DevSecOps Guide

Standard DevSecOps Platform Framework

Goal: Safer Software Sooner

The following document describes, at a high-level, the expectations, scope of responsibilities, maturity model, and metrics associated of any DevSecOps platform at GSA.

Baseline Definitions

DevSecOps: A cultural and engineering practice that breaks down barriers and opens collaboration between development, security, and operations organizations using automation to focus on rapid, frequent delivery of secure infrastructure and software to production. It encompasses intake to release of software and manages those flows predictably, transparently, and with minimal human intervention/effort.

Common Expectations

Successful DevSecOps teams have processes characterized by repeatability, low redundancy, high collaboration with dispersion of collective efforts; in order to achieve this most efficiently, automation and auditability is prized above subjective decision-making. The decisions that would drive successful release should be codified in code. If it is not feasible to capture in code, checklists with clear yes/no decision points are preferred to heavily documented standard operating procedures (SOPs). SOPs can be subjectively interpreted more so than these first options.

This does not mean that there aren’t scenarios in which that process can be disrupted and replaced by subjective decision-making; but those scenarios should be rare and driven by extreme conditions rather than the norm.

All of the components described below are going to imply the necessity for some foundational elements; for example, infrastructure-as-code, source control, automation, clear communication pipelines, and many others. Individual platforms may implement these differently, but we will see those common elements emerge as designed.

How to Use This Document

This document is not a framework describing any specific implementation. It describes the requirements that need to be met by any specific implementation before it can be considered a Standard GSA DevSecOps Platform. It should be used by owners of platforms in conjunction with the CTO, Deputy CIO, and CISO to define an implementation of the requirements described in this framework. It should be used by application developers to understand and find platform implementations. This framework is set alongside a template that captures the requirements for any platform implementation.

A platform can be anything from an IaaS-driven pipeline of software delivery to a PaaS to a SaaS-driven application deployment scheme. Applications are deployed on platforms and provide services to our users. Applications are accepted onto platforms via an intake process. In GSA, that could mean that our delivery of applications on Salesforce can (and should) align to the framework described below.

When a DevSecOps platform meets a certain level of maturity, it qualifies for a streamlined delivery and ATO process.

The DevSecOps platform standard is categorized into domains of responsibility. Each domain of responsibility has four components:

Description: A short description of the scope of that domain.

Maturity Model: A set of differing levels of maturity within that domain and their differentiators. In many cases, being below a certain level of maturity would preclude the platform from being considered a Standard DevSecOps Platform.

Metrics: Key metrics that any platform implementing the framework should be collecting (note: given GSA’s current level of maturity in DevSecOps, these metrics may not be immediately feasible to collect, but any platform owner should have a plan to begin that collection at the outset).

Artifacts: A list of artifacts (or artifact types) that should be provided by the platform owner to describe the implementation. The description, as mentioned above, is preferred to be executable, documented code. Platform owners should work expressly to keep textual artifacts minimal.

Each platform will assign responsibilities at the domain level and then the artifact level to ensure that individuals and organizations have clear understanding of who owns what.

Platform Domains and Responsibilities

Overarching DevSecOps Platform Considerations

Description

This domain encompasses the holistic nature of DevSecOps around the platform itself, capturing the flow of work into the environment and release of software out of it.

Maturity Model

Level 1 (Not considered viable for a DevSecOps platform): The platform is characterized by manual efforts, is not transparent about state, is not standardized across teams, and is heterogeneously configured on a per-project basis.

Level 2: Application developers have a pipeline that they can use to deploy software which is considerate of security and visible to operations. Intake into the platform may be manual or unpredictable. Steps to deploy or maintain operations may require manual effort or assessment.

Level 3: Application developers have a clear, self-service intake onto the platform and the ability to deploy and run security-compliant code in production through automation. The platform services are centralized in its infrastructure and pipeline implementation.

Metrics

There are countless possible metrics in a DevSecOps platform. The decision of which metrics to track is largely based on business need and compliance requirements. This framework labels individual metrics as “High-Value” or “Supporting”. High-Value metrics are those that provide the most critical insight into the performance of a DevSecOps platform, and should be prioritized for implementation. Supporting metrics are those that a team may find useful to improve their DevSecOps platform.

High-Value Metrics

These metrics should be implemented first in a new platform. Not all platforms will have these metrics immediately available, but a fully mature environment typically will have all of these metrics.

Metric Description Associated Domain(s)

Deployment frequency Number of deployments to production in a given time frame Application Deployment; Authority to Operate Processes

Change lead time (for applications) Time between a code commit and production deployment of that code Overarching; Authority to Operate Processes; Patch Management

Change volume (for applications) Number of user stories deployed in a given time frame Overarching

Change failure rate Percentage of production deployments that failed Application Deployment

Mean time to recovery (MTTR) (for applications) Time between a failed production deployment to full restoration of production operations Application Deployment; Backup and Data Lifecycle Management; Patch Management

Availability Amount of uptime/downtime in a given time period, in accordance with the SLA Availability and Performance Management; Network Management

Customer issue volume Number of issues reported by customers in a given time period Overarching

Customer issue resolution time Mean time to resolve a customer-reported issue Overarching

Time to value Time between a feature request (user story creation) and realization of business value from that feature Overarching; Authority to Operate Processes

Time to ATO Time between the beginning of Sprint 0 to achieving an ATO Overarching; Authority to Operate Processes

Time to patch vulnerabilities Time between identification of a vulnerability in the platform or application and successful production deployment of a patch Authority to Operate Processes

Supporting Metrics

These metrics provide useful information and it is recommended that metrics in this category be implemented after the High-Value metrics have been implemented.

Metric Description Associated Domain(s)

Test coverage Percentage of code that is covered by automated tests Application Development, Testing, and Operations

Change types Percentage of features vs fixes vs security patches Change Management; Patch Management

Time to availability of event information Time from an event to information about the event being available to the DevSecOps team or end users Logging, Monitoring, and Alerting

Developer onboarding Time from a developer joining the team to ability to commit code for production deployment Application Development, Testing, and Operations

Change resolution time Time between a change proposal and closing (implementation or rejection) Platform Governance

Change lead time (for the DevSecOps platform) Time between a change (e.g., code commit) and platform deployment of that change Change Management; Patch Management; Network Management; Platform Governance

Change volume (for the DevSecOps platform) Number of user stories deployed in a given time frame Change Management; Platform Governance

Change failure rate (for the DevSecOps platform) Percentage of platform deployments that failed Change Management; Platform Governance

Mean time to recovery (MTTR) (for the DevSecOps platform) Time from a failed platform deployment to full restoration of platform operations Change Management; Patch Management; Platform Governance

Change lead time for images Time from identification of need for a new/updated image to its availability for production use Image Management

Image publishing frequency Number of new/updated images published in a given time frame Image Management

Logging availability Amount of uptime/downtime of the logging system in a given time period Logging, Monitoring, and Alerting

Number of monitoring alerts Amount of monitoring alerts triggered in a given time period Logging, Monitoring, and Alerting

Number of unit/integration tests Number of automated unit or integration tests for an application Application Development, Testing, and Operations

Number of functional/acceptance tests Number of automated functional or acceptance tests for an application Application Development, Testing, and Operations

Mean recovery point Mean time range of data that is lost due to an incident Backup and Data Lifecycle Management

Retention control compliance Percentage of retention-related controls (e.g., AU-11, SI-12) that are automated Backup and Data Lifecycle Management; Authority to Operate Processes

Image instantiation Time between when a developer requests image instantiation and its actual instantiation Image Management

Security benchmark deviation Deviation between security benchmarks applied to an image and security benchmarks on an instantiated image Image Management, Authority to Operate Processes

Technical controls Number of technical security controls partially or fully in place Authority to Operate Processes

Vulnerability patching frequency How often vulnerability patches are regularly deployed to production Authority to Operate Processes

Vulnerability patching lead time Time between discovery of a new vulnerability (i.e., its publication) and patching in production Authority to Operate Processes

A&A lead time Time between initiating a security compliance assessment and completion of A&A processes Authority to Operate Processes

SAR findings count Number of findings in the Security Assessment Report (SAR) Authority to Operate Processes

POA&M count Number of POA&Ms Authority to Operate Processes

New architecture security review lead time Time between initiating a request for security review of a new architecture, and completion Authority to Operate Processes

Experiments and alternatives Number of technological experiments and alternative components tested Overarching

System logging count Number of systems (application, OS, services) in a DevSecOps platform that are logging Logging, Monitoring, and Alerting

AU security control compliance List and percentage of AU security controls that are satisfied via DevSecOps platform logging practices Logging, Monitoring, and Alerting

CM approval lead time Time between submitting a change request and its approval Change Management

Deployment approval lead time Time between requesting deployment of an approved change and the actual deployment to production Change Management

Notification lead time Time between any given step of the CM process and notification of all interested parties Change Management

User provisioning lead time Time between request for a new user on the platform and the user being able to log in Accounts, Privileges, Credentials, and Secrets Management

AC security control compliance List and percentage of AC security controls that are satisfied via DevSecOps platform account management practices Accounts, Privileges, Credentials, and Secrets Management

Privilege auditing frequency Number of times in a given time period that users and their privileges are audited Accounts, Privileges, Credentials, and Secrets Management

Administrator count List and number of users that have administrator-level privileges Accounts, Privileges, Credentials, and Secrets Management

Secret rotation frequency Number of times in a given time period that secrets are changed and updated where affected Accounts, Privileges, Credentials, and Secrets Management

Onboarding lead time Time between a request for a new application to use the DevSecOps platform and the application being deployed on the platform Agreements and Financial Management

Expense lead time Time between an expense and reporting of the expense Agreements and Financial Management

Artifacts

Platform description

Platform responsibilities allocation

Image Management

Description

Note: This is relevant to IaaS and PaaS capabilities, not necessarily SaaS-driven capabilities.

An image in the context of this framework is the definition of a component of computing infrastructure that can be instantiated for use by the platform or by application owners on that platform. Concretely, an image could be a VM image, AMI, a container image or definition, or similar products. Image management refers to lifecycle around the creation, maintenance, and delivery of those images to application developers.

Maturity Model

Level 1 (Not considered viable for DevSecOps): Application developers must make and secure all of their images from scratch or nearly so (e.g., they use their own AMIs which can vary from team to team).

Level 2: Application developers are provided base OS images; the owner of the image management process hardens those OS images, and includes any necessary agents. The application team cannot change the underlying hardened image, except to add application code and components.

Level 3: Applications developers are provided base OS images and images that provide component-level functionality that has also been hardened (e.g., standard images pre-packaged with hardened components i.e. databases or web/application servers). Images and components undergo automatic testing and are pre-approved by security and operations groups.

Artifacts

Repository of images (preferably a link to an open source repository to create image)

Process (code, checklist, and/or SOP) for adding a new image into the repository - the process should cover developer request, addressing of security considerations, and delivery to the platform

Process (code, checklist, and/or SOP) for instantiating an image in an application

Logging, Monitoring, and Alerting

Description

Logging, monitoring and alerting covers the domain of understanding and managing the health and security of an application’s operational state. This includes capturing what events have occurred (logging), providing information about those events (monitoring) and informing the appropriate parties when those events indicate issues to be resolved (alerting). Application teams need significant autonomy to manage the health of their own applications, but the enterprise at large also needs awareness of the health of applications within it.

Maturity Model

Level 1 (Not considered viable for a DevSecOps platform): Event capture is not existent, not granular, or not available to either application developers or platform owners

Level 2: Platform owners have awareness of platform health and perhaps health of individual applications. Application teams must create their own methods of health management, perhaps with some guidance from the platform owners.

Level 3: Application owners have full access to their application event information with monitoring and alerting flexibility for their own use. An enterprise-wide application logging and monitoring system is available.

Artifacts

Guide (code and/or document) to application owner access to logging, monitoring, and alerting services; use of the guide should suffice for an application owner to configure and manage their logs, monitoring, and alerts. The guide should also cover logging configuration for centralized security monitoring by SecOps.

Patch management

Is the process by which the operating system, software, and supporting services are upgraded. This is a key element of maintaining the security of systems.

Maturity Model

Level 1 (Not considered viable for a DevSecOps platform): Patching is both manual and is not enforceable as a requirement nor is the state of patches running machines discoverable

Level 2: Security teams inform the application owners of patches, and it is application owners’ responsibility to both be aware of those patches and implement them.

Level 3: Application owners are automatically notified when there are new security-related patches, and the platform owners are able to identify which applications are using which version of the relevant software via the platform.

Level 4: The platform automatically tests new patches on applications which run on it, informing the appropriate parties if decision points are reached (e.g., if a CVE is raised on an existing piece of software, the platform can automatically update that software, test it, and inform the application developers of the change if the tests pass or indicate that the patch needs to be applied in a particular timeframe). No downtime for patching.

Artifacts

Process (code, checklist or SOP) for patching a running system

Process (code, checklist or SOP) for introducing a patch into the platform [Note: This may overlap with image management]

Platform Governance

Description

Platform governance consists of the processes around and advertisement of changes to the platform, inclusive of managing the security and availability of the platform.

Maturity Model

Level 1 (Not considered viable for a DevSecOps platform): Changes are conducted ad hoc, without transparency, and unadvertised to users of the platform.

Level 2: Platform change is conducted through defined processes with significant manual components required to conduct, especially relying on consensus over objective criteria for making change

Level 3: Platform change is conducted through strictly defined processes with clear criteria defined that allow for rapid change; the platform automates changes and endeavors to impact the minimum number of application developers through that automation.

Artifacts

Change proposal intake process (intake form, process description, or SOP)

Change proposal evaluation process (process description or SOP)

Change proposal execution process (code, process description, or SOP)

A forum for communication between the platform owner and application users

Change Management

Description

Change management consists of all the standards and norms around version control of applications and the platforms itself.

Maturity Model

Level 1 (Not considered viable for a DevSecOps platform): No version control or standards related to version control

Level 2: Application teams are required to have version control, but there is no standard method of using version control or version control is entirely distinct from application management on the platform.

Level 3: Version control is a key method of managing application lifecycle, has well-defined standards for use such that any user of the platform has a baseline that is shared across all applications.

Artifacts

Version control repository (open source, unless there’s a good reason for it not to be, which is rare)

Version control standards for branching, merging, etc.

Application Development, Testing, and Operations

Description

These areas encompass the development of software by an application team, the unit and integration testing of that software, and the ability to manage that software in operation.

Maturity Model

Level 1 (Not considered viable for a DevSecOps platform): Application development has a poor user experience in which developers and operators of a system are strictly separated, development environments and operational environments differ, and manual processes to manage and release software are required. Testing is not implemented or required as part of release. Operators can make direct changes to the running system that may not be repeatable.

Level 2: Application teams have a set of tools that are provided to them that allow them to develop and test software. The development and operational environment may differ. Operators make changes to the system that can be scripted or manual, but all are documented.

Level 3: Development and operational environments are identical and immutable. Environments can be stood up and torn down via automation. All changes to the running system are logged and broadly conducted through scripting rather than actual access to the running system. All necessary tests, including security tests, are run as part of the deployment process. Development environments may be instantiated and torn down as needed.

Artifacts

Developer environment (code or image) that is usable by developers which aligns them as much as feasible to the operational environment, but does not require connecting to a database with real data

Operational procedures for updating running system

Testing tools (code) usable by developers for different project types

Testing standards best practices for the platform

Application Deployment

Description

Application deployment consists of the processes by which an application in development reaches production, most likely going through multiple environments to evaluate the correctness of deployment. Deployed products must be compliant with the relevant security and infrastructure considerations. Deployment should be performed with no downtime.

Maturity Model

Level 1 (Not considered viable for a DevSecOps platform): Deployment is manual, requires significant collaboration and consensus to release, and does not leverage any shared artifacts to facilitate compliant deployment. Getting an ATO and first deployment are independent processes, meaning assessments happen after development rather than during.

Level 2: Deployment has minimal manual steps and uses shared artifacts to facilitate deployment and compliance. ATO processes are independent rather than embedded in deployment.

Level 3: The only manual steps to deployment are those explicitly designed to meet application expectations (e.g., not every push to the master branch necessarily indicates a release, but a product that could be released, if there is a business reason to not automatically update).

Artifacts

Deployment pipeline (link to running pipeline, code for launching a pipeline, checklist for deployment and/or SOP)

Deployment playbook (code, checklist, or SOP)

Accounts, Privileges, Credentials, and Secrets Management

Description

This domain covers broadly the ability for user accounts to be created, given permissions to access, create, and destroy resources, and share secrets securely between the platform and the application as appropriate. This can, for example, include both the creation of user accounts but also the sharing of a database username and password between a provisioned database and the application using it. Maturity Model

Maturity Model

Level 1 (Not considered viable for a DevSecOps platform): User management is done manually and secrets are shared via person-to-person or person-to-platform information exchange without necessary consideration for least privilege access to those secrets. Secrets are hard-coded into configuration files or code in plain text.

Level 2: User management is done through automation with a fixed set of user types. Secrets are stored/passed through secure methods. Secrets are not required to be accessed by people in order to be useful, but could still be seen by people if necessary.

Level 3: User management is self-service with appropriate security limitations. Secrets are created/shared between parts of the platform, without people needing to set/interact with them.

Artifacts

User onboarding and offboarding guides

IAM definitions (code and/or documentation)

Secret management practices (tools and/or documentation)

Availability and Performance Management

Description

Availability and performance management covers the processes that allow application owners to be assured that the applications will be available, potentially in the face of disaster, and be responsive to user interactions. In order to achieve those goals, the application may deploy redundant capabilities, deploy across different hardware instances, or deploy into multiple regions. Further, application owners may need to manage specific performance characteristics of their applications.

Maturity Model

Level 1 (Not considered viable for a DevSecOps platform): The platform has no explicitly defined availability for itself. Application owners have little or no insight into the performance and health of their applications. Application owners have no tooling to support automated availability management and potentially no way to support true high availability requirements.

Level 2: The platform itself has defined availability expectation that is advertised to its users. Tenants are notified when there are downtime events, updates to those events, and resolution with port-mortems. Application owners have insight into application health and performance through tooling, but have to design their own architectures to support high availability.

Level 3: The platform manages availability for the application owners through automation based on application need. The platform provides direct insight into application health and performance. Applications can be seamlessly moved between hosting regions/zones in reaction to DR or threat activity.

Artifacts

Platform availability metrics

[If appropriate] Guide to configuring availability for applications

Method of accessing performance information

Network Management

Description

Network management consists of the creation, maintenance and management of the network structures (such as ingress and egress points, virtual private cloud construction) on which the platform resides and in which the applications are launched. Maturity Model

Level 1 (Not considered viable for a DevSecOps platform): Each application defines and manages its network structures, inclusive of security assessments.

Level 2: The platform governs the overarching network infrastructure supporting the applications, with defined and assessed separation of network concerns. Application deployment and development requires frequent update of the network configuration.

Level 3: The platform governs the overarching infrastructure supporting the applications, with defined and assess separation of network concerns. Application owners can make limited changes to their network environment sufficient to self-manage the deployment of their applications and creation of new components of their application, on their own, with appropriate compliance checks.

Artifacts

Network structure definition, ideally as code

Request process for network changes

Authority to Operate Processes

Description

The authority to operate (ATO) is the authority given by an authorizing official after assessment by the Chief Information Security Officer (CISO) that a system can “go live” with government data. It takes into consideration the holistic security posture of the application. Traditionally, ATO processes have come at the end of application development, but a DevSecOps environment requires that ATOs are achieved concurrently with development. Hence, the most mature environments will equate deployment with successful receipt of an ATO as the platform itself provides significant security assurances.

Maturity Model

Level 1 (Not considered viable for a DevSecOps platform): The ATO processes are distinct from the ability to deploy code. System security plans, assessments, and other artifacts are constructed manually. Existing security-approved code and process isn’t adopted.

Level 2: ATO processes are aligned with sprints, meaning partially developed systems can be launched into production. The deployment pipeline will provide notification if manual ATO processes need to be invoked. The first ATO may take longer than subsequent ATOs.

Level 3: ATO processes are highly automated. Compliant code and process is reused by multiple teams. All ATOs take the same amount of time for the same system and frequent deployment is only interrupted when specific risk triggers are raised through automation. Controls can be continuously monitored and measured with automation.

Artifacts

Process for achieving an ATO (code, checklist, or SOP) - this process should also dictate the constraints on the application which will break this process and turn it back into a traditional ATO

Templates for ATO artifacts with appropriate sections already filled in (code [e.g., OpenControl] or document template)

Backup and Data Lifecycle Management

Description

Backup and data lifecycle management allows application developers to ensure that their data is maintained over time and, in the case of failure of any subsystems, that it can be recovered with potentially some gap in transactional data. Lifecycle management of the data includes capabilities to archive and manage data over a long lifetime.

Maturity Model

Level 1: Manual management of data with little or no tooling provided by the platform directly.

Level 2: Backups are managed through automation with little intervention required from the application owner.

Level 3: The full data lifecycle is owned by the platform through automation.

Artifacts

Documentation on use of backup and data lifecycle management

Template policies or documentation for data retention - for example:

Log retention policies

Deployed image snapshot retention scripts/policies

Agreements and Financial Management

Description

Agreements and financial management consists of all the processes which allow new customers of the platform to come onboard through agreements (intake), allowing funding to be allocated by the customers for use, and tracking expenditures over time for both the platform owner and application owners.

Maturity Model

Level 1 (Not considered viable for a DevSecOps platform): No clear cost model, tracking of expenditures is manual. Onboarding a new customer is a laborious process.

Level 2: The cost model is defined and consistent and application owners get regular reports on expenditures to ensure they are within budget.

Level 3: Onboarding is largely self-service (within appropriate legal limits), and application owners have full access to their expenditures at any time. Application owners can set triggers on expenditures to manage their costs appropriately

Artifacts

Canonical documentation (ideally, a public web page) explaining the pricing and onboarding process

Links to spending dashboards

Process for onboarding

Process for allocating budget