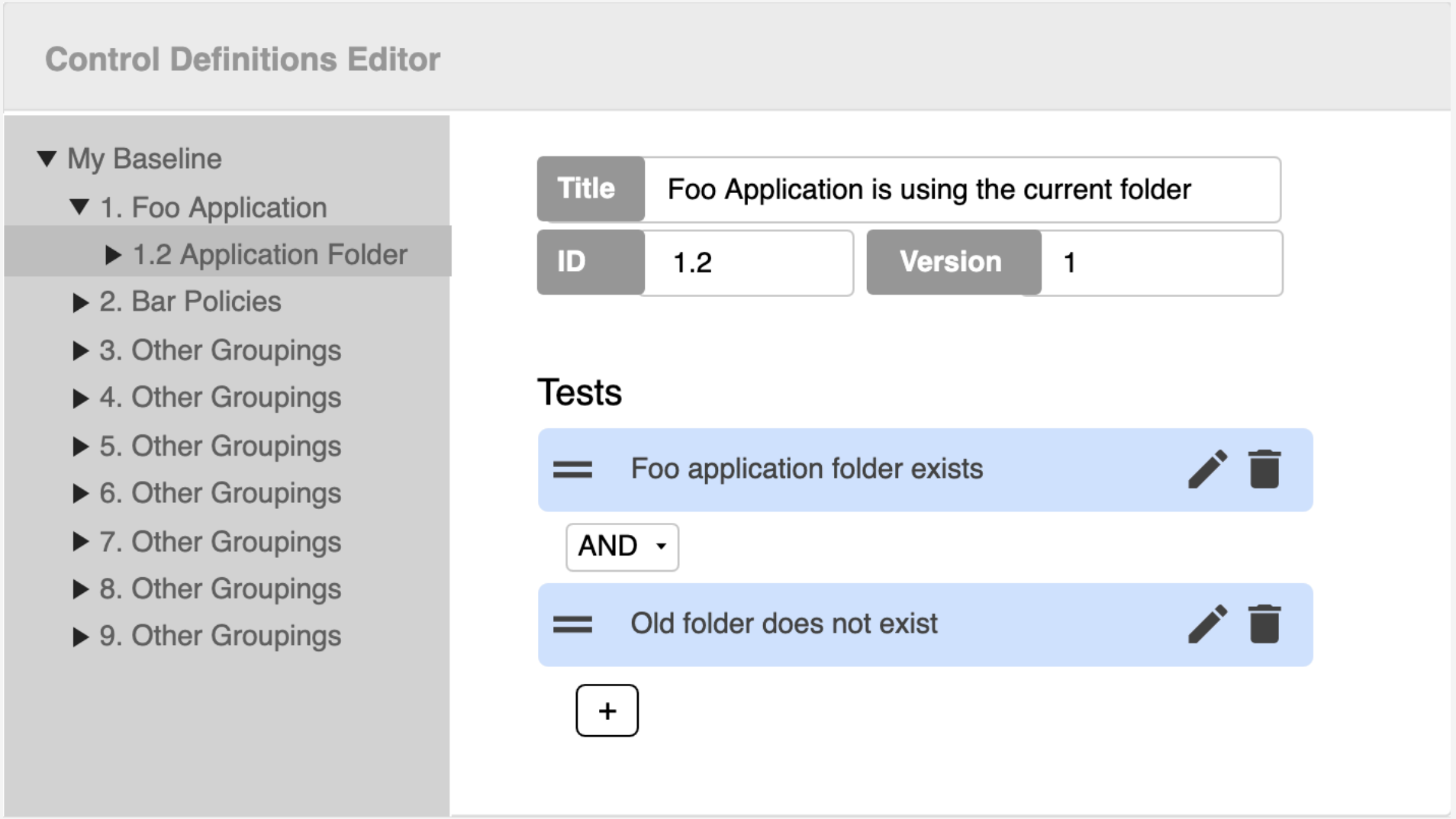
1. Application Layer

Layer 5: The Control Definition Layer (GUI)

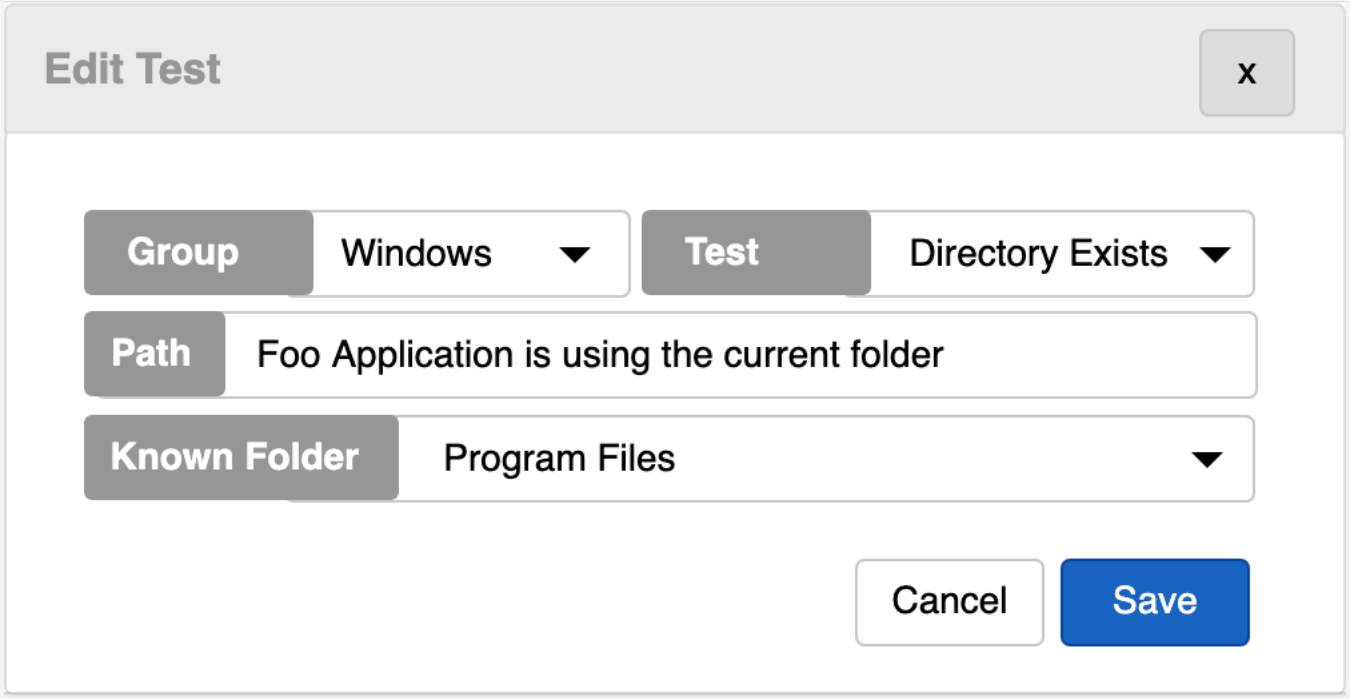
- A graphical user interface!
- Perhaps….
  - A navigation tree for each benchmark or grouping of OVAL definitions, with a leaf for each rule or top-level OVAL definition
  - For example it represents all YAML-expressable capabilities via form/UI controls with context-sensitive help.

5. Control Definition Layer (GUI)
4. Control Definition Layer
3. Plugin Layer
2. OVAL/SCAP Model
1. Application Layer

Layer 5: The Control Definition Layer (GUI)



Layer 5: The Control Definition Layer (GUI)



Feedback, Next Steps?

- Questions? Feedback?
- Does this seem like a useful approach?
- Next steps?