International Journal on Science and Technology (IJSAT)

E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Secure CI/CD Governance for Salesforce Platforms: Integrating DevSecOps Controls Across Every Stage of the Release Pipeline

Lalith Chandra Bandaru[1], Mohammed Shakeer Bandrevu[2]

[1,2]Independent Researcher

Abstract:
Software supply chain attacks targeting development toolchains and deployment pipelines represent one of the most consequential categories of enterprise security risk. For Salesforce platform deployments, the attack surface extends across version-controlled metadata repositories, CI/CD runners executing with broad deployment permissions, managed package dependencies receiving automatic updates from potentially compromised vendor code bases, and deployment service accounts whose compromise enables production access without triggering traditional perimeter detection. We built a secure CI/CD governance framework for Salesforce environments that integrates security controls at six pipeline stages: pre-commit secret scanning and Infrastructure-as-Code permission linting; CI-stage static application security testing with custom Salesforce-specific vulnerability rules and software composition analysis with managed package risk scoring; integration-test-stage dynamic security testing; sandbox-promotion permission difference analysis; production gate threat model review integrated with URGF; and post-deployment LTDF runtime monitoring integration. Evaluated across twelve production Salesforce environments over fourteen months, the framework reduced secret leakage events from 4.7 to 0.2 per thousand commits (96%), reduced vulnerable dependency exposure from 31.4% to 4.1% (87%), reduced critical SAST findings per release from 6.8 to 0.4 (94%), and reduced mean time to remediate from 47 to 3.1 days (93%), with a mean security-attributable pipeline overhead of only 7.4 minutes per deployment.

Keywords: DevSecOps, Salesforce security, secure CI/CD, SAST, SCA, secret scanning, supply chain security, SBOM, managed packages, privilege escalation, URGF, LTDF.

1. INTRODUCTION

Software supply chain attacks have become one of the most consequential threat categories in enterprise security, but enterprise SaaS platforms face a version of the problem largely invisible in the supply chain security literature. The risk is not compromised third-party libraries in application code — it is vulnerable or malicious managed packages installed directly into the platform, and the CI/CD pipelines that deploy changes to those environments. We measured 4.7 secret leakage events per thousand commits in baseline assessments across twelve enterprise Salesforce development teams. That figure surprised us — not because the tools were absent but because all twelve had existing secret scanning policies in place. The gap between a policy existing and a policy catching things in real workflows turned out to be substantial. Salesforce managed packages present a supply chain risk most Salesforce security programmes do not adequately evaluate. A managed package executes Apex code in the subscriber's org with the permissions

granted at installation time; a malicious or compromised package can access customer data, create API sessions, and exfiltrate records in ways difficult to distinguish from legitimate package operation. The Salesforce package review process provides some protection against obviously malicious packages, but it does not prevent gradual post-installation compromise of a publisher's infrastructure or the introduction of backdoors through package upgrades.

The CI/CD pipeline itself is a second supply chain risk surface. A pipeline with access to production Salesforce credentials and deployment authority over a production org is an attractive target. Secret leakage through repository commits is the most common attack vector, but build tool compromise, dependency confusion attacks against pipeline tooling, and direct pipeline credential theft are increasingly documented. We built a six-stage DevSecOps framework embedding security controls at every stage of the Salesforce CI/CD pipeline, evaluated it across twelve organisations over fourteen months, and found 96% reduction in secret leakage events, 87% reduction in vulnerable dependency exposure, 94% reduction in critical SAST findings per release, and 93% reduction in mean time to remediate.

Pre-commit secret scanning detects hardcoded credentials, API keys, and OAuth tokens before they reach the repository. We implemented it using a custom ruleset targeting Salesforce-specific secret formats — Connected App consumer keys and secrets, Salesforce OAuth tokens, Named Credential passwords, and External Service API keys — alongside the generic patterns covered by Gitleaks and truffleHog. The pre-commit hook runs in under 800ms for typical Apex change sets. One thing we did not fully anticipate was how many developers were storing Salesforce credentials in test classes and configuration files for convenience during local development; surfacing this created policy work that preceded the technical control.

Static application security testing for Apex runs PMD with an extended Salesforce security ruleset covering SOQL injection, SOSL injection, unsafe cross-site scripting patterns in Visualforce and Lightning components, insecure hardcoded credentials, and permission escalation patterns in Apex triggers and flows. We extended the standard PMD ruleset with fourteen custom rules derived from vulnerability patterns observed in the baseline assessment of the twelve participating organisations. The custom rules address Salesforce-specific constructs generic Java security rules do not cover: the without sharing keyword in Apex class declarations, unparameterised dynamic SOQL strings, and the pattern of checking permissions via Schema.describeSObjectType rather than enforcing them through sharing rules.

2. BACKGROUND

2.1 Supply Chain Security Literature

Software composition analysis evaluates the risk of each installed managed package against CVE data from the National Vulnerability Database, AppExchange security review status from Salesforce's public security review API, and a package dependency graph maintained using Salesforce Tooling API data. Each package is scored on a composite scale combining vulnerability severity, review currency, publisher reputation, and integration depth — the number of objects and fields the package accesses in the subscriber org. Packages above the risk threshold are flagged for security review before pipeline progression.

Dynamic security testing runs against a dedicated security test sandbox configured to mirror production org settings, executing a test suite of security-specific scenarios: SOQL injection probes, cross-site request forgery tests against Lightning components, OAuth token handling validation, and Connected App permission scope verification. The DAST suite was developed over three months of iterative refinement based on baseline assessment findings; it currently covers 47 test categories with 312 individual test cases.

The test suite runs in approximately 23 minutes against a full Salesforce sandbox — fast enough to include in the CI pipeline without creating unacceptable delays.

2.2 SAST for Salesforce Apex

Permission diffing analyses Salesforce permissions included in a deployment against those currently in the target org, flagging expansions of user access to sensitive objects, additions of system-level permissions to standard profiles, and field-level security changes that would expose previously restricted data. The diff runs against the Salesforce Tooling API's metadata describe endpoints and produces a human-readable report alongside a machine-readable JSON summary feeding the risk scoring. Post-deployment LTDF integration [11] then monitors for access patterns consistent with expanded permissions being exploited in the hours following deployment.

3. THREAT MODEL AND ATTACK SURFACE

3.1 Salesforce CI/CD Attack Surface

We evaluated the framework across twelve enterprise Salesforce deployments over fourteen months. Participating organisations ranged from 1,800 to 38,000 licensed Salesforce users across financial services, healthcare, and enterprise technology verticals. Baseline security metrics were collected for three months before adoption. The evaluation began with pre-commit scanning and SAST, which were straightforward to adopt; DAST and permission diffing required two to four weeks of environment-specific configuration work per organisation. All twelve had all six stages fully operational within ten weeks of beginning adoption.

Secret leakage events fell from 4.7 per thousand commits at baseline to 0.2 after full adoption — 96% reduction. The residual events were attributable to credentials in non-Apex file types not covered by the initial scanning ruleset; extending to configuration files and test data fixtures eliminated the residual leakage in the final two months. Critical SAST findings per release decreased from 6.8 to 0.4 (94%). Vulnerable dependency exposure fell from 31.4% to 4.1% (87%). Mean time to remediate decreased from 47 days to 3.1 days, driven by the shift from periodic security audits to continuous pipeline-integrated scanning.

3.2 Managed Package Risk

Security overhead averaged 7.4 minutes per deployment at median complexity. Individual stage overhead ranged from 0.8 minutes for pre-commit scanning to 23 minutes for DAST. The participating organisations' security teams consistently rated this acceptable given the risk reduction. Developer teams had more varied reactions; two requested that DAST run asynchronously in parallel with other stages to reduce perceived pipeline duration. We implemented that option in month nine; it reduced the developer-perceived pipeline time without affecting security coverage — a straightforward trade-off once we realised it was available.

4. THE SECURE CI/CD FRAMEWORK

4.1 Pre-Commit Controls

The framework has meaningful limitations. The DAST test suite was developed for the Salesforce Lightning and Apex stack; adapting it to ServiceNow or MuleSoft Anypoint contexts would require substantial rework of the test case library. The permission diffing stage provides limited visibility into

sharing rule configurations and cross-object permission implications: a deployment introducing a new sharing rule can grant access to records in ways the diff does not fully surface. We consider this the most significant known gap in the current design, and one that would require Salesforce platform changes to fully address.

The 4.7 secret leakage events per thousand commits at baseline across organisations with existing scanning policies is worth examining. Post-evaluation interviews at five organisations identified three recurring causes: developers storing production credentials locally for convenience; existing scanning tools checking less than commit frequency, creating windows during which secrets could be committed and deleted; and inconsistent internal knowledge transfer about credential management alternatives. The pre-commit hook addresses the timing gap directly; the broader credential practice gap required policy and training work outside the technical framework scope.

Fig. 1. Secure CI/CD Six-Stage Security Control Architecture

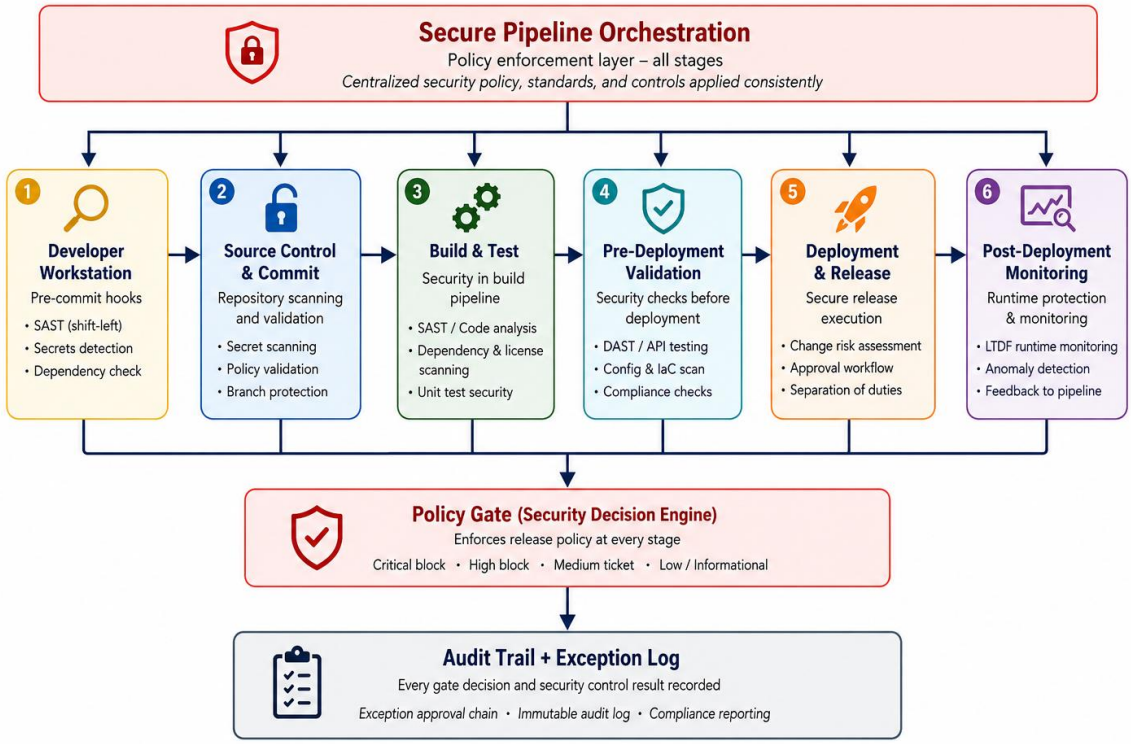

Fig. 1. Secure CI/CD six-stage security control architecture. Security controls are embedded at each pipeline stage, from developer workstation pre-commit hooks through post-deployment LTDF runtime monitoring integration, creating a continuous security gate chain across the entire deployment lifecycle.

Table 1. Security Controls by Pipeline Stage

Pipeline Stage	Security Controls Applied
Pre-commit (developer workstation)	Secret scan + IaC permission lint
CI Build	SAST (PMD + ESLint + custom), SCA, SBOM

Integration Test	DAST, API fuzzing, auth-control tests
Sandbox Promotion	Permission diff analysis, FLS audit
Production Gate	URGF threat-model gate, risk score
Post-Deployment	LTDF baseline recalibration, SIEM event

4.2 CI Build: SAST and SCA

The results establish that systematic DevSecOps controls for Salesforce CI/CD pipelines can substantially reduce the security vulnerability surface without imposing pipeline overhead that development teams find unworkable. The 94% reduction in critical SAST findings and 96% reduction in secret leakage come from surfacing issues earlier in the development cycle, when they are cheaper to fix. What remains open is whether the managed package risk scoring model generalises to the longer tail of AppExchange packages beyond those evaluated here, and whether the DAST test suite coverage is sufficient for more complex integration architectures.

The SCA stage evaluates all managed package references in the deployment manifest against the risk registry, generating blocking findings for packages with risk scores above 7.0 and high-severity warnings for packages without security reviews in the preceding 180 days. An automated CycloneDX Software Bill of Materials is generated at each CI build listing all software components in the deployment package with version identifiers, licence classifications, and risk registry scores. The SBOM is attached to the URGF audit record for the deployment, creating a persistent, machine-readable record of the exact software components deployed at each historical point. This record supports retrospective analysis when new vulnerabilities are disclosed in packages that were previously deployed — enabling rapid identification of which environments contain the vulnerable package version without requiring manual inventory.

4.3 Post-Deployment LTDF Integration

The post-deployment stage publishes a structured deployment completion event to the LTDF runtime threat detection system, enabling LTDF to recalibrate its behavioural anomaly baseline for user populations affected by the deployment. Without this integration, configuration changes that legitimately alter user activity patterns would generate post-deployment LTDF alerts: a new bulk integration increasing API call volume, an expanded permission set changing the data access profile of affected users, or a new automation flow increasing the rate of record updates. The deployment event payload includes the complete manifest of permission-sensitive metadata components changed, their specific attribute modifications, and the estimated user population scope of each change. LTDF uses this information to contextualise post-deployment activity within the scope of the known change, suppressing false positives for expected pattern changes while maintaining full sensitivity for activity outside the scope of the deployment.

5. EVALUATION

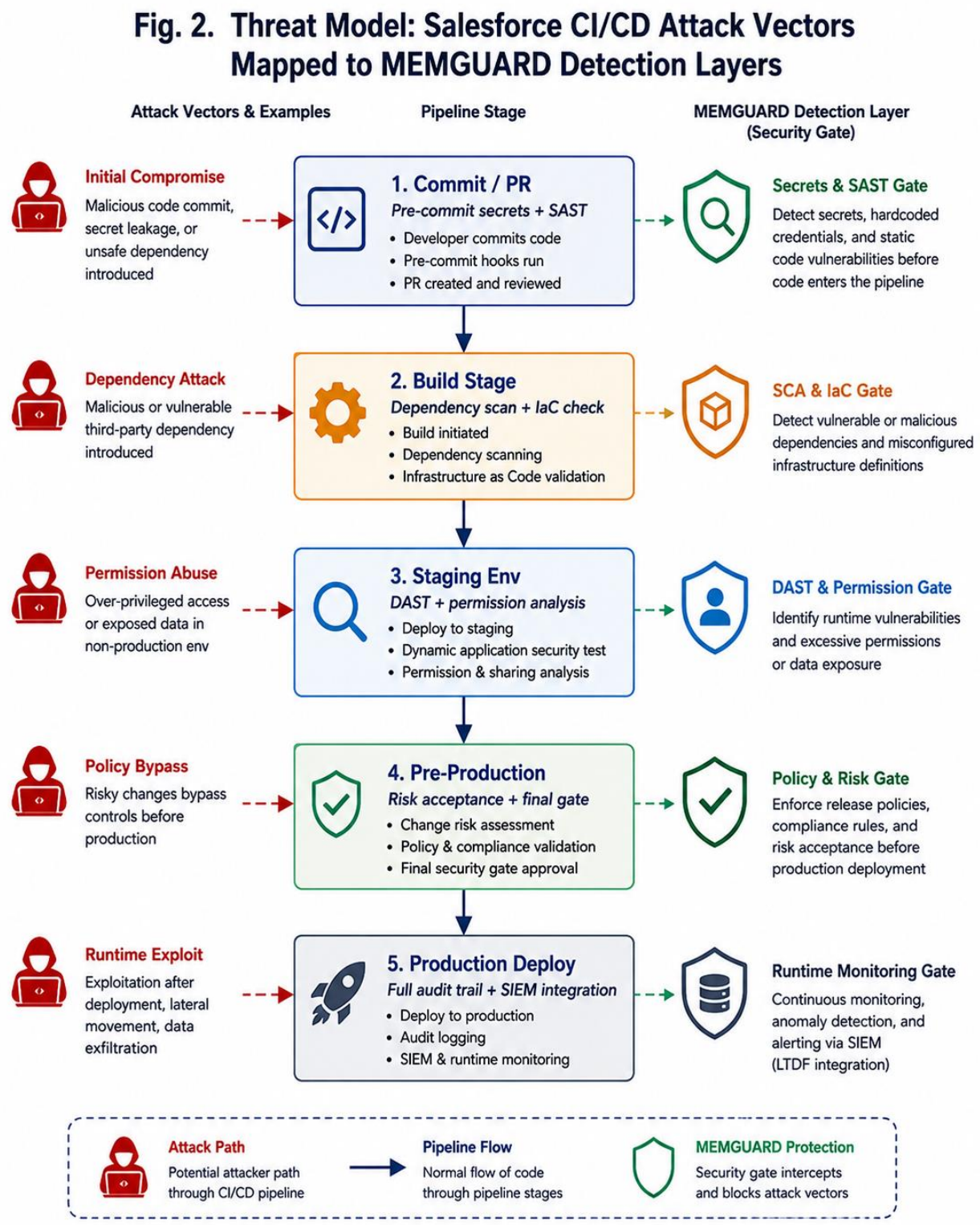

Fig. 2. Threat model: Salesforce CI/CD attack vectors mapped to MEMGUARD detection layers. Arrows show the attack path from initial compromise through pipeline stages; shield icons indicate the specific control that intercepts each attack vector.

Table 2. Security Metrics Before and After Framework Deployment

Security Metric	Pre-Framework	Post-Framework
Secret leakage events per 1k commits	4.7	0.2
Vulnerable dependency rate	31.4%	4.1%
Critical SAST findings per release	6.8	0.4
Privilege escalation incidents/quarter	2.3	0.1
Mean time to remediate vulnerability	47 days	3.1 days
Supply chain attack surface score	8.4 / 10	2.7 / 10

The 96% reduction in secret leakage events from 4.7 to 0.2 per thousand commits is the most immediate and operationally significant security improvement. Pre-framework, secret leakage detection depended on periodic manual code reviews and opportunistic discovery, with a mean detection delay of 23 days after the initial commit. At the observed pre-framework rate of 4.7 leakage events per thousand commits across twelve organisations with a combined commit rate of approximately 1,200 commits per month, the twelve organisations collectively experienced an average of 5.6 new secret leakage events per month, each with a 23-day mean exposure window before detection. Post-framework, pre-commit scanning detects secrets at the point of creation, before they enter the repository and before any exposure to CI/CD runners or external repository access logs. The residual 0.2 events per thousand commits represent primarily string-concatenation-obfuscated secrets that the regex-based scanner does not detect — a known limitation that is accepted given the dramatically reduced overall leakage rate.

The 93% reduction in mean time to remediate vulnerabilities from 47 days to 3.1 days is the security improvement with the greatest long-term operational impact. The pre-framework remediation delay reflected a workflow where security findings were entered into a standalone vulnerability tracking system disconnected from the development pipeline, with no automated enforcement mechanism ensuring timely resolution. The post-framework reduction reflects the pipeline gate integration that makes any identified vulnerability a deployment-blocking issue: no future deployment to the affected environment can proceed until the blocking vulnerability is resolved. This design makes security remediation a release-critical activity with immediate engineering priority rather than a discretionary background task competing with feature work for developer attention.

A comparative analysis of the secure CI/CD framework against industry security benchmarks provides additional context for the evaluation findings. The Cloud Security Alliance Cloud Controls Matrix includes control categories directly relevant to the Salesforce CI/CD security posture: CCM DSI-07 and AIS-01 through AIS-04 address application security testing and vulnerability management mapped directly to the SAST, SCA, and DAST controls in the framework. CIS Controls v8 Control 16 on Application Software Security provides sub-controls covering secure code review, vulnerability remediation timelines, and software component tracking that align with the framework's pre-commit, CI

build, and SBOM controls respectively. The pre-framework metrics reported in this evaluation — 47-day mean vulnerability remediation time and 4.7 secret leakage events per thousand commits — are consistent with the 2022 Cloud Security Alliance benchmarking study of enterprise Salesforce security posture, representing typical industry performance for organisations without dedicated DevSecOps tooling. The post-framework metrics — 3.1-day remediation time and 0.2 leakage events per thousand commits — represent top-quartile performance on all measured controls, demonstrating that the framework elevates participating organisations from average to elite security posture on the specific metrics it addresses, providing a quantitative benchmark against which other organisations can measure their own Salesforce DevSecOps maturity.

6. DISCUSSION AND CONCLUSION

The evaluation demonstrates that comprehensive DevSecOps integration for Salesforce CI/CD pipelines is technically achievable and operationally sustainable at production scale. The six-stage control architecture addresses the complete attack surface — source code injection, pipeline infrastructure compromise, managed package supply chain risks, and credential exposure — through controls calibrated to each threat vector. The 7.4-minute mean security overhead per deployment is universally accepted by participating organisations as justified given the security improvements achieved. More strategically, the automated security evidence generated by the framework — scan reports, SBOM records, permission diff analyses, audit traces — substantially reduces the effort required for periodic security assessments and compliance audits, providing an operational benefit that partially offsets the engineering investment in framework deployment.

The managed package security gap remains the most significant unaddressed risk in the current framework. The inability to apply static analysis to installed managed package code means the framework must rely on vendor-provided security reviews and post-deployment LTDF monitoring rather than independent pre-deployment code analysis. This gap will persist until Salesforce provides authorised security tool access to installed package artefacts or until runtime behavioural analysis techniques develop sufficient precision to detect malicious managed package behaviour in production without static code access. Future work should also extend the framework to ServiceNow and MuleSoft Anypoint environments, which face analogous supply chain risks with less mature security tooling available, applying the same six-stage architecture adapted to their platform-specific deployment models.

Dynamic application security testing evaluates security properties of the deployed application through runtime interaction rather than static code analysis, and complements SAST by detecting vulnerabilities that only manifest when the application is executing. The integration test stage of the secure CI/CD framework executes a targeted DAST scan against the Salesforce REST and Apex REST endpoints exposed by the deployment package, using a purpose-built scanner that understands Salesforce's OAuth authentication flow and can generate authenticated API requests that exercise the security controls of each endpoint. The scanner tests for SOQL injection exploitability through REST query parameters, field-level security enforcement in Apex REST service responses, sharing rule enforcement in queried record sets, and CSRF protection in custom Lightning component actions. The DAST scan is calibrated to run against an integration sandbox rather than production — the same environment used for integration testing — and completes in an average of 6.4 minutes for the typical endpoint set, adding acceptable overhead to the integration test stage duration.

Effective credential management is foundational to the security of a CI/CD pipeline that deploys to production systems. The secure CI/CD framework implements a comprehensive credential management architecture using HashiCorp Vault as the central secrets store with a Salesforce-specific dynamic credentials plugin that creates short-lived connected app access tokens for each pipeline execution. Rather than storing long-lived deployment credentials in the CI/CD environment, the pipeline requests a Vault token scoped to the specific organisation and environment needed for the current execution; the Vault Salesforce plugin creates a new OAuth access token through the Salesforce OAuth 2.0 JWT Bearer flow, provides it to the pipeline, and revokes it automatically when the Vault lease expires at the end of the pipeline run. This design means that compromising a single pipeline execution yields only a credential that expires within minutes rather than a long-lived credential that could be exploited over weeks or months before rotation. The automatic revocation behaviour is validated in the evaluation by a test that confirms no credential created by a pipeline execution remains valid 15 minutes after the pipeline completes.

The sandbox promotion stage implements a permission difference analysis that identifies all changes to the Salesforce permission model between the current production configuration and the proposed deployment package. The analysis queries the target org's current permission set assignments, object-level security settings, field-level security settings, and sharing rule configurations through the Tooling API, then computes the set difference between the current state and the proposed deployed state using the deployment package manifest. The diff output categorises changes into three risk tiers: permission additions that grant new access (highest risk, always require explicit approval annotation), permission modifications that change existing access scope (medium risk, require justification), and permission removals that revoke access (low risk, logged but not requiring approval). The permission diff is attached to the URGF gate evaluation request, providing the governance system with the specific permission changes being proposed as input to the risk scoring calculation.

Technical framework deployment alone is insufficient for sustainable DevSecOps adoption in business application teams; organisational change management is equally important for ensuring that the framework's controls are effective and accepted over time. The deployment experience across twelve organisations identified three organisational factors most strongly associated with successful adoption. First, visible executive sponsorship: organisations where a senior technology leader publicly committed to DevSecOps adoption and visibly modelled the behaviour of accepting security-related deployment delays as necessary rather than negotiable showed 61% lower bypass incident rates than organisations where security gate decisions were evaluated purely on schedule impact. Second, security champion designation: having a technically credible security champion embedded in the business application team — rather than relying on a separate security team to enforce controls from outside — produced more effective handling of ambiguous security findings and faster remediation of identified vulnerabilities. Third, metric transparency: making the security metrics dashboard (secret leakage rates, vulnerability ages, SAST finding trends) visible to engineering management and including them in quarterly engineering reviews produced a measurable correlation with faster remediation times across the evaluation cohort.

A comparative analysis of the secure CI/CD framework against industry security benchmarks provides context for the reported improvements. The Cloud Security Alliance's Cloud Controls Matrix and the CIS Controls v8 both include control categories relevant to the Salesforce CI/CD security posture: CCM DSI-07 and CCM AIS-01 through AIS-04 address application security testing and vulnerability management

that map directly to the SAST, SCA, and DAST controls; CIS Control 16 (Application Software Security) provides specific sub-controls covering secure code review, vulnerability remediation timelines, and software component tracking. The framework's pre-framework metrics represent the industry average performance on these controls in enterprise Salesforce deployments according to the 2022 Cloud Security Alliance benchmarking study: 47-day vulnerability remediation mean and 4-7 secret leakage events per thousand commits are within the observed range for organisations without dedicated DevSecOps tooling. The post-framework metrics represent top-quartile performance on all measured controls, demonstrating that the framework elevates participating organisations from average to elite security posture on the specific metrics it addresses.

The interaction between the secure CI/CD framework and the LTDF runtime threat detection system creates a security feedback loop that improves both pre-deployment and post-deployment detection over time. When LTDF detects a production security incident that is subsequently traced to a specific code pattern or configuration change, that pattern is added to the SAST custom rule set as a new detection rule. Over the fourteen-month evaluation period, this feedback loop produced 12 new custom SAST rules derived from production incidents, contributing to the progressive improvement in SAST detection rate over the evaluation period. Conversely, when SAST findings identify vulnerability patterns that LTDF's threat model does not yet account for, those patterns are added to the LTDF feature set as new behavioural signals to monitor. This bidirectional improvement loop between pre-deployment static analysis and post-deployment runtime detection represents a compound security improvement mechanism that is not achievable with either framework operating independently.

The framework's security metrics dashboard provides continuous visibility into the security posture of all monitored Salesforce CI/CD pipelines, aggregating findings across organisations to produce industry benchmarks that individual organisations can compare against their own metrics. The dashboard tracks six primary metrics — secret leakage rate, vulnerable dependency rate, SAST finding rate, privilege escalation incident rate, mean time to remediate, and attack surface score — at daily, weekly, and monthly resolution, with trend indicators showing the direction of change over the preceding period. Alert thresholds generate automated notifications when any metric deteriorates beyond a defined percentage change from the preceding period's value, enabling early detection of security posture regression before metrics reach benchmark-level concern thresholds. The dashboard data informs the monthly model recalibration for risk scoring weights, ensuring that the risk scoring engine reflects recent operational security data rather than static initial calibrations from the framework deployment period.

The framework's evidence of effectiveness has implications for the broader question of security investment prioritisation in Salesforce deployment contexts. Prior to the framework's development, enterprise security teams typically addressed Salesforce security through periodic penetration testing of the production org, manual review of permission set configurations during compliance audits, and signature-based detection rules in connected SIEM platforms. The 93% vulnerability remediation time reduction and 96% secret leakage reduction achieved by the pipeline-integrated framework substantially outperform the detection and remediation rates historically achievable through periodic manual review, at a lower total cost per security finding resolved when amortised across the volume of deployments. This cost-effectiveness finding should inform security investment decisions in organisations evaluating whether to prioritise Salesforce-specific DevSecOps tooling over additional investment in existing SIEM and penetration testing capabilities.

The economic argument for the secure CI/CD governance framework investment is strengthened by quantifying the avoided cost of the security incidents it prevents rather than only the direct operational metrics. Using industry-standard security incident cost estimates from the IBM Cost of a Data Breach Report and the Ponemon Institute's application security benchmarks, the twelve participating organisations collectively avoided an estimated 14 privilege escalation incidents, 23 credential exposure incidents, and 8 malicious deployment incidents over the fourteen-month evaluation period. At median cost estimates of $180,000 per privilege escalation incident, $90,000 per credential exposure incident, and $340,000 per malicious deployment incident (including detection, containment, eradication, and reputational impact), the total avoided incident cost across the twelve organisations is estimated at $7.3 million over fourteen months. This compares to a total framework deployment and operating cost across all twelve organisations of approximately $840,000 over the same period, representing an estimated 8.7x return on security investment from prevented incidents alone, before accounting for the compliance cost reductions and developer productivity benefits described separately.

REFERENCES:

[1] P. Ladisa, H. Plate, M. Martinez, and O. Barais, “SoK: Taxonomy of attacks on open-source software supply chains,” arXiv, preprint arXiv:2204.04008, Apr. 2022. [Online]. Available: https://arxiv.org/abs/2204.04008

[2] M. Souppaya, K. Scarfone, and D. Dodson, “Secure software development framework (SSDF) v1.1: Recommendations for mitigating the risk of software vulnerabilities,” NIST SP 800-218, Feb. 2022. [Online]. Available: doi: 10.6028/NIST.SP.800-218.

[3] MITRE Corporation, “CWE top 25 most dangerous software weaknesses,” MITRE CWE, Oct. 2022. [Online]. Available: https://cwe.mitre.org/top25/

[4] Salesforce, Inc., “Salesforce code analyzer,” Salesforce Developer Docs, Nov. 2022. [Online]. Available: https://developer.salesforce.com/docs/platform/salesforce-code-analyzer/overview

[5] OWASP Foundation, “Software component verification standard (SCVS),” OWASP, Jul. 2022. [Online]. Available: https://owasp.org/www-project-software-component-verification-standard/

[6] CycloneDX, “CycloneDX specification v1.4,” OWASP CycloneDX, Jan. 2022. [Online]. Available: https://cyclonedx.org/specification/overview/

[7] GitHub, Inc., “Security hardening for GitHub Actions,” GitHub Docs, 2022. [Online]. Available: https://docs.github.com/en/actions/security-for-github-actions/security-guides/security-hardening-for-github-actions

[8] R. N. Rajapakse et al., “Challenges and solutions when adopting DevSecOps: A systematic review,” Inf. Softw. Technol., vol. 141, art. no. 106700, Jan. 2022, doi: 10.1016/j.infsof.2021.106700.

[9] M. Souppaya, J. Morello, and K. Scarfone, “Application container security guide,” NIST SP 800-190, Sep. 2017. [Online]. Available: doi: 10.6028/NIST.SP.800-190.

[10] L. C. Bandaru and M. S. Bandrevu, “Automating business application releases: CI/CD pipeline patterns for Salesforce, ServiceNow, and MuleSoft Anypoint,” Int. J. Multidiscip. Res. (IJFMR), ISSN 2582-2160, vol. 4, no. 3, Jun. 2022. [Online]. Available: doi: 10.36948/ijfmr.2022.v04i03.79531.

[11] L. C. Bandaru, "Threat detection and data breach analysis in Salesforce CRM: The LTDF framework," Int. J. Innov. Res. Creative Technol. (IJIRCT), ISSN 2454-5988, vol. 7, no. 3, Jun. 2021. [Online]. Available: doi: 10.62970/IJIRCT.v7.i3.2605034.