Open Security Controls Assessment Language
The Anatomy of OSCAL Models
- Implementation Layer -

OSCAL 101 Series - Lecture #3

THE LECTURE WILL START SOON!

Presenter: Dr. Michaela Iorga
OSCAL Strategic Outreach Director
NIST is hosting a series of monthly educational workshops, on the third Tuesday of each month, 11:00-12:00 EST.

Purpose: improve OSCAL adoption by expanding the OSCAL community of interest (COI) through the onboarding of members who have no previous knowledge of OSCAL.


Welcome to the Lecture #3

Agenda
- Brief Recap of Lecture #1 & #2
- Continue with the Anatomy of OSCAL models
  - The Component Definition Model
  - The System Security Plan Model
Open Security Controls Assessment Language
The Anatomy of OSCAL Models
- Implementation Layer -

OSCAL 101 Series - Lecture #3

Presenter: Dr. Michaela Iorga
OSCAL Strategic Outreach Director
OSCAL is a standardized, flexible, open-source language designed to express security controls and their associated implementations and assessment methods in machine-readable formats (XML, JSON, and YAML). OSCAL content can be easily transformed into human-friendly formats.

OSCAL:

- Enables automated traceability
- Provides a standards-based foundation for the next generation GRCs
- Helps improve the risk management posture, consistency, and interoperability.
OSCAL is a standardized, flexible, open-source language designed to express security controls and their associated implementations and assessment methods in machine-readable formats (XML, JSON, and YAML). OSCAL content can be easily transformed into human-friendly formats.

OSCAL:

- Enables automated traceability
- Provides a standards-based foundation for the next generation GRCs
- Helps improve the risk management posture, consistency, and interoperability.
OSCAL is a standardized, flexible, open-source language designed to express security controls and their associated implementations and assessment methods in machine-readable formats (XML, JSON, and YAML). OSCAL content can be easily transformed into human-friendly formats.

OSCAL:
- Enables automated traceability
- Provides a standards-based foundation for the next generation GRCs
- Helps improve the risk management posture, consistency, and interoperability.

Recap: Lectures #1 & #2
OSCAL: the Open Security Controls Assessment Language

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.
Recap: OSCAL Models’ Outline


Complete v1.0.4 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.

```
<catalog uid="uid" [1]
  <metadata> ... </metadata> [1]
  <param id="token" class="token" depends-on="token"/> </param> [0 to 1]
  <control id="token" class="token"/> </control> [0 to 1]
  <group id="token" class="token"/> </group> [0 to 1]
  <back-matter> ... </back-matter> [0 or 1]
</catalog>
```

Complete v1.0.4 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.

```
<catalog>
  <metadata> ... </metadata>
  <import href="uri-reference"/> </import> [0 to 1]
  <merge> ... </merge> [0 or 1]
  <back-matter> ... </back-matter> [0 or 1]
</catalog>
```
Recap - OSCAL Content Validation
https://pages.nist.gov/OSCAL/concepts/validation/

"well-formed" vs "valid" OSCAL content

XML Schema Validators:
https://www.w3.org/XML/Schema#Tools

JSON Schema Validators:
https://json-schema.org/implementations.html#validators
Recap - Common OSCAL Structure

Complete v1.0.4 JSON Format Outline

The following outline is a representation of the JSON format for the combination of all OSCAL models. For each property, the cardinality and data type are also provided for each property where appropriate.

```json
catalog [1]: {
    uuid [1]: uuid,
    metadata [1]: { ... },
    params [0 or 1]: [ ... ],
    controls [0 or 1]: [ ... ],
    groups [0 or 1]: [ ... ],
    back-matter [0 or 1]: { ... },
},
profile [1]: {
    uuid [1]: uuid,
    metadata [1]: { ... },
    imports [1]: [ ... ],
    merge [0 or 1]: { ... },
    modify [0 or 1]: { ... },
    back-matter [0 or 1]: { ... },
},
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]: { ... },
}
OSCAL Implementation Layer

Component Definition Model
System Security Plan (SSP) Model
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.

```json
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 - Body


```
component-definition
  object
    (global-definition)

Component Definition

DESCRIPTION
A collection of component descriptions, which may optionally be grouped by capability.

▲ Constraints (2)

INDEX for component an index index-system-component-uuid shall list values returned by targets component using keys constructed of key field(s) @uuid
IS UNIQUE for capability; any target value must be unique (i.e., occur only once)

▲ Properties (6)

uuid
  uuid
    [1]

Component Definition Universally Unique Identifier

DESCRIPTION
A machine-oriented, globally unique identifier with cross-instance scope that can be used to reference this component definition elsewhere in this or other OSCAL instances. The locally defined UUID of the component definition can be used to reference the data item locally or globally (e.g., in an imported OSCAL instance). This UUID should be assigned per-subject, which means it should be consistently used to identify the same subject across revisions of the document.

metadata
  object
    (global-definition)

Publication metadata

DESCRIPTION
Provides information about the publication and availability of the containing document.

▲ Constraints (13)

INDEX for role an index index-metadata-role-ids shall list values returned by targets role using keys constructed of key field(s) @id
IS UNIQUE for document-id: any target value must be unique (i.e., occur only once)
IS UNIQUE for prop: any target value must be unique (i.e., occur only once)
```
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]: { ... },
}

Component Definition

- **Metadata**
  - Title, Version, Date, Document Labels, Revision History, Prepared By
- **Import Component Definition**
  - URI pointing to other component definition files
- **Component**
  - Individual component information, and information about controls the component is able to satisfy
- **Capability**
  - A grouping of related components into a larger capability
- **Back Matter**
  - Citations and External Links
  - Attachments and Embedded Images
The Anatomy of a Component Definition - Body


Loads a component definition from another resource.

Component Definition

- **Metadata**
  - Title, Version, Date, Document Labels, Revision History, Prepared By

- **Import Component Definition**
  - URI pointing to other component definition files

- **Component**
  - Individual component information, and information about controls the component is able to satisfy

- **Capability**
  - A grouping of related components into a larger capability

- **Back Matter**
  - Citations and External Links
  - Attachments and Embedded Images

---

```json
from another component
```

---

Individual component information, and information about controls the component is able to satisfy.

---

URI pointing to other component definition files.
A defined component that can be part of an implemented system.

```
  import-component-definitions [0 or 1]: [
    An array of import-component-definition objects [1 to ∞] {
      href [1]: uri-reference
    }
  ]

  ▼ components [0 or 1]: [
    An array of component objects [1 to ∞] {
      uuid [1]: uuid,
      type [1]: string,
      title [1]: markup-line,
      description [1]: markup-multiline,
      purpose [0 or 1]: markup-line,
      ▶ props [0 or 1]: [ ... ],
      ▶ links [0 or 1]: [ ... ],
      ▶ responsible-roles [0 or 1]: [ ... ],
      ▶ protocols [0 or 1]: [ ... ],
      ▶ control-implementations [0 or 1]: [ ... ],
      ▶ remarks [0 or 1]: markup-multiline,
    }
  ]
```

**Component Definition**

**Metadata**
- Title, Version, Date, Document Labels, Revision History, Prepared By

**Import Component Definition**
- URI pointing to other component definition files

**Component**
- Individual component information, and information about controls the component is able to satisfy

**Capability**
- A grouping of related components into a larger capability

**Back Matter**
- Citations and External Links
- Attachments and Embedded Images
The Anatomy of a Component Definition - Components


Defines how the component or capability supports a set of controls.

```
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,
      }
    }
  }
]
```
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,
      }
    ],
  },
},
]
The Anatomy of a Component Definition

```
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,
            },
        },
    },
]},
```

Identifies which statements within a control are addressed.
The Anatomy of a Component Definition - Capabilities


```javascript
\[\text{capabilities [0 or 1]}:\]
  \[
  \text{An array of capability objects [1 to \(\infty\)]} \{
    \text{uuid [1]: uuid,}
    \text{name [1]: string,}
    \text{description [1]: markup-multiline,}
    \text{props [0 or 1]: [ ... ],}
    \text{links [0 or 1]: [ ... ],}
    \text{incorporates-components [0 or 1]: [ ... ],}
    \text{control-implementations [0 or 1]: [ ... ],}
    \text{remarks [0 or 1]: markup-multiline,}
  \}
\]
```

A grouping of other components and/or capabilities.

---

**Component Definition**

- **Metadata**
  - Title, Version, Date, Document Labels, Revision History, Prepared By

- **Import Component Definition**
  - URI pointing to other component definition files

- **Component**
  - Individual component information, and information about controls the component is able to satisfy

- **Capability**
  - A grouping of related components into a larger capability

- **Back Matter**
  - Citations and External Links
  - Attachments and Embedded Images
An Example of a Component Definition Instance


```
"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. Provides a NoSQL database service",
      "responsible-roles": [...],
      "protocols": [...],
      "control-implementations": [
        {
          "uuid": "49f0b690-ed9f-4f32-aee0-625b77aa6d27",
          "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 authorized ports"
        }
      ]
    }
  ]
}
```

NOTE: some properties (e.g. `responsible-roles`, `protocols`, etc.) are collapsed
OSCAL Implementation Layer

Component Definition Model
System Security Plan (SSP) Model
The Anatomy of the SSP Model – Body


System Security Plan 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.

```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]: { ... }
}
```
The Anatomy of the SSP Model – Body

system-security-plan [1]: { Root Element & Root UUID uuid [1]: uuid,
metadata [1]: { ... },
import-profile [1]: { ... },
system-characteristics [1]: { ... },
system-implementation [1]: { ... },
control-implementation [1]: { ... },
back-matter [0 or 1]: { ... }
}
The Anatomy of the SSP Model – **Body**


**System Security Plan 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.

```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]: { ... }
    }
```

**System Security Plan (SSP)**

- **Metadata**
  - Title, Version, Date, Document Labels, Revision History, Prepared By/For Roles, People, Teams, Locations
- **Import Profile**
  - URI pointing to a Profile
- **System Characteristics**
  - System ID, Name, Description
  - Sensitivity/Impact Level
  - System Information
  - Service & Deployment Models
  - Diagrams: Authorization Boundary, Network, Data Flow
- **System Implementation**
  - Users, Components, Inventory
  - Ports, Protocols, & Services
  - Interconnections
- **Control Implementation**
  - Responsible Parties, Status, Origination
  - Parameter Values, Implementation
  - Description, Inheritance, Consumer Responsibilities
- **Back Matter**
  - Laws/Regulations, Standards/Guidance
  - Citations and External Links
  - Attachments and Embedded Images
system-security-plan [1]: {
    uuid [1]: uuid,
    ▶ metadata [1]: { ... },
    ▼ import-profile [1]: {
        href [1]: uri-reference,
        remarks [0 or 1]: markup-multiline
    },
    ▶ system-characteristics [1]: { ... },
    ▶ system-implementation [1]: { ... },
    ▶ control-implementation [1]: { ... },
    ▶ back-matter [0 or 1]: { ... }
}
The overall information system sensitivity categorization, such as defined by FIPS-199.

Contains details about all information types that are stored, processed, or transmitted by the system, such as privacy information, and those defined in NIST SP 800-60.

The overall level of expected impact caused by unauthorized disclosure, modification, or loss of access to the information.


```
<system-security-plan>
  <uuid>uuid</uuid>
  <metadata>
    ... contents ...
  </metadata>
  <import-profile>
    ... contents ...
  </import-profile>
  <system-characteristics>
    <system-ids>...</system-ids>
    <system-name>...</system-name>
    <system-name-short>...</system-name-short>
    <description>...</description>
    <props>
      ... contents ...
    </props>
    <links>
      ... contents ...
    </links>
    <date-authorized>...</date-authorized>
    <security-sensitivity-level>...</security-sensitivity-level>
    <system-information>
      ... contents ...
    </system-information>
    <security-impact-level>...</security-impact-level>
    <status>
      ... contents ...
    </status>
    <authorization-boundary>
      ... contents ...
    </authorization-boundary>
    <network-architecture>
      ... contents ...
    </network-architecture>
    <data-flow>
      ... contents ...
    </data-flow>
    <responsible-parties>
      ... contents ...
    </responsible-parties>
    <remarks>
      ... contents ...
    </remarks>
  </system-characteristics>
  <system-implementation>
    ... contents ...
  </system-implementation>
</system-security-plan>
```
A description of another authorized system from which this system inherits capabilities that satisfy security requirements. Another term for this concept is a common control provider.

```json
system-security-plan [1]: {
    uuid [1]: uuid,
    metadata [1]: { _ },
    import-profile [1]: { _ },
    system-characteristics [1]: { _ },
    system-implementation [1]: {
        props [0 or 1]: [ _ ],
        links [0 or 1]: [ _ ],
        leveraged-authorizations [0 or 1]: [ _ ],
        users [1]: [ _ ],
        components [1]: {
            uuid [1]: uuid,
            type [1]: string,
            title [1]: markup-line,
            description [1]: markup-multiline,
            purpose [0 or 1]: markup-line,
            props [0 or 1]: [ _ ],
            links [0 or 1]: [ _ ],
            responsible-roles [0 or 1]: [ _ ],
            protocols [0 or 1]: [ _ ],
            control-implementations [0 or 1]: [ _ ],
            remarks [0 or 1]: markup-multiline,
        },
        inventory-items [0 or 1]: [ _ ],
        back-matter [0 or 1]: { _ },
    },
    control-implementation [1]: { _ },
}
```
The Anatomy of the SSP Model – Body


Describes how the system satisfies a set of controls.

Describes how the system satisfies the requirements of an individual control.

Identifies which statements within a control are addressed. The information can be provided for each component of the system.

Defines how the referenced component implements a set of controls.

System Security Plan (SSP)

Metadata
- Title, Version, Date, Document Labels, Revision History, Prepared By/For
- Roles, People, Teams, Locations

Import Profile
- URI pointing to a Profile

System Characteristics
- System ID, Name, Description
- Sensitivity/Impact Level
- System Information
- Service & Deployment Models
- Diagrams: Authorization Boundary, Network, Data Flow

System Implementation
- Users, Components, Inventory
- Ports, Protocols, & Services
- Interconnections

Control Implementation
- Responsible Parties, Status, Origination
- Parameter Values, Implementation
- Description, Inheritance, Consumer Responsibilities

Back Matter
- Laws/Regulations, Standards/Guidance
- Citations and External Links
- Attachments and Embedded Images
The Anatomy of the SSP Model – Body

Simple OSCAL SSP Examples

- Please visit https://github.com/usnistgov/oscal-content/tree/main/examples/ssp for some simple SSP examples in:
  - XML
  - JSON
  - YAML
OSCAL is a community-driven program! Please join us!

Contact us at: oscal@nist.gov
Subscribe to our mailing lists: oscal-dev@list.nist.gov or oscal-updates@list.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
OSCAL News / Events

May 23-24, 2023

4th Annual OSCAL Conference and Workshop

Herbert C. Hoover Federal Building
1401 Constitution Avenue, NW, Washington
Open Floor Discussion

Ground Rules of Engagement

- Keep the discussion respectful by:
  - using welcoming and inclusive language
  - being respectful of differing viewpoints and experiences
  - gracefully accepting constructive criticism
  - wait for one speaker to finish before speaking
- Speak from your own experience instead of generalizing.
- Do not be afraid to respect fully challenge one another by asking questions focused on ideas not on the company or presenter.
- The final goal is not to always agree but rather gain a deeper understanding.