harness

EBOOK

Modern Software Delivery Platforms:

Achieving SOX Compliance While Maintaining Business Agility

Contents

Contents

Executive Summary

"We need to ship faster, but we're a public company." This common refrain echoes through engineering organizations grappling with Sarbanes-Oxley (SOX) compliance and its ilk. Development teams push for rapid deployment, while audit committees demand rigorous controls.

This tension often results in overly restrictive processes that stifle innovation or insufficient controls that put the organization at risk. This white paper demonstrates how modern software delivery platforms resolve this conflict, enabling both speed and compliance through automation, verification, and comprehensive audit trails.

Introduction

When WorldCom's $3.8 billion accounting fraud came to light in 2002, it triggered the creation of the Sarbanes-Oxley Act (SOX). This legislation mandated strict internal controls over financial reporting for public companies. Two decades later, those controls extend far beyond accounting practices - they reach deep into our software delivery processes. Every code change to a financial system, every deployment that could affect reporting accuracy, and every access control that protects financial data must meet SOX requirements. This paper examines how modern delivery platforms transform these requirements from potential roadblocks into guardrails that protect organizations while enabling rapid, confident software delivery.

The Challenge

To deliver software quickly, efficiently, and effectively while staying compliant, it is important to understand key controls associated with SOX, and how to address those controls. Here we look at seven controls and how to think about them in terms of software delivery. Teams and applications that are more mature may be able to satisfy the controls with more automation,while others may need more manual steps. In this paper, we'll look at the spectrum of options. To ground the options with an example, we will highlight how the Harness Software Delivery Platform supports its users in meeting their SOX compliance requirements for each control.

Overview of 7 Key Controls

This paper examines seven critical controls for SOX compliance in software delivery:

1. **Segregation of Duties (SoD):**

Prevent anyone from having unchecked power to modify systems with financial impact.

2. **Change Management:**

Ensuring all changes are documented, reviewed, and authorized appropriately.

3. **Version Control & Audit Trails:**

Maintaining tamper-proof records of all changes to code and configurations.

4. **Testing & Release Validation:**

Verifying changes won't compromise system integrity or reliability.

5. **Deployment Controls:**

Restricting production access and maintaining recovery capabilities.

6. **Documentation & Retention:**

Maintaining evidence of control execution for audit purposes.

7. **Audit & Continuous Improvement:**

Regularly verifying controls remain effective and addressing gaps.

For each control, we'll examine both traditional and modern implementation approaches and then explore how modern delivery platforms can help satisfy these requirements while maintaining development velocity.

Control 1: Segregation of Duties (SoD)

The Control

SoX, and similar regulatory frameworks, demand segregation of duties as part of their controls. A challenge for implementing organizations is that there is a lack of firm guidance. Exactly what divisions are required is not always clear.

However, a good heuristic is that it should take multiple people to working together to commit fraud or make a serious financial error. Therefore, no single person should have enough access to both develop and deploy code changes unilaterally. This fundamental control prevents both intentional fraud and innocent mistakes from reaching production systems.

Typically to achieve this, organizations will establish different groups and roles and assign individuals to those roles using an identity management system. Policies will demand that managers audit the assignment of individuals to those groups roles on a routine basis to ensure that accuracy has been maintained.

Modern Implementation Approaches

Organizations can achieve segregation of duties through various approaches, from traditional to highly automated:

- Role-based access control (RBAC) is important. It should clearly control who edits pipelines that create artifacts and deploy them to production. Deployment to production should also be controlled, especially if automated quality control is not especially robust and human decision-making is more important.
- Programmatic updates of identity management systems based on meta-data in HR systems and asset management systems like ServiceNow to partially automate security role configuration.
- Pull/Merge Request workflows ensure all code changes receive peer review before merging. This provides a basic "four eyes" requirement even where developers can deploy to production.

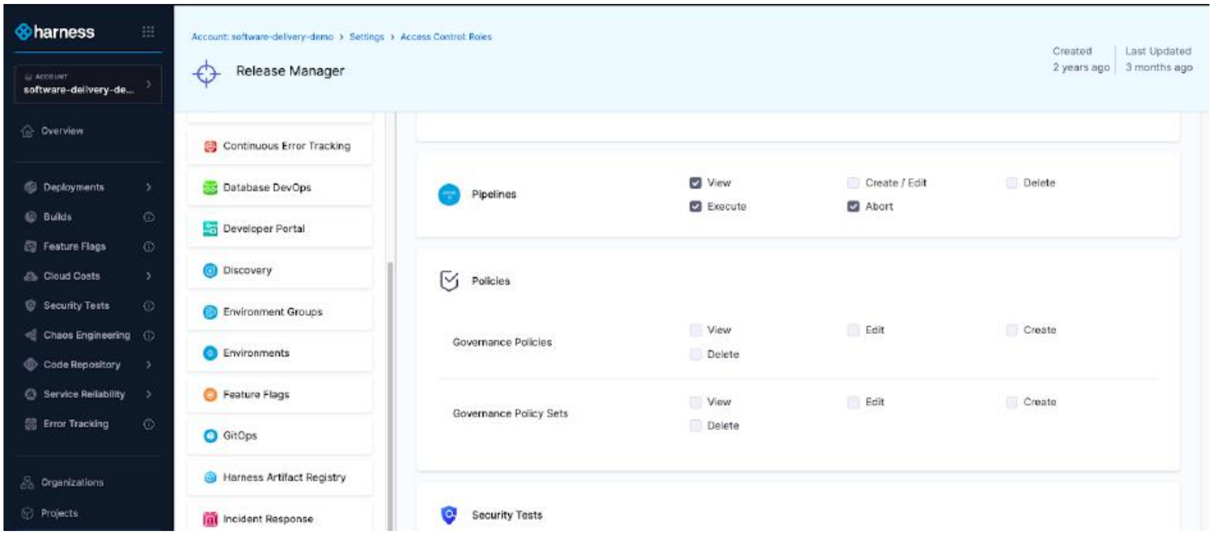
- Service accounts used in lower environments or for builds should be different than those used for production deployments. This helps protect against privilege escalation.
- Policy-as-code frameworks over the CI/CD system can provide an additional layer of protection. For example, the Git repository containing pipeline code may require pull reviews. To protect against misconfiguration, policy-as-code over the pipelines can make the same check - or validate that if the PR contains commits from several developers none of those developers was the reviewer. What might otherwise be a written policy that is awkward to audit, can be enforced automatically.

While a process with many checkboxes and manual sign-offs is generally compliant, it can be error-prone and easy to circumvent while being slow and cumbersome. Aim to automate what you can, ensuring strong audit logs of that automation to reduce both compliance toil and risk. The goal isn't to eliminate human oversight, but to make it meaningful and focused where it matters most.

How Harness Helps

Harness is a good example of a software delivery platform that provides the key capabilities to support the segregation of duties requirements:

- Fine-grained RBAC allows organizations to precisely control who can perform which actions across the platform. This can vary from application to application and environment to environment.
- Integration with SCM platforms maintains the separation between code review and deployment and provisioning processes
- Audit logs capture who performed which actions, providing evidence that SoD controls are consistently followed
- SSO integration ensures access control policies align with corporate identity management and just-in-time account provisioning
- Policy-as-code is implemented with Open Policy Agent (OPA) with key entities that are easily exposed to be operated on.



Account: software-delivery-demo › Settings › Access Control: Roles

Release Manager

Created 2 years ago | Last Updated 3 months ago

Pipelines: ☑ View ☐ Create / Edit ☐ Delete ☑ Execute ☑ Abort

Policies

Governance Policies: ☐ View ☐ Edit ☐ Create ☐ Delete

Governance Policy Sets: ☐ View ☐ Edit ☐ Create ☐ Delete

Security Tests

Control 2: Change Management

The Control

Every change to a system affecting financial reporting must be documented, reviewed, and authorized. This includes maintaining evidence of the change's purpose, its risk assessment, and approval trail.

Modern Implementation Approaches

A Change Advisory Board (CAB) is still a fact of life in many organizations. Today, changes are happening more frequently and CABs meet more often, and make more decisions with more data than ever before. The most modern teams are able to deliver without a CAB. In ITIL terms, they have made software delivery so routine and low risk, that it is a normal, pre-approved changed. Modern organizations can satisfy this control through several approaches:

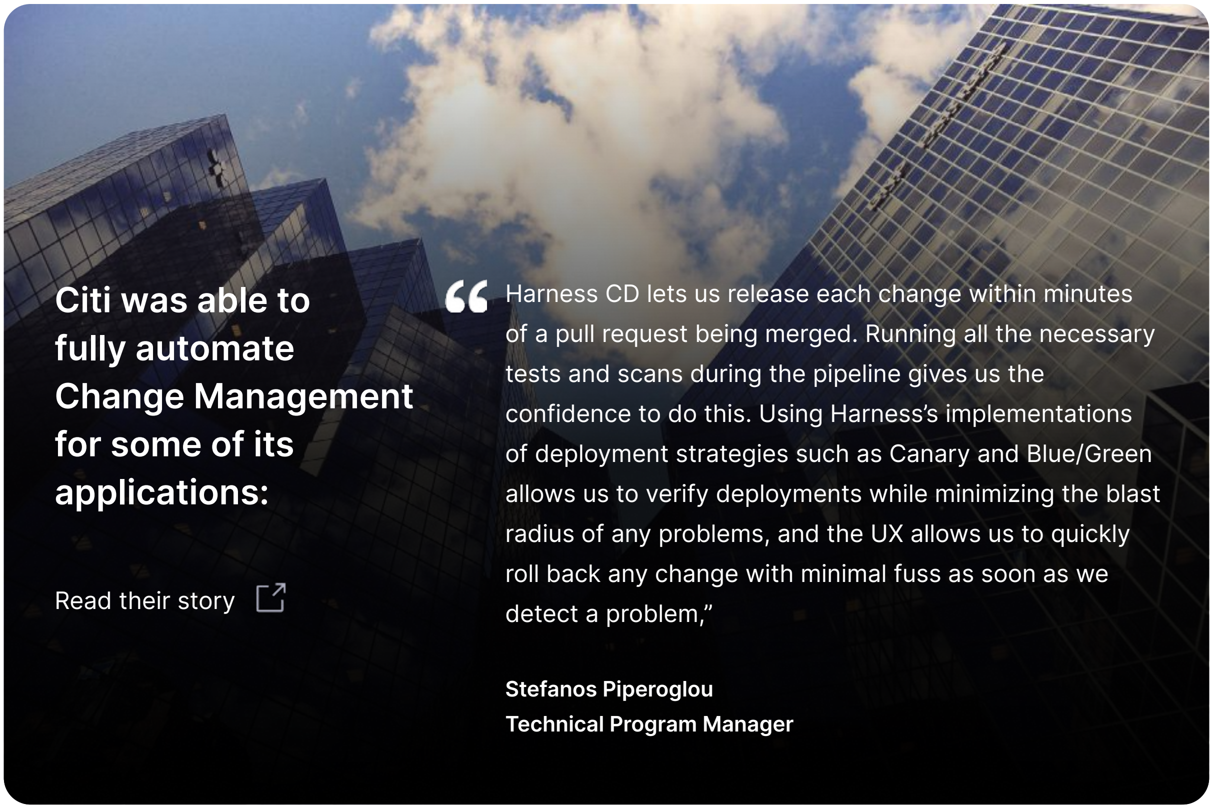
- Pipelines integrated with ITSM systems like ServiceNow to capture approvals from a body like a Change Advisory Board (CAB). At a minimum, adding a field to a pipeline execution for an approval ticket number can create a linkage. A full link that only executes the deployment in response to activity in the ITSM is more secure.
- Automated linking between commits, build artifacts, and deployments to maintain clear traceability
- Automated collection of change evidence (test results, security scans, performance impacts) to inform approvals
- Flexible approval workflows that can scale from lightweight peer review to formal CAB processes based on risk level
- Fully automated change management (post peer review) where all tests and security scans are automatically run and evaluated. If the application is sufficiently high quality it is progressively released into production, and monitored. If there is trouble in production, it is automatically rolled back, otherwise the release progresses.

While traditional change management often relies on manual documentation and meetings, modern platforms can automatically collect the most required evidence during the delivery process. This shifts human effort from paperwork to actual risk assessment and informed decision-making.

How Harness Helps

Modernizing change management is fundamentally a human endeavor - not a technical one. To help your organization modernize, you will take them on a journey by grounding them in the controls, like those in this document and understanding what the stakeholders in existing ceremonies like CABs believe are being achieved. The goal, would be to seek consensus on, “Would it be possible to achieve a similar end if....” while proposing an automated or semi-automated approach and offering a low-risk set of applications to try it out on.

With measurements on the outcomes, such as downtimes and security incidents incurred in applications going through traditional and new approaches, the organization can take a scientific and dispassionate approach to streamlining governance. A technology like Harness then, must offer the building blocks for your automated solution.



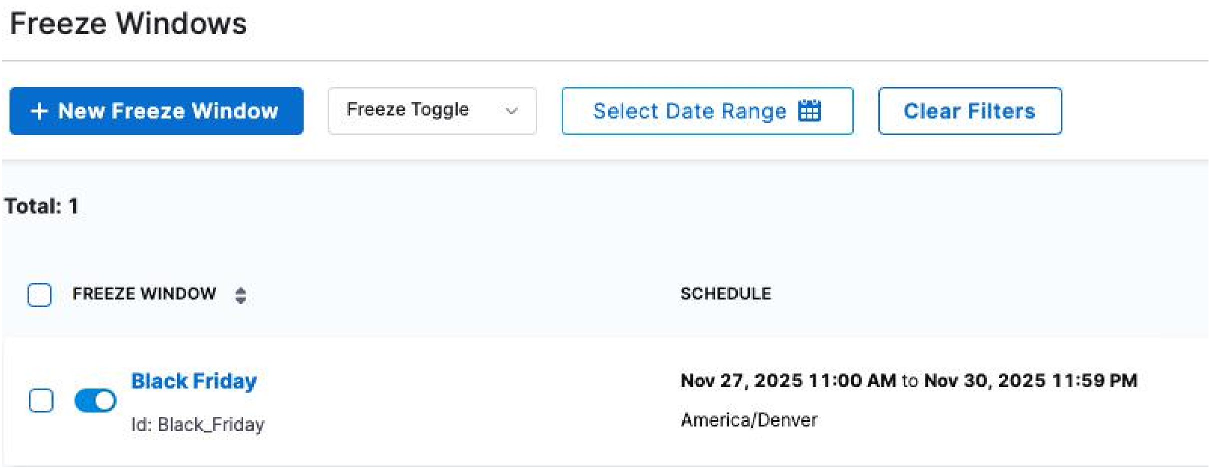
Citi was able to fully automate Change Management for some of its applications:

Read their story

> Harness CD lets us release each change within minutes of a pull request being merged. Running all the necessary tests and scans during the pipeline gives us the confidence to do this. Using Harness’s implementations of deployment strategies such as Canary and Blue/Green allows us to verify deployments while minimizing the blast radius of any problems, and the UX allows us to quickly roll back any change with minimal fuss as soon as we detect a problem,”
>
> **Stefanos Piperoglou**
> **Technical Program Manager**

Here’s how

- Change tracking across the entire software delivery lifecycle maintains a complete record of what changed and why
- Pipeline governance features ensure deployment approvals come from authorized individuals or teams. Harness offers a native approval management framework, and also integrates with third-party systems such as ServiceNow for organizations running traditional CABs.
- Open Policy Agent (OPA) integration enables automated policy enforcement, rejecting changes that don’t meet compliance requirements such as:
  - Applications with new high-severity security vulnerabilities
  - Infrastructure changes that exceed cost thresholds
  - Deployments missing required documentation or approvals
- Flexible approval workflows support both automated and manual governance:
  - Native approval steps within pipelines
  - Integration with external approval systems like ServiceNow and Jira
  - Support for uploading manual evidence (like signed documents) as pipeline inputs
- Comprehensive audit trails capture all changes, approvals, and policy evaluations
- Integration with popular ticketing systems maintains change documentation and traceability
- Deployment freeze windows and scheduling controls enforce change blackout periods



Total: 1

□	FREEZE WINDOW	SCHEDULE
□	Black Friday Id: Black_Friday	Nov 27, 2025 11:00 AM to Nov 30, 2025 11:59 PM America/Denver

Control 3: Version Control & Audit Trails

The Control

Organizations must maintain a complete, tamper-proof record of all system changes affecting financial reporting. This includes tracking who made each change, when it was made, and maintaining an unbroken chain of custody from development through production.

A naive automation setup for CI/CD or for infrastructure-as-code (IaC) can lack sufficient security scanning - or enforcement that prevents critical security issues from being propagated to production. The pipeline itself may be insecure as well. Poisoned pipeline attacks where the pipeline or the build infrastructure is compromised, can allow an attacker to compromise software built or deployed through the system with catastrophic results.

Fundamentally, you need to provide a record that you have ensured the security of your code and its dependencies and that none of the artifacts involved have been tampered with at any stage.

Modern Implementation Approaches

Version control and audit requirements extend beyond just tracking application code changes. A comprehensive approach must cover all elements that could affect system behavior:

- Application code version control ensures:
    - Immutable commit history with cryptographic validation
    - Signed commits to verify author identity
    - Protected branches preventing direct modifications to production code
    - Required reviews and approvals before merging changes

- Infrastructure and configuration management:
    - Infrastructure-as-Code stored in version control
    - Configuration changes tracked and versioned
    - Environment-specific configurations securely stored and versioned

- Deployment pipeline version control:
    - Pipeline definitions stored in version control or equipped with comparable audit capabilities
    - Changes to build and deployment processes tracked and reviewed
    - Pipeline configuration changes logged and attributable to specific individuals

The goal is to maintain a complete chain of custody showing not just what code was changed, but how it was built, configured, and deployed. Modern platforms can automate much of this audit trail collection, making it easier to demonstrate compliance while reducing manual record-keeping.

How Harness Helps

Harness provides comprehensive audit capabilities across the entire software delivery lifecycle, going well beyond basic version control tracking:

Pipeline Governance: Harness treats pipelines themselves as critical infrastructure requiring the same level of audit scrutiny as application code. Every pipeline change is tracked with:

- Complete version history of pipeline definitions
- Detailed logs of who modified pipeline configurations and when
- Ability to compare pipeline versions and track evolution over time
- Required approvals for pipeline changes that could impact compliance
- Option to store pipeline definitions in Git for additional governance

Deployment Audit Trails Each deployment captures a complete record of the delivery process:

- Every artifact involved in the deployment, from application binaries to configuration files
- Infrastructure changes made during deployment
- All approval decisions, whether automated or manual
- Policy evaluations and their outcomes
- Performance and security test results
- Integration with external tools and systems

Immutable Audit Logs Harness maintains tamper-proof audit logs that capture:

- All user actions within the platform
- System-generated events and automated decisions
- Integration activities with external systems
- Failed attempts to circumvent controls
- Changes to access permissions and security settings

These logs are:

- Retained according to configurable compliance requirements
- Searchable for audit investigations
- Exportable for external review
- Protected from unauthorized modification

Infrastructure-as-Code State Files

- IaCM "State Files" for OpenTofu and TerraForm are stored securely by Harness
- A history of state file versions is maintained, so administrators can see the state of infrastructure at any time in the past.
- Automated drift detection identifies when the infrastructure does not match desired state

Verification and Reporting To support both ongoing compliance and audit preparations:

- Real-time dashboards show compliance status across projects
- Built-in reports highlight control effectiveness
- API access enables integration with governance reporting tools
- Historical trending helps identify control degradation
- Drill-down capabilities support detailed investigations

Control 4: Testing & Release Validation

The Control

Changes to systems affecting financial reporting must be adequately tested before reaching production. While SOX doesn't prescribe specific testing methods, it requires organizations to demonstrate that changes are validated and won't compromise data integrity or system reliability.

Modern Implementation Approaches

Testing for SOX compliance isn't just about checking boxes - it's about providing evidence that changes are safe and working as intended. Modern organizations can approach this through:

- Automated test suites providing consistent, repeatable validation:
  - Unit tests verifying individual components
  - Integration tests checking system interactions
  - End-to-end tests validating critical business processes
  - Performance tests ensuring system stability
  - Security scans identifying vulnerabilities
- Quality gates enforcing minimum standards:
  - Test coverage thresholds
  - Security vulnerability limits
  - Performance benchmarks
  - Code quality metrics

- Evidence collection and retention:
    - Test execution logs and results
    - Coverage reports
    - Security scan findings
    - Performance test data
    - Manual test documentation when required

The key is establishing clear acceptance criteria and maintaining evidence that changes met these criteria before deployment. While manual testing may still play a role, automated testing provides more consistent, repeatable evidence of proper validation, especially for regression testing.

How Harness Helps

Harness supports comprehensive testing and validation throughout the delivery process:

Testing Solutions

- Chaos Engineering capabilities for stress testing critical applications
- An AI Test agent to improve UI-driven functional testing

Test Orchestration and Integration

- Harness provides seamless integration with popular testing tools and frameworks
- Parallel test execution for faster feedback
- Distributed test execution across multiple environments

Security Test Orchestration

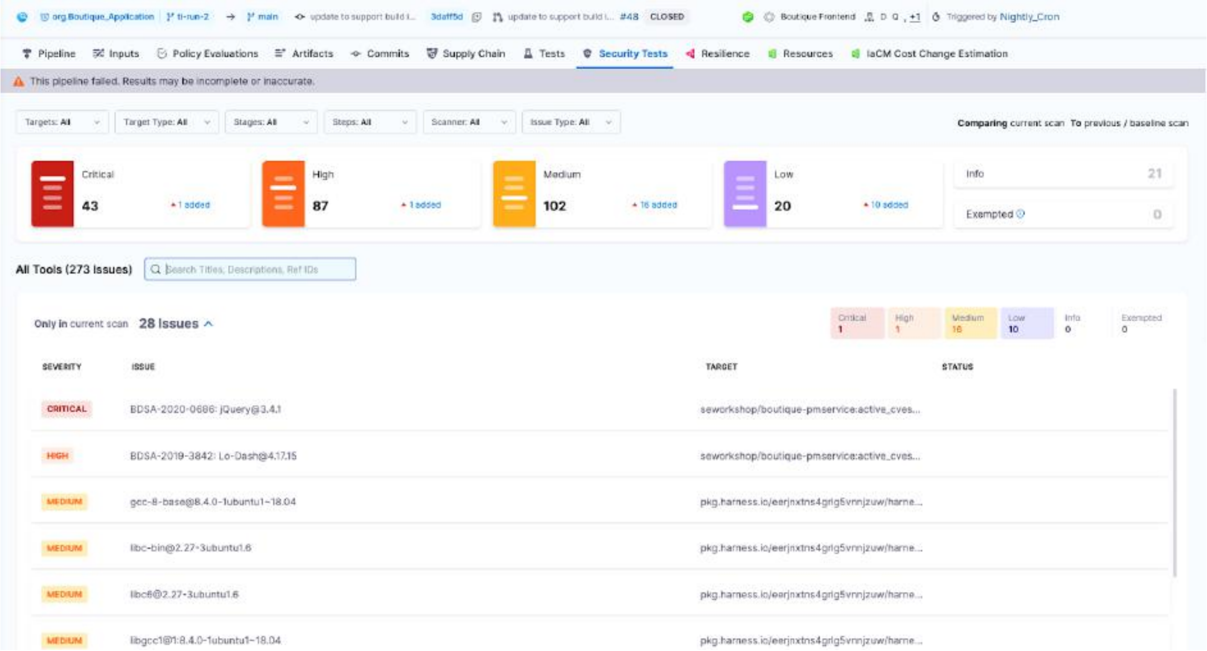
- Deduplication and priority normalization across multiple security scanners
- AI-native remediation guidance for security issues
- Tracking of new, existing, and false-positive issues

Quality Gates and Policy Enforcement

- Configurable quality gates based on test results, coverage, and other metrics
- Integration with security scanning tools to enforce vulnerability limits
- Performance validation through integration with APM tools
- Custom criteria definition using OPA policies
- Automatic pipeline termination when quality gates fail

Evidence Collection and Retention

- Detailed test execution logs
- Test result history and trending
- Security scan reports
- Performance test data
- Custom evidence upload support
- Integration with external test management tools



This pipeline failed. Results may be incomplete or inaccurate.

Targets: All | Target Type: All | Stages: All | Steps: All | Scanner: All | Issue Type: All

Comparing current scan To previous / baseline scan

Critical 43 ▲1 added
High 87 ▲1 added
Medium 102 ▲16 added
Low 20 ▲10 added
Info 21
Exempted 0

All Tools (273 Issues)

Only in current scan 28 Issues

Critical	High	Medium	Low	Info	Exempted
1	1	16	10	0	0

SEVERITY	ISSUE	TARGET	STATUS
CRITICAL	BDSA-2020-0686: jQuery@3.4.1	seworkshop/boutique-pmservice:active_cves...	
HIGH	BDSA-2019-3842: Lo-Dash@4.17.15	seworkshop/boutique-pmservice:active_cves...	
MEDIUM	gcc-8-base@8.4.0-1ubuntu1~18.04	pkg.harness.io/eerjnxtns4grig5vnnjzuw/harne...	
MEDIUM	libc-bin@2.27-3ubuntu1.6	pkg.harness.io/eerjnxtns4grig5vnnjzuw/harne...	
MEDIUM	libc6@2.27-3ubuntu1.6	pkg.harness.io/eerjnxtns4grig5vnnjzuw/harne...	
MEDIUM	libgcc1@1:8.4.0-1ubuntu1~18.04	pkg.harness.io/eerjnxtns4grig5vnnjzuw/harne...	

Control 5: Deployment Controls

The Control

Organizations must maintain strict control over production deployments, ensuring only authorized and validated changes reach production environments. This includes controlling who can deploy, when deployments can occur, and maintaining the ability to quickly recover from problems.

Modern Implementation Approaches

Deployment controls need to balance safety with the ability to respond quickly to business needs and critical issues. Modern approaches include:

- **Environment Access Controls:** When SoX was new, segregation of duties meant that only trusted operations personel had credentials to log into production machines and run to deployments. While that is compliant today, the best practice is to limit production deployments to only trusted, automated systems:
    - Restrict production access to authorized deployment mechanisms
    - Separate credentials for development and production systems
    - Just-in-time access for emergency fixes
    - Environment-specific deployment approvals

- **Deployment Safety Mechanisms:** Change always carries a degree of risk. To manage that risk responsibly, safety mechanisms should be built into your release plans. Traditionally, deployments of important applications would be accompanied by a war-room mentality where specialists would assemble to watch the release for signs of trouble and react. Today, this can be automated into the pipeline with:
    - Automated rollback capabilities built into every pipeline
    - Blue-green or canary deployment strategies
    - Integration between continuous delivery pipelines and observability stacks to trigger rollbacks automatically.
    - Configuration verification before deployment
    - Drift detection between environments

- **Operational Controls:**
  - Deployment windows and blackout periods
  - Emergency change procedures
  - Environment promotion paths (ensuring a version visits QA before production)

Whether using traditional deployment methods or continuous deployment, the key is maintaining clear evidence of who deployed what, when, and with what authorization, while ensuring the ability to rapidly recover from issues.

How Harness Helps

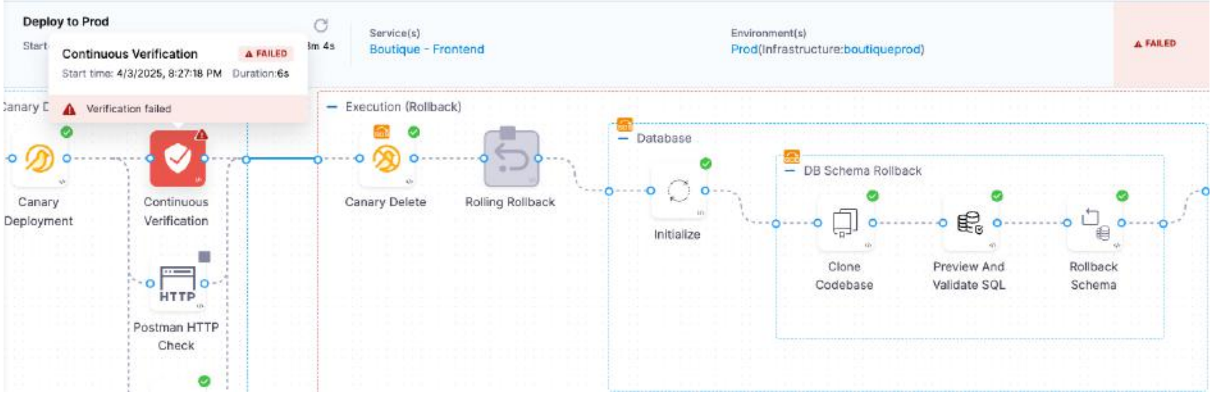
Where the best practice is to strictly limit access to production systems, ideally only to automated systems, the demand is that those automated systems both make change easy and provide excellent logging and telemetry. Harness provides comprehensive deployment control and verification capabilities:

Deployment Management

- Environment-specific deployment strategies and controls
- Automated progressive delivery with verification gates
- Blue-green and canary deployment support out of the box
- Automatic creation of rollback processes with every deployment process
- Infrastructure drift detection and remediation
- Deployment window enforcement

Recovery and Rollback

- One-click rollback to last known good state
- Automated rollback on failed deployments
- Integrations with observability and logging tools are analyzed with machine learning algorithms to detect problem releases, automatically triggering rollbacks to minimize risk to business systems
- Configuration version control
- State capture and restoration
- Recovery procedure documentation



Deploy to Prod

Start

Continuous Verification ▲ FAILED

Start time: 4/3/2025, 8:27:18 PM Duration:6s

▲ Verification failed

3m 4s

Service(s)
Boutique - Frontend

Environment(s)
Prod(Infrastructure:boutiqueprod)

▲ FAILED

Control 6: Documentation & Retention

The Control

Organizations must maintain documentation of their controls, processes, and evidence of their execution. This documentation must be retained for the period required by SOX (typically 7 years) and be readily accessible for audit purposes.

Modern Implementation Approaches

Documentation doesn’t have to mean static documents gathering virtual dust on a shared drive. Modern approaches can make documentation more dynamic and valuable:

- **Process Documentation:**
  - Pipeline definitions serve as living documentation of deployment processes
  - Infrastructure-as-Code captures environment configurations
  - Policy-as-Code documents security and compliance rules
  - Automated documentation generation from code and configurations
  - Version control of all documentation

- **Evidence Collection:**
    - Automated gathering of approval records
    - Test results are captured and stored automatically
    - Deployment logs and audit trails
    - Configuration change history
    - Access control modifications
    - Security scan results
- **Retention Management:**
    - Configurable retention policies
    - Immutable storage of compliance evidence
    - Automated archival processes
    - Quick retrieval for audit purposes
    - Regular validation of stored evidence

The goal is to make documentation and evidence collection an automated byproduct of your delivery process rather than a separate manual effort while ensuring everything needed for compliance is properly retained and accessible.

How Harness Helps

Harness supports comprehensive documentation and evidence management throughout the software delivery lifecycle:

Automated Evidence Collection

- Real-time capture of all platform activities
- Integration with external tools and systems for complete coverage
- Structured data format enabling easy querying and reporting
- Automated tagging and categorization of evidence
- Chain of custody tracking for all artifacts

Retention Management

- Configurable retention policies meeting SOX requirements
- Immutable storage preventing tampering
- Automated archival and cleanup
- Quick search and retrieval
- Evidence integrity verification
- Export capabilities for external review

Process Documentation

- Pipeline definitions document deployment processes
- Template management maintains standardization
- Built-in documentation of security policies
- Configuration and infrastructure documentation
- Integration with external documentation systems
- Version control of all platform configurations

Audit Support

- Pre-built compliance reports
- Custom report generation
- Evidence export capabilities
- Audit trail visualization
- Relationship mapping between changes, approvals, and deployments
- Integration with GRC platforms

Control 7: Audit & Continuous Improvement

The Control

Organizations must regularly review their controls to ensure they remain effective and address any gaps or weaknesses identified. This includes both responding to formal audit findings and proactively improving processes to reduce risk.

Modern Implementation Approaches

Continuous improvement shouldn't wait for annual audits. Modern organizations can take a data-driven approach to ongoing control enhancement:

- **Control Effectiveness Monitoring:**
  - Real-time metrics on control execution
  - Automated detection of control failures
  - Trend analysis of process compliance
  - Early warning indicators of control degradation
  - Regular testing of control effectiveness

- **Process Analysis:**
  - Data-driven identification of bottlenecks
  - Metrics on process efficiency
  - Developer satisfaction measurement
  - Security trend analysis
  - Compliance overhead assessment
- **Improvement Management:**
  - Systematic tracking of audit findings
  - Root cause analysis of control failures
  - Impact analysis of proposed changes
  - Measurement of improvement initiatives
  - Knowledge sharing across teams

The key is moving from reactive responses to audit findings toward proactive, continuous improvement based on real data about how controls are working in practice.

How Harness Helps

Harness provides comprehensive capabilities for monitoring, measuring, and improving your software delivery controls:

Software Engineering Insights

Harness's Software Engineering Insights provides comprehensive visibility into your delivery processes, analyzing workflow patterns to identify where improvements can have the greatest impact. The platform examines the time spent in each delivery phase, highlighting bottlenecks and inefficiencies that might otherwise go unnoticed. By analyzing trends across teams and projects, it offers data-driven recommendations for process improvements. This analytical approach goes beyond simple metrics, examining resource utilization and comparing performance across teams to identify best practices and opportunities for optimization. The impact of process changes can be measured and verified, ensuring that improvements actually deliver their intended benefits.

Internal Developer Portal

Harness's Internal Developer Portal enables organizations to establish clear benchmarks for application quality and compliance. Through customizable scorecards, teams can track their progress toward these standards across key metrics like test coverage, security posture, and deployment reliability. Management gains visibility into benchmark achievement across teams, making it easy to identify both centers of excellence and areas needing additional support. This data-driven approach transforms continuous improvement from a vague goal into a measurable journey, with clear evidence of control effectiveness and maturation over time.

Compliance Monitoring

Harness generates audit trails of changes to key entities in the system. By enabling streaming of the audit trail to Security Incident and Event Management (SIEM) tools, this trail can be used to generate alerts and detect anomalies.

Conclusion

“SOX compliance or software delivery speed” is a false choice. Modern software delivery platforms demonstrate that rigorous controls can actually enable greater speed and confidence in software delivery. By automating evidence collection, enforcing controls consistently, and providing deep visibility into the delivery process, these platforms reduce both compliance overhead and operational risk.

The key is shifting from manual processes and documentation to automated, consistently enforced controls:

Instead of manual change review boards, automated policies enforce standards while capturing evidence. Rather than periodic control testing, continuous monitoring ensures ongoing compliance. Where traditional processes relied on human memory and diligence, automated platforms enforce controls consistently. Manual documentation gives way to automated evidence collection throughout the delivery process.

This transformation requires a comprehensive security approach that addresses three critical domains:

- **Security OF the pipeline:** Ensuring the delivery platform itself is secure and trustworthy
- **Security IN the pipeline:** Embedding security testing and verification throughout the delivery process
- **Security AROUND the pipeline:** Implementing proper access controls and governance of the overall environment

Modern platforms also enable implicit controls through automated linkages to systems of record. When code commits are automatically tied to requirements, test results are automatically associated with deployments, and approvals are consistently logged, the evidence chain becomes an inherent byproduct of the process rather than a separate documentation exercise.

Perhaps the most transformational is the opportunity to fundamentally rethink change management. By implementing a “license to operate” model where teams earn autonomy through demonstrated compliance, organizations can move from heavyweight approval processes to lightweight verification. The most advanced organizations are embracing a zero-trust approach to production access, where even administrators don’t have direct access to production systems – changes flow exclusively through thoroughly validated automated pipelines.

This shift doesn’t just satisfy auditors - it creates better software delivery practices. When controls are automated and evidence collection is continuous, teams can focus on delivering value rather than proving compliance. Quality gates don’t slow teams down; they give them confidence to move faster. Audit trails aren’t just for auditors; they help teams understand and improve their processes.

The future of SOX compliance isn’t about more manual controls - it’s about smarter automated ones. Modern platforms provide the foundation for both rigorous compliance and rapid delivery, turning what was once a trade-off into a synergy that benefits both the business and its auditors.

harness

The AI-Native Software Delivery Platform™

Follow us on

𝕏 /harnessio

in /harnessinc

Contact us on

www.harness.io