Elevating Security Resilience: An Overview of OSCAL-based Automation

Dr. Michaela Iorga, OSCAL Director
April 19, 2024
Are you familiar with OSCAL and its ability of supporting security automation?

1. I never heard of OSCAL
2. I heard some peers mentioning it
3. I read about it but never tried to use it
4. I am using it, I am a pro
Agenda

- Why NIST started OSCAL?
- What is OSCAL?
- Shifting Left Risk Management and Continuous ATO with OSCAL
- Mini demo
Compliance & Risk Management Are Hard!

- Information technology is complex & calls for automation
- Regulatory frameworks are burdensome & need interop auto GRC tools
- Paper-based A&A doesn’t scale & calls for auto updates
- Security vulnerabilities are everywhere & calls for auto updates
- Risk management is hard & experts need automation
- DevOps & IaC is hard in multi-clouds & calls for interoperability & standardization
What was needed?

A (Cyber) Machine-readable Esperanto that enables actors, tools and organizations to exchange information via automation:

OSCAL sets the foundation for automation and interoperability.
OSCAL in a Nutshell

• OSCAL is a standardized, flexible, open-source language that expresses security controls and their associated implementations and assessment methods in both, machine-readable and human formats.

• Layers: 3
• Models: 7 released + 2 prototypes
• Formats: 3 (XML, JSON, YAML)
OSCAL provides a common/single machine-readable language, expressed in XML, JSON and YAML for:

- multiple compliance and risk management frameworks (e.g. SP 800-53, ISO/IEC 27001&2, COBIT 5)
- software and service providers to express implementation guidance against security controls (Component definition)
- sharing how security controls are implemented (System Security Plans [SSPs])
- sharing security assessment plans (System Assessment Plans [SAPs])
- sharing security assessment results/reports (System Assessment Results [SARs])

OSCAL enables automated traceability from selection of security controls through implementation and assessment.
A Closer Look at OSCAL Models

Legend:
• Information flow:
• Traceability:
• Data source:
Providing control-related information in machine-readable formats.

NIST, in collaboration with industry, is developing the Open Security Controls Assessment Language (OSCAL). OSCAL is a set of formats expressed in XML, JSON, and YAML. These formats provide machine-readable representations of control catalogs, control baselines, system security plans, and assessment plans and results.
Complete v1.1.2 XML Format Outline

The following outline is a representation of the XML format for the combination of all OSCAL models. For each element or corresponding entry in the XML Format Reference, the cardinality and data type are also provided for each element or attribute.

```xml
<catalog uid='uid' [1]
  > <metadata ... /></metadata>
  > <import ... /></import>
  > <group ... /></group>
  > <back-matter ... /></back-matter>
</catalog>

<profile uid='uid' [1]
  > <metadata ... /></metadata>
  > <import ... /></import>
  > <merge ... /></merge>
  > <modify ... /></modify>
  > <back-matter ... /></back-matter>
</profile>

<component-definition uid='uid' [1]
  > <import-component-definition href='uri-reference' /></import-component-definition>
  > <capability uid='uid' name='string' /></capability>
  > <back-matter ... /></back-matter>
</component-definition>

<system-security-plan uid='uid' [1]
  > <metadata ... /></metadata>
  > <import-profile href='uri-reference' /></import-profile>
  > <system-characteristics ... /></system-characteristics>
  > <system-implementation ... /></system-implementation>
  > <back-matter ... /></back-matter>
</system-security-plan>
```

Complete v1.1.2 JSON Format Outline

The following outline is a representation of the JSON format for the combination of all OSCAL models. For each element or corresponding entry in the JSON Format Reference, the cardinality and data type are also provided for each property where applicable.

```json
{ catalog: { [uid]: { uid: ... }, metadata: [ ... ], params: [ ... ], controls: [ ... ], groups: [ ... ], back-matter: [ ... ] }, import: [ ... ], merge: [ ... ], modify: [ ... ], back-matter: [ ... ] }, profile: { [uid]: { uid: ... }, metadata: [ ... ], imports: [ ... ], modify: [ ... ], back-matter: [ ... ] }, component-definition: { [uid]: { uid: ... }, imported-components: [ ... ], imports: [ ... ], modify: [ ... ], back-matter: [ ... ] }, system-security-plan: { [uid]: { uid: ... }, import-profile: [ ... ], system-characteristics: [ ... ], system-implementation: [ ... ], back-matter: [ ... ] } }
```
### Complete v1.1.3 XML Format Reference

The following is the XML format reference for the complete set of all OSCAL models, which is organized hierarchically. Each entry represents the corresponding XML element or attribute in the model's XML format, and provides details about the semantics and use of the element or attribute. The XML Format Outline provides a streamlined, hierarchical representation of this model's XML format which can be used along with this reference to better understand the XML representation of this model.

**XML Namespace**
http://csrc.nist.gov/ns/oscal/1.0

This format represents a combination of all the OSCAL models.

<table>
<thead>
<tr>
<th>catalog</th>
<th>element (global definition)</th>
</tr>
</thead>
</table>

**Description**
A collection of controls.

**Remarks**
Catalogs may use one or more `group` objects to subdivide the control contents of a catalog.

**Constraints**
- `xmlns:oscal/<name>`:
  - The value must be one of the following:
    - `resolution-tool`: The tool used to produce a resolved profile.
    - `source-profile`: The tool used to produce a resolved profile.

**Attributes**
- `uuid`:
  - `uuid`:
    - Catalog Universally Unique Identifier

**Elements**
- `metadata` (global definition)

### Complete v1.1.3 JSON Format Reference

The following is the JSON format reference for the complete set of all OSCAL models, which is organized hierarchically, represents the corresponding JSON property in the model's JSON format, and provides details about the semantics and property. The JSON Format Outline provides a streamlined, hierarchical representation of this model's JSON format which can be used along with this reference to better understand the JSON representation of this model.

**JSON Namespace**
http://csrc.nist.gov/ns/oscal/1.0

This format represents a combination of all the OSCAL models.

<table>
<thead>
<tr>
<th>catalog</th>
<th>object (global definition)</th>
</tr>
</thead>
</table>

**Description**
A collection of controls.

**Remarks**
Catalogs may use one or more `group` objects to subdivide the control contents of a catalog.

**Constraints**
- `xmlns:oscal/<name>`:
  - The value must be one of the following:
    - `resolution-tool`: The tool used to produce a resolved profile.
    - `source-profile`: The tool used to produce a resolved profile.

**Attributes**
- `uuid`:
  - `uuid`:
    - Catalog Universally Unique Identifier

**Elements**
- `metadata` (global definition)
FIRST OSCAL 1.0.0
RELEASED ON JUNE 7, 2021
LATEST: OSCAL 1.1.2

https://github.com/usnistgov/OSCAL/releases
“...First official, major release of OSCAL provides a stable OSCAL 1.0.0 for wide-scale implementation ...”
Latest OSCAL patch release : 1.1.2 - backwards compatible.
OSCAL Content Maintained by NIST

https://github.com/usnistgov/oscal-content
What can one do with the OSCAL models?

Actors:
- Product Engineer
- Component Supplier
- Solution Evaluator
- Catalog Creators
- System Architect
- System Engineer
- System Owner
- Auditor
- Assessor
- System Operator
- Authorizing Official
- System Engineer
- System Operator

Risk Management & OSCAL content:

RMF steps: PREPARE  CATEGORIZE  SELECT  IMPLEMENT  ASSESS  AUTHORIZE  CON-MON
Shifting Left with OSCAL Implementation Layer

Component Definition Model
System Security Plan (SSP) Model

The place where the true innovation starts!
Component Definition Model v1.0.4 JSON Format Outline

The following outline is a representation of the JSON format for this model. For each property, the name links to the corresponding entry in the JSON Format Reference. The cardinality and data type are also provided for each property where appropriate.

```
component-definition [1]: {
  uuid [1]: uuid,
  metadata [1]: { ... },
  import-component-definitions [0 or 1]: [ ... ],
  components [0 or 1]: [ ... ],
  capabilities [0 or 1]: [ ... ],
  back-matter [0 or 1]: { ... },
}
```
The Anatomy of a Component Definition - Components


**control-implementations [0 or 1]** [
An array of control-implementation objects [1 to ∞] {
  uuid [1]: uuid,
  source [1]: uri-reference,
  description [1]: markup-multiline,
  ▶ props [0 or 1]: [ ... ],
  ▶ links [0 or 1]: [ ... ],
  ▶ set-parameters [0 or 1]: [ ... ],
  ▶ implemented-requirements [1]: {
    An array of implemented-requirement objects [1 to ∞] {
      uuid [1]: uuid,
      control-id [1]: token,
      description [1]: markup-multiline,
      ▶ props [0 or 1]: [ ... ],
      ▶ links [0 or 1]: [ ... ],
      ▶ set-parameters [0 or 1]: [ ... ],
      ▶ responsible-roles [0 or 1]: [ ... ],
      ▶ statements [0 or 1]: [ ... ],
      ▶ remarks [0 or 1]: markup-multiline,
    }
  }
}]

Defines how the component or capability supports a set of controls.
Snippet of a Component Definition Instance in JSON


```
{
  "component-definition": {
    "uuid": "a7ba800c-a432-44cd-9075-0862cd66da6b",
    "metadata": {
    },
    "components": [
      {
        "uuid": "91f646c5-b1b6-4786-9ec3-2305a044e217",
        "type": "software",
        "title": "MongoDB",
        "description": "MongoDB is a source-available, cross-platform document-oriented database program."
      },
      "protocols": [
      ],
      "control-implementations": [
        {
          "uuid": "49f0b690-ed9f-4f32-aae0-625b77aa6d27",
          "description": "MongoDB control implementations for NIST SP 800-53 revision 5."
        },
        {
          "uuid": "cf8338c5-fb6e-4593-a4a8-b3c4946ee2a0",
          "control-id": "sc-8.1",
          "description": "MongoDB supports TLS 1.x to encrypt data in transit, preventing unauthorized access."
        },
        {
          "uuid": "cf8338c5-fb6e-4593-a4a8-b3c4946ee2a0",
          "control-id": "sa-4.9",
          "description": "Must ensure that MongoDB only listens for network connections on authorization.
```

NOTE: some properties (e.g. `responsible-roles`, `protocols`, etc.) are collapsed
OSCAL Component Definition (Cdef) instances are like the maritime shipping containers!

OSCAL Cdef Model allows:

- Vendors to document the security controls implemented by their products
- System owners or policy makers to define ‘playbooks’ for system components
- System owners to test, review and provisionally authorize system components
- Reuse the components for different systems
- Ease the system’s documentation generation
- Human-intensive labor of generating SSP to be semi-automated

*Courtesy of a collaborator requesting anonymity.*
System Security Plan Model JSON Format Outline

The following outline is a representation of the JSON format for this model. For each property, the name links to the corresponding entry in the JSON Format Reference. The cardinality and data type are also provided for each property where appropriate.

```json
system-security-plan [1]: {
  uuid [1]: uuid,
  metadata [1]: { ... },
  import-profile [1]: { ... },
  system-characteristics [1]: { ... },
  system-implementation [1]: { ... },
  control-implementation [1]: { ... },
  back-matter [0 or 1]: { ... }
}
```
Even Container Ships Can Sink

Shifting left the security by pre-assessing the system’s components does not mean the system is secure by default!

- Review the components in the context of the respective system
- Review the interaction between components
- Get the max return in your investment – be granular with the data and the assessment
I think OSCAL can help me because...

1. I can implement DevSecOps processes and demonstrate compliance
2. I can document the security posture of my systems
3. I can automate my security assessments and continuously authorize systems
4. All of the above
When a Leveraging System is also a Leveraged System

Leveraging System:
The leveraging system's SSP should:
- identify what is inherited from a leveraged system
- identify any addressed responsibilities (as identified by the leveraged system)

In addition to:
- identifying what may be inherited by the leveraging system's customers
- any responsibilities the leveraging system's customers must address to fully satisfy a control
When a Leveraging System has more than one Leveraged System

The same syntax is used
- It is simply replicated for each leveraged system

The Leveraging System's SSP:
- Has a separate "leveraged-authorization" assembly for each leveraged system.
- Has a separate "component" representing each leveraged system.
- Has a separate "component" representing the leveraged system components associated with inherited capabilities.
### Assessment Plan (AP) & Assessment Results (AR)

- **OVERLAPPING SYNTAX**
- **SIMILAR BUT DISTINCT PURPOSE**
- **UNIQUE to AR:** Results and Evidence

#### Continuous Assessment Approach

- **Assessment Plan:** What should be tested/inspected, how, and with which frequency
- **Assessment Results:** Time-slice of results

#### Planed activities ↔ Actual activities

### Assessment Plan (AP)

- **Metadata**
  - Title, Version, Date, Document Labels, Revision History, Prepared By/For Roles, People, Teams, Locations
- **Import SSP**
  - URI pointing to an SSP
- **Local Definitions**
  - When information in the linked SSP is missing or inaccurate, assessors may define it here
- **Terms and Conditions**
  - Rules of Engagement, Disclosures, Limitation of Liability, Assumption Statements, and Methodology
- **Reviewed Controls**
  - Controls to include in the assessment we well as associated Control Objectives and Assessment Methods
- **Assessment Subject**
  - Identifies what will be assessed, including Components, Inventory Items, Locations, and User Types, as well as Parties to be interviewed
- **Assessment Assets**
  - Tools used to perform the assessment
- **Assessment Action**
  - Enumerates the actions for performing the assessment, including procedures for performing the assessment action
- **Task**
  - Intended schedule of milestones and assessment actions

### Assessment Results (AR)

- **Metadata**
  - Title, Version, Date, Document Labels, Revision History, Prepared By/For Roles, People, Teams, Locations
- **Import AP**
  - URI pointing to an Assessment Plan
- **Local Definitions (Overarching)**
  - When results contain an activity or control objective not defined by the assessment plan, define it here

### Back Matter

- **Laws/Regulations, Standards/Guidance**
  - May include artifacts to review Other Attachments as Needed

### Results (Current)

- **Local Definitions**
  - When information in the linked AP or SSP is missing or inaccurate, assessors may define it here
- **Reviewed Controls**
  - Controls included in the assessment
- **Assessment Subject**
  - Identifies what was assessed, including Components, Inventory Items, Locations, and User Types, as well as Parties to be interviewed
- **Assessment Assets**
  - Tools used to perform the assessment
- **Attestation**
  - Assertions made by the assessor
- **Assessment Log**
  - Log of performed assessment actions
- **Observation**
  - Individual observations and evidence
- **Risk**
  - Enumerates and characterizes risks and weaknesses, provides risk status
- **Finding**
  - Identified findings, Objective Status

### Results (Last Cycle)

- **Evidence Descriptions and Links**
  - Disposition Status

### Results (Initial Cycle)

- **Evidence Descriptions and Links**
  - Disposition Status

### Evidence Attachments:

- Reviewed Artifacts: Interview Notes, Screen Shots, Photos, Tool Reports, Raw Output
- Penetration Test Report
- Other Attachments as Needed
OSCAL Adopters Around the Globe
After this OSCAL introduction OSCAL, I think ...

1. I learned enough
2. I would like to learn more about representing policies and regulatory frameworks in OSCAL
3. I would like to learn more about creating my system security documentation in OSCAL
4. I would like to learn more about implementing a GRC tool that supports OSCAL formats
OSCAL is a community-driven program! Please join our forces!

Contact us at: oscal@nist.gov
Chat with us on Gitter: https://gitter.im/usnistgov-OSCAL/Lobby
Collaborate with us on GitHub: https://github.com/usnistgov/OSCAL
Join our COI meetings: https://pages.nist.gov/OSCAL/contribute/#community-meetings

Thank you!

https://www.nist.gov/OSCAL