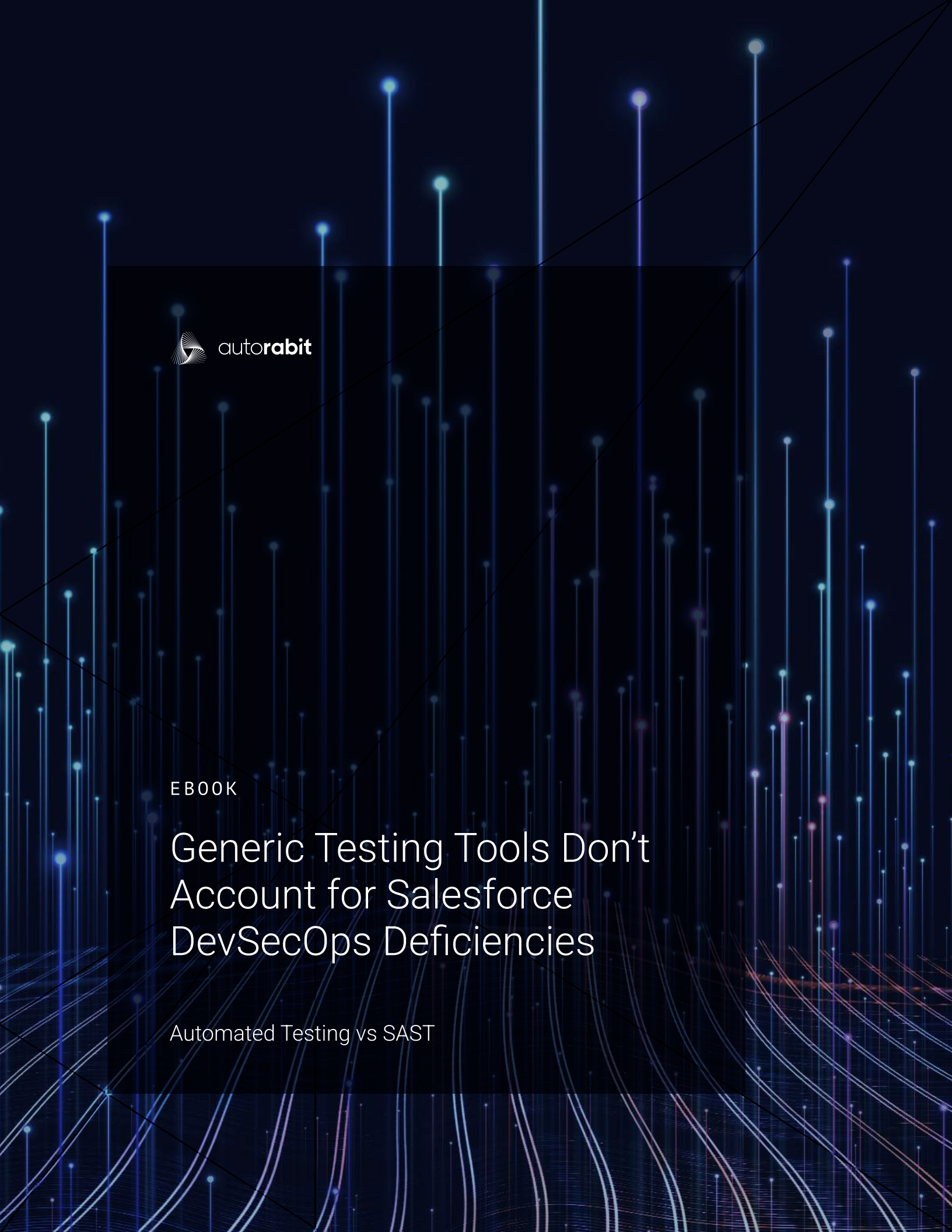


EBOOK

Generic Testing Tools Don't Account for Salesforce DevSecOps Deficiencies

Automated Testing vs SAST

INTRODUCTION

Salesforce aims to secure their platform, not your environment. That's left up to you. And to do this, you need to be aware of potential security vulnerabilities to choose the best strategy and tools to secure your Salesforce instance.

Unfortunately, that's not as easy as it might seem. Data security risks come from numerous intrinsic aspects of Salesforce's architecture.

For example, here are some recent security vulnerabilities reported in Salesforce:

1. **CVE-2023-34362 and CVE-2023-35036:** These two vulnerabilities could lead to unauthorized access to the MOVEit file transfer product and environment. However, there is no impact to Salesforce customer data at this time.
2. **CVE-2022-22128:** This issue affected the Tableau Server Administration Agent.
3. **Tableau Security Update:** This was an issue with Tableau Server logging Personal Access Tokens into internal log repositories.
4. **CVE-2022-22127:** This was a broken access control vulnerability in Tableau Server.
5. **Heroku Security Notification:** There was an issue with GitHub repositories connected to Heroku.
6. **Spring4Shell Security Update:** The Spring4Shell vulnerability published in March 2022 affected multiple Salesforce products including Tableau, Slack, Service Cloud, Salesforce Einstein, Salesforce Core, Sales Cloud, Quip, Pardot, MuleSoft, Marketing Cloud, Hyperforce, Heroku, Experience Cloud, Commerce Cloud, ClickSoftware.

Automated testing is a critical aspect of a comprehensive DevSecOps strategy, but generic automated testing tools that use RPA or a metadata layer do not cover inherent constraints within Salesforce. Static application security testing (SAST) and Salesforce security posture management (SSPM) account for these constraints and support a comprehensive DevSecOps strategy.

We’ll explore these three types of limitations, why test automation isn’t sufficient, and how SAST addresses these concerns:

1. Salesforce Limitations 004

2. Salesforce Complexity 007

3. Salesforce’s Unreliable Structure 010

01

Salesforce Limitations

SALESFORCE LIMITATIONS

Salesforce was originally built as a CRM. Its functionality continues to grow in impressive ways, but the architecture lacks some essential capabilities that can severely limit the potential for a DevOps team.

Automated testing—while being essential for catching mistakes in certain areas of the DevOps pipeline—isn’t able to sufficiently cover these deficiencies. SAST, on the other hand, is more suited to address these constraints.

Constraint: Limited API Calls

- Why Test Automation Fails in Salesforce: Salesforce has limits on the number of API calls that can be made within a 24-hour period. If automated tests are making API calls, they can quickly eat up these limits, causing tests to fail or affect the production environment negatively.
- Why SAST Makes More Sense: As SAST doesn't need to make API calls to Salesforce, it doesn't contribute to API usage.

Constraint: Governor Limits

- Why Test Automation Fails in Salesforce: Salesforce imposes certain governor limits like execution time limits, limits on the number of records that can be processed, etc., that can affect the ability to perform extensive regression and automated testing.
- Why SAST Makes More Sense: SAST and shift-left practices are proactive and thus can help write efficient code that respects Salesforce's governor limits.

Constraint: Multi-tenancy

- Why Test Automation Fails in Salesforce: Salesforce is a multi-tenant architecture where resources are shared among multiple users. This can lead to variable test performance, making it difficult to establish a reliable automated test suite.
- Why SAST Makes More Sense: SAST and shift-left can help identify issues that might arise in multi-tenant architectures before they become problems in production.

Constraint: Flaky Tests

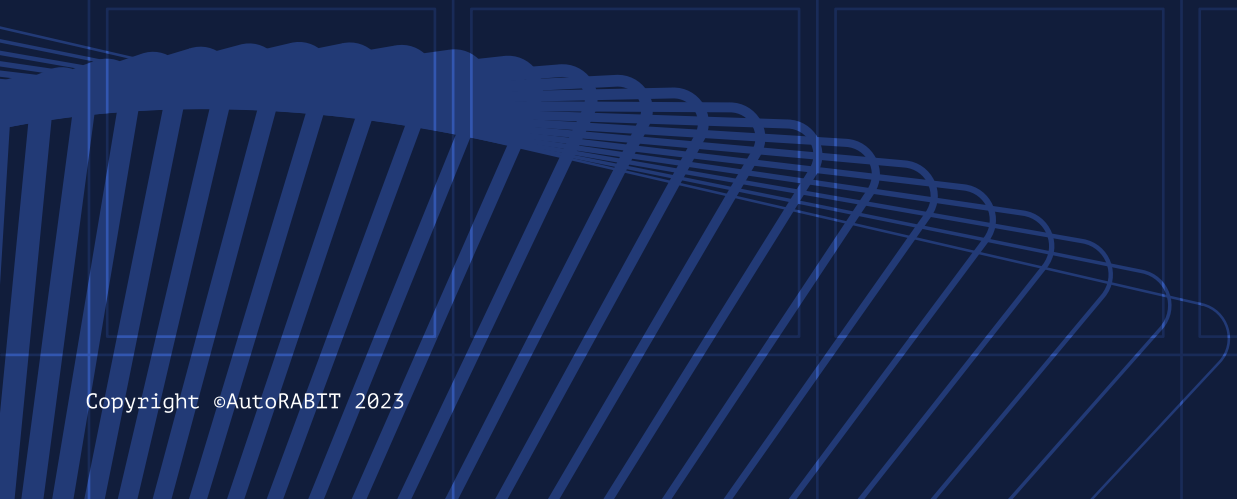
- Why Test Automation Fails in Salesforce: Due to the reasons mentioned above, automated tests can become flaky - they might pass or fail unpredictably.
- Why SAST Makes More Sense: As SAST is not affected by changes in UI or other dynamic elements, it provides more consistent and reliable results compared to some automated testing.

Constraint: Version Control

- Why Test Automation Fails in Salesforce: As Salesforce does not offer built-in version control, tracking changes and maintaining regression test suites across different versions can be difficult.
- Why SAST Makes More Sense: SAST tools often have features that can help with version control, ensuring secure code across different versions.

02

Salesforce Complexity



SALESFORCE COMPLEXITY

Overly complex processes lead to manual mistakes, wasted time, and preventable errors. Some of Salesforce’s greatest strengths can actually increase the possibility of these mistakes, which many users assume will be flagged through test automation. However, these tests are often not specialized to meet unique needs.

SAST is more flexible and therefore better equipped to address the increasingly complex nature of Salesforce architecture.

Constraint: High Customizability

- Why Test Automation Fails in Salesforce: Salesforce's strength lies in its customization abilities. However, this can make it challenging to write automated tests that will work correctly across different configurations, especially in regression testing when assessing new changes against previous functionality.
- Why SAST Makes More Sense: Even with a high degree of customization, SAST tools can consistently scan code for vulnerabilities. This ensures that secure coding practices are followed, regardless of how customized the solution is.

Constraint: Data Dependencies

- Why Test Automation Fails in Salesforce: Salesforce environments often contain complex data relationships and dependencies. Creating and managing test data that accurately reflects these dependencies can be challenging, and tests can fail if the data is not set up correctly.
- Why SAST Makes More Sense: As SAST operates at the code level, it is not dependent on specific datasets and can work irrespective of data complexities.

Constraint: Code Complexity

- Why Test Automation Fails in Salesforce: Salesforce's Apex code can be complex and difficult to write unit tests for, especially when trying to achieve the required 75% code coverage.
- Why SAST Makes More Sense: SAST tools can automatically scan the source code, irrespective of its complexity, for known vulnerabilities. This makes managing complex Apex code easier.

Constraint: Environment Synchronization

- Why Test Automation Fails in Salesforce: Ensuring that all testing, staging, and production environments are consistently synchronized can be a tough task, leading to ineffective or erroneous test results.
- Why SAST Makes More Sense: With shift-left, testing happens early and often, which makes it easier to identify and address discrepancies across environments.

Constraint: Asynchronous Processes

- Why Test Automation Fails in Salesforce: Testing asynchronous Apex code or batch jobs, which are common in Salesforce, can be challenging to automate because it requires additional logic to wait for the asynchronous process to complete.
- Why SAST Makes More Sense: SAST does not depend on running the code, and thus asynchronous processes do not affect its operation.

Constraint: Time and Resource Constraints

- Why Test Automation Fails in Salesforce: Writing comprehensive test cases and maintaining them require significant time and resources. Teams often underestimate the time necessary for this, causing tests to be rushed, incomplete, or nonexistent.
- Why SAST Makes More Sense: By catching issues early, shift-left can save time and resources in the long run. Issues are generally cheaper and faster to fix earlier in the development cycle.

03

Salesforce’s Unreliable Structure

SALESFORCE'S UNRELIABLE STRUCTURE

The problems with Salesforce's architecture go to the very root of how the platform is set up. These constraints make it more difficult to achieve DevOps goals and can make room for seemingly simple mistakes that grow into data security issues.

Frequent updates, customizations, and additional features that are needed to flesh out Salesforce's capabilities introduce unexpected consequences that can't be sufficiently addressed with basic test automation.

Constraint: Metadata Changes

- Why Test Automation Fails in Salesforce: Salesforce environments frequently undergo metadata changes. Keeping automated test scripts updated to reflect these changes can be an arduous task.
- Why SAST Makes More Sense: SAST tools identify poor coding practices that could lead to vulnerabilities during metadata changes.

Constraint: User Interface Changes

- Why Test Automation Fails in Salesforce: Salesforce often updates its user interface. If your automated tests are UI-based (like Selenium), they may break when these updates occur.
- Why SAST Makes More Sense: SAST scans the source code and is not affected by changes in the UI.

Constraint: Third-Party Integration

- Why Test Automation Fails in Salesforce: Salesforce often integrates with various third-party systems, making it challenging to create a comprehensive automated test scenario.
- Why SAST Makes More Sense: SAST tools identify insecure coding practices that could expose vulnerabilities during third-party integrations.

Constraint: Licenses and Features

- Why Test Automation Fails in Salesforce: Different Salesforce orgs might have different licenses and features enabled, making the automation process complicated as tests might behave differently across orgs.
- Why SAST Makes More Sense: Security is a priority regardless of licenses and features. SAST tools and shift-left practices ensure that code is secure, no matter the license or Salesforce org feature.

Constraint: Workflow Rules and Triggers

- Why Test Automation Fails in Salesforce: Salesforce automation often involves workflow rules and triggers, which can make automated testing complicated due to their dynamic nature and potential for side effects.
- Why SAST Makes More Sense: Shift-left practices encourage early and frequent testing of all elements, including workflow rules and triggers, leading to earlier detection and resolution of issues.

Constraint: Salesforce Updates

- Why Test Automation Fails in Salesforce: Salesforce updates its platform three times a year, potentially breaking existing automated test cases.
- Why SAST Makes More Sense: Shift-left and regular SAST help keep up with Salesforce updates, supporting the detection of any new vulnerabilities or issues that arise due to platform changes.

CONCLUSION

Salesforce is the leading CRM because it is a reliable and adaptable platform. However, inherent structural attributes and a wide capacity for customizations introduce various data security vulnerabilities and functional problems.

DevSecOps tools are essential in streamlining Salesforce DevSecOps and accounting for Salesforce’s constraints. However, specialized tooling should be utilized to best address these issues and minimize their harmful impacts on quality, security, and productivity.

SAST is much better equipped to account for these constraints and help DevSecOps teams produce error-free, reliable updates and applications.

CodeScan offers these adaptable services along with policy management for true Salesforce security posture management. Clean code, proper permission settings, and compliance considerations are automated to ensure total coverage and enable team members to focus on what they do best.

These problems require a proactive approach. Working with generic automated testing tools that use RPA or metadata layer simply don’t offer the specialized coverage needed to address Salesforce issues. They’ll never solve these problems. Investing in CodeScan’s SAST and SSPM solutions brings real results.

ABOUT AUTORABIT

AutoRABIT is a DevSecOps suite for SaaS platforms that automates and accelerates the entire application development and release process. This enables continuous integration and delivery by providing fast, simple, and secure end-to-end automation across all Salesforce implementations. AutoRABIT tools help enterprises achieve higher release velocity and faster time to market.

AutoRABIT features static code analysis, automated metadata deployment, version control, advanced data loading, orgs and sandbox management, test automation, and reporting. Its services complement and extend Salesforce DX.

AutoRABIT Vault is a backup and recovery solution that streamlines Salesforce data, simplifies data backup challenges, offers disaster recovery, and provides end-point data protection in the cloud.

CodeScan gives Salesforce developers and administrators full visibility into code health from the first line written through final deployment into production, along with automated checks of Salesforce policies.

FlowCenter helps Salesforce delivery teams achieve heightened release velocity and DevSecOps maturity goals with flow automation, DevOps metrics, and visualization tools that can't be found anywhere else.

Visit us at www.autorabit.com to learn more.

CERTIFIED
+ COMPLIANT

SOC 2 TYPE II CERTIFIED

CALIFORNIA CONSUMER PRIVACY ACT

GDPR

ISO 27001 INFORMATION SECURITY CERTIFIED

HIPAA COMPLIANT