Centralized Scan Factory for Application Security

Rohit Chouhan
Application Security Engineer
PepsiCo, TX, US
Email:rchouhan@cs.stonybrook.edu

Sera Kim
Application Security Engineer
PepsiCo, TX, US
Email:serakim1105@gmail.com

Chad Haley
Application Security Engineer
PepsiCo, TX, US
Email:chad.haley443@gmail.com

***Abstract*—**

automate, orchestrate, and scale application security scanning across enterprise repositories. The system integrates static application security testing (SAST), software composition analysis (SCA), and secrets detection into a unified pipeline. By leveraging event-driven orchestration through message queues and on-demand containerized agents, the platform dynamically allocates compute resources for scanning tasks while maintaining centralized control, visibility, and reporting. This architecture ensures consistent scanning coverage with minimal manual intervention, optimizing both performance and cost efficiency.

1. Introduction

Application security (AppSec) in the modern software landscape encompasses a wide range of domains, including static code analysis, API security, edge and container security, AI security, and web application firewalls (WAFs). Modern applications are complex, distributed systems composed of numerous interconnected components and services. Securing each of these layers is a daunting task, making Application Security a critical pillar of enterprise security strategy.

An important aspect of AppSec is static source code scanning - the process of analyzing source code to detect potential security vulnerabilities before deployment. Since source code forms the basis of all applications, identifying and remediating security issues early in the development lifecycle is crucial.

Traditionally, static code analysis has been performed using commercial off-the-shelf (COTS) scanners. These tools inspect application source code to uncover programming flaws that could lead to vulnerabilities such as SQL injection, cross-site scripting (XSS), or insecure deserialization. They also perform software composition analysis (SCA) to detect vulnerabilities in third-party dependencies and flag license compliance issues, while secrets scanners identify hardcoded credentials within the codebase.

However, effectively operationalizing these scanners at scale presents significant challenges for large organizations. Continuous scanning, correlating results, and surfacing actionable findings to the respective code owners remain persistent bottlenecks. A common approach is to embed security scans directly within CI/CD pipelines. While this appears natural and convenient, it introduces several operational limitations. Integrating scanners across thousands of repositories requires close coordination with multiple DevOps teams, often leading to inconsistencies in scan coverage and configuration drift. Moreover, embedding scans directly in CI/CD pipelines limits central control, reduces visibility across projects, and introduces dependencies on individual pipelines' health and configuration.

To address these challenges, we propose the Centralized AppSec Scan Factory, a cloud-native framework designed to

2. Background

COTS tools in the application security domain typically provide capabilities for SAST, DAST, SCA, container, API, AI, and secrets scanning. These tools also offer integrations with cloud platforms, source code management (SCM) systems, DevOps pipelines, and incident-management solutions. Using these integrations, organizations can automate security scans and route findings to the appropriate teams for remediation.

While such integrations can be effective for small organizations with only a handful of development teams, they do not scale well. In smaller environments, AppSec responsibilities can be delegated to developers or DevOps teams, and the operational overhead remains manageable even when multiple tools are used for different types of scans. This decentralized approach breaks down as the size of the engineering organization grows. As the number of repositories, teams, and toolchains increases, maintaining consistent and reliable scanning becomes significantly more challenging. Several problems quickly emerge:

1) Managing access and permissions across numerous teams
2) Enforcing consistent scanning policies and remediation standards
3) Increasing operational burden on developers and DevOps teams
4) Creating confusion when multiple scanning tools are used simultaneously
5) Ensuring full scan coverage across all assets
6) Prioritizing scans based on risk or business importance

These challenges are common in large enterprises attempting to establish an effective application security scanning program. Most organizations rely heavily on the default integrations provided by COTS tools. However, while these integrations work reasonably well in smaller settings, they often fail to deliver comprehensive coverage in large organizations with diverse use cases and heterogeneous development workflows. Existing literature on enterprise-scale AppSec scanning is sparse, and there is limited empirical data on the effectiveness of vendor-provided integrations in achieving complete scan coverage.

In our effort to deploy SAST, SCA, and secrets scanning for a Fortune 500 company with over 20,000 repositories, we confronted the same core question: How can we guarantee complete scan coverage from the very first day of tool adoption? Each scanner integrates differently with SCM and CI/CD platforms, and many are opinionated regarding scan frequency, prioritization logic, and alerting workflows. To overcome these limitations, we designed a centralized "scan factory" architecture capable of orchestrating and managing scans at scale. The remainder of this paper details the architecture, implementation, and evaluation of this centralized approach.

3. Problem Definition

In this section, we outline the requirements for an effective enterprise-scale AppSec program and explain why the default integrations offered by COTS security scanners are insufficient for achieving organization-wide coverage.

A fundamental objective of any AppSec program is to ensure that all relevant assets are continuously scanned and that identified vulnerabilities are remediated in a timely and consistent manner. A key metric in this process is *scan coverage*, defined as:

$$\text{Coverage} = \frac{\text{Number of scanned assets}}{\text{Total number of scannable asset}}$$

For static code analysis, the total number of scannable assets is typically equivalent to the number of source code repositories maintained by the organization. Ideally, every repository should be scanned regularly, with findings reliably surfaced to the appropriate engineering teams.

Many COTS AppSec tools provide native integrations with SCM platforms to support automated and periodic scanning. While these integrations reduce the operational burden on development teams, our experience indicates several critical limitations that prevent them from meeting the needs of a large enterprise. Specifically, default integrations often:

- fail to scan all branches of a repository,
- run an complete scan on a zero-day vulnerability,
- restrict the ability to configure scan frequency,
- do not automatically import new or recently created repositories,
- lack support for on-demand scanning (e.g., pre-production checks),
- cannot trigger scans based on custom business workflows,
- provide incomplete or inconsistent integrations with platforms used across the organization,
- lack flexibility in defining how findings should be ranked, filtered, or interpreted,
- prevent customization of scan strategies,
- offer no mechanism to prioritize scanning targets based on business risk or operational needs.

When multiple COTS tools are used for SAST, SCA, secrets scanning, or other AppSec functions, these limitations compound. Integrations become fragmented, findings remain siloed, and the organization is ultimately constrained by the overlap—or lack thereof—among the integrations supported by each tool. This leads to gaps in coverage, inconsistent scanning practices, and ineffective remediation pipelines, especially when vulnerabilities are not routed to the appropriate teams.

From this assessment, it becomes clear that the default integrations provided by COTS scanners are not sufficient for large organizations. Achieving complete and consistent AppSec coverage requires a more flexible and centralized approach—one that aligns with enterprise workflows, supports scalable automation, and ensures comprehensive visibility across all assets.

4. Design

An enterprise-scale scan factory must address the shortcomings of default COTS integrations while enabling centralized, automated, and reliable application security scanning at scale. Unlike decentralized models— where individual development teams configure and maintain their own scans - a scan factory centralizes orchestration, monitoring, and policy enforcement to ensure complete coverage and consistency. A cetralized scan factory will have the following components -

4.1. Inventory Service

This service will help the system maintain an up-to-date inventory of all scannable assets, such as source code repositories, container images, APIs, and infrastructure-as-code artifacts. This requires continuous synchronization with SCMs and other asset stores. Newly created repositories must be automatically discovered and added to the inventory database.

4.2. Scan Orchestration Service

This service is responsible for scheduling and launching scans on the assets discovered by the inventory system. Scans may be triggered event-based or executed on a configurable schedule, depending on organizational requirements. All scheduled scans can be placed into a managed queue to ensure orderly, reliable processing at scale.

In addition, the scan orchestration service enforces the organization's scanning policy - such as determining which assets should be scanned based on their last update time or other priority criteria - ensuring consistent and policy-driven scan coverage across the environment.

4.3. Scan Service

This will be main service which will pick up messages from the queue and launch scans of them by making using of a COTS scanner. The architecture of this service should be such that that COTS scanner can easily be replace by any other scanner as and when need arises.

4.4. Security Findings Processing

Once a scan is completed, the resulting security findings must be processed, normalized, and enriched before being forwarded to the organization's centralized findings management system. Centralizing findings ensures consistent formatting, prioritization, and categorization, enabling them to be routed to the appropriate asset or code owners with the necessary context for effective remediation. This unified workflow supports timely risk reduction across the organization and provides clear visibility into the overall security posture.

5. Implementation

The centralized scan factory can be implemented as a collection of modular services, each responsible for one of the core functions described in the section above. These services will be bespoke to the organization's environment and must be designed to integrate with its specific source code management platforms, DevOps platoforms, container registries, cloud providers, and centralized findings management systems. Together, these components abstract the complexity of scanning away from developers and other stakeholders, enabling full centralization while still allowing complete flexibility and customization.

In this model, the COTS scanning tool is used solely for executing scans, while all supporting workflow-scheduling, orchestration, prioritization, normalization, and reporting are fully managed by the scan factory. Application security engineers and tool administrators will continue to use the COTS scanner's native portal for tool configuration, license management, and operational settings.

These services can be deployed on any cloud provider depending on organizational preference. The specific cloud architecture, infrastructure components, and deployment models are beyond the scope of this paper, as these implementation details will naturally vary across organizations and should be tailored to meet their unique operational and security requirements.

6. Evaluation

In this section, we compare three distinct approaches to implementing an application security scanning program -

- The traditional decentralized model, where different types of scans are performed by separate COTS tools integrated into CI/CD pipelines and maintained by developers or DevOps engineers;
- The default-integration model, where organizations rely solely on the built-in connectors and automation features provided by each COTS tool; and
- Our proposed centralized scan factory architecture.

The traditional pipeline-based approach places significant operational responsibility on development teams, who must configure, maintain, and troubleshoot scanner integrations while also interpreting scan results. The default-integration approach reduces this burden but is limited by the features and constraints of the vendor-provided integrations, often lacking flexibility in scheduling, scan-trigger mechanisms, prioritization logic, and multi-tool coordination.

In contrast, the centralized scan factory aims to provide full coverage, consistency, and flexibility by abstracting all scanning workflows into a unified system independent of development teams and individual tool limitations.

To evaluate these three models, we present a qualitative comparison using common industry workflows and real-world use cases encountered in large enterprises. We analyze how each approach performs in scenarios involving scan coverage, operational overhead, prioritization, scalability, and findings management. Additionally, we provide case studies drawn from a Fortune 500 environment to illustrate how each approach behaves in practice and to highlight the specific advantages of the centralized scan factory design.

6.1. Case Study: NPM shai-hulud

7. question for COTS scanners

Why should we trust them with our data, who is scanning the scanner, closed-source, scan as a library.