Shifting Left the Right Way with OSCAL

A Case Study using the Open Security Controls Assessment Language

Chris Compton, Alexander Stein, Nikita Wootten
Information Technology Laboratory
Computer Security Division
Case Study
Managing Risk With OSCAL

A research pilot for secure information exchange between multiple organizations that is continuously monitored, assessed and authorized to operate.
Case Study

The Question

Controls

SP 800-53
Catalog of Security and Privacy Controls for Information Systems and Organizations

Process

RMF
Risk Management Framework

nist.gov/rmf
OSCAL
Open Security Controls Assessment Language
- Open Source Project on GitHub
- First Official Release: June 7, 2021
OSCAL Overview

Open Security Controls Assessment Language

Produce and interpret machine-readable security documentation using a common specification that promotes interoperability.
Enables **automated traceability** from selection of security controls through implementation and assessment.
OSCAL Overview

Open Security Controls Assessment Language

Models in Development: Control Mapping, Shared Responsibility

https://pages.nist.gov/OSCAL/
Certain products may be identified on this web page, but such identification does not imply recommendation by the US National Institute of Standards and Technology or other agencies of the US Government, nor does it imply that the products identified are necessarily the best available for the purpose.
Give OSCAL a Try!

Not Just for Government...

Also: Get Involved!

• Past OSCAL Workshops
• Reference Documentation
• Community Teleconferences
• Future “Office Hours”

• Contributions of Code, Experiences and Expertise!

https://pages.nist.gov/OSCAL/contribute/
Workflow
How do we participate earlier in the development process, with constructive feedback, and documentation that contributes to the momentum of the project?
GitHub Workflow
.github/workflows/ci.yaml

Application Unit Testing
All OSCAL Model Content
OSCAL Assessment Plan
OSCAL Assessment Result

CUSTOM ACTION
OSCAL Validation
.github/actions/oscal-validation

CUSTOM ACTION
OSCAL Assessment
.github/actions/oscal-assess

CUSTOM ACTION
OSCAL Validation
.github/actions/oscal-validation

INPUT
OSCAL Profile + Full Catalog
Execute Plan Tasks
Findings Comment Added to Pull Request

Profile Resolution
Result Generation

INPUT
OSCAL Catalog
OUTPUT
OSCAL Assessment Result
AC-8 System Use Notification

a. Display [Assignment: organization-defined system use notification message or banner] to users before granting access to the system that provides privacy and security notices consistent with applicable laws, executive orders, directives, regulations, policies, standards, and guidelines and state that:

1. Users are accessing a U.S. Government system;
2. System usage may be monitored, recorded, and subject to audit;
3. Unauthorized use of the system is prohibited and subject to criminal and civil penalties; and
4. Use of the system indicates consent to monitoring and recording;

[...SNIP...]
NIST 800-53

Over 1100 Controls and Enhancements
The 'Old Way'

OSCAL System Security Plan

AC-8

<table>
<thead>
<tr>
<th>Responsible Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter AC-8(a):</td>
</tr>
<tr>
<td>Parameter AC-8(c)-1:</td>
</tr>
</tbody>
</table>

Implementation Status (check all that apply):
- Implemented
- Partially implemented
- Planned
- Alternative implementation
- Not applicable

Control Origination (check all that apply):
- Service Provider Corporate
- Service Provider System Specific
- Service Provider Hybrid (Corporate and System Specific)
- Configured by Customer (Customer System Specific)
- Provided by Customer (Customer System Specific)
- Shared (Service Provider and Customer Responsibility)
- Inherited from pre-existing FedRAMP Authorization. Click here to enter text, Date of Authorization

AC-8 What is the solution and how is it implemented?

Part a
Part b
Part c
Automated Assessment Plan

```yaml
27 tasks:
28   - uuid: 6b7e6a29-4588-46be-b242-a0bd0b0d0eccc
29     title: Validate System Use Notification Presence from Python Script
30     description: Check system use notification presence.
31     type: action
32     props:
33       - name: ar-check-method
34         ns: https://www.nist.gov/itl/csd/ssag/blossom
35         value: system-shell-return-code
36       - name: ar-check-result
37         ns: https://www.nist.gov/itl/csd/ssag/blossom
38         value: "0"
```

```
ci.yaml
on: pull_request

- application_test
  1m 5s
- oscal_validate
  32s
- oscal_assess
  39s
```

- Developer Change
- Software Testing
- OSCAL Content Validation
- Assessment Plan Execution
Demonstration
Appendix: The Demo App

Welcome to the application!

Enjoy your stay!

Main | Documentation
Appendix: Automated Assessment

tasks:
- uuid: 6b7e6a29-4588-46be-b242-a0bda0092eec
  title: Validate System Use Notification Presence from Python Script
  description: Check system use notification presence.
  type: action
  associated-activities:
  - activity-uuid: d85636e6-0d9d-4c94-a924-5a612a119040
    subjects:
    - type: component
      include-all: {}
  props:
  - name: ar-check-method
    ns: https://www.nist.gov/itl/csd/ssag/blossom
    value: system-shell-return-code
  - name: ar-check-result
    ns: https://www.nist.gov/itl/csd/ssag/blossom
    value: "0"

```python
#!/usr/bin/env python3
# Consumed by the oscal-workflow harness
import os
import textwrap
from urllib import request
from bs4 import BeautifulSoup

# The system use notification text
expected_use_notification = os.getenv('SSP_PARAM_AC_8_PRM_1')
if expected_use_notification is None:
    raise Exception('ac-8_prm_1 must be defined in the SSP')

# running via docker-compose.yml
response = request.urlopen(http://127.0.0.1:10000')
soup = BeautifulSoup(response, 'html.parser')

# Drill into the element that the system use notification text lives
raw_use_notification = soup.body.div.p.text

clean_use_notification = textwrap.dedent(raw_use_notification).strip().replace('\n', ' ')

assert clean_use_notification == expected_use_notification
```
Appendix: Bringing the System Into Compliance

```
24  @enroller.get("/")
25  async def read_root(request: Request):
26    return views.TemplateResponse("warning/non_conforming.html", {"request": request})
```