DRYRUN SECURITY

How Commerce Replaced SAST Noise with Code Security Intelligence

Commerce™

Customer
Commerce

Industry
Ecommerce, Technology

Employees
1,000+

Country
United States

About Commerce

Commerce is a global ecommerce company that unifies BigCommerce, Feedonomics, and Makeswift into one brand, powering storefronts and omnichannel selling for merchants ranging from fast-growing small businesses to large global enterprises. BigCommerce provides an open, API-first ecommerce platform for B2C and B2B brands, while Feedonomics handles large scale product feed optimization and syndication across marketplaces like Amazon, Walmart, and other major channels.

Security and compliance are core to why customers choose Commerce over alternatives like Salesforce Commerce, Shopify, WooCommerce, and Magento. The organization invests heavily in certifications and maintains strict, audit-ready payment security controls, including PCI DSS compliant infrastructure that helps merchants offload significant portions of their compliance burden.

The Problem

Commerce operates in a heavily regulated ecommerce environment with PCI, SOC, and ISO expectations, and a codebase that spans more than a dozen programming languages. Traditional SAST tooling had become a bottleneck. After three years on Snyk Code, the team was dealing with a high volume of false positives and noisy rule matches that did not map to real exploitability.

Pattern based scanners would either miss risky changes entirely or flag low value findings like specific function usage without explaining why they mattered. Developers saw “this function is bad” but not the underlying risk, and security teams had to spend time triaging issues that were either not exploitable or were better described as code smells.

Configuration options in their prior tool, like ignoring specific file paths or files and overriding default rules with custom ones - were easy to misapply, forcing tradeoffs between coverage and configuration complexity.

CHALLENGES

- Commerce operates under PCI, SOC, and ISO expectations, with a codebase spanning more than a dozen programming languages.
- After 3 years on Snyk Code, the team faced a high volume of false positives and noisy rule matches that did not map to real exploitability.
- Pattern-based scanning either missed risky changes or flagged low-value findings without explaining impact, forcing security teams into constant triage.
- Configuration tradeoffs were easy to misapply, creating a choice between coverage and complexity.

> *“The context that you get within the pull request being surfaced to engineers is outstanding.*
>
> *It provides them with direct feedback so they know this line of code is introducing a potential vulnerability or a valid vulnerability, and here’s why.”*

The Solution

Commerce turned to DryRun Security to get out of the pattern matching trap and into contextual analysis, starting at the pull request (PR) layer. Installation through the GitHub App was straightforward: point DryRun Security at the right repositories and let DryRun agents begin evaluating each change with contextual, AI-powered analysis, without maintaining rules or regular expressions.

From the first rollout, the team saw a "dramatic improvement" in the quality of PR comments. Where Snyk Code often reported that everything was fine, DryRun Security would surface nuanced guidance to help developers understand security risk. That behavior is driven by the Code Review Agent, which uses Contextual Security Analysis to distinguish between immediately exploitable issues and code paths that are structurally risky even if current guards make them hard to trigger.

Commerce values that distinction. Instead of treating these as noisy false positives, they view them as high value code smells that level up secure coding habits. Developers get real time feedback in the pull request, tied to specific lines of code and backed by clear explanations of "here is what this line does, here is how it could be abused, and here is how to tighten it." That is a very different experience from legacy SAST, where findings often read like generic rule text.

> ***"The context that you get within the pull request being surfaced to engineers is outstanding. It provides them with direct feedback so they know this line of code is introducing a potential vulnerability or a valid vulnerability, and here's why."***

Looking ahead, Commerce plans to experiment with the DryRun Security Custom Policy Agent and Natural Language Code Policies to encode organization specific rules in plain language rather than DSLs. They are especially interested in mapping findings to exploitability to support internal severity re-rating and PCI timelines across whole repositories, not just individual pull requests.

Summary

Commerce needed an application security approach that could keep pace with AI-driven, "vibe coding" development across a large, polyglot ecommerce platform while standing up to strict PCI and audit scrutiny. DryRun Security delivered fast time to value through an easy GitHub App rollout, low configuration overhead, and significant improvements in signal to noise on pull requests.

By replacing pattern-only scanning with Contextual Security Analysis, Commerce now gets clear, actionable feedback that differentiates exploitable issues from code smells and explains the "why" directly to developers. This lowers friction between security and engineering and builds a stronger secure coding culture in a high throughput environment.

From the first rollout, the team saw a "dramatic improvement" in the quality of PR comments.

SOLUTIONS

- Evaluate each PR without maintaining rules or regular expressions for 12+ programming languages.
- Use the "exploitable vs. structurally risky" distinction to replace noisy false positives with high-value guidance that improves secure coding habits.
- Use Contextual Security Analysis to explain what changed, why it matters, how it could be abused, and how to fix it, directly in the pull request.
- Test natural language code policies to encode organization-specific guardrails and map findings to exploitability for severity re-rating and PCI timelines.

RESULTS

- Dramatically higher-quality PR comments that developers can act on immediately
- Less time burned on triage and "rule text" findings that do not match real risk
- Clearer alignment between security review and developer workflow, improving collaboration
- A path to plain-language guardrails and exploitability-based prioritization across repositories