Learn you an oscal-cli
for better security data!

NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

Information Technology Laboratory
Computer Security Division

Slides and examples

You can download the slides and examples content from github.com/aj-stein-nist/oscal-cli-demo/releases/.

Goals

- Today's presentation has the following goals.
  - Locate, download, and install oscal-cli releases
  - Use the tool for its four high-level functionalities
  - Understand architecture for advanced usage to extend or adapt those features

Non-goals

- We enjoy helping our users, but we won't have time to present today on the topics below.
  - Setup of oscal-cli prerequisities
  - Writing Java to extend or adapt oscal-cli
  - Advanced OSCAL usage patterns with or without the oscal-cli

Who is this for?

- Software developers
- System engineers
- Technologists
- Others welcome (but more to learn)

What is 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."

- OSCAL's website

What is oscal-cli?

- OSCAL is a set of data formats.
- Software uses data formats.
- So what kind of software is oscal-cli?

What is oscal-cli?

- The oscal-cli (github.com/usnistgov/oscal-cli) is a reference software implementation with four high-level functionalities.
  - Data validator
  - Data converter
  - Data processor
  - Data modeler

Setup for oscal-cli

- The oscal-cli supports mutliple operating systems.
  - Linux
  - macOS
  - Windows

Setup for oscal-cli

- The only prerequisite is a Java runtime.
  - JRE or JDK 11 or newer is required.
  - We use Eclipse Temurin to build our releases.

Setup for oscal-cli

- Download the latest release per the project's instructions.
  - Development snapshots
  - Stable releases
- Check GPG signature.
- Extract the latest release.
- Check the version and test the install.

Functionalities of oscal-cli

- There are several key points to remember with oscal-cli.
  - The conversion and validation functionalities function the same for all models.
    - `./bin/oscal-cli $modelname convert example.xml --to=json`
    - `./bin/oscal-cli $modelname validate example.xml`
  - Data processing is specific to each model (e.g. only profile resolution at this time).
  - Data modeling is a generic sub-system (with the `oscal-cli metaschema` subcommand).

Data Validation

```
./bin/oscal-cli ssp validate /mnt/oscal-cli-demo/content/example_ssp.json
```

```
./bin/oscal-cli ssp validate /mnt/oscal-cli-demo/content/example_ssp.xml
```

```
./bin/oscal-cli ssp validate /mnt/oscal-cli-demo/content/example_ssp.custom --as=yaml
```

Interpreting Error Messages

```
./bin/oscal-cli ssp validate --as=xml /mnt/oscal-cli-demo/content/example_ssp_error.xml
Validating '/mnt/oscal-cli-demo/content/example_ssp_error.xml' as XML.
Validation identified the following in file '/mnt/oscal-cli-demo/content/example_ssp_error.xml'.
[ERROR] cvc-complex-type.4: Attribute 'state' must appear on element 'implementation-status'.
[file:///mnt/oscal-cli-demo/content/example_ssp_error.xml{260,41}]
[ERROR] cvc-complex-type.4: Attribute 'state' must appear on element 'implementation-status'.
[file:///mnt/oscal-cli-demo/content/example_ssp_error.xml{260,41}]
```

Data Conversion

```
./bin/oscal-cli ssp convert /mnt/oscal-cli-demo/content/example_ssp.xml --to=json
```

```
./bin/oscal-cli ssp convert /mnt/oscal-cli-demo/content/example_ssp.yaml --to=json
```

Data Processing

```
./bin/oscal-cli profile resolve /mnt/oscal-cli-demo/content/example_profile.json --to=xml
```

```
./bin/oscal-cli profile resolve /mnt/oscal-cli-demo/content/example_profile.xml --to=json
```

Data Modeling

```
./bin/oscal-cli metaschema generate-schema /mnt/oscal-cli-demo/content/computer_metaschema.xml --as=json
```

```
./bin/oscal-cli metaschema generate-schema /mnt/oscal-cli-demo/content/computer_metaschema.xml --as=xml
```

Data Modeling

```
./bin/oscal-cli metaschema validate /mnt/oscal-cli-demo/content/computer_metaschema.xml
```

Data Modeling

```
./bin/oscal-cli metaschema validate-content -m=/mnt/oscal-cli-demo/content/computer_metaschema.xml /mnt/oscal-cli-demo/content/computer.xml
```

```
./bin/oscal-cli metaschema validate-content -m=/mnt/oscal-cli-demo/content/computer_metaschema.xml /mnt/oscal-cli-demo/content/computer.json
```

Architecture

- The oscal-cli software has a three-layer architecutre.
  - metaschema-java
  - liboscal-java
  - oscal-cli

Advanced Usage

- Can I use oscal-cli features in my own software? Yes.
  - Yes, use metaschema-java and liboscal-java.
  - Write your own similar or different oscal-cli alternative.

Giving back to oscal-cli

- How can you help the NIST OSCAL Team with oscal-cli?
  - Use it.
  - Provide feedback at github.com/usnistgov/oscal-cli.
    - Document use cases
    - Report bugs (run commands with `--show-stack-trace`)
    - Request new features
  - Rinse and repeat.

Conclusion and Announcements

- How to keep current and contribute to OSCAL overall?
  - Project website at pages.nist.gov/OSCAL/
  - Code repository at github.com/usnistgov/OSCAL/
  - Contribution guidelines at pages.nist.gov/OSCAL/contribute/
  - Contact methods at pages.nist.gov/OSCAL/contact/