Smart Compliance with Component Oriented Security Thinking and OSCAL

A stackArmor, Inc. White Paper

Author: Johann Dettweiler, CISO

stackArmor

Contents

stackArmor

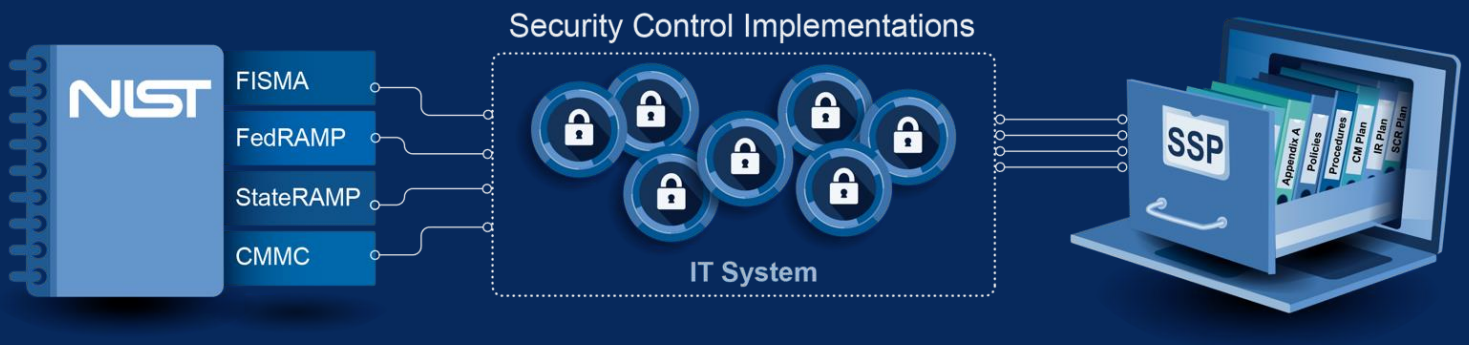
Smart Compliance with Component Oriented Thinking and OSCAL

Decomposing complex compliance documents into objects & components, encapsulation of controls and leveraging inheritance can help reduce duplicative documentation, assemble accurate artifacts, and streamline the creation of digital artifacts by adopting the Component Definition Model Approach to Open Security Controls Assessment Language (OSCAL).

Introduction

National Institute of Standards and Technology (NIST)-based cybersecurity governance frameworks are increasingly being used to protect critical digital assets and comply with government regulations. To meet these requirements, NIST established a series of Special Publications that includes (SP) 800-53 and 800-171 which together outline a collection of security controls that satisfy Federal Risk and Authorization Management Program (FedRAMP), Federal Information Security Modernization Act (FISMA)/Risk Management Framework (RMF), State Risk and Authorization Management Program (StateRAMP) and Cybersecurity Maturity Model Certification (CMMC) v.2.0 requirements.

All required NIST security controls for a given governance framework must be translated from the aforementioned NIST publications and implemented by cybersecurity architects, engineers, and operators in a way that meets the requirements of each control. A critical artifact that demonstrates how these requirements are satisfied is the System Security Plan (SSP). Required by nearly all compliance frameworks, the SSP is a critical document to understand a system's purpose, authorization boundary, and control implementation details. This centralized source of information about a system's architecture and security posture is the lynchpin of a quality security program. As such, the importance of this document's quality cannot be over stated.



Traditionally, SSPs have been crafted using narrative-driven implementation statements, where security controls are described in prose. While this method provides flexibility, it can introduce material issues into a security program.

Specific issues are:

- **Inconsistencies** – The complexity of an SSP is such that multiple resources are typically tasked with its creation. These resources often struggle to maintain a consistent view of the system which can lead to inaccuracies when documenting control implementations.
- **Redundancies** – Many control implementations are similar in nature and leverage the same components to implement the various requirements. In a narrative approach to writing implementation statements, the same language must be reused throughout the SSP to describe use of these components to address the requirements.
- **Maintenance Challenges** – Systems are in a constant state of change. These changes often require updates to documentation as they impact how a certain security control has been implemented. When working with a document that can be upwards of 700-800 pages, it is exceedingly difficult to identity all areas that require updates.

These issues tend to contribute to an overall lack of standardization and lead to ambiguities that make it difficult to ensure comprehensive security coverage, as well as complicating the auditing process.

Recognizing these challenges, NIST introduced OSCAL, a standardized, machine-readable framework designed to enhance the efficiency and accuracy of compliance documentation. By providing a structured format, OSCAL facilitates automation, improves consistency, and enables better integration with various security tools and processes. This shift towards structured documentation marks a significant advancement in the field of compliance management and provides a framework to address the material issues noted above.

Component Oriented Compliance with OSCAL CDEF

Within OSCAL, the Component Definition Model offers a novel approach to SSP development. This model allows for the modular representation of system components and their associated control implementations. The approach can be adapted to align compliance with the system development using the four pillars of Object-Oriented Analysis & Design (OOAD):

- **Abstraction** - Find areas of your system that are similar in form and function and can serve to implement multiple control requirements.
- **Encapsulation** - Segregate your system into modules (components) that have the appropriate level of privacy and responsibility regarding implementing control requirements.
- **Inheritance** - Consider the reusability of components to implement control requirements and the overall “functionality” of the component.

- **Polymorphism** - Determine the ability for a single component to implement various control requirements throughout the system security and privacy architecture.

By adapting the principles of the OOAD approach, we can start to define system components for generating SSPs. Information System Owners (ISOs) can expect various benefits including:

- **Modularity** – complex implementation of controls is now broken down into manageable component modules.
- **Reusability** – components can be utilized across an entire system or across multiple systems within an organization to implement controls.
- **Maintenance** – when change occurs, impacted components are easily identified and updated to account for the nature of the change.
- **Scalability** – the baseline of components is hierarchical in nature and can be "built on top of" when new features are added to the system, without impacting what already exists.

By defining discrete components that encapsulate specific control implementations, organizations can create reusable modules that streamline the development and maintenance of SSPs. This modularity facilitates clearer traceability and more efficient updates, as changes to a component automatically propagate to all instances where it is utilized, be it within a single system or across all systems within the organization.

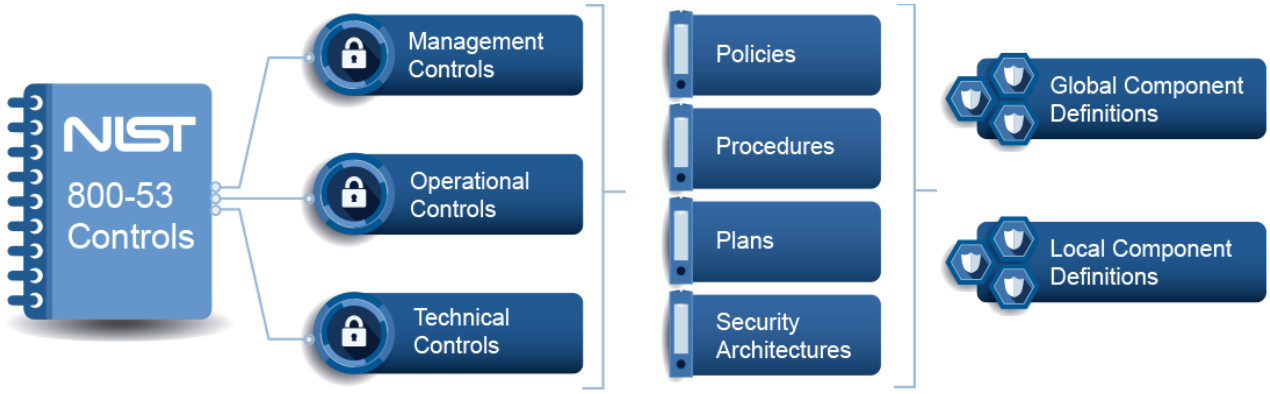


Figure 1. - Infographic showing the relationship between various artifacts and elements that make up Component Oriented Compliance. Copyright of stackArmor, Inc. All rights reserved.

However, this approach is not without challenges:

- **Granularity** - Determining the appropriate level of granularity for components is crucial; overly granular components may lead to complexity due to fragmentation, while overly broad components can reduce reusability and clarity.
- **Front-Loaded Resource Expenditure** - The initial effort required to define and organize components can be substantial, as it requires a re-education in how to approach the SSP writing process. Understanding what "is" and "is not" a component, and managing

the interdependencies introduced by these definitions necessitates careful planning and constitutes an entirely new way of thinking about compliance.

Despite these challenges, adopting a Component-Oriented compliance approach offers significant benefits, including enhanced consistency, improved traceability, and increased efficiency in compliance documentation through re-use. By leveraging the Component Definition Model within OSCAL, organizations can achieve a more structured and maintainable SSP development process, creating a library of components that can be leveraged across multiple systems. This will ultimately lead to a more accurate and representative compliance posture without the need for mindless updating of Word documents that are constantly out of date.

This white paper delves into the intricacies of the Component Definition approach, exploring the application of OOAD principles to compliance, the benefits it offers, the challenges it presents, and best practices for its effective implementation in the development of SSPs.

Foundational Concepts

OSCAL provides a structured format for representing control catalogs, profiles (i.e., control baselines) system security plans, and other related documents in a machine-readable manner. OSCAL provides a “compliance-as-code” approach to the creation of a system’s authorization package that provides an avenue for integration into already existing processes such as configuration and change management and infrastructure-as-code. This integration of OSCAL into existing processes presents an opportunity to better represent the “state” of the system through its authorization documentation, which is often misrepresented in the traditional approach to authorization package creation.

The format was developed by NIST to address these inefficiencies in traditional compliance documentation. Historically, compliance documents like SSPs, Plans of Action and Milestones (POA&Ms), and Control Baselines were static and specially created to be human-readable. This format limits their adaptability in dynamic environments where security configurations often evolve rapidly and are increasingly guided by machine automation – outside of the immediate purview of humans. NIST focused on key areas in the development of OSCAL to provide some functionality that cannot be recognized in the common approach to documentation, such as:

- **Machine-Readable:** By structuring compliance documents in XML, JSON, or YAML, OSCAL facilitates integration with existing automation and security tools. Utilizing their outputs, these tools can be leveraged to perform tasks such as gap analysis, validation, and audit preparation without manual intervention.
- **Interoperability:** OSCAL’s standardized format enables seamless data exchange across tools, improving collaboration between system owners, auditors, and certifying bodies. This removes the “translation layer” that often opened the door for misinterpretations and communication errors between various stakeholders in a system’s lifecycle.
- **Continuous Authorization:** By linking machine-readable documents with real-time monitoring tools, organizations can ensure compliance dynamically, rather than relying on static, point-in-time documentation. As changes are made to the configuration of the

system, those configurations are represented, in near real-time, directly in the system's authorization package.

Comparison of Traditional Documentation versus Component Definition Methods

Traditional "Monolithic" Approach

Traditional compliance documentation is akin to monolithic programming: a single, lengthy narrative that describes system controls. While this provides flexibility, rapidly changing and evolving systems often struggle to maintain these "monolithic" authorization packages. Teams dedicated to a particular aspect of the system (e.g., Security Operations, Contingency Planning, etc.) may not be aware of or be able to easily discern the impact of changes on documentation.

These teams need to interface with the Compliance Team to communicate the system changes. The Compliance Team is then left to interpret and incorporate those changes into the authorization package - identifying all aspects of the package that may need to be updated based on those changes. The needed updates may be spread across multiple documents and/or multiple control families and their interoperability may not be immediately obvious. Depending on the skill set and capabilities of the resources dedicated to compliance, the likelihood of inconsistencies across the authorization package is typically very high.

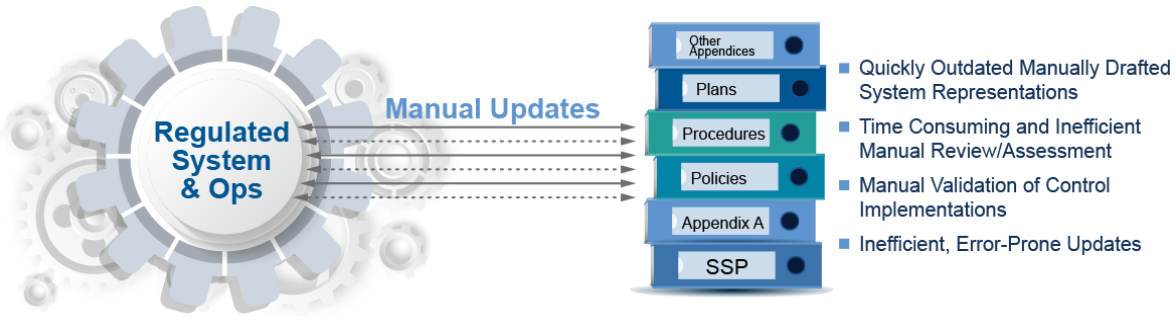


Figure 2. - Traditional Manual Approach to System Security Authorization Documentation

Component Definition "Microservice" Approach

In contrast to the traditional monolithic compliance approach, defining the system by the components that are utilized to implement the controls, the Component Definitions approach is more akin to a microservice architecture. When a change is made, instead of trying to review and understand the entire system to determine what aspects may have been impacted, a team can focus on the microservices (components) that are relevant to their area (e.g., the SecOps team focuses on the components utilized to implement system monitoring). By considering impacts to individual components, teams can quickly identify the impacts of change, without needing to understand the entire monolith.

The components become the centralized repository for updates to the system. The teams that are responsible for those components can update them directly and those changes are

immediately propagated across all controls that component touches. By directly involving the teams responsible for components, the changes are immediate, accurate, and relevant.

- **Example** – A team updates the version of the VPN software utilized to access the management plane of the environment. This version update includes an update to the cryptographic module utilized by the VPN to encrypt traffic. Instead of searching across the entire authorization package to identify all areas where this module is mentioned, the Team updates the Component Definition that defines the old version of the module to the new version. This change is now propagated throughout the SSP.

OSCAL allows for the complete control over the granularity of the components, allowing them to represent individual system elements or more complex, interdependent systems. This control over granularity allows for tailored levels of detail to fit the needs of the resources assigned to manage those components. It also lends itself to overall scalability. As a system grows in complexity, various components can be combined or further decomposed based on the changing needs of the system. As a system's complexity and scope changes resource distribution will change as well with centralized teams becoming decentralized, or vice versa.

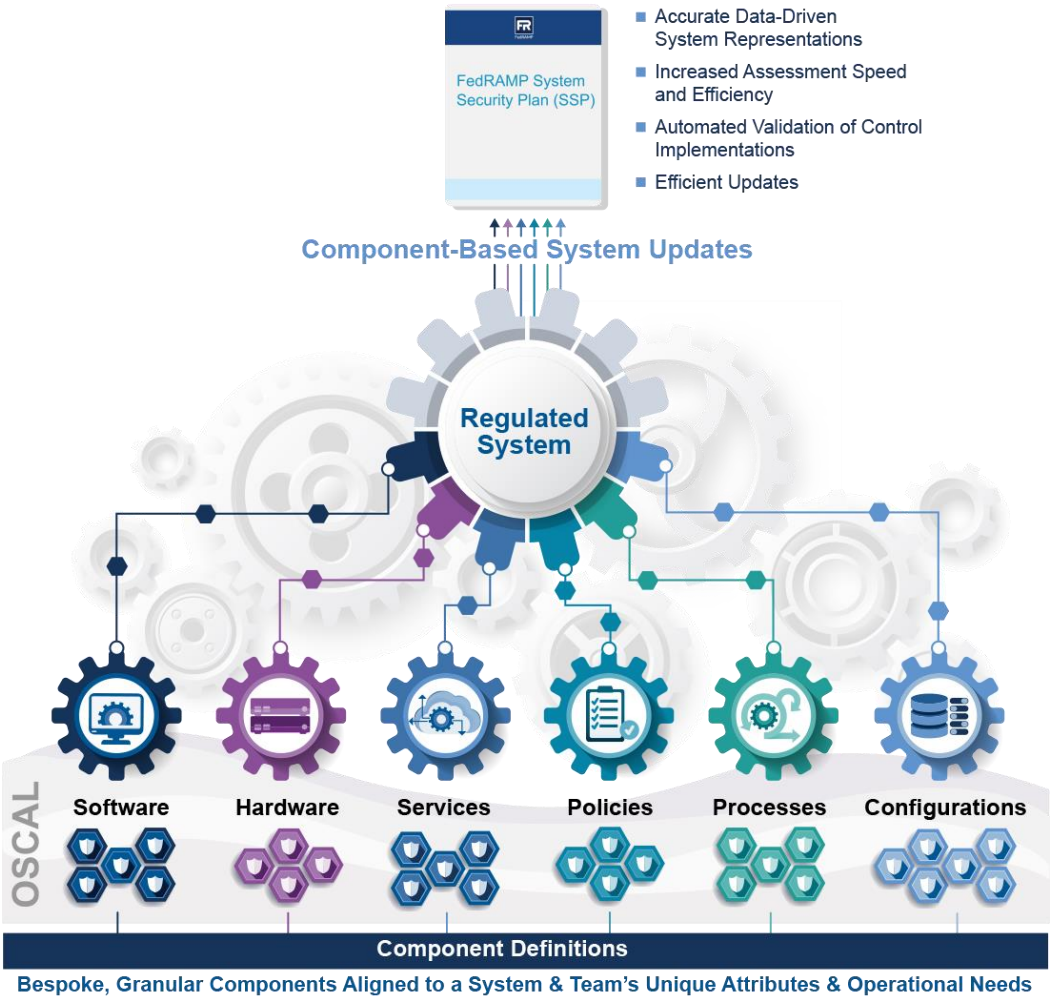


Figure 3. - Component-Based Approach to System Authorization Documentation Management

Components that were once managed by a single team may end up being managed by multiple teams and it may become beneficial to decompose a component into further subdivisions that better align to component ownership and responsibility.

Technical Framework for Component Definitions

The adoption of the Component Definition Model requires a clear and well-defined technical framework to ensure its success. This framework should address the lifecycle of components, from initial definition to deployment and maintenance. It should also provide guidelines for achieving consistency, scalability, and adaptability across various use cases.
Establishing clear guidelines for defining components is the cornerstone of an effective framework. The Component Definition approach allows for more meaningful input from all system stakeholders, but it is essential that those stakeholders follow a regimented approach to defining components. This regimented approach will ensure the interoperability of components throughout the system and will ensure the appropriate data points are captured. The following practices can help ensure components are both meaningful and reusable:

Granularity

Components should be defined at a level of granularity that captures their primary purpose without being overly fragmented. For example, a "Web Application Firewall (WAF)" component should encompass configurations and control mappings relevant to traffic filtering, without delving into broader network configurations. Control mappings can then be used as a guide to determine component scope. Components should cover specific controls or control families without overlapping unnecessarily.

Consistency in Metadata

Consistency in metadata is critical in the overall OSCAL ecosystem through the implementation of validation. Because OSCAL is machine-readable, it allows for fast and efficient review of a system's documentation package to ensure that the minimum datapoints have been collected and detailed. To ensure successful validation of an OSCAL package, an organization must:

- **Understand required metadata and establish a minimum baseline for Component Definitions** – depending on the OSCAL framework, there will be necessary metadata such as name, description, applicability, dependencies, and status, that must be captured for each component. This information must be included with each Component Definition and there should be processes and institutional knowledge in place to capture these critical datapoints.
- **Use standardization in taxonomy** – there should be a strategy in place to appropriately define key datapoints such as component type, description, purpose, and asset types. Regardless of the component, an approach needs to be developed to ensure that similar component types are identified, and the qualifying information is sufficiently detailed in accordance with the overall Component Definition strategy.

Documentation

A general format, whether that is JSON, XML or YAML should be selected at the start of the project. The appropriate OSCAL templates should be utilized by stakeholders to document the Component Definition ’s purpose, implementation details and control mappings in accordance with the required template.

Linking Components to Controls and Control Enhancements

A key aspect of the Component Definition Model is its ability to map components directly to the controls and enhancements they implement. This linkage ensures traceability and provides a structured view of the system's compliance posture. By building out a “Component Library”, components can be selected based on the controls they implement to ensure that all necessary controls in a security baseline are addressed by one or more components based on these established linkages.
A key feature of OSCAL’s Component Definition template is the ability to map components directly to the individual control requirements that they implement.

- **For example -** A "Firewall" component may link to SC-7 (Boundary Protection) and SC-7(4) (Remote Access Traffic Restriction). By mapping the single component to multiple control implementations, the reusability of that component has increased.

It is important to understand that there will be Cross-Component Dependencies in which a single component alone does not implement a control requirement but works in conjunction with other components to fully address the control requirement.

- **For example -** A "VPN" component may depend on an "Authentication" component that enforces multi-factor authentication (MFA).

These dependencies can become very complex in nature, and it is critical to avoid conflicts in overlapping information to maximize the modularity, reusability, and scalability of system components. As the Component Definition baseline is developed, stakeholders should manage dependencies carefully to avoid conflicts during updates or audits.

Versioning and Updates

Careful attention must be paid to the component versioning to ensure that each component is always representative of the current state of the system. It is critical to maintain a version history for components to track changes and their impact on compliance. For instance, updating an encryption module should trigger a version increment and a review of dependent components by the team that is responsible for that component.

Real-World Component Definition Examples

The best way to understand the value of Component Definitions and how they might be utilized to implement controls is to see a few real-world examples and how those examples map back to the object-oriented compliance approach.

Example: Defining Components that Implement the NIST 800-53 AU-2 Event Logging Security Control

Utilizing the AU-2: Event Logging NIST 800-53 security control, the following is an example of how one might decompose a control implementation according to the components that address the control requirements.

The NIST 800-53 AU-2 control has the following requirements:

A. *Identify the types of events that the system is capable of logging in support of the audit function: [Assignment: organization-defined event types that the system is capable of logging];*
B. *Coordinate the event logging function with other organizational entities requiring audit-related information to guide and inform the selection criteria for events to be logged;*
C. *Specify the following event types for logging within the system: [Assignment: organization-defined event types (subset of the event types defined in AU-2a.) along with the frequency of (or situation requiring) logging for each identified event type];*
D. *Provide a rationale for why the event types selected for logging are deemed to be adequate to support after-the-fact investigations of incidents; and*
E. *Review and update the event types selected for logging [Assignment: organization-defined frequency].*

Requirement “A” is a policy-based requirement and should typically be defined at the organizational or departmental level. This requirement is trying to achieve a definition of “what” regarding log events needs to be captured by the systems that are governed by this control requirement.

In addition, requirement “A” is not only asking to define “what” will be audited, but also “how” that logging will occur, therefore, two components are needed to address this requirement:

- **Component 1 - Auditable Events and Content**[1] – Defines the events that are required to be captured by information systems, and their content (which can then be utilized across AU-3 requirements).
- **Component 2** – **Security Information and Event Management (SIEM)** – Defines the SIEM that will be utilized to execute the logging of the events defined by the “Auditable Events and Content” component.

[1] Depending on how an organization creates policies (e.g., organization, system, department-level) and the granularity at which they are defined, this component could be achieved by a “Policy” component, but the feasibility of that is something that would have to be determined by the organization.

Requirement “B” is a somewhat unique requirement. It hasn’t been addressed by the components created thus far and is process-based in nature. There needs to be a process by which event logging functionality is coordinated with other organizational entities (e.g., security and risk strategy sessions), so a new component is needed:

- **Component 3 - Audit, Incident Response, and Contingency Coordination** – Defines the process by which event logging is coordinated with other entities (e.g., the incident and contingency planning teams) to achieve overall system awareness.

Requirements “C” and “D” can be addressed by the “Auditable Events and Content” component. Requirement “C” asks that a subset of events defined in AU-2a are identified as well as the frequency to audit those events. This is a policy-based definition and should be defined in the component that establishes event logging requirements. Requirement “D” asks that the rationale for these event logging requirements be defined, again, this is best achieved through the policy component that has already been associated with the definition of events. As with requirement “A” the “how” component should also be addressed, so requirements “C” and “D” are addressed by the following two components:

- **Auditable Events and Content** – The component is updated to reflect the information needed to address requirements “C” and “D”.
- **SIEM** – This component is providing the “how” regarding the logging events actually being captured and correlated.

Requirement “E” introduces another new element, which is a requirement to review the events defined in the “Auditable Events and Content” component with a defined frequency. Again, this is a process-based requirement. NIST 800-53 requires a review of authorization documentation throughout a system’s lifecycle (e.g., annual review of Security plans, and supplemental plans like the Incident Response and Contingency Plan, as well as policies and procedures). A component dedicated to these required reviews, of which a review of the “Auditable Events” would be a subset is the most efficient way to address. Therefore, an additional component is created:

- **Review and Update System Documentation** – There are numerous ways to define this component, but ideally this would apply to all controls where there is a requirement to review a document on an ongoing basis (e.g., annually). Including the review of auditable events in this process-oriented component will achieves maximum efficiency.

This is a real-world example of how components can be utilized to implement the AU-2 security control. This is by no means the only way to deconstruct the control requirements and associate them with Component Definitions, but it is one way to do so. The next section examines these components’ alignment with the principles of “object-oriented compliance” and where the benefits of this method can be better understood.

Aligning the AU-2 Event Logging Component Definition Example to Object-Oriented Compliance

This section demonstrates how the Component Definitions, defined in the section above, align to the principles of object-oriented compliance.

Modularity

The approach taken in the example deconstructs the AU-2 security control into its key requirements that can then be addressed using the modular approach of Component Definitions. Component definitions are used to define the singular aspects of each part of the control requirement. Control requirements that are similar in nature (e.g., processed-based, technical, etc.) are examined to look for commonalities that can be leveraged across components.

Reusability

The Component Definitions are created in such a way that they can be used to not only implement other controls in the AU family, but across the control baseline. This is best highlighted through examples:

- **SIEM –** The security information and event management component has potentially the most reusability across both a network and a system.
  - **System-Level –** The SIEM component can be used as an implementation component in numerous AU controls such as AU-6: Audit Record Review, Analysis, and Reporting, AU-7: Audit Record Reduction and Report Generation. The Component Definition's reuse is not limited to only the AU family. Traditionally, the SIEM plays a major role in implementing the monitoring of the system (SI-4: System Monitoring), Flaw Remediation (SI-2: Flaw Remediation), and Malicious Code Protection (SI-3: Malicious Code Protection). As such, the SIEM Component Definition can be utilized in those control implementations as well.
  - **Network-Level –** The true value of a SIEM is the ability to correlate audit information across several systems for trend analysis and overall network visibility. When multiple information systems leverage the same SIEM, the Component Definition can be re-used across those systems. So, the Component Definitions and mappings utilized by the example system (System A) in the section above, would also apply to any other system that is utilizing the same SIEM (System B). Both systems can then leverage the Component Definition for the organization's component registry and all of the implementation capabilities of the SIEM Component Definition.

Maintenance

Maintaining Component Definitions through a "Component Definition Registry" (i.e., centralized repository where the OSCAL files are stored) creates an ecosystem where the level-of-effort (LoE) associated with maintaining authorization documents is greatly reduced. A great example of this is the following scenario:

- The Chief Information Security Officer (CISO) for an organization has identified a new threat vector while conducting the annual risk assessment. This new threat vector requires that additional audit events be configured and collected to preemptively identify vector activity on organizational information systems.

In the past, this scenario would prompt the compliance team (in association with the technical implementation teams) to review and update each system's authorization documentation package, looking for every section where the auditable events are referenced and ensuring that the newly identified event is added.

Through Component Definitions, this update occurs in a singular location; the "Auditable Events and Content" Component Definition, stored in the component registry. Once updated in this single location, any system leveraging this component automatically has those updates pushed out through component versioning. The control mapping is already defined in the Component Definition, thus greatly reducing the manual effort to identify and update impacted areas in the authorization documentation.

Scalability

The hierarchical nature of Component Definitions is a prime feature that allows for fast-paced, real-world scalability needs. Again, this is best illustrated through a scenario:

- The organization is running two systems on the network, system A and B. The Information System Owner (ISO) of system B has identified key application metrics to better identify key performance indicators (KPIs) of the application. The ISO has determined that the organizational SIEM does not provide the full set of dashboarding and metric analysis capabilities needed and has implemented a third-party audit log collection service to capture these needed application-level metrics.

In this scenario, the baseline component of the SIEM is still being utilized to implement the logging of security events and so no updates are needed to the Component Definition in the component registry. However, the component registry requires an update to capture the control implementation for this new third-party audit log collection service. A new Component Definition is created and mapped to the controls being implemented on system B. This Component Definition builds on the control implementations already mapped to the system from the "SIEM" component.

In the future, should system A choose to leverage that same third-party service in addition to the organization's SIEM, the system would simply leverage the associated Component Definition that exists in the organization's component registry.

Actionable Steps to Smart Risk Management

The Whitehouse/OMB is mandating the use of OSCAL for FedRAMP Authorized systems, and this is a strong indicator that other such mandates for the management of all systems through OSCAL will be quick to follow.

Agencies must implement strategies for modernizing their RMF and Governance, Risk and Compliance (GRC) assets. The OMB memo presents agencies with the opportunity to extend OSCAL to their private cloud and digital infrastructure. The adoption of the Component Definition Model requires a clear and well-defined technical framework to ensure its success.

This framework should address the lifecycle of components, from initial definition to deployment and maintenance. It should also provide guidelines for achieving modularity, reusability, maintainability, and scalability. The following is a potential approach for agencies to implement a Component Definition approach to OSCAL.

Strategy for Implementing Component Definition OSCAL Model for Federal Agencies

The following section outlines an actionable strategy that Federal Agencies could adopt for implementing the Component Definition OSCAL Model into their existing RMF.

1. **Establish Leadership Buy-In and Stakeholder Collaboration** - Engage Leadership Early: Educate key decision-makers on the benefits of adopting OSCAL and the Component Definition Model, emphasizing its alignment with OMB mandates and its potential to streamline compliance processes.
    a. Form a Cross-Functional Team: The team should be led by the CISO or designee and have representatives from compliance, IT, security operations, and program management to ensure all perspectives are considered during implementation.
    b. Define Roles and Responsibilities: Clearly outline who owns the creation, maintenance, and review of components to prevent gaps or redundancies. The key is to identify areas of efficiency and which stakeholders will have the highest value impact.

2. **Define the Component Definition Strategy** - Understand the overall agency architecture and where the Component Definition approach may provide the most benefit. Utilize a hierarchical approach, identifying General Support Systems (GSSs), major applications, and minor applications and where the strategy will provide the most return on investment (ROI).

   a. Determine Granularity and Standard Taxonomy - Establish guidelines to define components at the appropriate level of abstraction. This includes developing a consistent taxonomy for naming, categorizing, and describing components to ensure standardization across systems and maximize potential reuse.

3. **Build a Component Registry** - Start by identifying a secure, centralized repository (e.g., GitHub, GitLab, or an internal platform) that is accessible by the appropriate stakeholders (via appropriate Access Control) to store Component Definitions.
   a. Version Control: Create a strategy to implement versioning for each component to track changes over time and manage dependencies effectively. Strong versioning is also necessary as components are leveraged by systems. Understanding current Component Definitions versions and applying them appropriately to systems is key to a successful approach.

4. **Leverage Automation and Integration** - Adopt OSCAL-compatible tools like DRTConfidence, Compliance Trestle or OSCAL-based templates to author and validate Component Definitions. Ensuring that OSCAL outputs are appropriately validated upon creation is essential to maximizing component reusability.

5. **Integrate with CI/CD Pipelines -** Incorporate OSCAL documents into existing DevSecOps pipelines to ensure seamless updates to authorization documentation as systems evolve.
   a. Automate Validation: Implement automated checks in CI/CD pipeline to validate the accuracy and completeness of Component Definitions, ensuring compliance with OSCAL schema and organizational standards.

6. **Create a Library of Reusable Components -** Identify common components that will have maximum reusability (e.g., GSS components will likely be leveraged across numerous major and minor applications). Define reusable components for shared resources like firewalls, VPNs, SIEMs, and IAM platforms.
   a. Establish Control Mapping: Link each component to the controls and enhancements it supports, ensuring traceability and reusability.
   b. Iterate and Expand: Continuously add to the component library as new systems or requirements emerge.

7. **Pilot and Refine the Model -** Start with a Single GSS system, where the components will have the maximum impact, and identify challenges and refine processes. The most common adjustments will be to the granularity of components. Creating the most efficient components with maximum modularity and reusability requires iterations.
   a. Incorporate Feedback: Collect feedback from stakeholders involved in the pilot to address gaps and improve implementation strategies.
   b. Scale Gradually: Expand the use of Component Definitions to additional systems, leveraging the hierarchical nature of the components to build a strong foundational baseline and add to that baseline once all lessons learned have been incorporated into the processes.

8. **Train Personnel and Promote Cultural Change** - This approach is going to be novel to most stakeholders and will require significant investiture of resources to provide impactful hands-on training. Training programs should be offered for both technical and compliance teams to familiarize them with OSCAL, the Component Definition Model, and relevant tools. This training will be a key piece in breaking down the “silos” that exist due

to past compliance approaches and promotes a culture of collaboration between compliance and technical teams to ensure alignment in component creation and updates.

9. **Monitor, Review, and Optimize -** Schedule periodic reviews of the Component Definitions and the overall model to ensure it remains aligned with organizational goals and compliance requirements. Incorporate these reviews into existing RMF stage gates (e.g., security control assessments).
    a. Utilize Continuous Monitoring: Integrate Component Definitions with real-time monitoring tools to dynamically reflect system changes in authorization documentation.
    b. Iterate and Improve: Use metrics from audits, compliance checks, and system changes to continuously improve the framework.

10. **Align with Broader Federal Initiatives -** Leverage OSCAL's machine-readable nature to move towards continuous monitoring and authorization, as recommended by federal guidelines. Share best practices, tools, and components with other federal agencies to promote consistency and efficiency across the government.

By following these guidelines, federal agencies can successfully implement the Component Definition OSCAL Model, positioning themselves for enhanced compliance management, increased efficiency, and improved alignment with evolving federal mandates. This strategic approach will not only address current compliance challenges but also set a foundation for scalable and sustainable risk management practices.

Components: A New Beginning for Compliance

The evolution of compliance documentation has reached a pivotal moment. Traditional, monolithic approaches to crafting SSPs have served their purpose, but they are no longer sufficient for the dynamic, complex systems of today's digital landscape. The introduction of OSCAL and the adoption of the Component Definition Model signal a profound shift in how compliance is managed—one that mirrors the transformation seen in software development with the advent of OOAD. This is not just an incremental improvement; it is a reimagination of how compliance is conceptualized, implemented, and sustained.

By decomposing compliance into modular, reusable components, organizations can achieve a level of precision, efficiency, and scalability that was unattainable in the narrative-driven approach. The Component Definition Model addresses the chronic issues of inconsistency, redundancy, and maintenance that have long plagued compliance efforts. Moreover, this model provides a framework that is not only better suited to the rapidly changing configurations of modern systems but also aligns with federal mandates pushing for automation and continuous monitoring.

However, this approach requires a shift in mindset. It calls for organizations to think of compliance not as a static deliverable, but as a living, breathing entity—a digital artifact that evolves and lives alongside the system it represents. This transition will not be without

challenges. The initial investment in defining components, building libraries, and training personnel can be significant. Yet, the long-term benefits far outweigh these costs. Once implemented, the Component Definition Model provides the foundation for a more agile and accurate compliance posture, one that can adapt to the ever-shifting landscape of threats and regulatory requirements.

In this transformation, the principles of OOAD provide a guiding philosophy. They remind us that simplicity and reusability are the keys to managing complexity. Just as OOAD revolutionized software development, the integration of these principles into compliance documentation offers the promise of a smarter, more sustainable future.

The adoption of OSCAL is not merely a technical advancement; it is a paradigm shift. It represents a movement toward a compliance ecosystem that is integrated, modular, and reflective of the true state of a system. This is compliance that works in real-time, driven by data, and seamlessly woven into the lifecycle of modern systems. For federal agencies and organizations striving to meet NIST, FedRAMP, or CMMC requirements, the Component Definition Model is not just a tool; it is the blueprint for the future of smart compliance. By embracing this approach, organizations can move beyond the limitations of the past and build a compliance framework that is representative of the dynamic state of the underlying systems. As we stand at the threshold of this new era, the message is unmistakable: compliance is no longer just about documentation—it is about enabling digital assurance. The future of compliance is smarter, faster, and more resilient, and it begins with the adoption of “Object-Oriented Compliance” through OSCAL. This is the way forward; this is the new beginning for compliance.

Notices

About stackArmor

stackArmor is a cybersecurity compliance industry leader with over ten years of security and compliance engineering experience helping public sector organizations accelerate their ability to meeting FedRAMP, FISMA/RMF, DoD Cloud Computing SRG, and CMMC requirements. At the heart of stackArmor's approach is the operationalization of NIST SP 800-53 security controls through OSCAL-Informed Component Definition models, compliance and engineering standardization, engineered automation, and specialized teams of experts – together helping to reduce the time and cost of ATO projects by as much as 40%.

stackArmor supports over 40 customers by helping them implement strong security and compliance controls based on NIST SP 800-53 or NIST SP 800-171 to align with FedRAMP, FISMA/RMF and CMMC 2.0 compliance requirements. stackArmor's ThreatAlert® ATO Accelerator for FedRAMP has been successfully deployed at Department of Justice, US Department of Education, General Services Administration (GSA) and DOD in addition to over 20 CSPs including Red Hat, Bluescape and Microstrategy amongst others.