



# The Orion JK21 Collaboration for Agile ATO: OSCAL, SDP, TIC 3.0, SAF

2022 ATARC Cloud Security Working Group Report

*March 2022*



Advanced Technology Academic Research Center  
designed by government • led by government • attended by government

## Acknowledgments

On behalf of the Advanced Technology Academic Research Center, ATARC is proud to announce the release of the 2022 Report titled “**The Orion JK21 Collaboration for Agile ATO: OSCAL, SDP, TIC 3.0, SAF**”, authored by the members of the **ATARC Cloud Security Working Group**.

ATARC would like to take this opportunity to recognize the following contributors - members of the ORION JK21 project who dedicated substantial time on a regular basis to research and develop different components of the project described by this document.

### **Government:**

Michaela Iorga (National Institute of Standards and Technology)  
Nikita Wootten (National Institute of Standards and Technology)  
Akash Shah (National Institute of Standards and Technology)  
Erik Johnson (Federal Reserve Board)

### **Academia:**

Selena Xiao (University of Maryland)  
Jed Shakarji (University of Maryland)  
Samuel Howard (University of Maryland)

### **Not For-Profit:**

Mari Spina (MITRE)  
Aaron Lippold (MITRE)  
Robert Clark (MITRE)

### **Industry:**

Juanita Koilpillai (Waverley Labs)  
Paul Swinton (Waverley Labs)  
Thanh Tong (2Twelve Solutions)  
Greg Elin (GovReady)  
Anil Karmel (C2Labs)  
Juliette Easley (C2Labs)  
Travis Howerton (C2Labs)  
Jasson Walker Jr. (cFocus Software)  
Alexander Stein (Flexion)  
Yolanda Darricarrere (AID Trans-Network)

Sincerely,

Tom Suder, Founder, Advanced Technology Academic Research Center (ATARC)

## Dedication

All of us here at ORION JK21 project and our collaborators from afar who are involved with this pilot are dedicating this work as well as the associated report to the memory of our colleague, collaborator, and friend, Juanita Koilpillai of Waverley Labs, who has tragically passed away on July 25, 2021. Juanita was the heart and soul of this project and spent many long hours pushing this effort forward by implementing and documenting the open Software Defined Perimeter (SDP) layer. She lived her life well, conscientiously acting upon her spiritual beliefs by assisting both friends and strangers in need, facing all challenges in her life bravely and with optimism, and teaching us all to be the best versions of ourselves. We will carry on her legacy and passion for cybersecurity in practice through this work and will forever keep her altruistic spirit alive in our hearts.

## Abstract

The ATARC (Advanced Technology Academic Research Center) Federal organization hosted an industry-government collaboration to address the technical concepts and commercially available products associated with delivering a secure cloud-based IT system implementing continuous automated assessment and authorization (A&A) capabilities. As Zero-Trust Architecture (ZTA) had become the modern principle driving cybersecurity in the cloud, the collaboration team aims to demonstrate A&A automation in the context of a live Software-as-a-Service (SaaS) system developed in, and operated from a Blue-Green Development, Security, and Operations environment (DevSecOps). The DevSecOps environment is protected by a Zero-Trust Platform-as-a-Service (PaaS) solution (a.k.a open-source Software Defined Perimeter) deployed on an Infrastructure-as-a-Service (IaaS). The MITRE Security Automation Framework (SAF) was employed to demonstrate automated real-time security assessment by integrating SAF with commercial Governance, Risk and Compliance (GRC) tools (e.g. GovReady, C2Labs' Atlasity, and cFocus Software's ATO as a Service (ATOaaS), to demonstrate the successful use of NIST Open security Controls Assessment Language (OSCAL) to automate the A&A process. Resulting artifacts from this collaboration are made available by ATARC on the project's GitLab repository ([Orion · Wiki · ATARC / Orion · GitLab](#)), and include a MITRE SAF plugin which supports GRC tools' integration and supports A&A data portability in OSCAL JSON format, an InSpec compliance assessment profile for the OpenSDP software, and an InSpec Trusted Internet Connection (TIC) 3.0 capabilities assessment profile for the OpenSDP's Policy Decision Point (PDP), also referred to as OpenSDP Gateway.

## Table of Contents

<b>Acknowledgments .....</b>	<b>i</b>
<b>Dedication .....</b>	<b>ii</b>
<b>Abstract .....</b>	<b>iii</b>
<b>Table of Contents .....</b>	<b>iv</b>
<b>Table of Figures .....</b>	<b>v</b>
<b>Table of Tables .....</b>	<b>vi</b>
<b>1 Introduction .....</b>	<b>1</b>
1.1 Background .....	1
1.2 Working Group Objectives & Outcomes.....	2
1.3 A Sample User Story.....	3
1.4 Pilot System Design Concepts, Data Flows, and Shared Responsibilities.....	4
1.4.1 System Data Flows.....	5
1.4.2 Shared Responsibility Model.....	5
1.4.3 Test System Components & Configuration .....	5
1.4.4 TIC 3.0 Capabilities of the OpenSDP .....	9
1.4.5 Cloud Infrastructure Services .....	9
1.4.6 Documentation Automation Process.....	10
<b>2 Deliverables.....</b>	<b>11</b>
2.1 Sprint #1: Deliverables .....	11
2.1.1 GovReady: OSCAL SSP Generation.....	11
2.1.2 GovReady: Lessons Learned.....	13
2.1.3 ATO as a Service: OSCAL SSP Generation .....	14
2.1.4 ATO as a Service: Lessons Learned .....	18
2.1.5 C2 Labs: Importing and Processing SSPs .....	19
2.1.6 Atlasity Integration of Inspec and Heimdall .....	19
2.1.7 C2 Labs: Lessons Learned.....	19
2.1.8 OSCAL – InSpec Integration Problems .....	20

<b>2.2</b>	<b>Sprint #2: Deliverables .....</b>	<b>21</b>
2.2.1	Demonstrated Tools .....	22
2.2.2	Workflow InSpec Enhancements.....	22
2.2.3	Towards the Concept of OSCAL Namespace .....	25
2.2.4	Workflow Demonstration: InSpec Profile for OpenSDP Compliance .....	27
2.2.5	Workflow Demonstration: TIC 3.0 Testing of OpenSDP .....	30
2.2.6	Workflow Demonstration: Atlasity Enhancements.....	35
2.2.7	Component Enhancements for SSPs.....	36
2.2.8	OSCAL Validation with AJV .....	36
2.2.9	Passing OSCAL Parameters to InSpec Profile.....	37
2.2.10	Atlasity – Loading the InSpec Results File.....	38
2.2.11	Parsing the InSpec Results File .....	39
2.2.12	Exporting the SAP/SAR.....	40
2.2.13	Sprint #2 Results Summary .....	41
<b>3</b>	<b>Recommendations and Conclusions .....</b>	<b>41</b>
<b>4</b>	<b>Appendix.....</b>	<b>43</b>
<b>4.1</b>	<b>Acronyms .....</b>	<b>43</b>

## Table of Figures

Figure 1.	Orion JK21 Collaboration Target Architecture.....	5
Figure 2	Network Host and Protocol Mapping for the Orion JK21 Test Environment .....	7
Figure 3.	E3 Labs 2Twelve IaaS Cloud.....	10
Figure 4.	ATO Automation Process for the Orion JK21 Collaboration.....	11
Figure 5.	Screenshot of GovReady Component Library.....	12
Figure 6.	Screenshot of GovReady Component Library.....	13
Figure 7.	ATO as a Service Add Component Workflow (Jasson please update per Michaela's email) .....	15
Figure 8.	ATO as a Service Security Categorization Workflow .....	16
Figure 9.	ATO as a Service Select & Tailor Security Controls Workflow .....	16
Figure 10.	ATO as a Service Controls Implementation Workflow .....	17
Figure 11.	ATO as a Service Validate & Publish SSP Workflow .....	18
Figure 12.	Example#1 SSP Parameter Comparison for InSpec Integration.....	21
Figure 13.	Example#2 SSP Parameter Comparison for InSpec Integration.....	21
Figure 14.	OSCAL Proof-of-Concept End-to-End Workflow .....	23
Figure 15.	Customizing the test using InSpec parameters allows the test to better reflect the expected state of the system .....	23
Figure 16.	The value set in OSCAL is read and incorporated when executing the InSpec Profile.....	24
Figure 17.	Proposed OSCAL Extension Model for InSpec Integration .....	25
Figure 18.	A Possible Namespace Model to Support OSCAL Integrations.....	26

Figure 19. Sample Creation of InSpec Overlay Profile used to Test the OpenSDP Platform .....	28
Figure 20. Heimdall Viewer Showing Baseline Ubuntu and OpenSDP InSpec Profile Test Results Side-by-Side.....	28
Figure 21: TIC 3.0 Testing Architecture .....	32
Figure 22: TIC 3.0 InSpec Profile Results Summary .....	34
Figure 23: TIC 3.0 InSpec Profile Results Details .....	34
Figure 24. Atlasity Enhancements for OSCAL/Heimdall Integration .....	36
Figure 26. OSCAL Conversion Error Report.....	37
Figure 27. OSCAL File Parameterized to Pass Testing Values for the Waverley Labs OpenSDP InSpec Profile.....	38
Figure 28. Atlasity Tool Integration Pane; MITER SAF Ingest.....	39
Figure 29. Atlasity Heimdall Ingest Results.....	40
Figure 30. Atlasity SAP, SAR Export Capability .....	41

## Table of Tables

Table 1. Hardware and Software Testbed configuration.....	8
Table 2. OSCAL Parameter -InSpec Input Mapping Issues.....	20
Table 4. OpenSDP Hardening Tests .....	30

# 1 Introduction

The Orion JK21 industry-government collaboration was promoted, hosted, and supported by the ATARC (Advanced Technology Academic Research Center) organization.

## 1.1 Background

On September 4<sup>th</sup>, 2019, ATARC launched several Cloud Computing Working Groups, each focusing on distinct areas of interest (e.g. cloud migration, cost modeling, cloud security, etc.). The Cloud Security working group, gathering around the table representatives from private and public sector, proposed a proof of concept or pilot that aimed to demonstrate trending technologies and concepts applied to cloud environments: (1) Zero-Trust Architectures (ZTA) and principles, (2) DevSecOps' integrated processes of development and operations, (3) agile Authorizations to Operate (ATO) through automation of the assessment and authorization (A&A) process with NIST's OSCAL and (4) Trusted Internet Connection (TIC) 3.0 – network requirements. Each of these trends were becoming accepted as the inevitable future by the federal government IT community, but demonstrable reference implementations were non-existent.

The pilot was named Orion and in 2021 it became Orion JK21 (see Dedication). In 2019, the topics the Orion members were interested in were still maturing, despite the fact that:

- Zero Trust Architecture was considered the best industry practice and NIST has released the first draft version of the NIST SP 800-207 on Zero Trust Architecture<sup>1</sup>.
- NIST had also released, in June 15, 2019, the first milestone (M1) of the Open Security Control Assessment Language (OSCAL) 1.0.0 ([OSCAL 1.0.0-M1](#)) and had focused primarily on representing control catalogs and profiles or baselines<sup>2</sup>.
- DHS CISA was in the process of finalizing Trusted Internet Connection (TIC) 3.0 guidance for Government agencies<sup>3</sup>.
- Department of Defense (DoD) published in August 2019 the [DoD Enterprise DevSecOps Reference Design](#).

The landscape of tools and technologies that existed to support data sharing of risk management information was limited in scope and maturity, with much of the assessment and ATO work still being done in siloed assessment technologies coupled with manual documentation requiring a large amount of labor and time to keep up to date.

The ORION JK21 team felt strongly that being able to provide a proof of concept for each of these emerging trends would move forward cloud security and compliance automation.

---

<sup>1</sup> [SP 800-207, Zero Trust Architecture | CSRC \(nist.gov\)](#)

<sup>2</sup> [OSCAL \(nist.gov\)](#)

<sup>3</sup> [TIC 3.0 Core Guidance Documents | CISA](#)



## 1.2 Working Group Objectives & Outcomes

The Orion JK21 pilot demonstrates A&A automation in the context of a live Software-as-a-Service (SaaS) system implementing an application developed in, and operated from a Blue-Green Development, Security, and Operations environment (DevSecOps).

The SaaS is protected by Waverley Labs' open-source implementation of Software Defined Perimeter (OpenSDP<sup>4</sup>) as the Zero-Trust Platform-as-a-Service (PaaS) solution deployed on the 2Twelve Solutions' E3 Labs Infrastructure-as-a-Service (IaaS). The DevSecOps and OpenSDP leveraged virtual machines (VMs) running Ubuntu 16.04.

The MITRE Security Automation Framework (SAF) was employed to demonstrate automated real-time security assessment by integrating SAF with commercial Governance, Risk and Compliance (GRC) tools including GovReady, C2Labs' Atlasity, and cFocus Software's ATO-as-a-Service (ATOaaS), to demonstrate the use of the NIST's Open Security Controls Assessment Language (OSCAL) for the production of the NIST Risk Management Framework (RMF) A&A artifacts in machine-readable format (i.e. JSON). The System Security Plan (SSP), System Assessment Plan (SAP), System Assessment Report (SAR), and Plan of Action and Milestones (POA&M), when applicable where generate for the PaaS (Ubuntu components).

Collaboration activities among our team members were set out to achieve the following outcomes:

1. Demonstrate Assessment and Authorization (A&A)/Governance Risk and Compliance (GRC) tools interoperability and data portability
2. Demonstrate the use of Zero Trust architecture for securing a DevSecOps environment used to develop and operate a simple app
  - Employ E3 Lab Infrastructure-as-a-Service (IaaS)
  - Deploy Waverley Labs OpenSDP which demonstrates core principles of Zero-Trust Architecture
3. Demonstrate that the OpenSDP Gateway operating as a ZTA Policy Enforcement Point (PEP) is compliant compliance with TIC 3.0 for cloud access network requirements.
4. Demonstrate the Agile Authorization-to-Operate (ATO) process
  - Leverage NIST OSCAL for information sharing and portability
  - Leverage GovReady for System Security Plan (SSP) generation
  - Leverage C2 Labs Atlasity for SSP, Components, System Assessment Plan (SAP) & System Assessment Result (SAR) generation
  - Leverage MITRE Security Automation Framework (SAF) & MITRE Heimdall for hardening and compliance

---

<sup>4</sup> <https://www.waverleylabs.com/open-source-sdp/>

The collaborative work shows that the NIST OSCAL can successfully be used by modern GRC tools to generate the A&A artifacts and effectively enables the sharing and porting of such artifact among GRC tools to convey the necessary security information in a standardized machine-readable format. Resulting artifacts from this collaboration include a MITRE SAF plugin to support integration with GRC tools that are able to exchange A&A data in NIST OSCAL formats, an InSpec compliance assessment profile for the OpenSDP software, and an InSpec Trusted Internet Connection (TIC) 3.0 capabilities assessment profile for the OpenSDP's Policy Decision Point (PDP), also referred to as OpenSDP Gateway. The artifacts are available at [Orion · Wiki · ATARC / Orion · GitLab](#).

### 1.3 A Sample User Story

The use case story that emerged for the working group was that of three (3) service providers working together to get an ATO for an IT system hosted in a cloud environment.

The Federal Information Security Modernization Act (FISMA) of 2014 (PL 113-283, 44 USC 3554) emphasized the importance of information security to the economic and national security interests of the United States and required each federal agency to develop, document, and implement an agency-wide program to provide information security protections commensurate with the risk and magnitude of the harm resulting from unauthorized access, use, disclosure, disruption, modification, or destruction of information and information systems. FISMA requires agency heads to report on the adequacy and effectiveness of the information security policies, procedures, and practices of their enterprise.

Based on FISMA requirements, a federal agency deploying an application in a cloud environment is responsible for assessing and authorizing to operate (ATO) each layer of the entire cloud stack not just for the application the agency manages. In our use case, the fictive federal agency deploying F-FORCE application as a service in a secured (OpenSDP) cloud environment becomes responsible for assessing and authorizing (A&A) all cloud layers. The system owner and the assessor assigned to complete the assessment are employing a process pipeline to automate the development of compliance documentation starting with the System Security Plan (SSP) and ending with the System Assessment Report (SAR). In the process, the agency works with GovReady and ATO-as-a-Service to develop the SSPs for the cloud services (IaaS and OpenSDP) and document the controls satisfaction. The GovReady's & ATOaaS' OSCAL SSPs are imported then into the C2 Labs Atlasity platform where data is used to generate the System Assessment Plan (SAP) and the SAR. Incorporating the MITRE Security Automation Framework (SAF) and Heimdall Data Format (HDF) and Heimdall visualization tools, Atlasity is able to render and visualize the system's security posture (the hardening test results) and the compliance status, including the Plan of Actions and Milestones (POA&M) conveying the findings that require remediations. By automating the A&A process, the agency is able to eliminate the human-driven, manual process and demonstrate an agile ATO process that can be repeated as often as needed.

## 1.4 Pilot System Design Concepts, Data Flows, and Shared Responsibilities

The high-level architecture developed for the Orion JK21 collaboration was chosen to specifically provide a complex access control environment necessary to exercise Zero Trust concepts as well as demonstrate operations in a DevSecOps environment that would allow the collaborators to examine realistic OSCAL-based (A&A) artifacts generation and automated testing capabilities.

A Blue-Green DevSecOps environment, a practice recommended by AWS<sup>5</sup>, provides essentially identical environments used as development and operations environments. The objective is to develop the new version of the application on the Green environment while the already deployed (older) version of the application operates on the Blue environment. When the Green-based system is completed, tested and ready for transition to production, the routing switch is flipped and all Green environment and application then become Blue (operational) and vice versa, for the previous Blue environment becomes Green, and ready for new application updates. Blue-Green deployments enable you to launch a new version (green) of your application alongside the old version (blue) and monitor and test the new version before you reroute traffic to it, rolling back on issue detection.

A Blue-Green DevSecOps solution presents a complicated access control challenge because elements of the system may be shared by application users, application administrators, and application developers. As systems are swapped from Green to Blue and back again, user access must swap in unison so that application users aren't given access to developmental systems and developers aren't allowed to disturb production systems. Such an environment provides a rich opportunity to examine Open Software Defined Perimeter (SDP) technologies.<sup>6,7</sup>

Access control data flows are illustrated in target architecture defined by the Orion JK21 collaboration team; reference Figure 2. In this diagram, the Blue-Green DevSecOps systems, dedicated to the development and hosting of the F-Force application, are secured by the Waverley OpenSDP solutions which provides an OpenSDP Controller (a Policy Decision Point (PDP)) and the OpenSDP Gateway (a Policy Enforcement Point (PEP)). The Controller acts much like an Identity-as-a-Service (IDaaS) solution providing onboarding capabilities for users and devices and management of access control. The Gateway allows and denies access to the Blue-Green environments based upon a Single Packet Authorization (SPA) message sent from the Controller via a dedicated communication channel.

---

<sup>5</sup> [Blue/Green Deployments - Introduction to DevOps on AWS \(amazon.com\)](#)

<sup>6</sup> Software-Defined Perimeter (SDP) Specification v2.0, Cloud Security Alliance (CSA), 2021

<sup>7</sup> SDP Working Group Software-Defined Perimeter Architecture Guide, Cloud Security Alliance (CSA), 2019

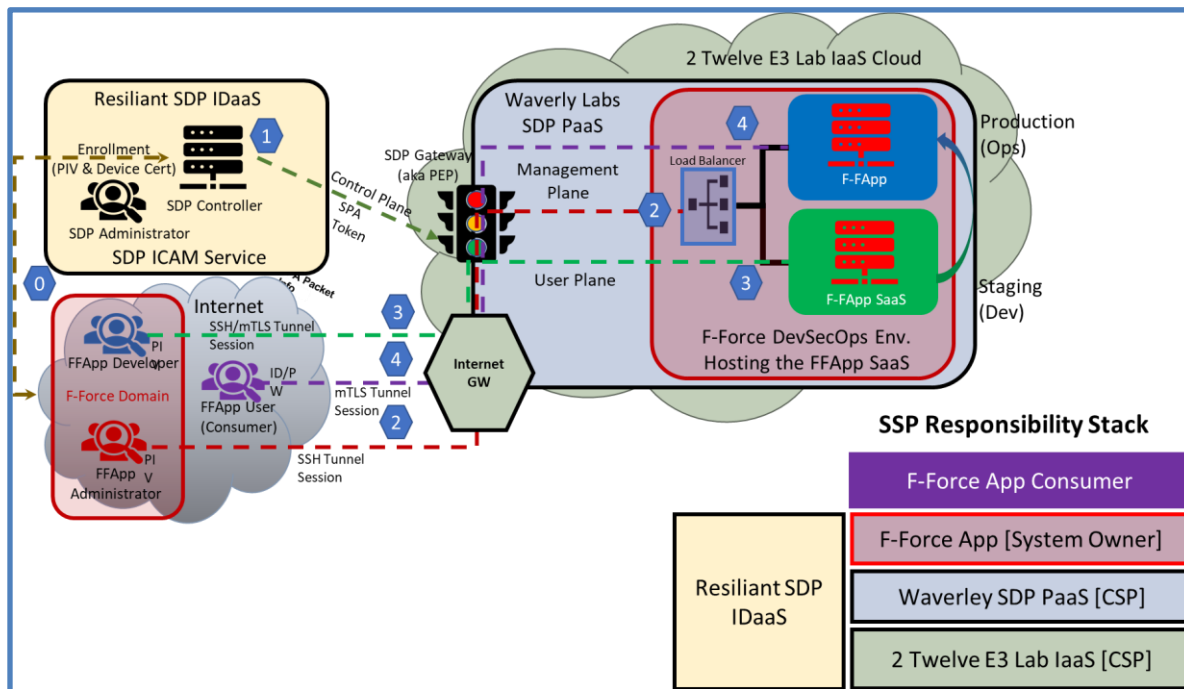


Figure 1. Orion JK21 Collaboration Target Architecture

#### 1.4.1 System Data Flows

System data flows are illustrated in Figure 1. Data flows consistent of:

- User On-boarding to the OpenSDP Controller (Flow #0)
- OpenSDP Controller to OpenSDP PEP Communications (Flow #1)
- Application Administrators (Flow #2)
- Application Developers (Flow #3)
- Application Users (Flow #4)

The intent of the OpenSDP's Gateway is to enforce control over access to Blue-Green environments in a manner consistent with TIC 3.0 Network Policy Enforcement Point (PEP) requirements. Only flows #2, 3, and 4" are allowed through the SDP Gateway PEP.

#### 1.4.2 Shared Responsibility Model

The Shared Security Responsibility Model (SSRM) for the overall system is illustrated in The high-level architecture developed for the Orion JK21 collaboration was chosen to specifically provide a complex access control environment necessary to exercise Zero Trust concepts as well as demonstrate operations in a DevSecOps environment that would allow the collaborators to examine realistic OSCAL-based (A&A) artifacts generation and automated testing capabilities.

A Blue-Green DevSecOps environment, a practice recommended by AWS, provides essentially identical environments used as development and operations environments. The objective is to develop the new version of the application on the Green environment while the already deployed (older) version of the application operates on the Blue environment. When the Green-based system is completed, tested and ready for transition to production, the routing switch is flipped and all Green environment and application then become Blue (operational) and vice versa, for the previous Blue environment becomes Green, and ready for new application updates. Blue-Green deployments enable you to launch a new version (green) of your application alongside the old version (blue) and monitor and test the new version before you reroute traffic to it, rolling back on issue detection.

A Blue-Green DevSecOps solution presents a complicated access control challenge because elements of the system may be shared by application users, application administrators, and application developers. As systems are swapped from Green to Blue and back again, user access must swap in unison so that application users aren't given access to developmental systems and developers aren't allowed to disturb production systems. Such an environment provides a rich opportunity to examine Open Software Defined Perimeter (SDP) technologies.'

Access control data flows are illustrated in target architecture defined by the Orion JK21 collaboration team; reference Figure 2. In this diagram, the Blue-Green DevSecOps systems, dedicated to the development and hosting of the F-Force application, are secured by the Waverley OpenSDP solutions which provides an OpenSDP Controller (a Policy Decision Point (PDP)) and the OpenSDP Gateway (a Policy Enforcement Point (PEP)). The Controller acts much like an Identity-as-a-Service (IDaaS) solution providing onboarding capabilities for users and devices and management of access control. The Gateway allows and denies access to the Blue-Green environments based upon a Single Packet Authorization (SPA) message sent from the Controller via a dedicated communication channel.

Figure 1. These responsibilities are important for properly securing logically stacked systems like ORION JK21 where upper layers inherit controls from the lower layers in the stack. Authorizing to Operate each layer as a separate service requires a clear understanding of the SSRM and uses Customer Responsibilities Matrixes (CRMs) when "leveraging" the ATOs of the lower layers in the stack in the A&A process of the upper layer or services.

The OpenSDP Gateway (operated by Waverly Labs) acts as a deny-all firewall unless a proper SPA packet with the access decision for a user and device is received.

The OpenSDP Controller provides Policy Decision Point (PDP) capabilities and acts as an Identity-as-a-Service (IDaaS) operated by the Resilient<sup>8</sup> company. The F-Force app end-user is required to onboard and secure the credentials. The F-Force App owner has development and administration responsibilities and manages its employees: the application developer and the application administrator.

The 2Twelve company manages the E3 Labs which provides the underlying IaaS.

---

<sup>8</sup> [Waverley Labs Launches RESILIENT™ Essential to Zero Trust Model | Waverley Labs](#)

### 1.4.3 Test System Components & Configuration

The Ubuntu 16.04 Virtual Machine (VM) host operating as OpenSDP Gateway and the networking configuration implemented using the 2Twelve's E3 Labs IaaS are illustrated in Figure 2.

To assess the environments for compliance against TIC 3.0 network requirements, a set of tests are designed and run using a Test Client (an Ubuntu 16.04 VM).

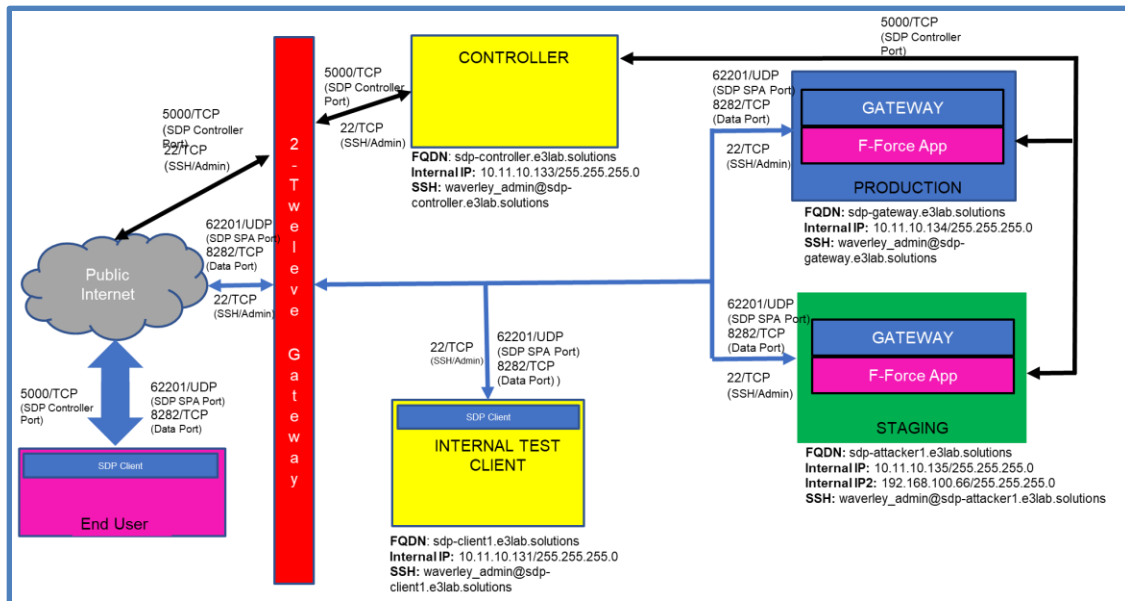


Figure 2 Network Host and Protocol Mapping for the Orion JK21 Test Environment

The Specific VM sizing, and software loads used for this collaboration are provided in Table 1.

**Table 1. Hardware and Software Testbed configuration**

Machine	Function	Specs	Software
1	Gateway-Prod (TIC 3.0 PEP)	CPU: 2cores RAM: 16 GB HD: 8 GB vNIC: 1Gb (*dual homed)	Ubuntu 16.04.3 LTS OpenSSL v1.0.2g 1 Mar 2016 conntracklibssl-dev libjson0 libjson0-dev libpcap-dev texinfo libtool autoconf
2	Gateway-Stage (TIC 3.0 PEP)	CPU: 2cores RAM: 16 GB HD: 8 GB vNIC: 1Gb (*dual homed)	Ubuntu 16.04.3 LTS OpenSSL v1.0.2g 1 Mar 2016 conntrack libjson0 libjson0-dev libpcap-dev texinfo libtool autoconf
3	Controller	CPU: 2 cores RAM: 8 GB HD: 500 GB vNIC: 1Gb	Ubuntu 16.04.3 LTS OpenSSL v1.0.2g 1 Mar 2016 nodejs v8.12.0 npm v6.4.1 mysql v14.14 Distrib 5.7.24
4	Client1	CPU: 2 cores RAM: 8 GB HD: 8 GB vNIC: 1 Gb	Ubuntu 16.04.3 LTS OpenSSL v1.0.2g 1 Mar 2016 libjson0 libjson0-dev libpcap-dev texinfo libtool autoconf

The deployed OpenSDP solution provides a simplified implementation of what a development team may want to deploy in order to limit access to Production (Blue) and Staging (Green) environments to only

authorized users. Users with authorization to update and/or work in the Staging environment are only permitted access the specific service(s) protected by the SDP Gateway for the Staging environment. The same is true for users with authorization to update and/or work in the Production environment. They are only permitted access to the specific service(s) protected by the SDP Gateway for the Production environment.

Because each SDP Gateway implements a deny all firewall policy, attempts to perform reconnaissance using port scans or similar techniques reveal no listening ports (aka services) available in either Production or Staging environments. It's not until the user, with legitimate SDP Client software and credentials sends a properly formatted and encrypted Single Packet Authorization (SPA) request to the SDP Gateway will the service port be made available for connection. The user has a limited window (30 seconds) to make the connection to the requested protected service before the SDP Gateway removes the listening port from the firewall ACCEPT list; re-instating the firewall deny all policy and again making all protected services invisible. As long as the user's established connection remains active (i.e. traffic traversing the connection), it will remain connected.

#### 1.4.4 TIC 3.0 Capabilities of the OpenSDP

Alongside the rest of the pilot, an InSpec Profile was developed to demonstrate automated testing of an OpenSDP implementation for TIC 3.0 Network PEP Capabilities.

TIC 3.0 is non-prescriptive cybersecurity guidance developed by DHS/CISA to provide agencies with the flexibility to secure distinctive computing scenarios in accordance with their unique risk tolerance levels. CISA encourages agencies to leverage the Security Capabilities Catalog, use cases, and overlays when implementing the TIC capabilities in their network environment. These documents, in conjunction with documents like National Institute of Standards and Technology (NIST) Cybersecurity Framework (CSF) and NIST Special Publication (SP) 800-53, help agencies design a secure network architecture and determine the appropriate requirements and service providers tailored to their agency.

Taking in consideration CISA's recommendations, the ORION JK21 team tested if the OpenSDP's Gateway that provides the PEP capability for the Blue-Green environments implements the TIC 3.0 Network PEP capabilities. Therefore, the tests were developed based on the descriptions given for the Network PEP Capabilities listed in Table 7 of the Security Capabilities Catalog (Volume 3 of TIC 3.0)<sup>9</sup>.

#### 1.4.5 Cloud Infrastructure Services

The E3 Lab (see Figure 2) is deployed on Nutanix Hyper-Converged-Infrastructure hardware within 2Twelve Solutions' Multi-Cloud environment. 2Twelve Solutions personnel manage E3 Lab and are responsible for engineering, securing and operating everything from the OS layer on down. The pilot implementation leverages security and management services that is owned and operated by 2Twelve Solutions. Authorized personnel from Waverley Labs have privilege access to the Ubuntu hosts that exist

---

<sup>9</sup> Trusted Internet Connections 3.0 Volume 3 Security Capabilities Catalog  
[https://www.cisa.gov/sites/default/files/publications/CISA%20TIC%203.0%20Security%20Capabilities%20Catalog%20v2.0\\_0.pdf](https://www.cisa.gov/sites/default/files/publications/CISA%20TIC%203.0%20Security%20Capabilities%20Catalog%20v2.0_0.pdf)



in the pilot implementation in order to configure and deploy the OpenSDP platform. Therefore, Waverley Labs shares in the responsibility to manage the Ubuntu VM hosts (Blue-Green environments).

The instances in the deployment and the layer 2 networking that it is connected to has been stripped down to the bare minimum. The instances have direct access to the Internet gateway and minimal protections have been enabled. This bare-bones design is meant to showcase the full capabilities of the software security overlays built and managed by our partners in the Pilot, such as the Software Defined Perimeter and Zero Trust Architecture that is to be installed on the provisioned instances.

With that said, 2Twelve Solutions, illustrated in Figure 3, is committed to the security of our environments so we have the necessary host-based monitoring, configuration hardening and system protections running within the instances. We continuously monitor for the availability and utilization of resources, privilege access management, anti-malware, system integrity, and denial of service attempts. We also employ inline NGFWs at the perimeter so that 2Twelve Solutions can act immediately to mitigate any abusive behaviors or threat actors that manage to bypass the software security overlays that are implemented, if such a scenario were to occur.

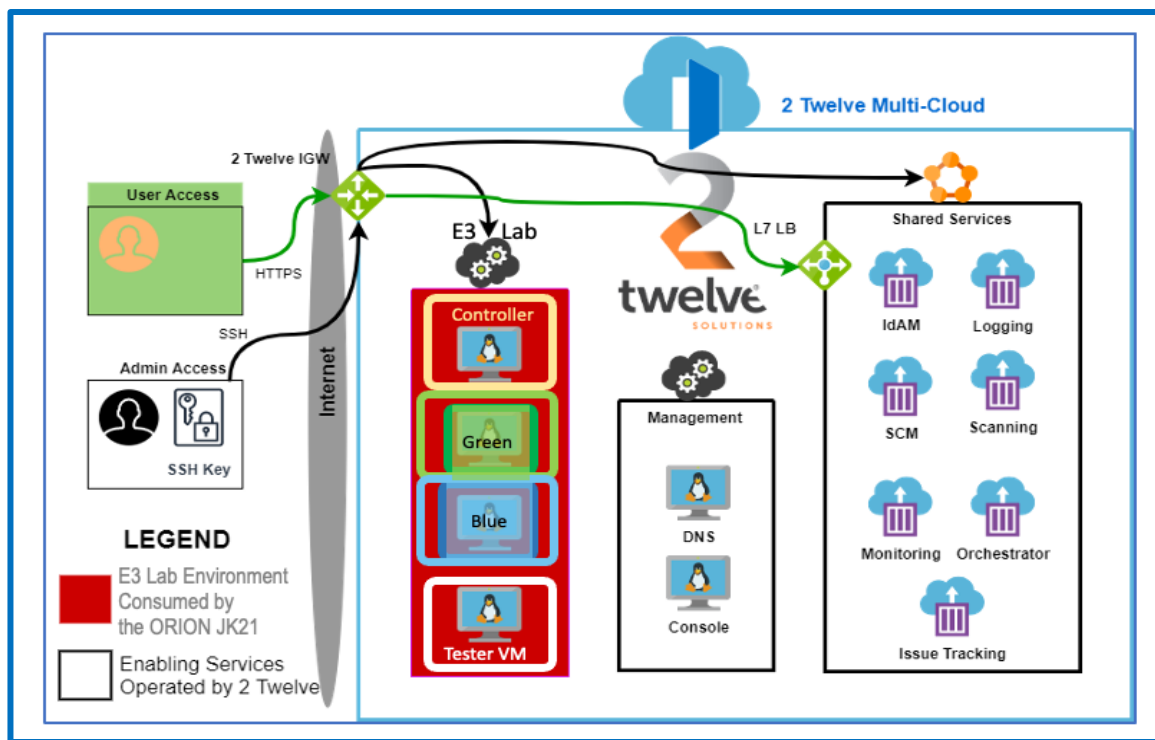


Figure 3. E3 Labs 2Twelve IaaS Cloud

As the responsible party for the cloud infrastructure services, we also documented the sets of security controls that were required for the scope of this pilot and the policy and procedures that apply to the instances that have been deployed as part of the pilot. Please refer to the E3 Lab IaaS System Security Plan for further details.

#### 1.4.6 Documentation Automation Process

To demonstrate tool interoperability using the NIST OSCAL, an RMF artifact sequencing was demonstrated. As illustrated in Figure 4, the process started with an understanding of the security

control implementation by the various employed systems and services. A Shared Security Responsibility Model (SSRM) was generated from the system layers including IaaS, OpenSDP Gateway, OpenSDP Controller, F-Force (FICAM), and application end user. The SSRM was then used to generate the Customer Responsibility Matrix (CRM) using available component SSP information for the IaaS, PaaS, and IDaaS systems. The impact level of each of the system layers was assumed to be FISMA moderate. From the CRM, an OSCAL representation of a demonstration SSP was then possible using the GRC tools. The process execution was then able to continue through OSCAL data sharing. GovReady and ATOaaS were employed for SSP generation, Atlasity for SAP and SAR generation, and the MITRE SAF InSpec and Heimdall Visualization tools for automated real-time assessment.

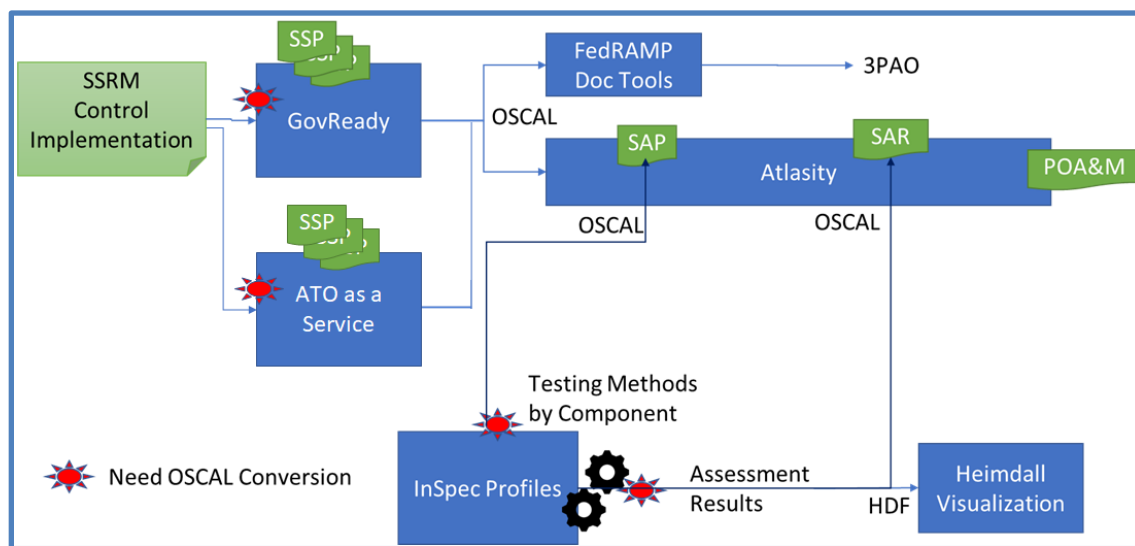


Figure 4. ATO Automation Process for the Orion JK21 Collaboration

## 2 Deliverables

### 2.1 Sprint #1: Deliverables

#### 2.1.1 GovReady: OSCAL SSP Generation

The working group began its first Sprint around compliance automation in early 2020, shortly after NIST released its first OSCAL draft in mid-2019.

One of the goals of this Sprint was to import spreadsheet-based control content describing the E3 Labs Environment controls into the early-stage OSCAL assessment layer data models. To accomplish this, ATARC Working Group member GovReady PBC used their open-source software, GovReady, as a host for an initial host of OSCAL.

Previous proof of concepts of the component-based, Compliance as Code approach to automatically generating machine-readable SSP had been accomplished using command-line tools such as

Compliance-Masonry<sup>10</sup> that worked with content stored in multiple files on the file the file system. These early versions attracted developers comfortable with the command-line and seemed to align with a DevOps and pipeline-oriented thinking. In practice, however, this quickly proved impractical for making SSP automation available to less technical users, such as ISSOs, ISSMs, and Assessors.

GovReady was one of the first available tools that used a persistent database and modern web interface to explore the concept explicitly allocating control implementation statements to individual system components in order to create machine-readable, reusable content for assembling SSPs and other ATO documentation. The Component Library provides a noteworthy demonstration of how component-based Compliance as Code will reduce the cumbersome and error-prone process manually maintaining SSPs.

During this Sprint, the existing spreadsheet content was parsed via a script and imported into GovReady. Additionally, a FedRAMP Word Document template was used as source material for generating a human readable SSP in GovReady. The successful importing of the content into GovReady and generation of machine-readable version of the SSP in OSCAL and human-readable version of the SSP represented an important milestone of showing components and OSCAL within the context of an GRC application environment and providing a foundation SSP in OSCAL to begin exploring.

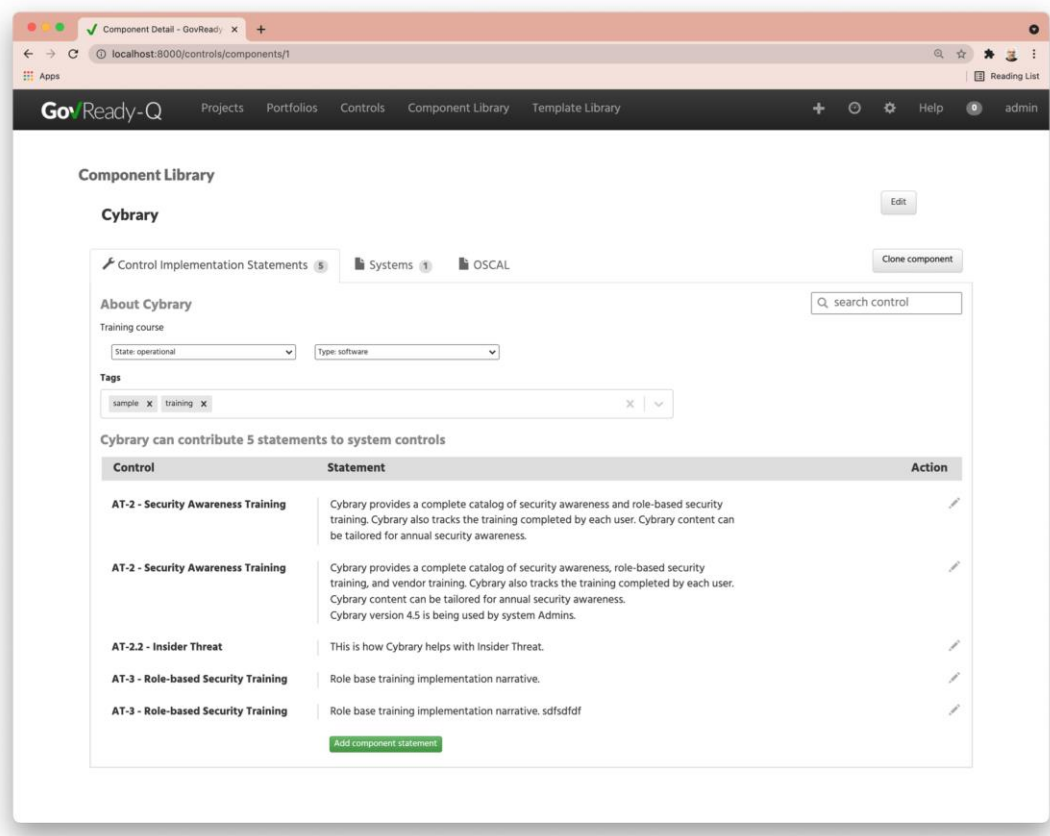
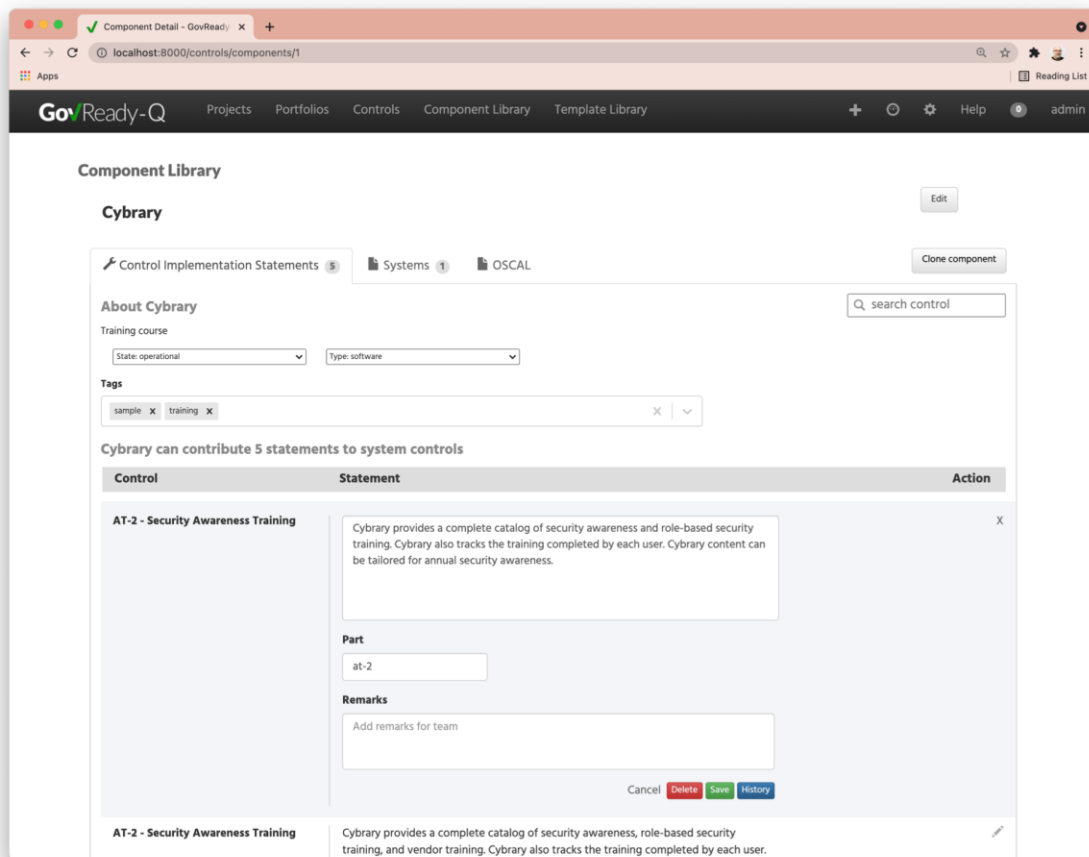


Figure 5. Screenshot of GovReady Component Library

<sup>10</sup> Footnote here on Compliance Masonry, Open Control and other tools



**Figure 6. Screenshot of GovReady Component Library**

### 2.1.2 GovReady: Lessons Learned

Specific lessons learned using the GovReady tool were:

- 1) Lesson: Convert an existing FedRAMP SSP template that was traditionally used for manually copying, pasting, and writing content to a template that supported being generated automatically and displaying control implementation statements assembled from multiple components.

Traditional SSPs contain control implementation statements that are unique snowflakes written for that specific control for that specific system. Component modularity is an OSCAL construct that facilitates content reuse by associating partial control implementation statements with individual system elements. To do so, the content itself must be written read well when reused and combined with content from other components. Also, the presentation on the human readable page must also be modified to represent the components and display as part of a coherent description of the fully implemented control.

- 2) Lesson: This initial work also revealed how much the software tools we use drive our thought process in preparing content (e.g., control implementation statements.)

Initial content had been prepared in spreadsheets by members of the team describing E3 Labs. As a tool, spreadsheets naturally support the two dimensions of Controls (rows) and Statements (column), encouraging the writing of a single statement to fully describe the control implementation. But spreadsheets do not easily represent the additional dimension of Components, making it much harder to write the content in a component-based format.

This lesson facilitated further development within the GovReady- Q software to make statement writing environment that helped the users think in all three dimensions of Controls, Statements, and Components.

3) Lesson: An OSCAL model supporting the Customer Responsibility Matrix (CRM) was proven necessary to accurately capture the fact that a customer is responsible for configuration of a component that is provided by a cloud service provider. To fully take advantage of the OSCAL potential of machine-readable SSPs, our tools will also need to help users think in a fourth dimension of who is responsible for an individual control. Going forward, it is important to add transparency to components and SSPs

### 2.1.3 ATO as a Service: OSCAL SSP Generation

In addition to using GovReady to develop the 2Twelve E3 Labs SSP in OSCAL format, examined another tool called “ATO as a Service” for rendering the SSP in OSCAL format. ATO as a Service is an AI-powered Software as a Service (SaaS) that automates both FedRAMP and the Risk Management Framework and creates RMF artifacts (e.g., SSP, SAP, SAR, POA&Ms, etc.) in OSCAL format. Since OSCAL enables data interoperability between A&A tools, and since both GovReady and ATO as a Service produce SSPs in OSCAL format, we can integrate both tools into our workflow.

The system assessor can use ATO as a Service to easily create an SSP in OSCAL format without knowing any OSCAL whatsoever. ATO as a Service features a user-friendly user interface and workflows that guide the assessor through Categorize, Select, and Implement Steps of the RMF as defined by NIST SP 800-37 rev. 2:


**RMF Categorize Step**—ATO as a Service automates the process of (1) Documenting the characteristics of a system; and (2) Categorizing a system according to FIPS 199 and NIST SP 800-60 vol. 1 & 2.

1. **Documenting the Characteristics of OpenSDP:** The system owner can easily document the OpenSDP in ATO as a Service by copy and pasting system information from the SSP in human readable format (e.g., Word document) into the ATO as a Service Components and Inventory page. Upon review of the E3 Labs SSP Word document, it appears that 2 components are consistently referenced in the controls implementations: Green and Blue Ubuntu Operating Systems. The system owner can go to the ATO as a Service Components and Inventory page to create these components manually, or if available, import an OSCAL formatted components definition as illustrated in Figure 7.

All components that are created and imported into the component library are available during the RMF Implement Step for controls implementation.

## Add/Edit System Component

Use the fields below to add/edit a system component for your SSP.

 **Component Basics**

Component Name

Operating System

Operational

Software or Hardware

Component Description

OpenSDP Operating System Component. The system is installed with the Ubuntu 1604 operating system. Once the base system is up with the use of the Ansible Tower, a number of Ansible playbooks are executed to ensure a repeatable installation process of the applications and its dependencies.

Component Purpose

The Operating System governs the functionality of the IaaS component.

**Figure 7. ATO as a Service Add Component Workflow**

2. Categorizing OpenSDP: The system owner already knows the information types that are relevant to the OpenSDP (PaaS) system and already determined the Moderate adverse impact with respect to the loss of confidentiality, integrity, and availability of the system and the information processed, stored, and transmitted by the system. ATO as a Service's workflow for the RMF Categorize step enables the system owner to add the appropriate information types to an information-types table and subsequently determines the appropriate system security categorization as illustrated in Figure 8.

### Security Categorization (FIPS 199)

What type of federal government information/data is populated or generated within the system?

**Information Types**

+ Add Row

Information Type	Identifier	Confidentiality	Integrity	Availability	
Research and Development	D.20.1	Low	Moderate	Low	Delete
System Development	C.3.5.1	Low	Moderate	Low	Delete
System Maintenance	C.3.5.3	Low	Moderate	Low	Delete

**Security Objective Categorization**

Security Objective	Low, Moderate, High
Confidentiality	Low
Integrity	Moderate
Availability	Low

**System Security Categorization**

Based on the information types you selected, your system is assigned to the following security categorization impact level:

System Security Categorization  
Moderate

Figure 8. ATO as a Service Security Categorization Workflow

**RMF Select Step**— Once the OpenSDP system security categorization has been reviewed and approved, ATO as a Service begins the RMF *Select* workflow where the system owner can select the appropriate security control baseline, common controls, overlays, compensating controls, etc. for the system as illustrated in Figure 9.

### Select & Tailor Security Controls

ATO as a Service™ enables you to tailor the security controls included in your System Security Plan.

Based on your system security categorization, ATO as a Service™ has selected and highlighted all of the baseline security controls for your System Security Plan

You may tailor in additional controls and or tailor out controls by checking and/or unchecking the appropriate security controls below. You may also apply one or more available overlays to your SSP.

If you want to undo all changes and revert back to the baseline set of security controls, click the 'Undo all Tailoring' button.

**Available Security Controls Baselines**

☐ NIST SP 800-53 rev4 High Baseline

☒ NIST SP 800-53 rev4 Moderate Baseline

☐ NIST SP 800-53 rev4 Low Baseline

**Available Overlays**

☐ CNSSI-1253-PHI-Privacy-Overlay-profile

☐ CNSSI-1253-PHI-High-Privacy-Overlay-profile

☐ CNSSI-1253-PHI-Low-Privacy-Overlay-profile

☐ CNSSI-1253-PHI-Moderate-Privacy-Overlay-profile

☐ CNSSI-1253-Space-Platform-Overlay-profile

**Selected SSP Security Controls**

Search security controls

Undo all Tailoring


Included?	Control ID	Control Title
<input checked="" type="checkbox"/>	AC-1	Access Control Policy and Procedures
<input checked="" type="checkbox"/>	AC-2	Account Management
<input checked="" type="checkbox"/>	AC-2(1)	Automated System Account Management
<input checked="" type="checkbox"/>	AC-2(2)	Removal of Temporary / Emergency Accounts

Figure 9. ATO as a Service Select &amp; Tailor Security Controls Workflow


**RMF Implement Step** – For each security control in the SSP, the system owner can navigate to the control using the ATO as a Service UI and copy and paste the implementation details from the SSP Word document. ATO as a Service is fully compliant with OSCAL 1.0.0 and supports leveraged authorizations and control inheritance between systems as illustrated in Figure 10. Using the leveraged authorization capability, the OpenSDP owner can identify the inherited controls and document how the OpenSDP owner’s responsibilities are met and how the controls are satisfied.

### AC-3 Access Enforcement


The information system enforces approved authorizations for logical access to information and system resources in accordance with applicable access control policies.

 **Supplemental Guidance**

Access control policies (e.g., identity-based policies, role-based policies, control matrices, cryptography) control access between active entities or subjects (i.e., users or processes acting on behalf of users) and passive entities or objects (e.g., devices, files, records, domains) in information systems. In addition to enforcing authorized access at the information system level and recognizing that information systems can host many applications and services in support of organizational missions and business operations, access enforcement mechanisms can also be employed at the application and service level to provide increased information security.

 **AC-3 Control Summary Information**

Responsible Role ▼

 **Implementation Status**

☐ Implemented

☐ Partially Implemented Explanation

**Figure 10. ATO as a Service Controls Implementation Workflow**

**Publish SSP in OSCAL Format**— When the system owner enters all the control implementation details, ATO as a Service enables the owner to validate the information in OSCAL is well formatted per OSCAL schema and publish the completed SSP in OSCAL format as illustrated in Figure 11.



**Validate & Publish SSP**

Click the 'Validate & Publish SSP' button below to publish your SSP package of documents. The SSP package includes:

- SSP in OSCAL format (JSON/XML/YAML)
- System Security Plan document
- Security Control Policies and Procedures documents
- System Security Plan attachments

Publishing your SSP can take up to 5 minutes to complete.

You will receive a notification when the package is ready to be downloaded.

**Publish SSP Details**

ATO as a Service™ automatically validates your SSP against the appropriate OSCAL schema and publishes your SSP in various formats (e.g. JSON, XML, Microsoft Office, etc.).

Select from the SSP publishing options below

☐ Publish to OSCAL file      OSCAL Language: JSON ▼

☐ Publish to Microsoft Office (Word and Excel) files

Figure 11. ATO as a Service Validate & Publish SSP Workflow

#### 2.1.4 ATO as a Service: Lessons Learned

Generating and handling OSCAL content is challenging. At cFocus Software, we have been integrating OSCAL into ATO as a Service over the past two years, and during that time, we have put into practice a number of valuable lessons related to working with OSCAL formats were learned:

1. **Understanding OSCAL data model hierarchy and data imports:** One of the basic elements of OSCAL's architecture is the hierarchy of data models that allow data to be imported rather than duplicated. It never behooves you to try to take a shortcut and duplicate data in a higher data model rather than importing the data from a lower data model. Be sure to understand how profile models import from catalog models, how SSP models import from catalog models, how SAP models import from SSP models, etc. We created import functions that allow us systematically import data from lower data models when necessary.
2. **Utilizing Components:** Components are perhaps the single most helpful data model to automate OSCAL generation. Component definitions are reusable OSCAL data models that allow you to specify component details, security controls that can be brought into compliance by the component, implementation details that can be added to an SSP, etc. For example, we used ATO as a Service to create components for the 2Twelve E3 Labs environment, and by doing so, the component was able to autofill the SSP narratives for all of the security controls that were satisfied by that component!
3. **Interoperability with standard tools:** Generating OSCAL is much easier when you integrate various standard tools for UUID generation, XML/JSON validation, XML/JSON path expression management, etc. Be sure to have solutions for all these objectives, as they will be necessary to successfully generate OSCAL content.

### 2.1.5 C2 Labs: Importing and Processing SSPs

C2 Labs' Atlasity played a centric role for the system's assessment and the integration of system's configuration and hardening scanning results. Atlasity was responsible for importing the SSPs generated by different tools.

Importing (a) the SSP for the IaaS layer managed by E3 Labs which used GovReady to generate the SSP and (b) the SSP for the OpenSDP layer (PaaS) managed by Waverley Labs that used the ATO as a Service to generate the SSP proved to be challenging due to the discrepancies between the implementations provided by the two tools.

Moreover, Atlasity was responsible for extracting the necessary security control parameters needed by InSpec profile used to test the Ubuntu VMs for proper hardening and secure configuration.

### 2.1.6 Atlasity Integration of InSpec and Heimdall

The Atlasity required enhancement to enable interoperability between compliance tools and security testing tools:

1. Atlasity was enhanced to strongly type parameters on our side so that they could be easily parsed by InSpec. While not required by OSCAL, this enhancement improved the interoperability.
2. Atlasity was modified to improve the use of the NPM InSpec library to perform the integration in a more robust/supportable manner using inspecjs version 2.5.1
3. Atlasity was enhanced to leverage Heimdall Data Format (HDF) JSON schema to import InSpec profiles

### 2.1.7 C2 Labs: Lessons Learned

For C2 Labs, there are several lessons we learned from the adoption of OSCAL at all layers (catalog, profile, security plan, component, SAP, SAR, and POAM) in our platform. These lessons include

1. **Architectural Flexibility:** OSCAL leverages a complex and detailed schema that is unlikely to be an exact match with existing commercial tools. The ability to quickly add new fields to our internal schema to ensure we had all required fields made this process smoother. The complexity varies by level and for some portions of OSCAL (i.e. Components) we did not have that object in our schema at all. It required significant investments of time and resources to close gaps in the schema. Also, what tool vendors label fields may differ slightly from the OSCAL naming conventions which can result in some user confusion when reviewing the OSCAL output. Finally, we were big fans of the flexibility in the standard to include optional properties. We collect a great deal early in the development process; it was hard to tell if we had valid OSCAL. The output would pass the eyeball test, as in it generally looked right, but data validation was a challenge. The NIST team provided out of the box validators using the AJV library that allowed us to integrate this tool into our platform and perform a complex validation based on an authoritative source from NIST. The AJV tool saved an incredible amount of time in what would have otherwise been a very complex and time-consuming custom validation library in our platform.
2. **Lack of Strongly Typed Parameters** – when passing parameter data via OSCAL between tools (i.e. from Atlasity -> MITRE Heimdall), the parameters were not really machine readable in a

consistent way. This caused a manual translation step between tools to perform the initial integration. In future MVPs, we were able to add support for namespaces to allow for stronger parameter typing by extending the model. This extension allowed a pure machine to machine translation of parameters between tools.

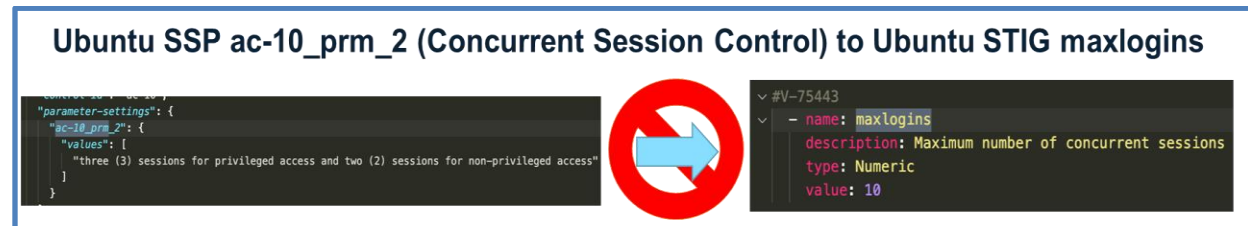
3. **Version Differences** – as an early adopter, we felt the growing pains of significant changes in the OSCAL model as it moved between releases and towards the ultimate 1.0.0 release. These changes required a fair amount of refactoring along the way to stay current with the standard. The lesson learned for us is that being an early adopter required significantly more resources than initially planned to support these changes. However, it also had the upside of allowing us to provide direct feedback to the NIST team on the standard’s development while also giving us an advantage as one of the first tools to adopt the standard.
4. **Parsing** – the flexibility in the OSCAL standard is a blessing for performing very complex system modeling. However, it was a curse for trying to consistently parse documents. There were differences in how data was laid out between layers and vast differences in how tools implemented the standard and where information might be stored. This complexity created early challenges in consistently parsing catalogs, profiles, and security plans in a way that allowed full automation of importing into Atlasity using OSCAL.

### 2.1.8 OSCAL – InSpec Integration Problems

OSCAL parameters and InSpec inputs have some differences that are important to understand. These differences have required the addition of new data into the OSCAL file using OSCAL props to support testing with InSpec. Similar considerations would be required for any tool that is attempting to extrapolate system configuration information from OSCAL.

**Table 2. OSCAL Parameter -InSpec Input Mapping Issues**

OSCAL Parameters	InSpec Inputs
Free form text fields	Strongly typed input values. <b>(String, Number, Boolean, Array, Key Value Pair)</b>
Values from OSCAL are passed through tools to the final document, not parsed by them.	Values are evaluated and used to affect the checks that InSpec performs.
Defines high-level requirements of securing a system.	Describes, specific to a particular IT system or application: <ul style="list-style-type: none"> <li>• Where to begin auditing for the correct technical implementation of a control, OR</li> <li>• What to compare a current configuration value to.</li> </ul>
Requires advanced Natural Language Processing for a computer to understand the context and translate into the technical values and audits.	Is already machine readable.



**Figure 12. Example#1 SSP Parameter Comparison for InSpec Integration**

These two parameters are related; however, they are not equivalent. The STIG requirement for this parameter is “The Ubuntu operating system must limit the number of concurrent sessions to ten for all accounts and/or account types.” The SSP parameter describes different session limits based on account type. The Ubuntu STIG sets an overall limit without considering account type.

Modifying the Ubuntu STIG InSpec check is not the correct answer since this is not a shortcoming of InSpec, this is a difference between the STIG and the organizational requirements reflected in the SSP. Properly addressing this requires the organization to write an organization-specific overlay that overwrites the STIG check, which requires coding effort and is not just a simple mapping.



**Figure 13. Example#2 SSP Parameter Comparison for InSpec Integration**

This parameter is used by InSpec to identify if the OS is currently supported. The OSCAL document does not include any parameters for this information. To extrapolate this information from OSCAL, it would require NLP which must understand the following steps:

1. That flaw remediation happens through software updates.
2. That the end of life of an Operating System is when it stops receiving software updates.
3. That the operating system in question is Ubuntu 16.04.
4. That it needs to lookup the end of life for Ubuntu 16.04 (which is April 2021)
5. Any additional organizational support programs that may extend this date (most OS manufacturers will allow the purchase of longer-term support for a fee).

## 2.2 Sprint #2: Deliverables

The implementation and tools’ integration initiated during Sprint #1 continued during the Sprint #2, and the team was able to complete the following tasks:

1. Implemented Atlasity support for strongly typed parameters, ports and protocols, and classification which were not previously supported. As of version 2.4.0, Atlasity supports all required fields of all layers of the OSCAL stack and can export compliant content at every layer.
2. Added data validation via AJV to Atlasity for machine generated OSCAL, including custom validation language to make sure reported errors provide additional information trouble shooting in the Atlasity UI.
3. Added an Atlasity Component Definition model with functionality to export data .to the MITRE SAF using the OSCAL 1.0.0 Component Definition model,
4. Demonstrated the ability to pass to the MITRE SAF an OSCAL Component Definition instance which uses OSCAL extension mechanisms where it was parsed to support a tailored InSpec/STIG scan.
5. Parsed the MITRE results using the NPM InSpecJS library (version 2.5.1) to generate an SAP/SAR based on automated scan results while self-updating the SSP paperwork.

### 2.2.1 Demonstrated Tools

During the Sprint #2, the team integrated the following tools for the listed purposes:

- [GovReady](#) – generation of the initial System Security Plan (SSP) for 2Twelve E3 Labs and native export to OSCAL
- [ATO as a Service](#) - generation of the initial System Security Plan (SSP) for 2Twelve E3 Labs and native export to OSCAL
- [Atlasity](#) – import of the NIST catalog, profile(s), GovReady/ATO as a Service SSP, generation of the component OSCAL, and parsing of MITRE results to auto-generate the Security Assessment Plan (SAP) and Security Assessment Results (SAR)
- [MITRE Security Automation Framework \(SAF\)](#) – ingest the OSCAL component file, tailored the scan based on OSCAL parameters/controls, and return the results in a HDF file that was parsed by the MITRE InSpecJS library in Atlasity.

### 2.2.2 Workflow InSpec Enhancements

The workflow executed for the Sprint #2 activities is illustrated in Figure 14. The workflow highlights the process developed to pass onto InSpec parameters found in an OSCAL SSP.

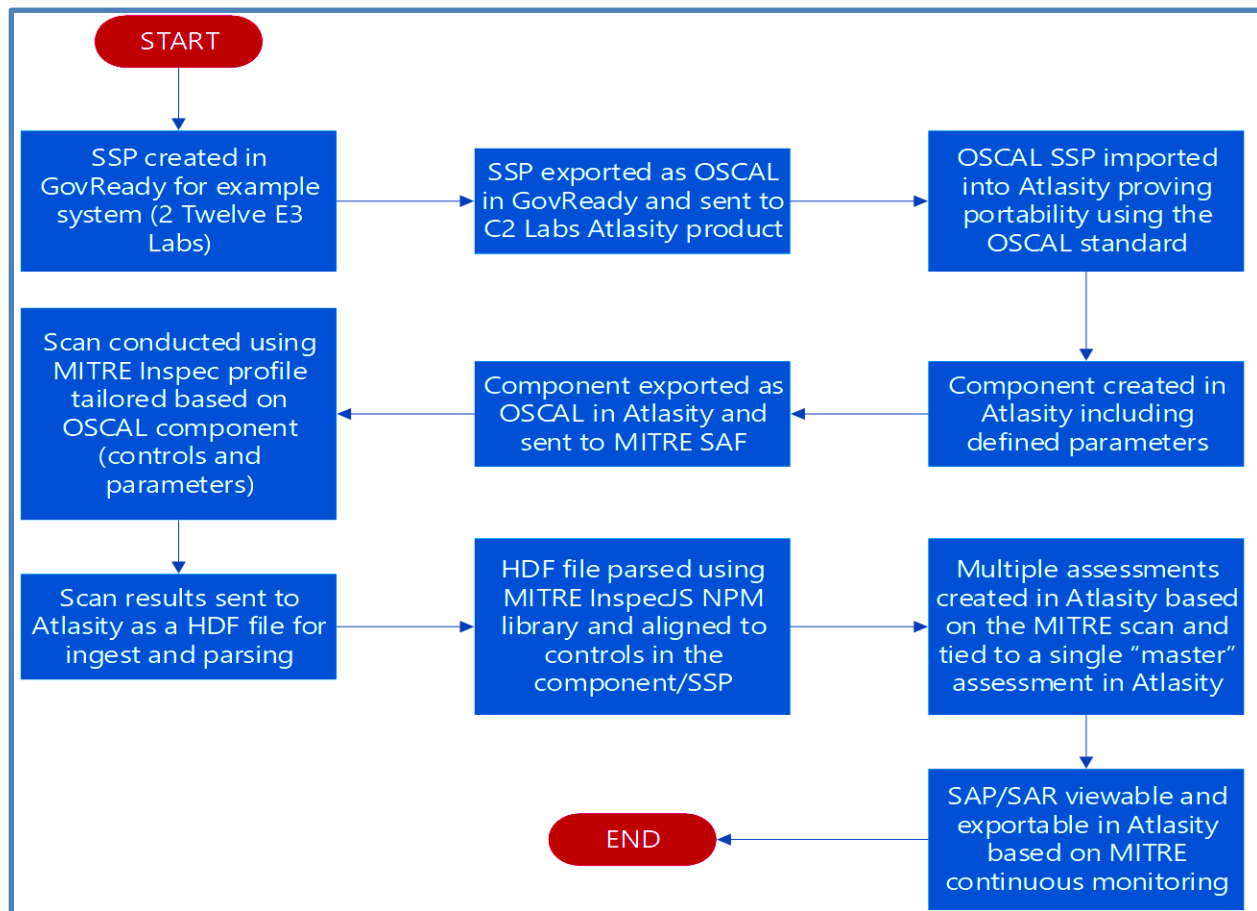


Figure 14. OSCAL Proof-of-Concept End-to-End Workflow

InSpec supports the concept of parameters (also called inputs) which can be used to customize the values used by an InSpec when checking the current state of a system. These parameters can be thought of “knobs” that can be used to tweak what values the system will be tested against.

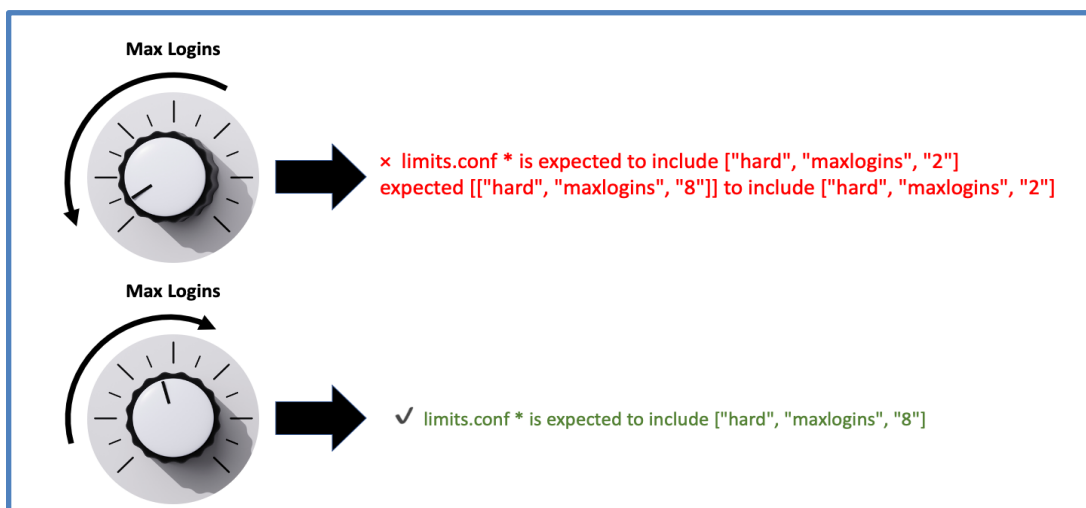
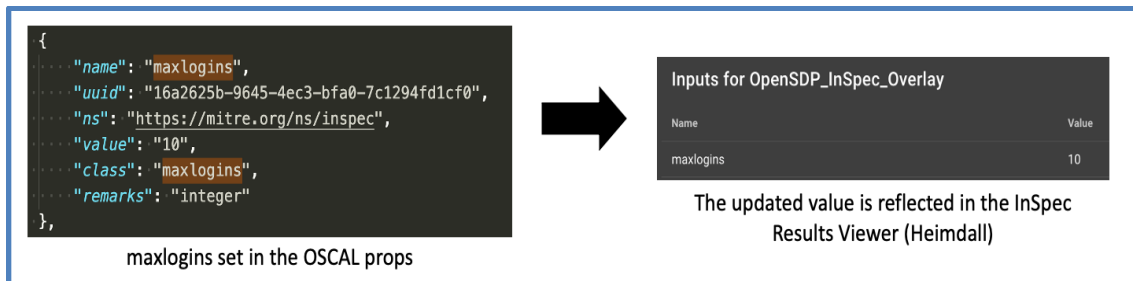


Figure 15. Customizing the test using InSpec parameters allows the test to better reflect the expected state of the system

To read information from OSCAL, a plugin had to be created that could read the appropriate parameters from OSCAL and incorporate their values into the InSpec run. To achieve this, an InSpec plugin was created called [inspec-oscsl](#).



**Figure 16. The value set in OSCAL is read and incorporated when executing the InSpec Profile.**

The [inspec-oscsl](#) plugin can read the appropriate OSCAL properties and pass-through appropriate data from OSCAL to InSpec. This integration provides for the ability to tailor InSpec profiles using only information stored in an OSCAL file and allows for all the information about the system to be stored in a single location.

To pass through the data to InSpec from OSCAL, the working group used the OSCAL Extension Model to design an extension instance. The Extension Model allows developers to extend core OSCAL data structures in a reusable way by developing an extension instance conforming to the Extension Model Schema.

The Extension Model Schema defines what, how, and where custom data can be added in specific OSCAL models, like in the case of our pilot, a component definition. Developers build an extension instance that follows this schema, and this extension instance can be analyzed to programmatically find the locations of custom data and their structure. Figure 16 illustrates this relationship between an extension instance, an OSCAL component definition, and the mapping to InSpec input parameters that are processed by the [inspec-oscsl](#) plugin. This approach allows developers to write code that looks for extension data only once and analyze any number of extension instances with that code later. They do not have to rewrite new code for each extension instance on a case-by-case basis.

While designing the extension instance, the team also had to map how low-level security baseline configurations (measured by InSpec tests) to strategic compliance goals of Risk Management Framework (defined in OSCAL). Customarily, there is not always a one-to-one mapping. InSpec, for example, will make granular checks to test a server running Ubuntu Linux around restricting permissive local authentication, permissive remote authentication via SSH, and hardening of the SSH daemon itself.



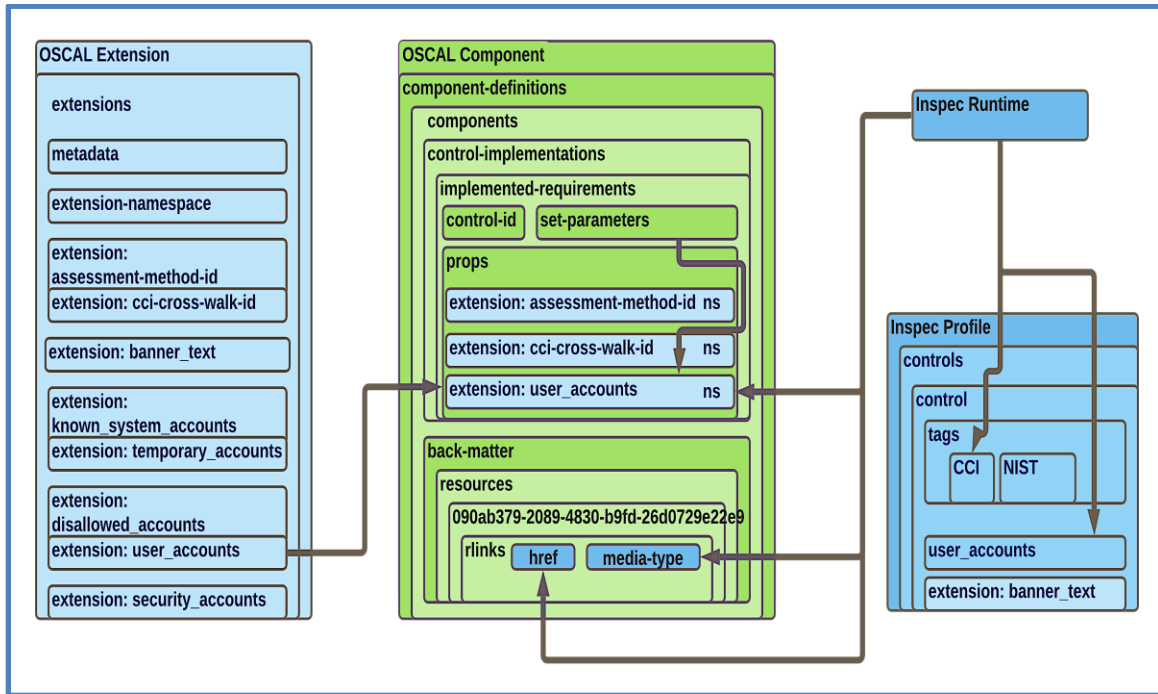


Figure 17. Proposed OSCAL Extension Model for InSpec Integration

Control requirements for the Risk Management Framework’s SP 800-53 baselines, on the other hand, are higher-level. For example, an SP 800-53 control will require that an information system enforces session length for authenticated users and do so for all sub-components of the information system. When such control is implemented by an Ubuntu Linux server, a set of granular details are necessary to be checked to confirm and collect the evidence associated with the higher-level control requirement. With OSCAL, these relationships are expressed in a machine-readable format, not implicitly by skilled analysts. Initially, the working group attempted to bridge the gap with the DISA Control Correlation Identifier (CCI) mappings present in the relevant InSpec profile. The working group attempted to crosswalk these CCI identifiers in code back to the SP 800-53 control identifiers. This approach, albeit efficient, led the working group to pivot after an important insight. CCI mappings, as the name suggests, only map at the control level, while OSCAL content requires more specificity, pointing to the individual statement level for requirements nested inside in a control. With that in mind, the working group continued to map all InSpec inputs in an Ubuntu Linux 16.04 LTS profile to two relevant RMF 800-53 control families at the requirements’ statements level.

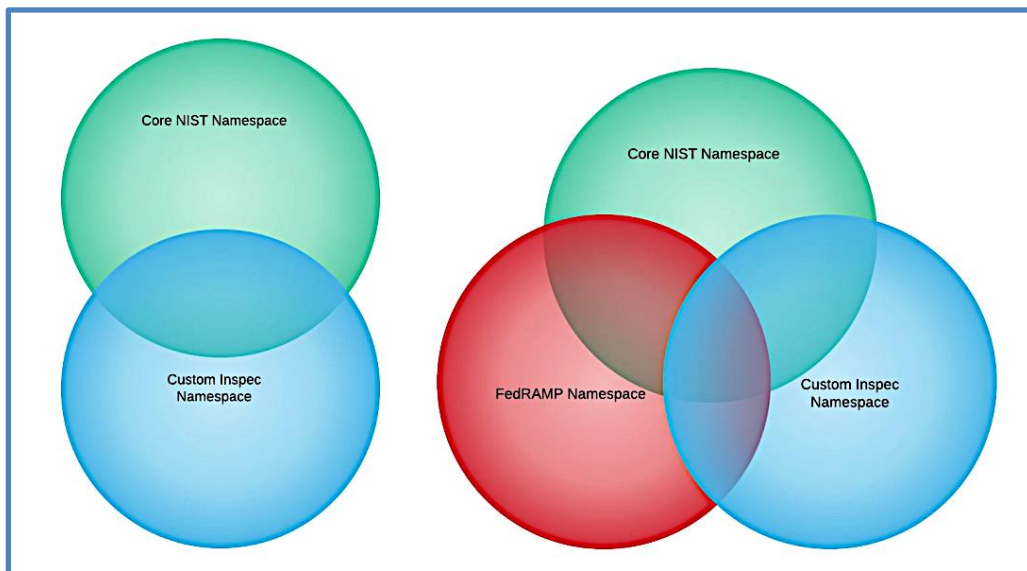
### 2.2.3 Towards the Concept of OSCAL Namespace

OSCAL is an information model for assessment of security controls. And that assessment, at its core, is cyclical communication between multiple parties. It is pivotal to organize who said what information and how the information fits together. To allow these parties need to add or modify data without erroneous overlaps, conflicts, ambiguity, or even resulting syntax errors, OSCAL has namespaces.

OSCAL provides developers and practitioners namespaces on certain kinds of data elements, allowing safe extensibility. Although a robust namespace mechanism exists in XML, the OSCAL namespace attribute is supported in all OSCAL data formats (XML, JSON, and YAML). It is different and distinct from



this XML-specific construct. It allows a developer to identify custom data granularly, preventing several classes of problem in OSCAL data modeling. Conceptually, OSCAL namespaces allow applications and their users to color-code data. They can recognize the color of data, opting in or opting out of analyzing or augmenting it.



**Figure 18. A Possible Namespace Model to Support OSCAL Integrations**

Namespaces allow OSCAL to support quick prototyping and production use of custom data elements, while maintaining sensible validation from OSCAL core. Otherwise, the NIST OSCAL development team would rigorously need to review, oversee, and add the inclusion of each new data element type into the core standard and update validations accordingly. For a diverse and heterogeneous security industry, this is a daunting obstacle. Moreover, this suboptimal approach only works so long as there are no conflicting data elements. Namespaces solves this problem as well.

Namespaces allow the use of similar data elements or duplicating identically named elements with differing constraints. Without distinct namespaces per-element, this would lead to duplicate data elements, likely resulting in syntax errors. Certain data elements have complex validation encoded into the core OSCAL information model. Without namespaces, developers cannot provide a data element with the same name, but with different validation constraints for a slightly different use case. Currently, programs like FedRAMP must use namespaces to express their unique data requirements within OSCAL using namespaces, or OSCAL would be inflexible and not suitable for purpose.

Namespaces also establishing simple logic and criteria for software to separate core OSCAL from optional data. Without namespaces or a similar mechanism, the burden of parsing and properly categorizing data elements and core OSCAL data from additional data significantly increases. Developers, in that case, would have to exhaustively detect any deviations from the core data model and write custom code for additional cases on a case-by-case basis. With namespaces, the developers of OSCAL-enabled software can opt-in or opt-out with an allow-list or deny-list approach, vastly reducing the complexity of application logic and increasing processing speed as a result.

A namespace can be any URI or URN developers and practitioners agree upon. The NIST OSCAL developers, by specification and practice, do not mandate a governance model, registry, or related approval workflow for any custom namespace, so multiple approaches exist. Pilot members experimented with them in several iterations. We determined three generalized categories with real-world examples.

- Namespace by frameworks:
  - ○ SP800-53
  - ○ SP800-171
  - ○ ISO 27001
- Namespace by assessment organizations (FedRAMP; DISA; federal agency):
  - ○ FedRAMP
  - ○ DISA
  - ○ NIST (or any federal agency)
- Specialty guidance from advisory bodies:
  - ○ RMF 800-53A automated application, infrastructure and cloud guidance specific to one agency
  - ○ RMF 800-53A automated application, infrastructure and cloud guidance specific from agency consortium
  - ○ RMF 800-53A automated application, infrastructure and cloud guidance from MITRE
  - ○ RMF 800-53A automated application guidance from community organizations (CIS, CSA, OWASP)

One of the key challenges for the long-term viability of OSCAL we identified in the pilot is the practical governance of namespaces and implied ownership by a body of developers and practitioners. Further real-world application, with diverse implementers and their uses cases will require subsequent analysis and yield recommended governance practices. These are beyond the scope of this pilot.

#### 2.2.4 Workflow Demonstration: InSpec Profile for OpenSDP Compliance

An InSpec profile can inherit the controls from another InSpec profile. This type of profile is called an “overlay profile”. An overlay profile allows InSpec users to customize any part of the inherited controls (e.g., the description field), or even skip some controls; illustrated in Figure 19.

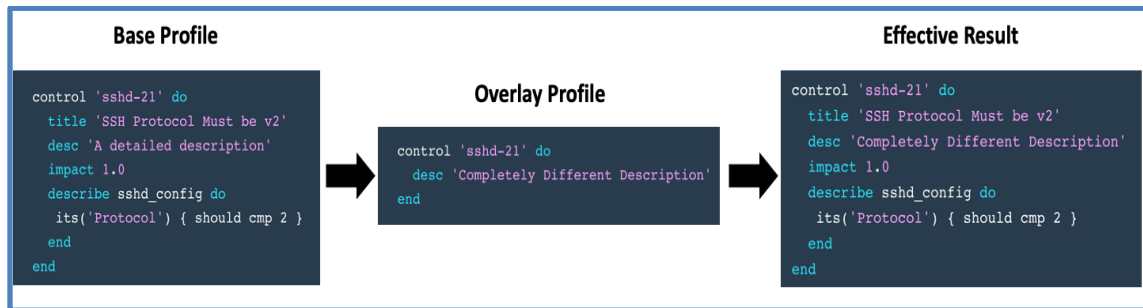


Figure 19. Sample Creation of InSpec Overlay Profile used to Test the OpenSDP Platform

The screenshot of Heimdall in Figure 20 illustrates the difference between a control being tested with the Ubuntu STIG baseline on the left and the OpenSDP compliance profile on the right. Note the description of the test and test result are different from the Ubuntu baseline.

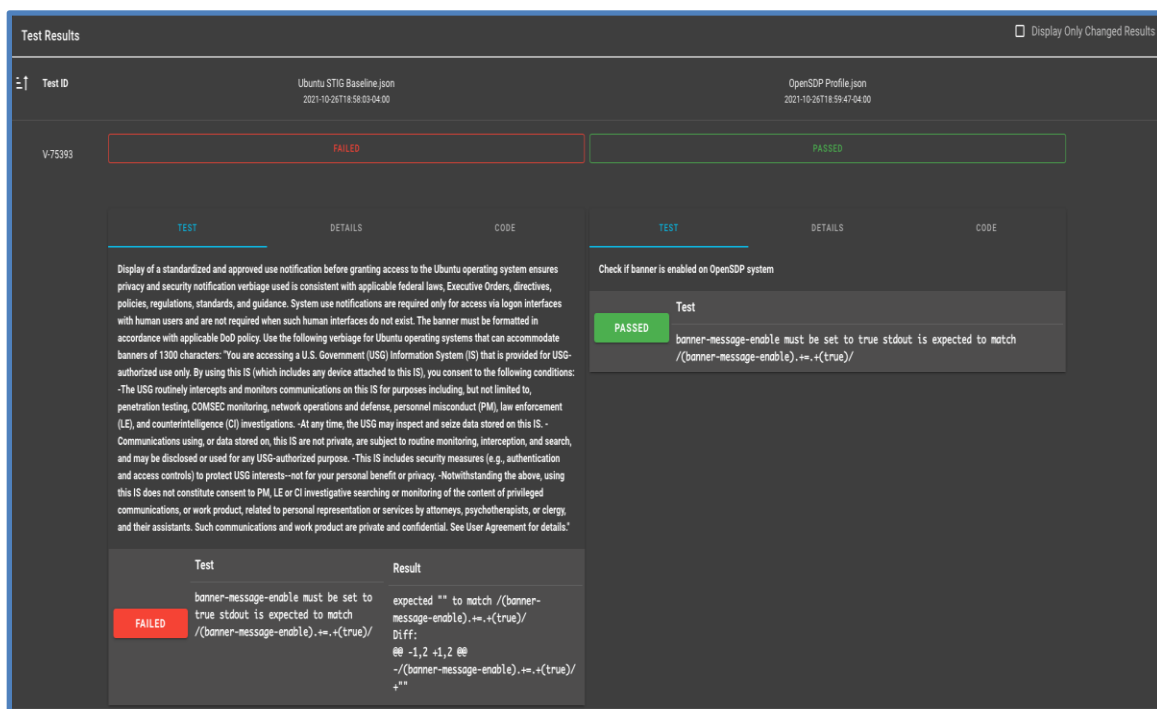


Figure 20. Heimdall Viewer Showing Baseline Ubuntu and OpenSDP InSpec Profile Test Results Side-by-Side

There are many InSpec profiles already developed and freely available for use. To save development time, InSpec allows its users to create overlay profiles that reuse existing profile controls without rewriting the same tests.

For this project, we created an overlay profile that used an [existing Ubuntu STIG profile](#) as a base. An overlay was created which customized those tests to match what was described in the OpenSDP environment documentation. The team collaborated with Waverley Lab to define which compliance tests are necessary for the OpenSDP environment.

Since the tests in the Ubuntu base profile were written for a STIG, they were all overridden as part of the overlay process to instead describe the FedRAMP Moderate baseline controls as implemented by the

OpenSDP. Many control tests were skipped as they did not apply to the controls in FedRAMP baseline. For these controls, separate controls assessment tests had to be designed and used. Four of control tests were custom developed for requirements that did not exist in the Ubuntu STIG base profile. Below is a table of all control tests used in our InSpec overlay profile for OpenSDP compliance.

**Table 3. OpenSDP Hardening Tests**

Control Description	Source
Checks usernames with sudo privileges	Custom developed
Checks file ownership permissions	Custom developed
Checks running process and network port associations	Custom developed
Checks firewall/iptables rules	Custom developed
Checks system for known system accounts and disallowed accounts	Ubuntu STIG base profile
Checks emergency accounts	Ubuntu STIG base profile
Checks temporary accounts	Ubuntu STIG base profile
Checks for duplicate user IDs on system	Ubuntu STIG base profile
Checks account inactivity system setting	Ubuntu STIG base profile
Checks auditing configuration for changes to account-related and password files	Ubuntu STIG base profile
Checks auditing of privilege escalation commands (e.g., sudo)	Ubuntu STIG base profile
Checks for usage of usermod command	Ubuntu STIG base profile
Checks for automatic logout system setting	Ubuntu STIG base profile
Checks setting for account lockout after wrong password too many attempts	Ubuntu STIG base profile
Checks for delay of time between log-in prompts following a failed logon attempt	Ubuntu STIG base profile
Checks for system and SSH banner display	Ubuntu STIG base profile
Checks for number of max concurrent sessions	Ubuntu STIG base profile
Checks for max inactivity time during SSH sessions	Ubuntu STIG base profile
Checks for users that have no password set	Ubuntu STIG base profile
Checks if firewall package “ufw” is installed and enabled	Ubuntu STIG base profile
Checks SSH key strength	Ubuntu STIG base profile
Checks for allowed network interfaces	Ubuntu STIG base profile
Checks for automount setting for USB devices	Ubuntu STIG base profile
Checks system password requirements	Ubuntu STIG base profile

This overlay profile was used in conjunction with the InSpec OSCAL plugin described above to create a fully tailored testing profile while simultaneously reusing previous work to shorten our development time.

### 2.2.5 Workflow Demonstration: TIC 3.0 Testing of OpenSDP

Since the OpenSDP implementation has many logical components, there are several ways to test for each capability, depending on which component is being tested. For example, when testing if a system blocks outbound connections, one approach could be to make an outbound connection from an internal machine and check for an appropriate response, and another approach would be to check the firewall rules directly. These tests were performed on the following locations of the OpenSDP implementation:

- An external machine acting as an authenticated or unauthenticated client.
- The SDP Controller, which contains a database which tracks active users, and makes all access control decisions.
- The SDP Gateway, which is what ultimately protects the services behind the OpenSDP implementation.
- The internal “blue” and “green” machines, which host the “F-Force App” test applications and are separated by the gateway into two environments.

The TIC 3.0 Network PEP Capabilities and their accompanying tests are listed and defined below:

1. **Access Control:** Access control protections prevent the ingest, egress, or transiting of unauthorized network traffic.

The profile tests for access control ingress protections as the external machine (location 1) by verifying that an unauthenticated client cannot access the production environment, but an authenticated machine can.

The profile tests for access control egress and transiting protections as the internal machine (location 4). For access control egress it verifies that the internal machine cannot send an http request to some external address (such as Google). For access control transiting, it verifies that the internal production machine in the “blue” environment cannot access the development machine in the “green” environment.

2. **IP Denylisting:** IP denylisting protections prevent the ingest or transiting of traffic received from or destined to a denylisted IP address.

The profile tests for egress and transiting by acting a client (location 1) and checking whether a list of inputted denylists were accessible externally and internally.

3. **Host Containment:** Host containment protections enable a network to revoke or quarantine a host’s access to the network.

The profile tested host containment as an authenticated client (location 1). First, the user was authenticated, then the user was temporarily marked as invalid in the SDP Client’s database, finally tests are performed verifying that the invalid user had no access to the production environment.

4. **Network Segmentation:** Network segmentation separates a given network into subnetworks, facilitating security controls between the subnetworks, and decreasing the attack surface of the network.

The profile tested for network segmentation ingress, egress, and transiting protections as the SDP gateway (location 3) by checking if the Iptables rules on the gateway were properly configured to drop incoming and outgoing connections by default.

5. **Microsegmentation:** Micro segmentation divides the network, either physically or virtually, according to the communication needs of application and data workflows, facilitating security controls to protect the data.

The profile tested for micro segmentation protections as the SDP controller (location 2). The profile checked that each service defined by the SDP controller's database had a unique address. Additionally, the profile checks that each client has been defined services.

InSpec was used to test active (state changing) tests. InSpec is generally designed to check static configurations of systems, whereas here it is being used to check if a system can react properly while actions are being done to that system. This was accomplished by assigning each InSpec control a "state" tag (Authenticated, Unauthenticated, Invalid).

The tests were managed by a script that handled the state transitions (authenticating the user, invalidating the user) and ran the InSpec profiles filtered by the appropriate state tags. This script also managed the testing infrastructure. This script ran all tests through a running InSpec Docker container. Tests targeting an authenticated or unauthenticated user were run through a Docker container with the SDP Client installed.

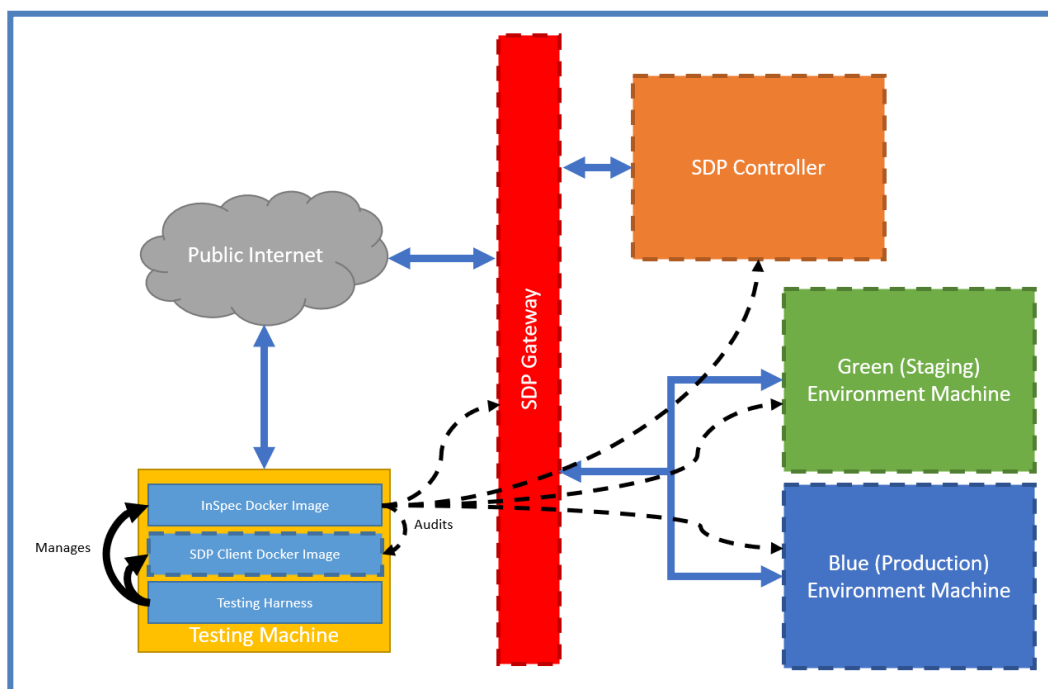


Figure 21: TIC 3.0 Testing Architecture

### 2.2.5.1 Testing Results

Upon execution of the script, a JSON file for each profile which ran will be generated. The results are as listed below:

- Access Control:
  - Ingress: **Passed**
    - Unauthenticated client was denied access.
    - Authenticated client was granted access.
  - Egress: **Failed**
    - Dev machine *was able* to access external site.

- Note: This SDP implementation is setup on a test environment.
- Transiting: **Passed**
  - Dev machine was able to access defined services on prod.
  - Note: The InSpec profile only checks that the dev machine was able to access their defined services, not whether it was denied other services.
- Host Containment:
  - Ingress: **Failed**
    - An invalidated (previously authenticated) host *was still* granted access.
    - Note: Since there was no systematic way to invalidate a host, the profile did so manually by modifying the controller's database. The SDP implementation was not designed to check for this change while a client was already connected.
    - Note: Occasionally the profile passed this test, but only because the session had already expired.
- Network Segmentation/IP Deny-listing:
  - Ingress: **Passed**
    - Before Authentication, the Gateway's IP Tables had a Drop All Input Rule, and did not have an Accept All Input Rule
    - After Authentication, the Gateway's IP Tables did have a new Accept Rule specifically for the authenticated client
  - Egress: **Failed**
    - Gateway's IP Tables *did not* have a Drop All Output Rule.
    - Note: This SDP implementation is setup on a test environment.
- Micro segmentation:
  - Unique Connection: **Failed**
    - Each service on the controller *was not* assigned a unique port.
  - Restricted Scoped: **Passed**
    - Each client was assigned services

The set of JSON files can be viewed using the InSpec results visualizer, Heimdall. See screen shots displaying those result in Figure 22 and Figure 23.





Figure 22: TIC 3.0 InSpec Profile Results Summary

Results View Data							
<div>Show Only Unviewed Sync Tabs Single Expand Expand All</div>							
Status	Result Set	ID	Impact	Title	800-53 Controls & CCIs	Run Time	0/10 Controls Viewed
Passed	Client-Authenticated.json	AC-Ingress-Authenticated	HIGH	Access Control Ingress - Authenticated Client	UM-1	0.923s	<input type="checkbox"/>
Failed	Client-Invalidated.json	HC-Ingress-UC	HIGH	Hostname Containment Ingress	UM-1	5.792s	<input type="checkbox"/>
Passed	Client-Unauthenticated.json	AC-Ingress-Unauthenticated	HIGH	Access Control Ingress - Unauthenticated Client	UM-1	4.854s	<input type="checkbox"/>
Passed	GW-Authenticated.json	NS-Ingress-Authenticated	HIGH	Network Segmentation Ingress - Authenticated Client	UM-1	0.066s	<input type="checkbox"/>
Failed	GW-NoState.json	NS/IP-Egress	HIGH	Network Segmentation/IP Denysiting Egress	UM-1	0.255s	<input type="checkbox"/>
Passed	GW-Unauthenticated.json	NS/IP-Ingress-Unauthenticated	HIGH	Network Segmentation/IP Denysiting Ingress - Unauthenticated Client	UM-1	0.181s	<input type="checkbox"/>
Failed	Internal.json	AC-Egress	HIGH	Access Control Egress	UM-1	0.121s	<input type="checkbox"/>
Passed	Internal.json	AC-Transiting	HIGH	Access Control Transiting	UM-1	4.235s	<input type="checkbox"/>
Failed	SDP-NoState.json	MS-Unique_Connection-SDP	HIGH	Microsegmentation: Unique Connection - SDP Controller	UM-1	0.015s	<input type="checkbox"/>
Passed	SDP-NoState.json	MS-Restricted_Scope-SDP	HIGH	Microsegmentation: Unique Connection - SDP Controller	UM-1	0.000s	<input type="checkbox"/>

Figure 23: TIC 3.0 InSpec Profile Results Details

### 2.2.5.2 Results Analysis:

These test results show many positive results for this SDP implementation. Namely, the profile was able to detect that the IP Tables rules on the gateway were temporarily updated whenever making a request from an Authenticated client. This is at the core of what SDP does, and it seems that it is working properly.

Amongst the failures, some of them can be easily attributed to the fact that this OpenSDP implementation is set up on a test environment. The general IP Table rules may be setup based on other concerns for testing. That being said, our InSpec profile is properly exposing the flaws in this SDP implementation. Less failures are expected on a production version of this implementation.

An important failure listed above is that each service is not assigned a unique port. This SDP implementation opens the ports for each client based off which services that client has access to. If two services have the same port, then a client will be able access all services that are assigned to that port.

### 2.2.5.3 Future Enhancements

If the InSpec profile were to continue development, first, versatility would be improved. Due to time constraints, the profile was written in a way where it only works for this specific SDP implementation. For this profile to test other SDP implementations, some more thought would have to be considered on how generalizing the tests

## 2.2.6 Workflow Demonstration: Atlasity Enhancements

This project built upon previous work that was demonstrated at the NIST workshop in February 2021 via the Atlasity platform's support for OSCAL. At that workshop, we demonstrated the ability to import controls from NIST 800-53 Rev 4 and 5 based on the NIST OSCAL documents, building multiple profiles (i.e. High, Moderate, and Low), and import of a SSP into Atlasity from GovReady using an earlier version of the OSCAL standard (pre-1.0 release). In addition, we were able to demonstrate automation by taking scanning results from MITRE Heimdall, parsing them from the JSON output, and documenting the results as assessment of a set of controls within the GovReady provided SSP.

While this initial work served as a basic proof of concept, it was insufficient to support the goals of the ATARC project. As the OSCAL standard was refined into the final version 1.0 release, there were significant schema gaps between Atlasity and the standard. In addition, there was a heavy focus on real-world usability in how consumers would ultimately want to build a SSP in Atlasity and export in OSCAL as part of their workflow.

To improve quality of life for the consumer, we added control inheritance functionality to allow SSPs to be nested where they can inherit the controls of their parent security plan(s) at any arbitrary "n" number of layers. We also implemented a responsibility model to indicate whether controls were customer, provider, or a shared responsibility to better model modern cloud systems. In addition, we added a classification system to allow each individual record in the system to receive a classification stamp so they are marked appropriately in the OSCAL export. Finally, we refactored the MITRE SAF integration to remove custom code for data parsing and instead leveraged the inspec.js NPM package from MITRE to provide a more robust and consistent parsing of the data.

At the data layer, significant enhancements were also required. We updated to our schema to capture ports and protocol data for the SSP. In addition, we added namespace support to our “Admin panel” to allow for separating and strongly typing parameters within the OSCAL output. We also enriched the OSCAL data model from the initial demo to bring in optional OSCAL fields consisting of additional metadata that is tracked by Atlasity but not required by OSCAL; generating a more complete model of the SSP. For lineage, we also added functionality to track the parent catalog and profile that generated a SSP as previous versions of Atlasity did not store this information. Finally, we implemented the SAP/SAR layer of OSCAL to allow assessment results to be published whether leveraging automated assessments (i.e. MITRE Heimdall) or manual assessments.

### 2.2.7 Component Enhancements for SSPs

The previous version of Atlasity had no model for components in its schema. The platform was updated to allow Components to be created as child records to SSPs and then instantiated from profiles to allow each component to address a discrete set of controls. Similar to SSPs, we also added the ability to store the lineage of the profile used and to track ports and protocols at the component level. Lastly, we added the ability to add strongly typed parameters to namespaces allowing the OSCAL export of the component to be passed to MITRE Heimdall, parsed consistently, and then used to initiate a scan for the SAP/SAR.

Figure 24. Atlasity Enhancements for OSCAL/Heimdall Integration

### 2.2.8 OSCAL Validation with AJV

Another challenge that had to be overcome in Atlasity was to ensure that the OSCAL exported was fully compliant with the standard. Data validation of such a complex model would be complicated and brittle if performed just within C2 Labs’ platform. However, NIST published a set of validation tools with AJV that provided the ability to automate validation using a trusted source from NIST. AJV was fully

integrated into all Atlasky exports to identify issues in data validation before exporting. The screenshot below shows an example set of data validation issues in pink:

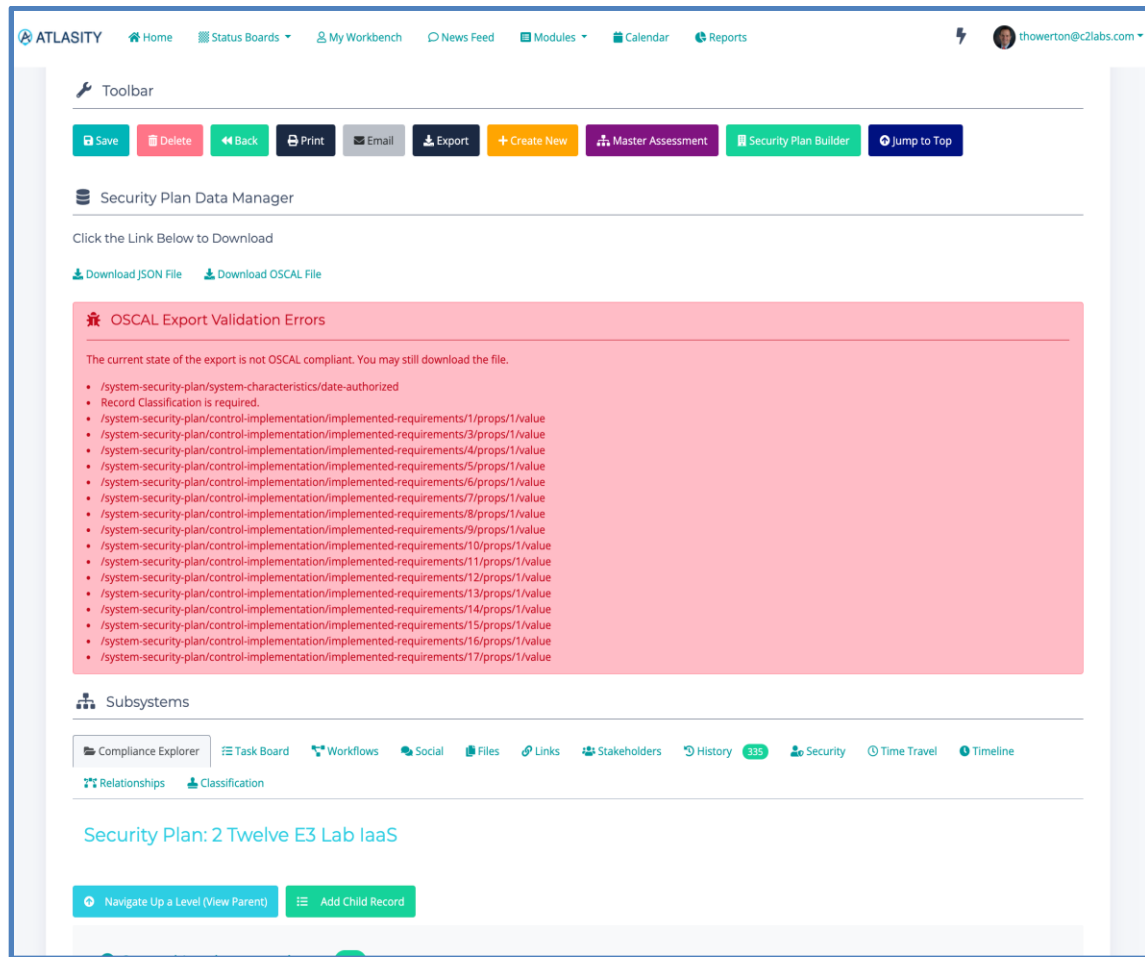


Figure 25. OSCAL Conversion Error Report

### 2.2.9 Passing OSCAL Parameters to InSpec Profile

After completing validation, C2 Labs needed to pass the Component OSCAL to MITRE Heimdall to perform an automated scan. This OSCAL had strongly-typed parameters using namespaces that allowed us to consistently send parameters to the MITRE team that could be programmatically parsed. Once parsed, MITRE could run the scan against the infrastructure using parameters provided while tying the results back to the discrete controls in the component with OSCAL as the intermediary for data exchange. The component export is shown in the screenshot below:

```

1 {
2   "component-definition": {
3     "uuid": "52796e48-805e-4379-8318-4e7242086bf3",
4     "metadata": {
5       "title": "Ubuntu 16.04 LTS",
6       "published": "2021-06-14T10:17:17.640537",
7       "last-modified": "2021-06-28T14:57:03.169405",
8       "revisions": [
9         {
10          "title": "Version 2",
11          "version": "2",
12          "oscal-version": "1.0.0",
13          "last-modified": "2021-06-14T10:17:17-04:00",
14          "remarks": "MDS Hash: 6AE950130A0BFD247BAA93AF84547B36"
15        },
16        {
17          "title": "Version 1",
18          "version": "1",
19          "oscal-version": "1.0.0",
20          "last-modified": "2021-06-28T14:57:03-04:00",
21          "remarks": "MDS Hash: 1FD16D32F7216D0FEEC3653B2F0D8C63"
22        }
23      ],
24      "version": "Version 3",
25      "oscal-version": "1.0.0"
26    },
27    "components": [
28      {
29        "uuid": "52796e48-805e-4379-8318-4e7242086bf3",
30        "type": "software",
31        "title": "Ubuntu 16.04 LTS",
32        "description": "Ubuntu 16.04 LTS",
33        "control-implementations": [
34          {
35            "uuid": "085f825e-e6de-4635-be4b-23e95bd807af",
36            "source": "https://atlas-dev.c2labs.com/catalogues/form/109",
37            "description": "<div>Ubuntu operating system management includes the ability to control the number of users and user sessions that",
38            "implemented-requirements": [
39              {
40                "control-id": "ac-10",
41                "uuid": "085f825e-e6de-4635-be4b-23e95bd807af",
42                "description": "",
43                "statements": [
44                  {
45                    "statement-id": "status_smt",
46                    "uuid": "cac8d68f-f68e-4197-a936-3bf820eff46c",
47                    "description": "Not Applicable"
48                  },
49                  {
50                    "statement-id": "implementation_smt",
51                    "uuid": "4088111d-2114-4d82-b07e-8718950358b3",

```

Figure 26. OSCAL File Parameterized to Pass Testing Values for the Waverley Labs OpenSDP InSpec Profile

### 2.2.10 Atlasity – Loading the InSpec Results File

Once the MITRE scan was completed, the results were exported into a MITRE HDF file format that is based on JSON. In the initial NIST demo in February, C2 Labs was able to parse this raw JSON file using custom Python scripts to load the assessment results into Atlasity via the platform's APIs. While this was sufficient for a feasibility demonstration, the architecture was brittle and there was a fair amount of "fuzzy" matching controls that lacked precision.

In the ATARC project, the legacy Python scripts were retired in favor of using MITRE published and supported ["InSpecJS" NPM package](#). This approach allowed the results to be consistently parsed using the MITRE libraries. In addition, the InSpecJS library is more robust allowing multiple profiles to be parsed in a single scan result file. Also, because it was published as a NPM package, C2 Labs was able to build this integration into the Atlasity platform natively versus parsing in a "sidecar" script. The native integration lowers the barrier to entry for consumers to leverage this integration as it can now easily be configured in the GUI without the need for developer/scripting support. A screenshot of the MITRE SAF integration is shown below:

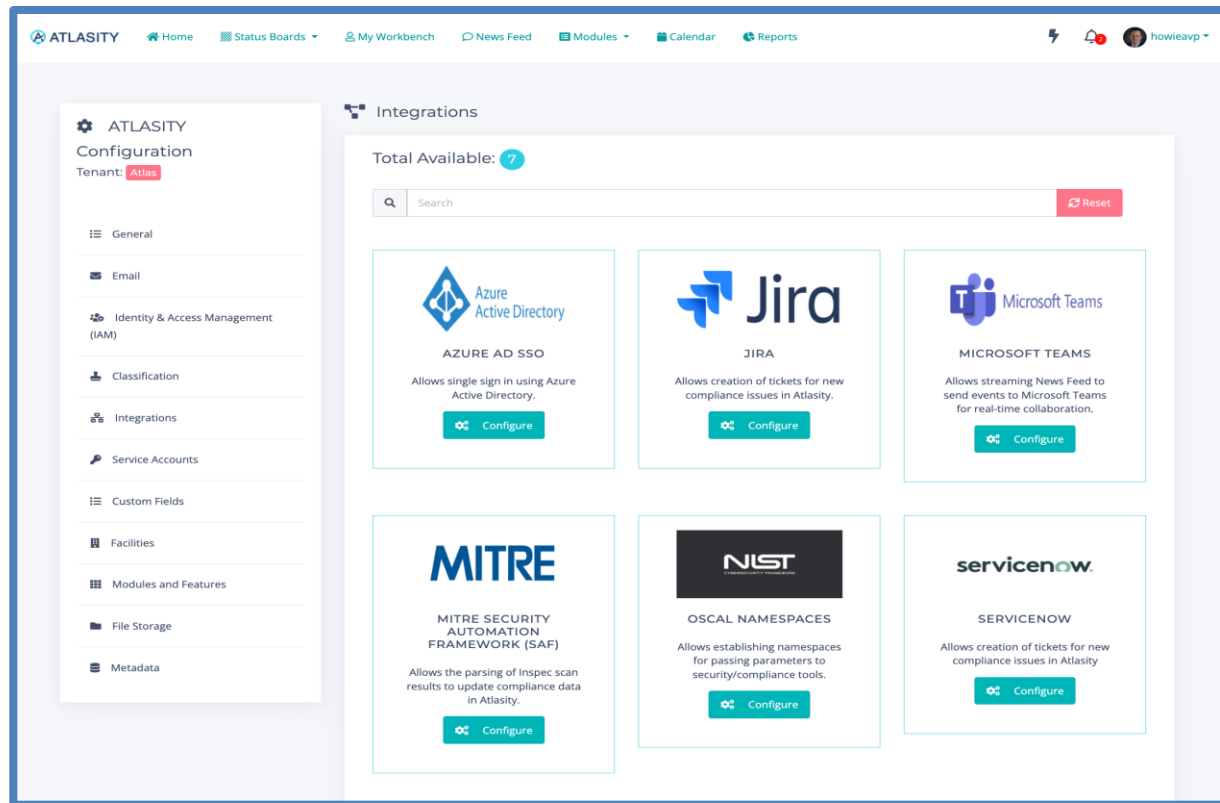


Figure 27. Atlasity Tool Integration Pane; MITER SAF Ingest

### 2.2.11 Parsing the InSpec Results File

The user experience of parsing the MITRE Heimdall results is now more straight forward and intuitive. Atlasity provides an upload tool to allow the consumer to take their scan result file and upload it into Atlasity for parsing. Atlasity then uses the inspecjs library to parse the results, provide a preview panel of the results, and then a confirmation before the final upload of the results into Atlasity. Once uploaded, the results can be exported as a SAP/SAR using OSCAL. See the screenshot below for the Atlasity upload tool for MITRE Heimdall scan results:

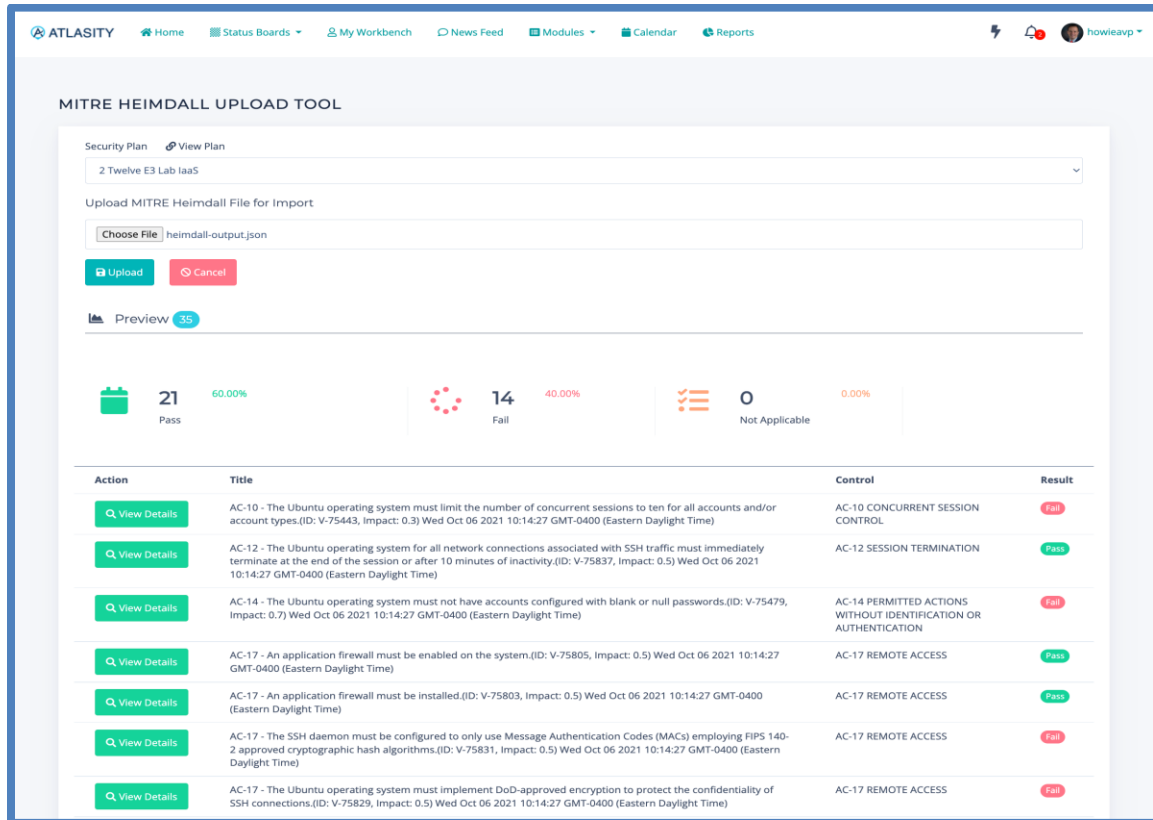


Figure 28. Atlassian Heimdall Ingest Results

### 2.2.12 Exporting the SAP/SAR

Within the Atlassian platform, the system supports bulk scheduling control assessments as part of a group; known as a “Master Assessment” in Atlassian. The Master Assessment consists of one or more controls for an SSP/Component that are logically grouped together as an audit/assessment. C2 Labs was able to extend this functionality to work with the MITRE integration.

When parsing the scan results, a new Master Assessment is created that has a collection of child assessments against each in scope control. The results of the individual control tests are then aggregated into the Master Assessment module within Atlassian. This functionality proved convenient for modeling the SAP/SAR layers of OSCAL. By combining the profiles from MITRE for the scan with the parameters in Atlassian for the controls, a SAP could be derived. By parsing the results of the scan, the SAR was then derived and tied back to the SAP layer. The final result is the ability to programmatically generate the SAP/SAR using the integration between Atlassian and MITRE. A screenshot showing the ability to export the SAP/SAR in Atlassian is shown below:

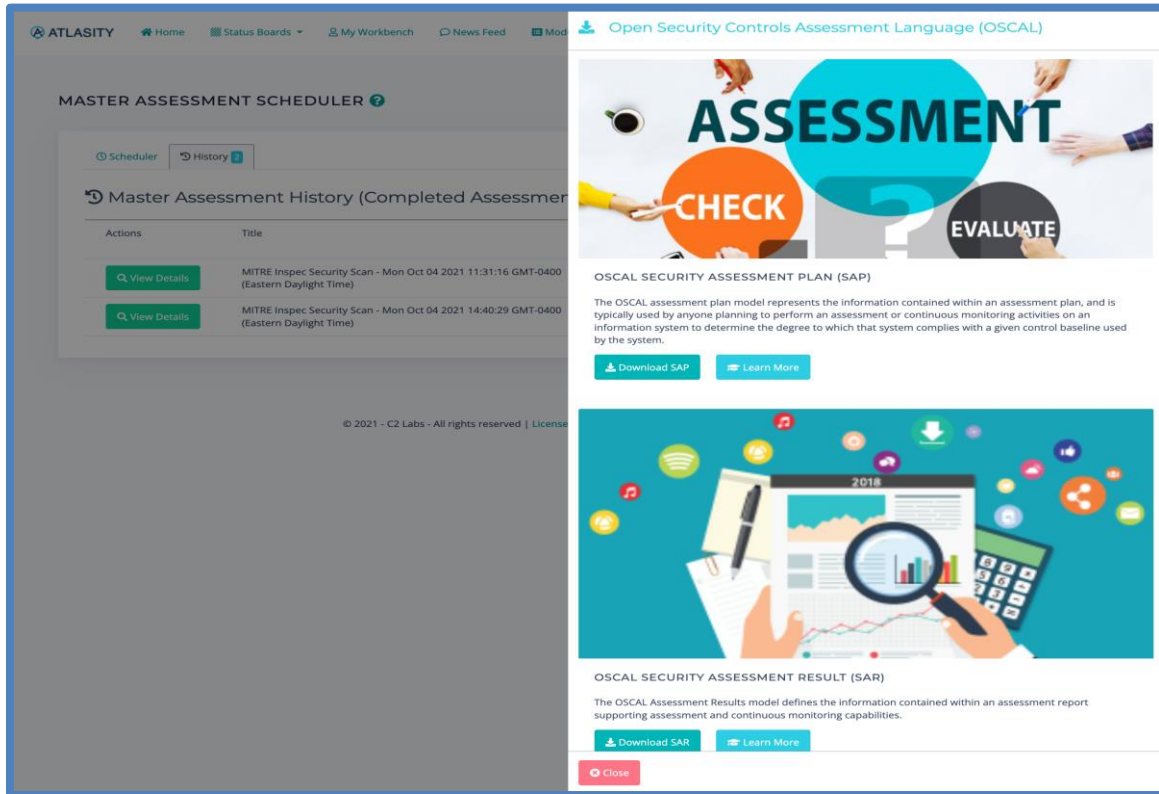


Figure 29. Atlacity SAP, SAR Export Capability

### 2.2.13 Sprint #2 Results Summary

C2 Labs was able to build on previous support for OSCAL to enhance the platform to fully align to the 1.0 version of OSCAL at the catalog, profile, SSP, component, SAP, and SAR layers. These enhancements were then integrated with GovReady and MITRE Heimdall to provide an end-to-end demonstration of using OSCAL to ingest an SSP using OSCAL, develop a component, define its parameters, export as OSCAL, perform an automated scan, publish the results, and then parse and export as a OSCAL SAP/SAR. This project exercised every layer of the OSCAL stack using multiple commercial tools to demonstrate viability. While this implementation was tool specific, most of the architecture and approach would be reusable with any other commercial tools that support the OSCAL standard.

## 3 Recommendations and Conclusions

As the market for Compliance-as-Code and Policy-as-Code continues to grow, the supply and demand for them present a series of ongoing challenges inside. US federal government projects are a focal point for many of those challenges. Such projects are both highly regulated and resource intensive. Despite that, they are often intended for public use. So, there is a great deal of scrutiny and interest regarding how compliance and security standards' oversight is done, and how it can be optimized. This working group and the pilot simulated complex real world project requirements, with the necessary security architecture for a properly risk-managed information system.



The major challenge is that much of the US federal information security and privacy requirements, specifically those designed by the NIST Risk Management Framework, target high-level, strategic objectives that are not easily translated into actionable technical requirements for system implementers. Even the subset of those strategic objectives that are seemingly straightforward or have tangible technical details are very broad in scope. Automating those requirements, from the least technical to the most technical, is a more complex endeavor beyond what most software solutions are capable of expressing. This pilot designed and implemented several novel solutions in support of demonstrating one way of automating the A&A process while providing a toolchain for verifying the practical implementation of a secure, Zero Trust architecture. The pilot efforts will inform future improvements to how system implementers will use improvements to OSCAL, which expresses these strategic objectives, and map to operational metadata from software that implement security architecture and verify its ongoing correctness.

Beyond addressing this challenge, the pilot did experiment and put forth solutions that allowed an ecosystem of risk management and security testing tools to exchange data with standards in a way that was only hypothetical in the years prior. The pilot and its artifacts will pave the way for expansive automated exchange of A&A data across heterogeneous GRC platforms from greenfield experiment to commonplace industry practice.

With the arrival of NIST OSCAL 1.0.0 with modern GRC tools and assessment platforms, the time has finally arrived to achieve what was once thought impossible – an agile / continuous ATO (cATO). As with any new language or technology, there will need to be a cultural transformation to move from point in time compliance to continuous compliance, requiring education of stakeholders throughout the lifecycle of the ATO in how these new languages, technologies and approaches can enable and truly deliver a continuous ATO (agile ATO/cATO). Educational curricula around this new approach coupled with best practices around how to implement a cATO are recommended next steps.

Automation is needed now more than ever and as such, “X”-as-Code has become the *de facto* norm in how we tie and integrate systems together in today’s API-sharing Internet connected economy. ATARC built an industry-government coalition in this working group to study and refine this problem space, coupling best-of-breed technologies with government scientists to collectively advance the state of the art of cloud- and cybersecurity. These types of public-private partnerships are critical to ensure America’s continued competitive edge to stay one step ahead of our adversaries.

By coupling modern Zero Trust solutions from Waverly Labs (OpenSDP), IaaS platforms like 2Twelve Solutions, the Rosetta Stone for compliance in NIST’s Open Security Controls Assessment Language, MITRE’s cutting-edge Security Assessment Framework and modern GRC platforms like GovReady, ATO-as-a-Service and Atlasity, the government can effectively now shift both security and compliance left to deliver a Zero Trust continuous Authority to Operate.

## 4 Appendix

### 4.1 Acronyms

**AJV** – Asynchronous Validation (JSON Schema Validator)

**ATARC** – Advanced Technology Academic Research Center

**AO** – Authorization or Authorizing Official

**ATO** – Authority or Authorization to Operate

**CIS** - Center for Information / Internet Security

**CRM** – Customer Responsibility Matrix

**CSA** – Cloud Security Alliance

**DISA** - Defense Information Systems Agency

**FedRAMP** - Federal Risk and Authorization Management Program

**FFRDC** – Federally-funded R&D Center

**FICAM** - Federal Identity, Credential, & Access Management

**FISMA** - Federal Information Security Management Act of 2002

**HDF** – Heimdall Data Format

**IDaaS** – Identity-as-a-Service

**ISO** – International Standards Organization

**JSON** - JavaScript Object Notation

**OSCAL** –Open Security Controls Assessment Language

**OWASP** - Open Web Application Security Project

**PEP** – Policy Enforcement Point

**RMF** – Risk Management Framework

**RVA** - Risk and Vulnerability Assessment

**SAF** – Security Automation Framework

**SAR** - Security Architecture Review

**SAP** – Security Assessment Plan

**SDP**- Software Defined Perimeter

**SSP** – System Security Plan

**SSRM** – Shared Security Responsibility Matrix

**STIG** - Security Technical Implementation Guide

**TIC** – Trusted Internet Connection

**URI** – Uniform Resource Identifier

**URL** – Uniform Resource Locator

**URN** – Uniform Resource Name

**UUID** - Universally Unique Identifier

**ZTA** – Zero Trust Architecture