WHITEPAPER

Complying with Executive Order 14028 Software Security Requirements

Security Compass

Contents

Introduction

Organizations that provide software to US federal agencies face new requirements regarding software security. By early 2023, the Federal Acquisition Regulation (FAR) Council will require compliance with NIST's Secure Software Development Framework (SSDF). This paper helps readers understand the potential impact of SSDF compliance on their organizations and steps they can take to meet SSDF requirements.

Supply Chain Attacks Are Growing

In December 2020, FireEye researchers discovered "a supply chain attack trojanizing SolarWinds Orion business software updates". The backdoor in Orion – a platform for centralized monitoring and management of IT infrastructure – allowed the attackers full administrative access to Orion customers' infrastructure. The attack has been attributed to Russian nation state actors and affected over 100 private sector entities and at least nine federal agencies, including the departments of Defense, Commerce, Energy, Justice, Homeland Security, State, and Treasury and the National Institute of Health.

The SolarWinds attack was a supply chain compromise. A supply chain attack compromises software used by an organization, instead of targeting an organization directly . In this case, the attacker inserted back doors into legitimate software and waited for SolarWinds to distribute the attack through "trusted" updates. Shortly after the SolarWinds event, a back door that affected thousands of organizations was discovered in Microsoft Exchange (and attributed to China).

Supply chain attacks need not insert backdoors into the supply chain. They can just as effectively leverage coding errors, misconfigurations, and other security weaknesses in commercial and open source applications used by the targeted organizations. The end result is the same as in the SolarWinds breach - an attack vector through which attackers can compromise an application or system.

Executive Order 14028

In light of these events, and the Colonial Pipeline ransomware attack in early 2021, the Biden Administration issued Executive Order (EO) 14028 - "Improving the Nation's Cybersecurity." Included in the EO is the requirement that "the Federal Government must take action to rapidly improve the security and integrity of the software supply chain."

> **"The development of commercial software often lacks transparency, sufficient focus on the ability of the software to resist attack, and adequate controls to prevent tampering by malicious actors."**
>
> EO 14028, Section 4
> May 12, 2021

The EO comprises 11 sections covering topics from general policy declarations to information sharing to improving the U.S. federal government's investigative and remediation capabilities. In each, the EO requires various government agencies to produce plans, policies, and guidelines within specified timeframes "to identify, deter, protect against, detect, and respond to" threats against the public and private sectors. Section 4 – Enhancing Software Supply Chain Security – addresses the federal government's security requirements for software it uses.

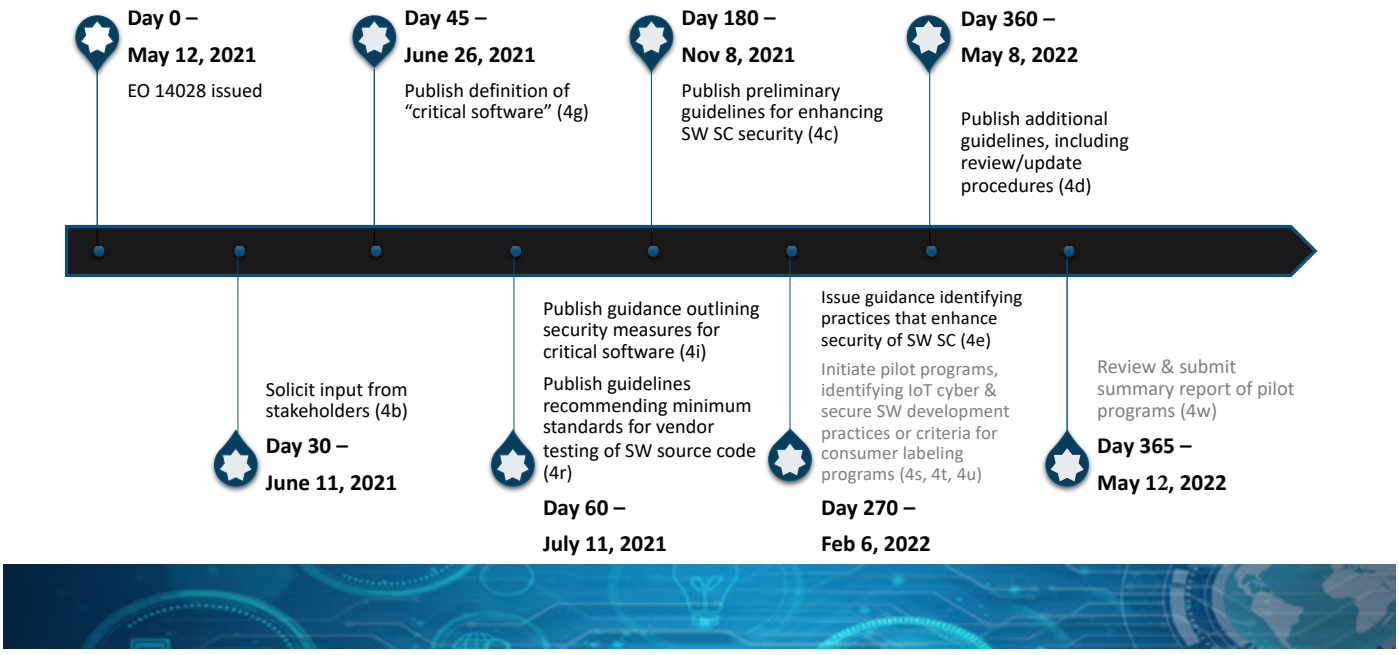
Section 4 acknowledges that the government lacks sufficient information on the software it procures to resist attacks and a "pressing need to implement more rigorous and predictable mechanisms for ensuring that products function securely."

It includes orders to the Director of NIST to "Within 180 days…publish preliminary guidelines…drawing on existing documents as practicable, for enhancing software supply chain security."

The EO prioritizes "critical software" and requires the guidelines to include standards for several practices, including secure development environments, using automated tools to identify vulnerabilities, and "attesting to conformity with secure software development practices."

> "The [secure software] guidelines shall include criteria that can be used to evaluate software security, **include criteria to evaluate the security practices of the developers and suppliers themselves…"**
>
> Executive Order 14028
> May 12, 2021

EO Section 4 Tasks and Timelines



Source: https://www.nist.gov/itl/executive-order-improving-nations-cybersecurity

The Executive Order and Authority to Operate (ATO)

The Federal Information Security Modernization Act already requires federal agencies to achieve Authority to Operate (ATO) by having systems in place to assess and monitor security and privacy risks. This includes compliance with NIST's Risk Management Framework.

Organizations selling software to government agencies should expect security requirements to change and align with the EO. Section 2 of the EO orders a review of the Federal Acquisition Regulation (FAR) and Defense Federal Acquisition Regulation (DFAR) Supplement and recommendations to the FAR Council to standardized contract language for cybersecurity requirements. At the request of the Department of Defense, General Services Administration, and NASA, legislation is also in process to amend the FAR's cybersecurity contractual requirements across Federal agencies for unclassified information systems.

Secure Software Development Framework

The Executive Order orders NIST to identify "existing or develop new standards, tools, and best practices for complying" with the security requirements. Fortunately, NIST had already started work on such a framework. In 2019, NIST published "Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework (SSDF)" which defined secure software development practices and tasks for software producers. The white paper included most of the itemized requirements in the EO. An update in 2021 covered the remaining items, resulting in SP 800-218, Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities.

The SSDF builds from and references several other industry efforts, including the Cloud Native Computing Foundation's (CNCF) Software Supply Chain Best Practices, OWASP's Open Software Assurance Maturity Model (OpenSAMM), NIST's Guidelines on Minimum Standards for Developer Verification of Software, the Building Security In Maturity Model (BSIMM), and SAFECode's Fundamental Practices for Secure Software Development.

> "The SSDF presents an opportunity to measurably improve the cybersecurity posture of U.S. federal, state, and local government agencies. Security Compass embraces and contributes to this standard."
>
> Rohit Sethi
> CEO, Security Compass

The SSDF is a set of high-level secure software development practices that can be integrated with an organization's development process. The practices are organized into four groups.

- **Prepare the Organization (PO):** Ensure that the organization's people, processes, and technology are prepared to perform secure software development at the organization level and, in some cases, for each individual project.
- **Protect the Software (PS):** Protect all components of the software from tampering and unauthorized access.
- **Produce Well-Secured Software (PW):** Produce well-secured software that has minimal security vulnerabilities in its releases.
- **Respond to Vulnerabilities (RV):** Identify vulnerabilities in software releases and respond appropriately to address those vulnerabilities and prevent similar vulnerabilities from occurring in the future.

SSDF Implications on Existing Software Development Processes

The SSDF provides high-level secure software activities for integration into an organization's software development life cycle (SDLC). The activities or practices are intended to minimize the number of vulnerabilities in software, mitigate the impact of exploits of undetected or unaddressed vulnerabilities, and "address the root causes of vulnerabilities to prevent future recurrences."

The SSDF does not require a specific SDLC. Its activities can be applied in waterfall, agile, or DevOps models. It is not prescriptive in its recommendations, instead focusing on the outcome of the practices. This allows organizations of any size or security maturity to implement and benefit from the practices. The practices can be applied to traditional software development, IT, Internet of Things (IoT), or Industrial Control Systems (ICS) programs.

While applicable across any SDLC, the SSDF does include several themes.

Shift left

Meeting the goal of fewer vulnerabilities can be achieved in many ways. Traditionally, organizations would run automated scans later in the development process. This increases remediation costs and slows releases. The SSDF encourages organizations to "shift left" and implement security activities early in the SDLC.

Take a risk-based approach

Not all projects warrant the same level of security scrutiny. Each will have different requirements, scales, scopes, budgets, and problems. Bugs and flaws in some projects can result in devastating outcomes, while others may present "acceptable risk" for an organization. The SSDF acknowledges that risk, cost, and feasibility are considerations when deciding which practices to adopt for each project.

Adopt a common language

To improve communication between business owners, security teams, development, and operations, the SSDF provides a common vocabulary to describe secure software development practices. This common language also allows software acquirers to describe and define the required security characteristics of software in their acquisition process. Commercial software companies can use the vocabulary to describe their security practices to customers.

The SSDF provides mapping from the requirements in the EO to the practices in the SSDF to help business, development, and security resources better communicate why specific activities are required.

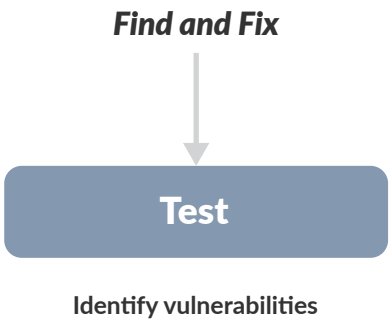
SSDF Practices Corresponding to EO 14028 Subsections

EO 14028 Subsection	SSDF Practices and Tasks
4e(i)(A)	PO.5.1
4e(i)(B)(B)	PO.5.1
4e(i)(C)	PO.5.1, PO.5.2
4e(i)(D)	PO.5.1
4e(i)(E)	PO.5.2
4e(i)(F)	PO.3.2, PO.3.3, PO.5.1, PO.5.2
4e(ii)	PO.3.2, PO.3.3, PO.5.1, PO.5.2
4e(iii)	PO.3.1, PO.3.2, PO.5.1, PO.5.2, PS.1.1, PS.2.1, PS.3.1, PW.4.1, PW.4.4
4e(iv)	PO.4.1, PO.4.2, PS.1.1, PW.2.1, PW.4.4, PW.5.1, PW.6.1, PW.6.2, PW.7.1, PW.7.2, PW.8.2, PW.9.1, PW.9.2, RV.1.1, RV.1.2, RV.2.1, RV.2.2, RV.3.3
4e(v)	PO.3.2, PO.3.3, PO.4.1, PO.4.2, PO.5.1, PO.5.2, PW.1.2, PW.2.1, PW.7.2, PW.8.2, RV.2.2
4e(vi)	PO.1.3, PO.3.2, PO.5.1, PO.5.2, PS.3.1, PS.3.2, PW.4.1, PW.4.4, RV.1.1, RV.1.2
4e(vii)	PS.3.2
4e(viii)	RV.1.1, RV.1.2, RV.1.3, RV.2.1, RV.2.2, RV.3.3
4e(ix)	All practices and tasks consistent with a risk-based approach
4e(x)	PS.2.1, PS.3.1, PS.3.2, PW.4.1, PW.4.4

Source: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf

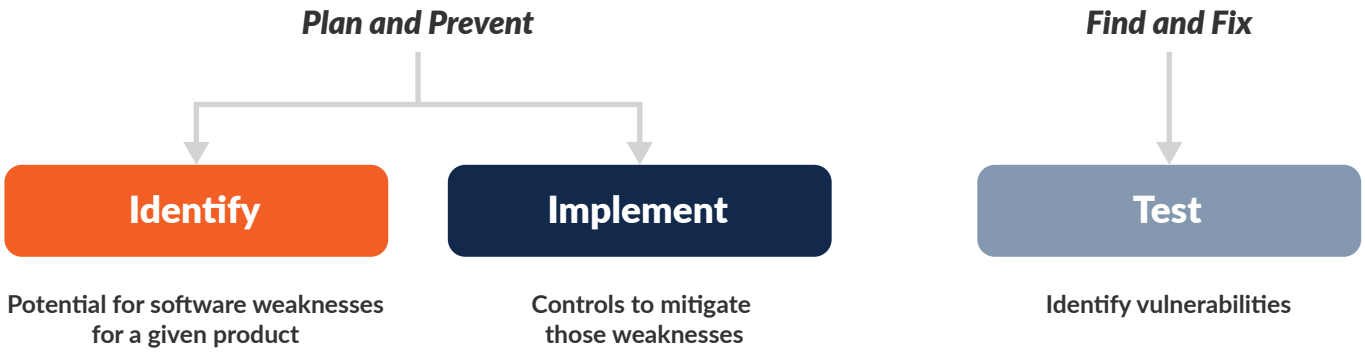
How SD Elements Helps You Follow SSDF Recommendations

As mentioned earlier, most organizations test for security by running automated scans late in the SDLC to identify coding errors and design flaws that could be exploited by an attacker. This "Find and Fix" approach is reactive and slows down developers.



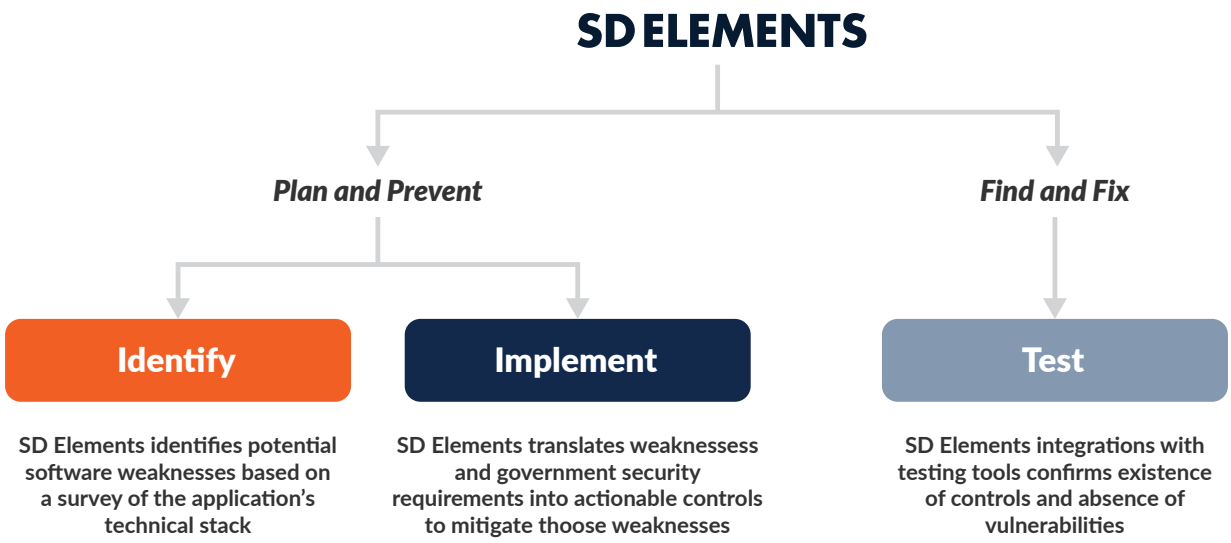
A better approach is "Plan and Prevent." In this approach, teams anticipate and identify weaknesses in the software, frameworks, and deployment environment (plan) then implement mitigation controls for those weaknesses during the normal development process (prevent).

By adopting a "Plan and Prevent" strategy, teams proactively *avoid* vulnerabilities. Security testing truly becomes a validation exercise to confirm all required mitigation controls are properly implemented.



In many organizations, the "Plan and Prevent" exercise consists of lengthy security requirements maintained on spreadsheets. Others may create manual threat models. Threat modeling teams can spend days mapping an application's data flow, diagramming trust boundaries, and prescribing mitigations for implementation by development teams. The investment in time from scarce security and development resources limits manual threat models to a few critical applications.

SD Elements automates the "Plan and Prevent" exercise. A brief survey provides information on the application's technology stack, including programming languages, frameworks, deployment environment, and applicable regulatory requirements. SD Elements enumerates potential weaknesses from this then translates these and other federal government security requirements into actionable tasks to be implemented by development, security, and operations. Its integrations with development and DevOps tooling, issue trackers, security testing tools validates that all controls are properly implemented and provides near real-time reporting on the status of each item.



SD Elements scales a "Plan and Prevent" strategy across the organization's entire software portfolio. Its extensive content library supports a broad range of technologies, platforms, programming languages, and regulatory standards. Just-in-Time Training (JITT) delivered directly to developers' desktops provides brief, informative guidance on secure coding practices.

Examples: Prepare the Organization (PO)

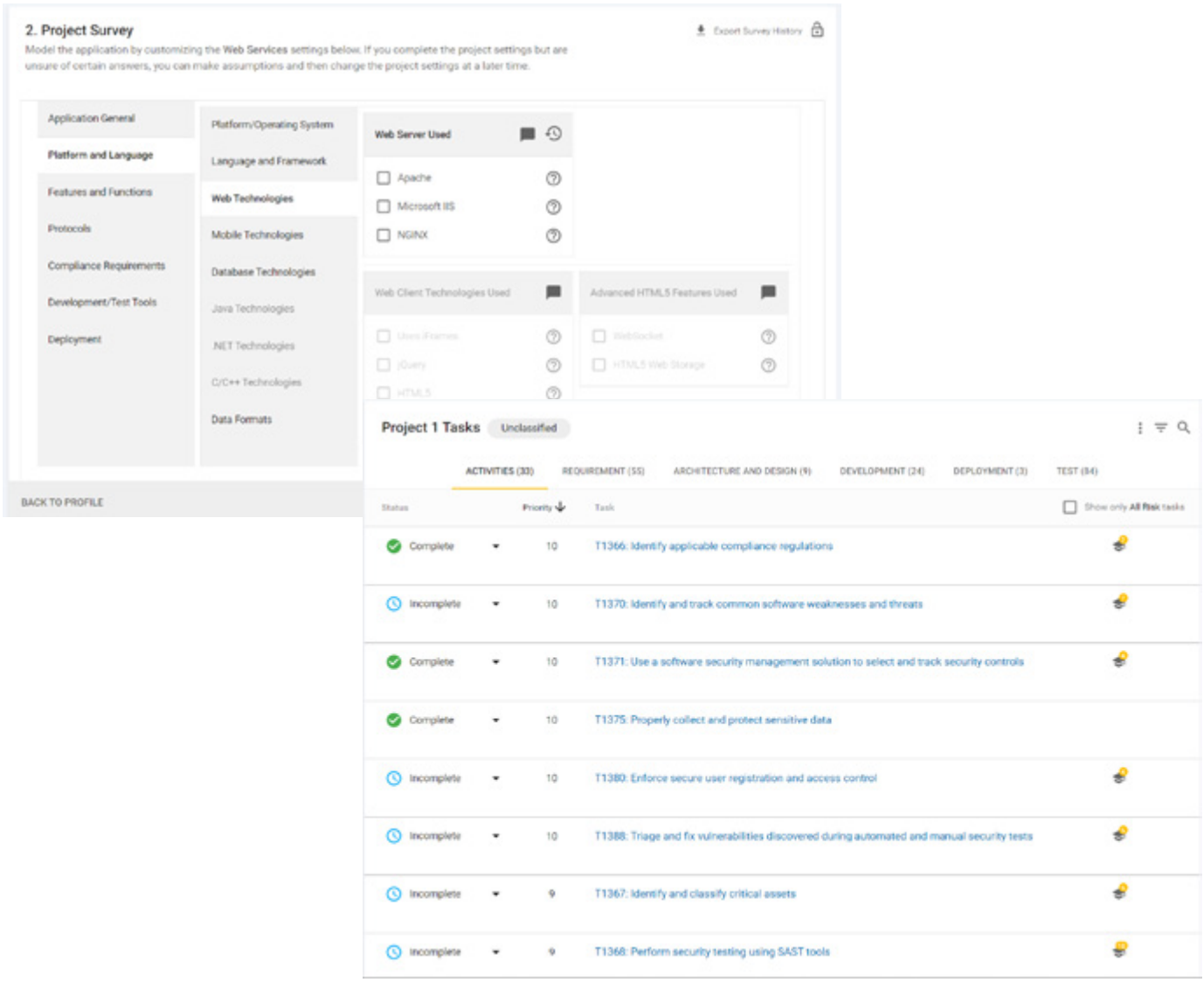
PO.1: Define Security Requirements for Software Development

The first practice in the Prepare the Organization group (PO.1) is *"Define Security Requirements for Software Development": "Ensure that security requirements for software development are known at all times so that they can be taken into account throughout the SDLC, and duplication of effort can be minimized because the requirements information can be collected once and shared. This includes requirements from internal sources (e.g., the organization's policies, business objectives, and risk management strategy) and external sources (e.g., applicable laws and regulations).*

SD Elements supports two of three tasks within PO.1.

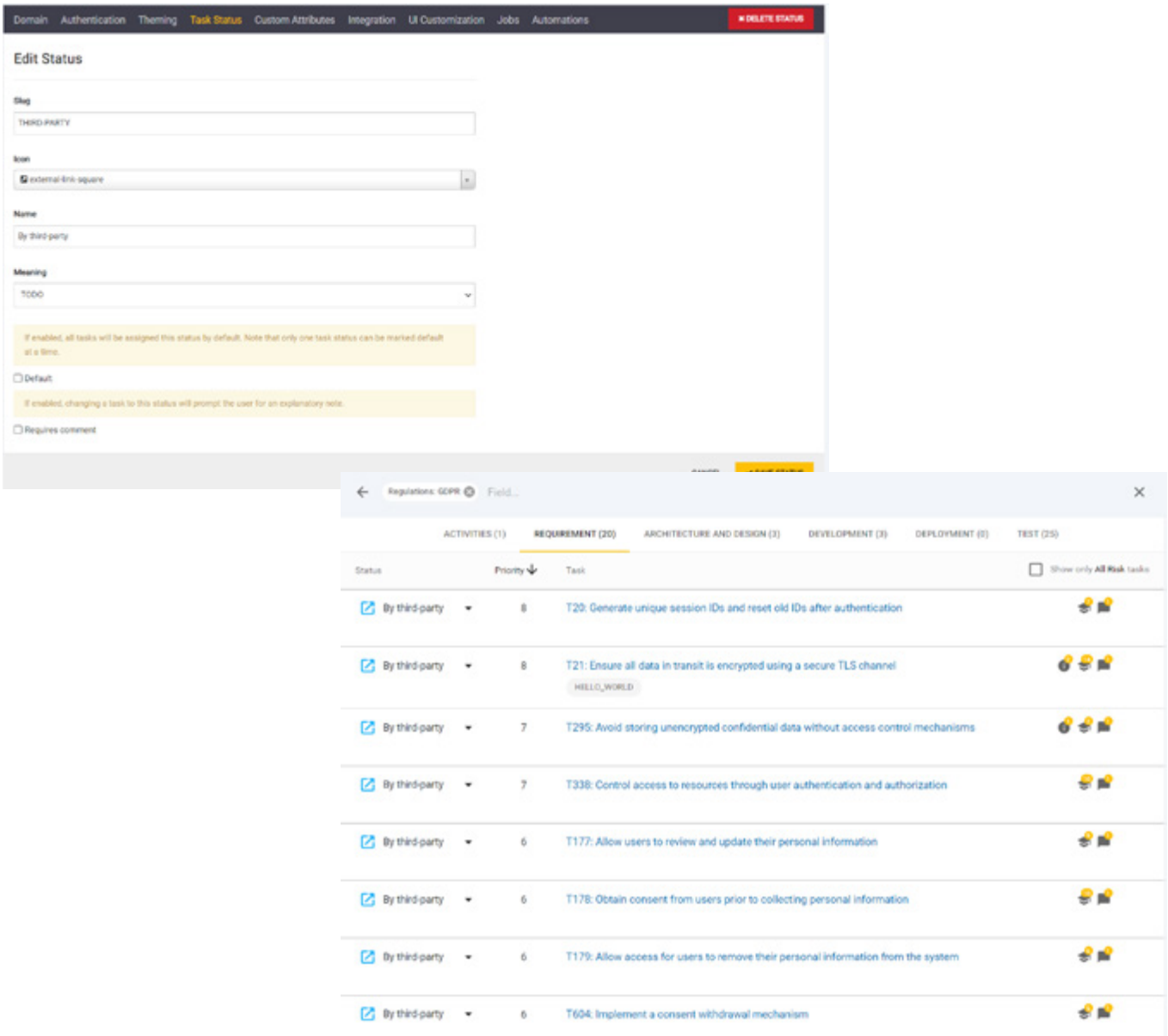
- **PO.1.2: Identify and document all security requirements for organization-developed software to meet and maintain the requirements over time.** Examples include defining policies that specify risk-based software architecture and design requirements, analyzing the risk of applicable technology stacks, and defining policies that specify the security requirements for the organization's software, and verify compliance at key points in the SDLC

  - SD Elements automatically enumerates security requirements for an application based on a survey of the application's technical stack including its deployment environment and applicable regulatory standards. It then translates those requirements into actionable tasks for development, security, and operations. Tasks are communicated through integrations with issue trackers and other common development tools.



- **PO.1.3: Communicate requirements to all third parties who will provide commercial software components to the organization for reuse by the organization's own software.** Organizations should define a core set of security requirements for software components and ensure that the requirements are included in all contracts. These should include vulnerability disclosure policies and incident response capabilities.

  » For software that requires third-party libraries, organizations can mandate the software vendors comply with a list of controls. This can be done with an export model of selected controls from the survey, direct integration into the third party's Jira workflow, or through a shared Jira instance.



PO.2: Prepare the Organization: Implement Roles and Responsibilities

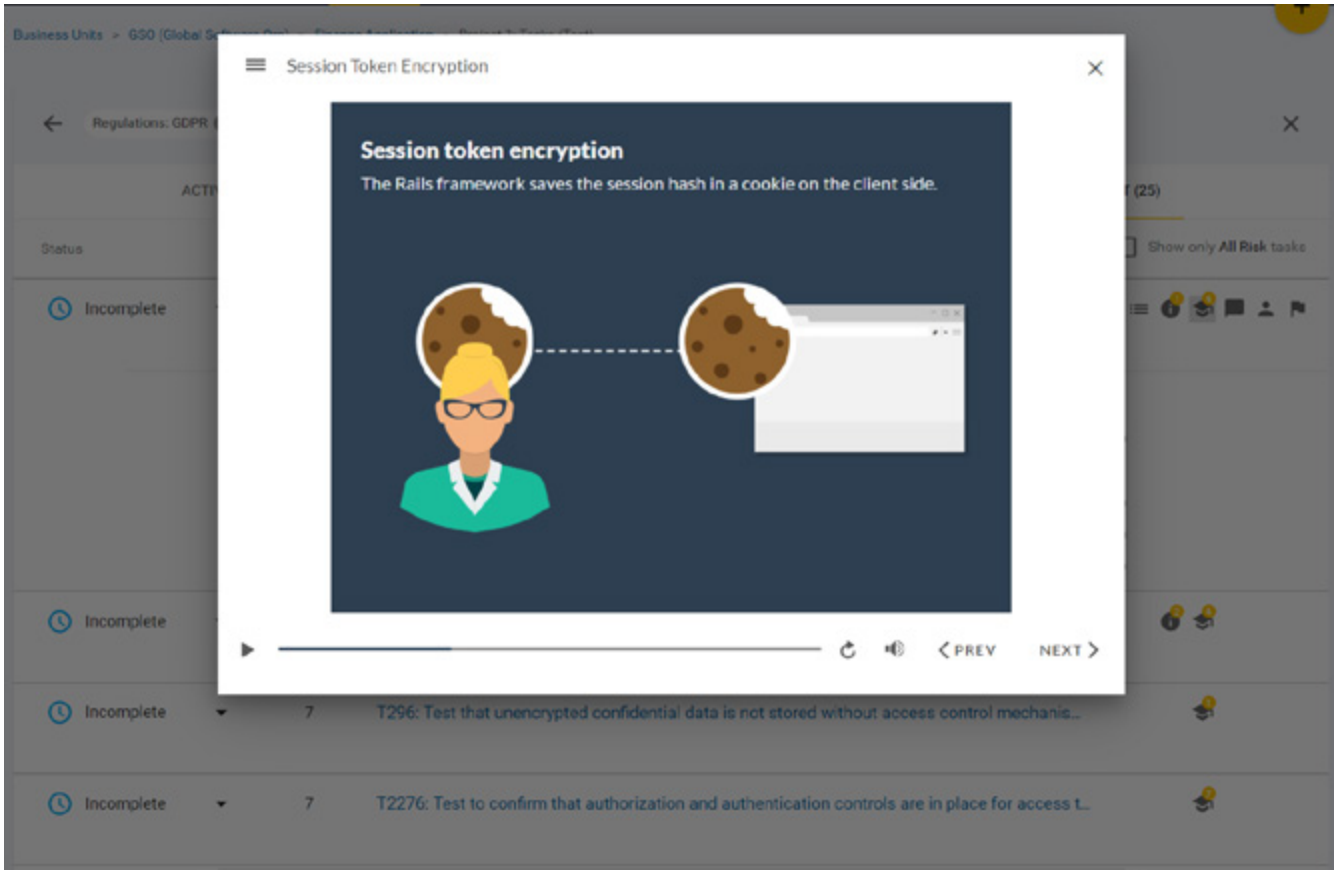
The second practice in Prepare the Organization (PO.2) requires "Implement Roles and Responsibilities"

"Ensure that everyone inside and outside of the organization involved in the SDLC is prepared to perform their SDLC-related roles and responsibilities throughout the SDLC."

- **PO.2.2: Provide role-specific training for all personnel with responsibilities that contribute to secure development. Periodically review role-specific training and update it as needed.** This task requires organizations to document the desired outcome of training, define the curriculum, and create a training plan for each role in the development process,

  - Security Compass offers an extensive on-demand eLearning library that supports every role in software development and deployment. Courses for software developers, software architects, QA engineers, and project managers cover fundamental elements of software security and language-specific secure coding practices. Our Software Security Practitioner (SSP) Suites are pre-selected sets of courses for specific coding languages or specific roles within the development team.

  - To reinforce secure coding training, SD Elements Just-In-Time Training (JITT) provides short videos that support the implementation of software security and privacy requirements during development.



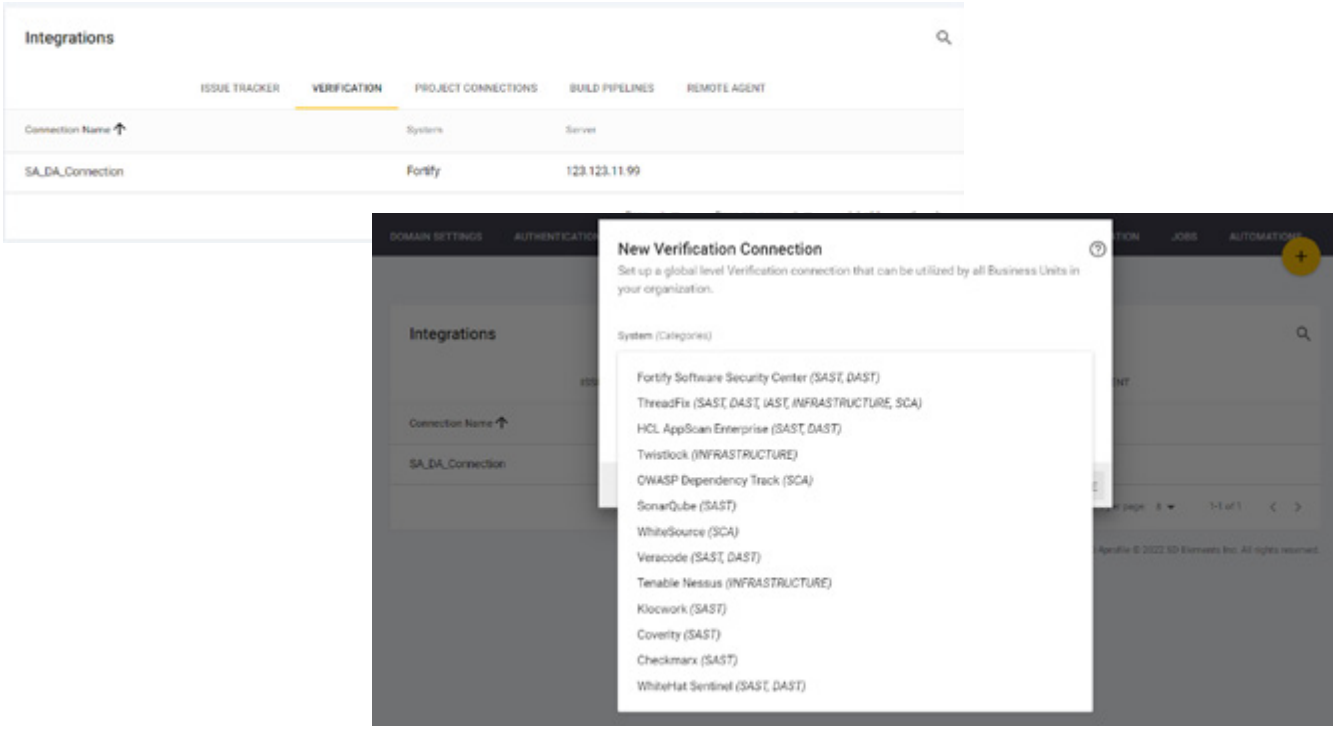
PO.3: Prepare the Organization: Implement Supporting Toolchains

PO.3 advocates for automation across the SDLC:

"Use automation to reduce human effort and improve the accuracy, consistency, usability, and comprehensiveness of security practices throughout the SDLC, as well as provide a way to document and demonstrate the use of these practices. Toolchains and tools may be used at different levels of the organization, such as organization-wide or project-specific, and may address a particular part of the SDLC, like a build pipeline. "

- **PO.3.3: Configure tools to collect evidence and artifacts of their support of the secure software development practices as defined by the organization.** Organizations will want traceability of all activity, including issue tracking and validation of controls. This is useful for internal use and can also provide evidence of compliance with SSDF to auditors and customers.

  » Unlike spreadsheet-based models that are subject to error and lack traceability, SD Elements provides a centralized repository for all activity and full, evidentiary quality auditing for all actions. Teams have near real-time reporting on the status of each project with granularity to individual controls. Integrations with issue trackers allows organizations to assign and track each task for completion. Integrations with security testing tools like static application security testing (SAST) and dynamic application security testing (DAST) tools enable fast, consistent validation of security control implementation status and sharing of results directly with developers.



Examples: Protect the Software (PS)

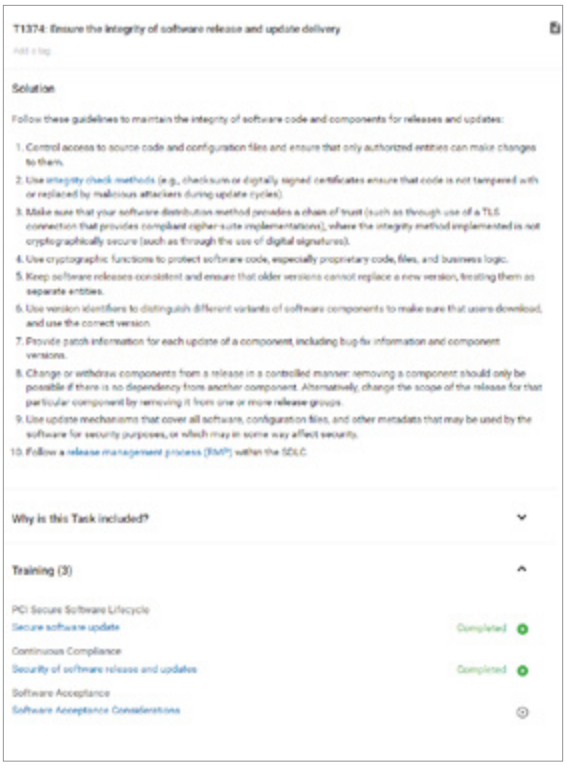
PS.1: Protect All Forms of Code from Unauthorized Access and Tampering

PS.1 acknowledges that building secure applications requires organizations to protect their development and build environments. Recent attacks on SolarWinds and CodeCov demonstrated the disastrous impact of poor practices.

"Help prevent unauthorized changes to code, both inadvertent and intentional, which could circumvent or negate the intended security characteristics of the software. For code that is not intended to be publicly accessible, this helps prevent theft of the software and may make it more difficult or time-consuming for attackers to find vulnerabilities in the software."

- PS1.1: Store all forms of code, including source code and executable code, based on the principle of least privilege so that only authorized personnel, tools, services, etc. have the necessary forms of access. This includes maintaining repositories that are protected for confidentiality and integrity, the use of code and commit signing.

  - SD Elements' risk mitigation controls include recommendations for strict access control and secure storage rules, as well as the use of obfuscation and checksum or digitally signed certificates to ensure that code is not tampered with or replaced by malicious attackers during update cycles. It provides controls for using cryptographic functions to protect software code, files, and business logic, and the use of back-out positions so applications can recover from failed changes or unexpected results.



Examples: Produce Well-Secured Software (PW)

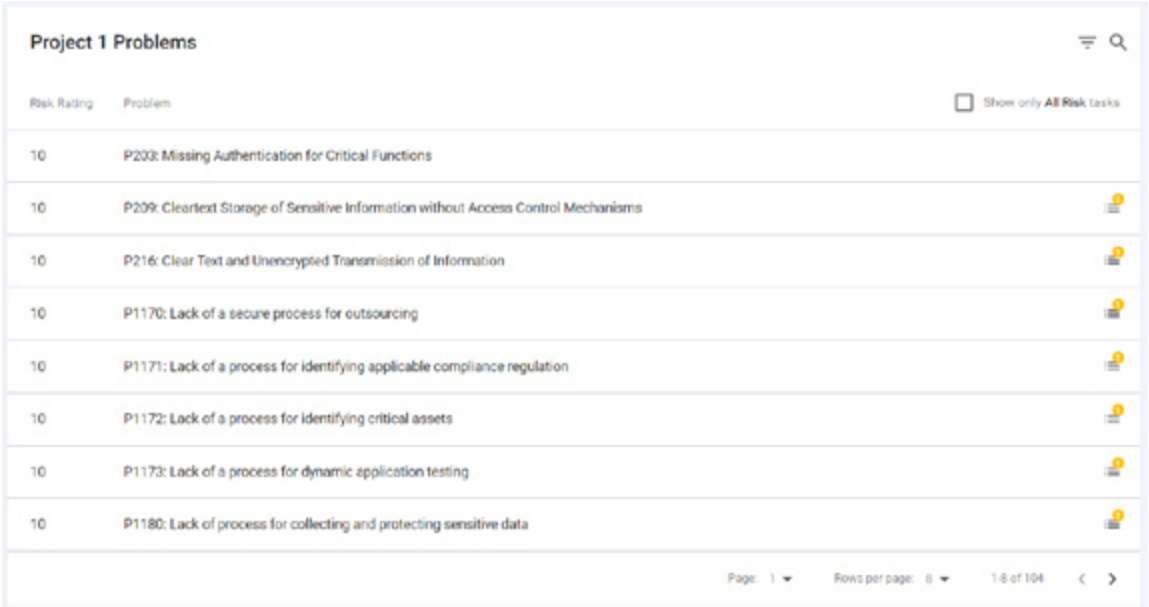
PW.1: Design Software to Meet Security Requirements and Mitigate Security Risks

PW.1 requires organizations to adopt a "Plan and Prevent" strategy to anticipate weaknesses in an application and proactively adopt risk mitigation controls.

"Identify and evaluate the security requirements for the software; determine what security risks the software is likely to face during operation and how the software's design should mitigate those risks; and justify any cases where risk-based analysis indicates that security requirements should be relaxed or waived. Addressing security requirements and risks during software design (secure by design) helps make software development more efficient."

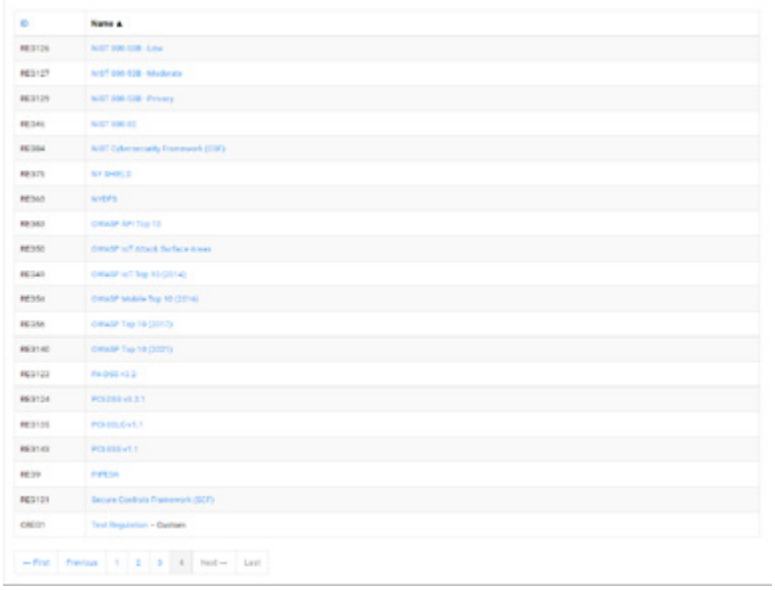
- **PW.1.1: Use forms of risk modeling, such as threat modeling, attack modeling, or attack surface mapping, to help assess the security risk for the software.** This includes identifying potential weaknesses and using a risk-based approach to address the risks and implement mitigations.

  - SD Elements automates threat modeling, reducing the time required from weeks to hours. After the completion of a project survey, SD Elements identifies weaknesses that threats target and enables the delivery of mitigation controls directly to those responsible in development, security, and operations. By anticipating threats and building mitigations as part of the normal development process, security testing is simplified, more proactively, and easily scaled across an entire software portfolio.



- **PW.1.2: Document the software's security requirements, risks, and design decisions.** This includes the requirement to document the response to each risk, including how mitigations are to be achieved.

  » SD Elements' content library includes dozens of regulatory standards and best practices frameworks and translates these into easy-to-follow instructions for development, assurance, and deployment teams. It can be customized to accommodate secure coding policies of an individual company or project.



PW.4: Reuse Existing, Well-Secured Software When Feasible Instead of Duplicating Functionality

PW.4 encourages organizations to identify and reuse "known-good" components and microservices.

"Lower the costs of software development, expedite software development, and decrease the likelihood of introducing additional security vulnerabilities into the software by reusing software modules and services that have already had their security posture checked. This is particularly important for software that implements security functionality, such as cryptographic modules and protocols."

- **PW.4.2: Create well-secured software components in-house following SDLC processes to meet common internal software** development needs that cannot be better met by third-party software. Secure development requirements apply equally to applications and reusable components or micro-services.

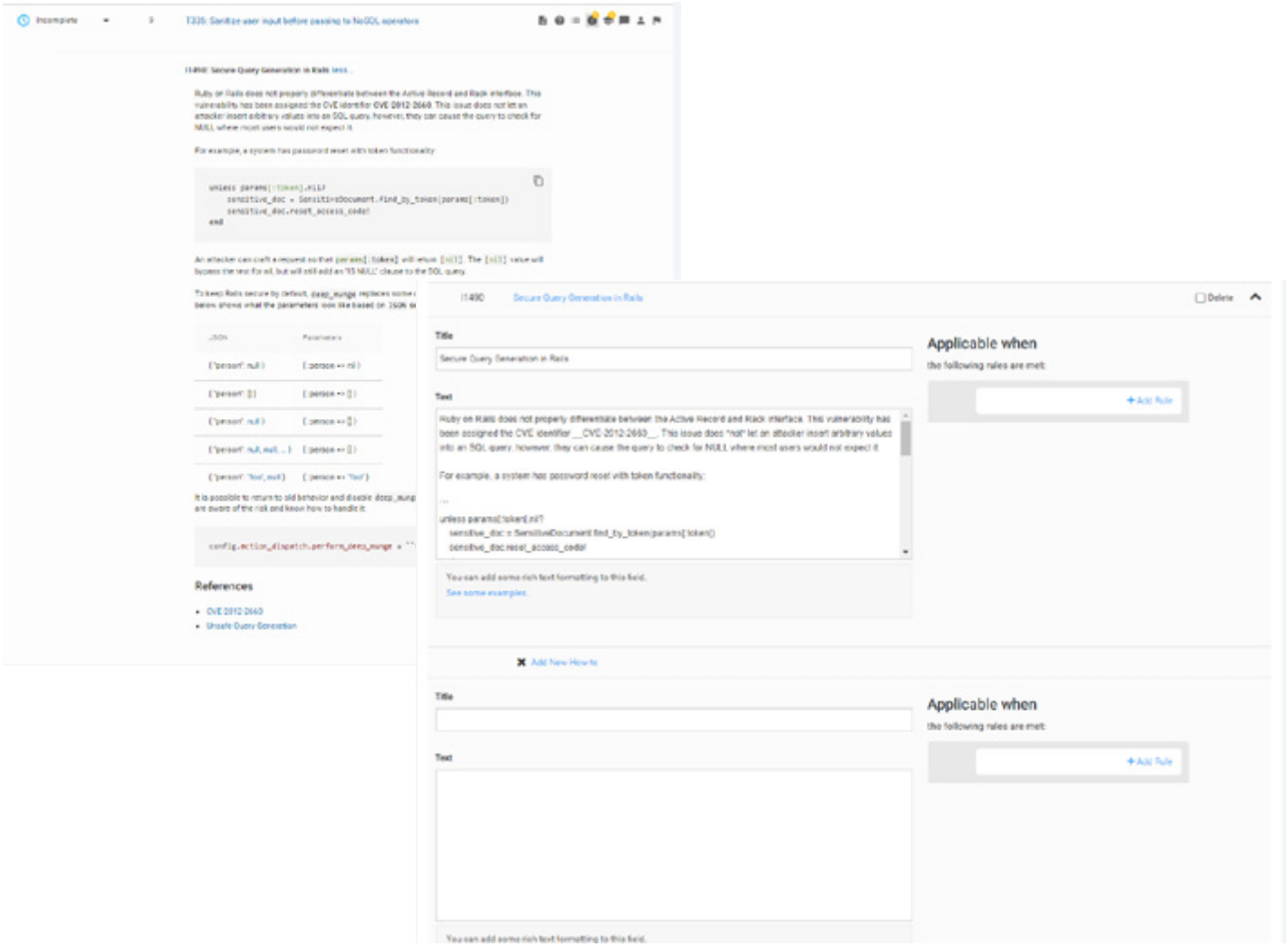
  » SD Elements can be used on projects of any size. As a project evolves, updates to the survey will update any required controls.

PW.5: Create Source Code by Adhering to Secure Coding Practices

PW.5 covers the core practices of building secure software. It requires organizations to consider weaknesses that may be inherent to specific programming languages and deployment environments.

- **PW.5.1: Follow all secure coding practices that are appropriate to the development languages and environment.** This task covers all secure coding best practices, including input validation, avoiding unsafe functions and calls, ensuring complete logging, and code reviews.

  » SD Elements enables secure development by translating language and platform specific secure development policies into specific tasks. A team of security experts continuously updates security controls, including coding samples and test plans, to ensure that teams apply consistent and effective controls. Extensive secure coding policies are included with SD Elements, or organizations can add their own policies.



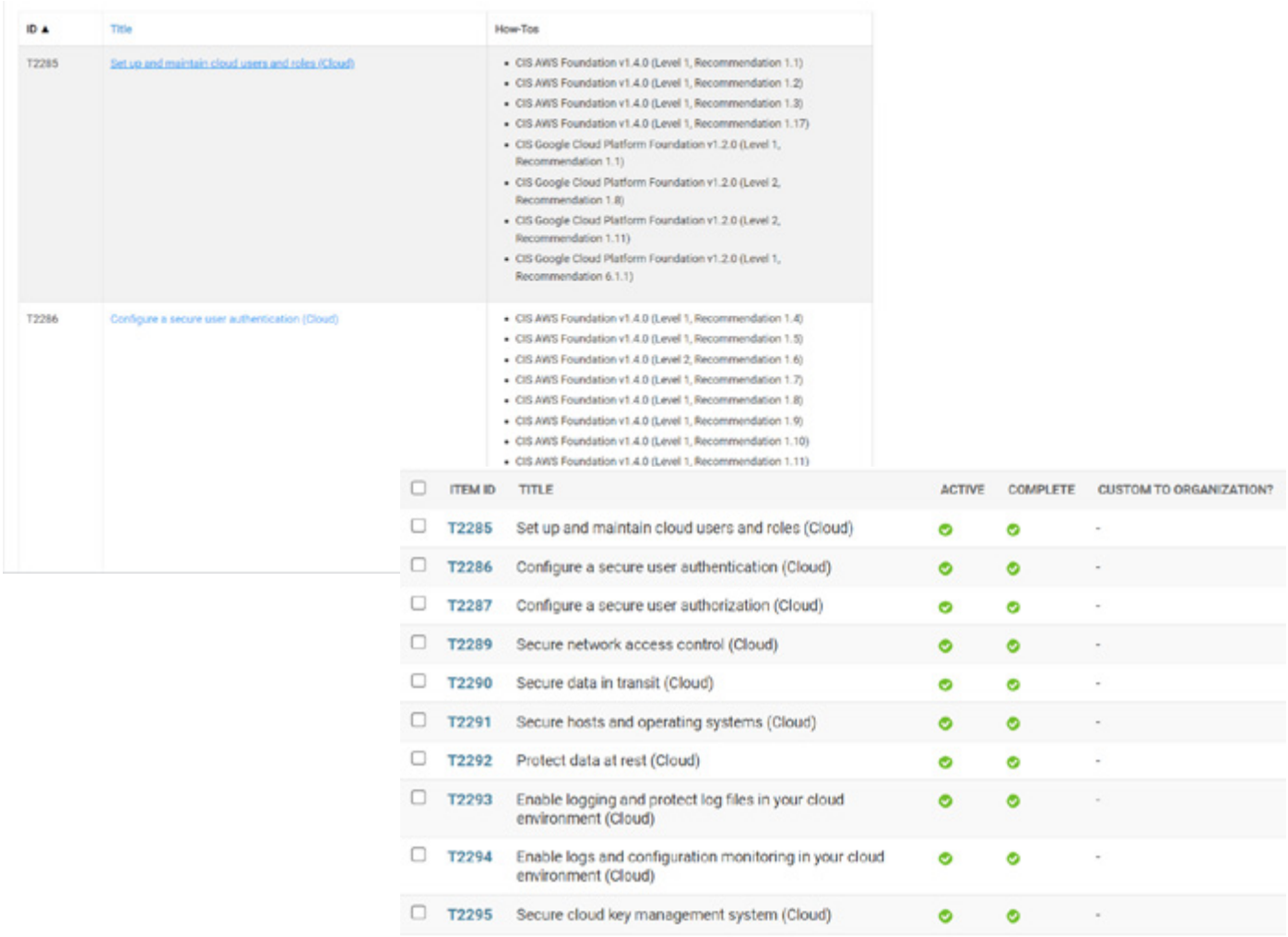
PW.9: Configure Software to Have Secure Settings by Default

PW.9 recognizes that weaknesses can enter an application from multiple points.

"Help improve the security of the software at the time of installation to reduce the likelihood of the software being deployed with weak security settings, putting it at greater risk of compromise."

- **PW.9.1: Define a secure baseline by determining how to configure each setting that has an effect on security so that the default settings are secure and do not weaken the security functions provided by the platform, network infrastructure, or services.** Misconfigurations of servers and storage can result in data leakage and provide simple attack vectors to attackers. These settings are often missed by automated scanners and must be explicitly confirmed by security and/or operations.

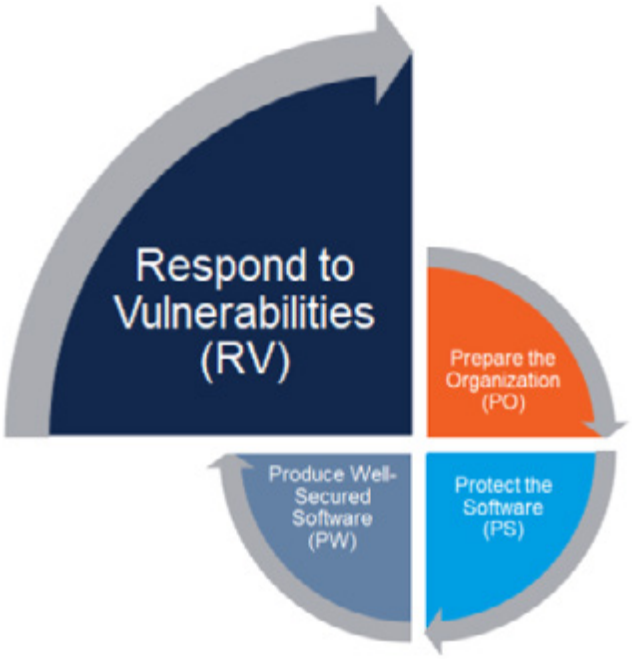
  » SD Elements' content library includes secure configurations as well as security standards for cloud deployments from the Cloud Security Alliance.



ID ▲	Title	How-Tos
T2285	Set up and maintain cloud users and roles (Cloud)	• CIS AWS Foundation v1.4.0 (Level 1, Recommendation 1.1) • CIS AWS Foundation v1.4.0 (Level 1, Recommendation 1.2) • CIS AWS Foundation v1.4.0 (Level 1, Recommendation 1.3) • CIS AWS Foundation v1.4.0 (Level 1, Recommendation 1.17) • CIS Google Cloud Platform Foundation v1.2.0 (Level 1, Recommendation 1.1) • CIS Google Cloud Platform Foundation v1.2.0 (Level 2, Recommendation 1.8) • CIS Google Cloud Platform Foundation v1.2.0 (Level 2, Recommendation 1.11) • CIS Google Cloud Platform Foundation v1.2.0 (Level 1, Recommendation 6.1.1)
T2286	Configure a secure user authentication (Cloud)	• CIS AWS Foundation v1.4.0 (Level 1, Recommendation 1.4) • CIS AWS Foundation v1.4.0 (Level 1, Recommendation 1.5) • CIS AWS Foundation v1.4.0 (Level 2, Recommendation 1.6) • CIS AWS Foundation v1.4.0 (Level 1, Recommendation 1.7) • CIS AWS Foundation v1.4.0 (Level 1, Recommendation 1.8) • CIS AWS Foundation v1.4.0 (Level 1, Recommendation 1.9) • CIS AWS Foundation v1.4.0 (Level 1, Recommendation 1.10) • CIS AWS Foundation v1.4.0 (Level 1, Recommendation 1.11)

ITEM ID	TITLE	ACTIVE	COMPLETE	CUSTOM TO ORGANIZATION?
T2285	Set up and maintain cloud users and roles (Cloud)	✓	✓	-
T2286	Configure a secure user authentication (Cloud)	✓	✓	-
T2287	Configure a secure user authorization (Cloud)	✓	✓	-
T2289	Secure network access control (Cloud)	✓	✓	-
T2290	Secure data in transit (Cloud)	✓	✓	-
T2291	Secure hosts and operating systems (Cloud)	✓	✓	-
T2292	Protect data at rest (Cloud)	✓	✓	-
T2293	Enable logging and protect log files in your cloud environment (Cloud)	✓	✓	-
T2294	Enable logs and configuration monitoring in your cloud environment (Cloud)	✓	✓	-
T2295	Secure cloud key management system (Cloud)	✓	✓	-

Examples: Respond to Vulnerabilities (RV)



RV.1: Identify and Confirm Vulnerabilities on an Ongoing Basis

RV.1 highlights the importance of threat awareness. It requires organizations to monitor public sources for newly disclosed vulnerabilities and adjust security controls accordingly.

"Help ensure that vulnerabilities are identified more quickly so that they can be remediated more quickly in accordance with risk, reducing the window of opportunity for attackers."

RV.1.1: Gather information from purchasers, consumers, and public sources on potential vulnerabilities in the software and third-party components that the software uses, and investigate all credible reports. Thousands of new vulnerabilities are disclosed publicly each year. Teams should monitor vulnerability mailing lists and other public disclosures to avoid adding simple attack vectors to their code base.

- SD Elements' research team monitors multiple sources to maintain timely and accurate content on vulnerabilities and attack patterns, including CIS Benchmarks and databases for weaknesses and vulnerabilities like CVE and CAPEC. As new vulnerabilities are disclosed (e.g., Log4shell), the research team immediately acts and provides the necessary tasks to mitigate the vulnerability in subsequent releases.

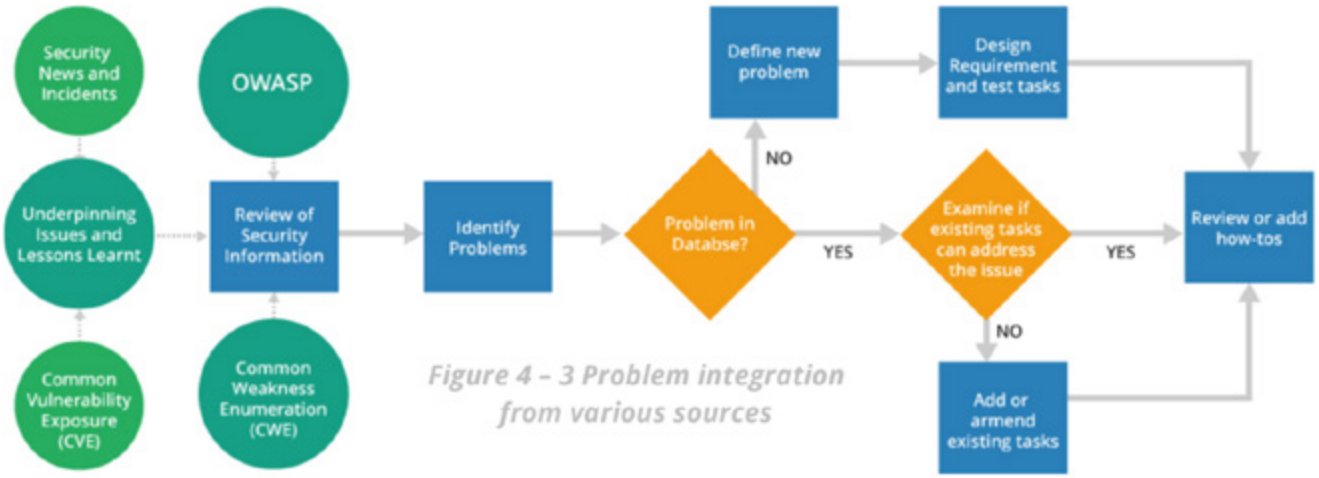


Figure 4 – 3 Problem integration from various sources

RV.2: Assess, Prioritize, and Remediate Vulnerabilities

RV.2 emphasizes the need to prioritize vulnerabilities using a risk-based approach. This includes remediating vulnerabilities, risk mitigation, and risk acceptance.

"Help ensure that vulnerabilities are remediated in accordance with risk to reduce the window of opportunity for attackers."

- **RV.2.2: Plan and implement risk responses for vulnerabilities. Appropriate controls can mitigate the risk of most vulnerabilities.** Having consistent, effective controls improves security and makes software maintenance simpler.

    - SDE identifies and prioritizes vulnerabilities based on the technical stack to inform risk-based decisions (e.g., risk acceptance, risk transference). When a permanent mitigation is unavailable, mitigations are provided to reduce the attack risk.

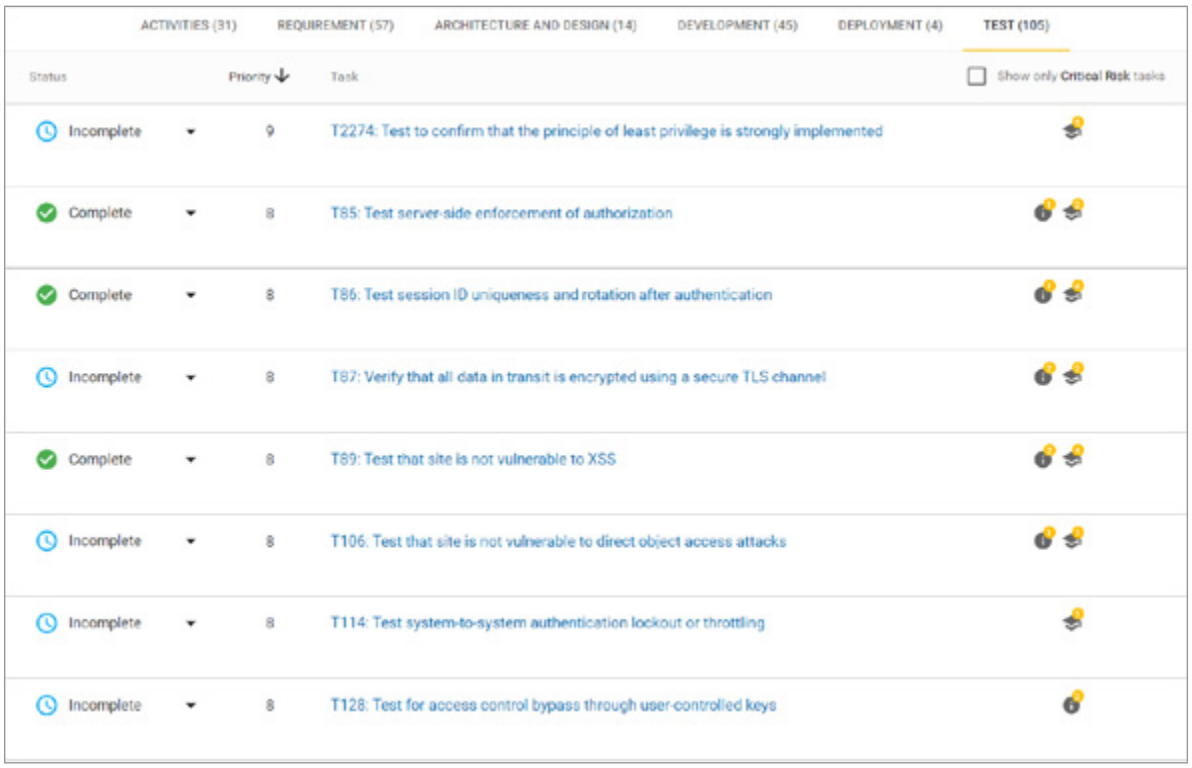
RV.3: Analyze Vulnerabilities to Identify Their Root Causes

RV.3 cites the importance of continuous improvement through observation. By identifying the root cause of vulnerabilities, teams can improve their secure coding skills.

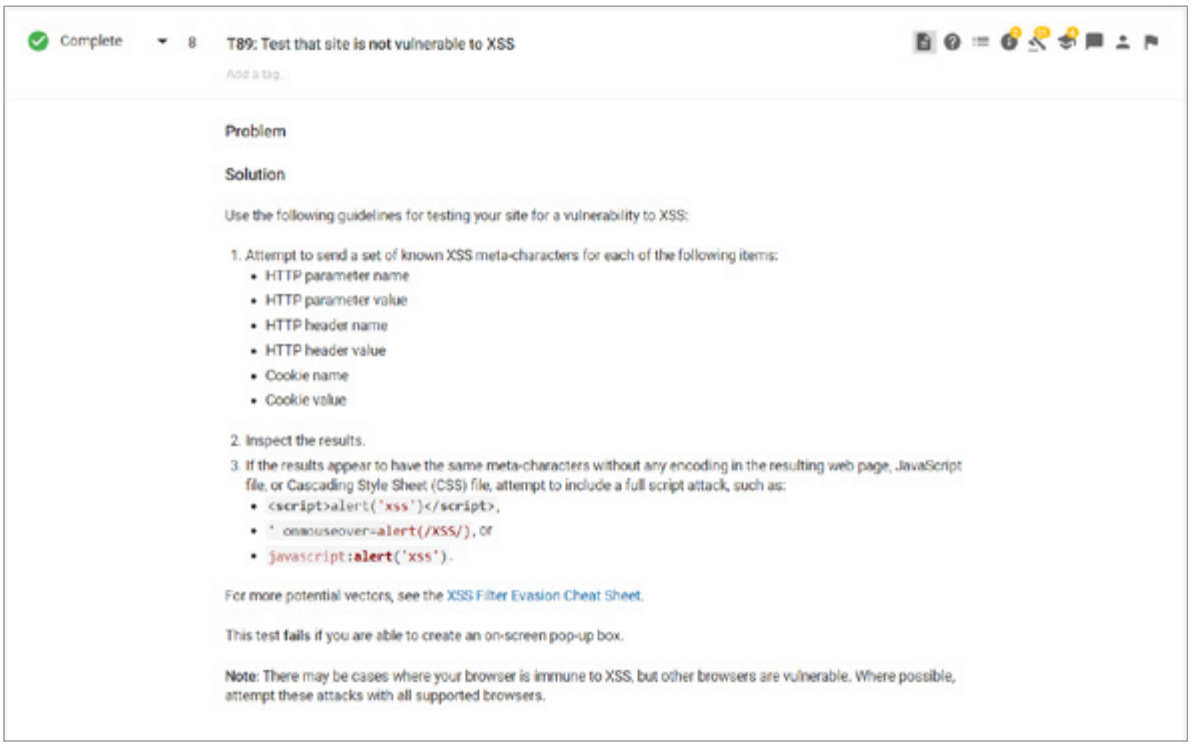
"Help reduce the frequency of vulnerabilities in the future."

- **RV.3.3: Review the software for similar vulnerabilities, and proactively fix them rather than waiting for external reports.** The guidance for RV.3.3 references practices 7 and 8: Test Executable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements.

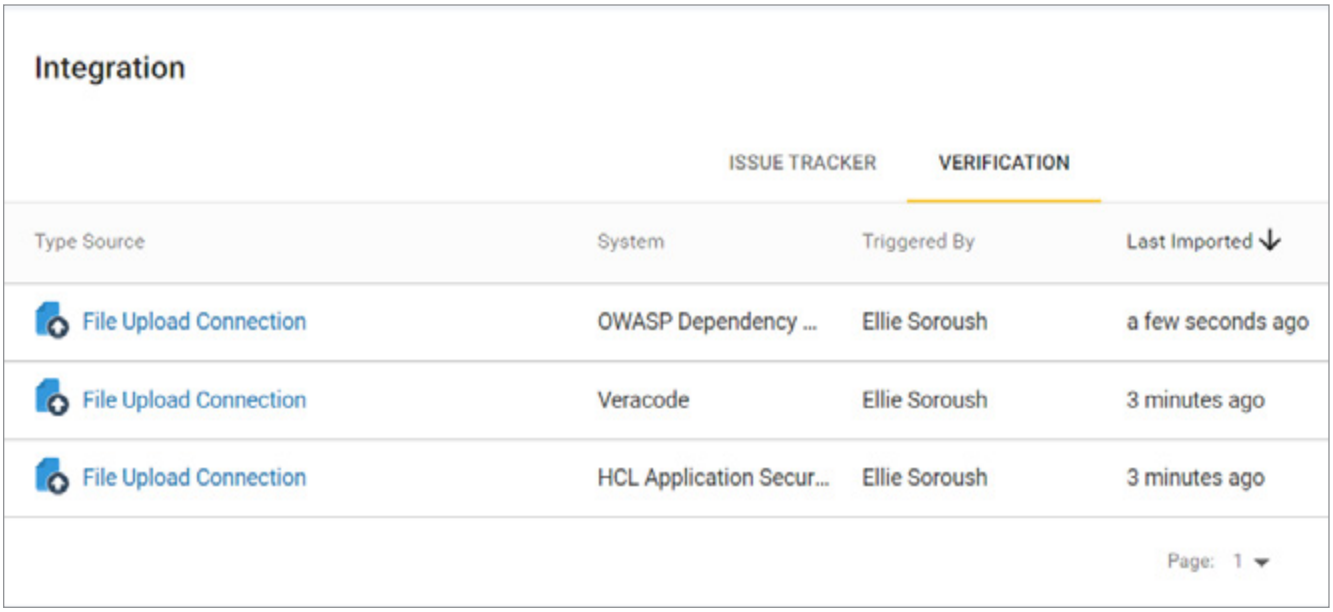
    - SD Elements provides guidelines and instructions for building processes that ensure applications meet security verification requirements. It integrates with security testing tools like SAST, DAST, and SCA to import the results of scanning, verify results, and automatically close some tasks.



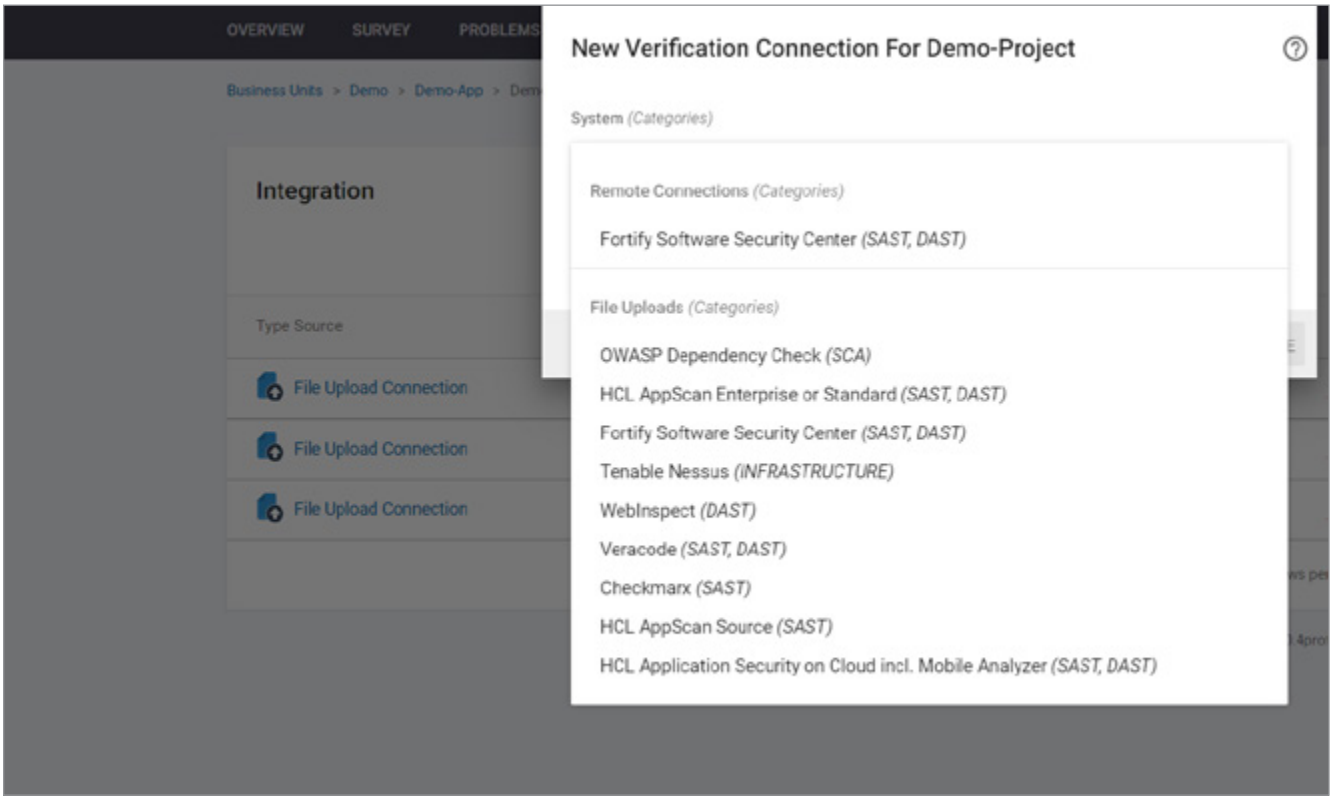
R.V.3.3 (1) is a list of test tasks that "provides guidelines and instructions... that ensure applications meet security verification requirements."



R.V.3.3 (2) is the instruction in one of the test tasks.



R.V.3.3 (3) shows the integration of SDE with three verification tools.



R.V.3.3 (4) shows the list of verification tools that can be integrated with SDE.

Next Steps

EO 14028 applies specifically to organizations providing software to US government agencies and Authority to Operate. As commercial sector demand grows for improved security in software supply chains, it also provides a useful framework for improving software security for any organization building applications.

You can learn more about the EO and how to begin aligning to the best practices by watching our two-part on-demand webinar series:

- Part 1: Executive Order 14028: Guidelines for Enhancing Software Supply Chain Security
- Part 2: Using SD Elements to Comply with US Executive Order 14028 Secure Software Development Recommendations

You can also speak to us about how SD Elements can organizations building software for U.S. federal, state, and local government agencies adhere to SSDF recommendations.

Security Compass

Go Fast. Stay Safe.

Security Compass, a pioneer in application security, enables organizations to shift left and build secure applications by design, integrated directly with existing DevSecOps tools and workflows. Its flagship product, SD Elements, helps organizations accelerate software time to market and reduce cyber risks by taking an automated, developer-centric approach to threat modeling, secure development, and compliance. Security Compass is the trusted solution provider to leading financial and technology organizations, the U.S. Department of Defense, government agencies, and renowned global brands across multiple industries. For more information, please visit www.securitycompass.com.

1.888.777.2211
info@securitycompass.com
www.securitycompass.com

@SECURITYCOMPASS
SECURITY COMPASS

OFFICES

GLOBAL HEADQUARTERS
1 Yonge Street
Suite 1801
Toronto, Ontario
Canada M5E 1W7

TORONTO
390 Queens Quay W
2nd Floor
Toronto, Ontario
Canada M5V 3A6

NEW JERSEY
621 Shrewsbury Avenue
Suite 215
Shrewsbury, New Jersey
USA 07702

CALIFORNIA
600 California Street
San Francisco, California
USA 94108

INDIA
#4.07
4th Floor, Statesman House
Barakhamba Road, New Delhi
India 110001