Scenario: **AppSec Goals for DevNest Tech**

Application Security Foundations Level 1

From Semgrep

**Company Name:** DevNest Tech  
**Size:** 25–30 employees  
**Type:** Small-scale software development startup  
**Work Culture:** Fully remote  
**Environment:** Cloud-based (AWS)  
**Teams:** Frontend Developers, Backend Developers, Database Admins, IT Operations  
**Security Team:** Only one person—me, the sole AppSec Engineer

I have set 3 main goals for the integration of security in the SDLC environment in the company:

**Goal #1: Integrate Security in CI/CD Pipelines Across All Projects**

* Ensure every software project at DevNest Tech integrates automated security scans in its CI/CD pipeline within 3 months.
* This includes

1] Static Application Security Testing (SAST),

2] Secret scanning

3] Open-source dependency analysis.

**STEP 1: In this first step, I will try to understand the development environment completely. [1 week]**

**Action**: Initiate informal 1:1 coffee chat with each team lead (Frontend, Backend, DevOps). Ask questions regarding how the current CI/CD pipeline looks, what tools are already in use, and what is the average duration of sprints and deployments. Also, I would ask them what part of the process they think needs improvement or what part of the whole process of development is slower and frustrating.

**Reason:** Understanding workflows helps me avoid friction. If DevOps is already using GitHub Actions or GitLab CI, I tailor integration scripts accordingly. If frontend teams push directly to S3 or CloudFront without a build pipeline, I need to work with them differently.

**STEP 2: Selecting the right tools and optimizing the whole process of integration of the development cycle and security. [2 weeks]**

* **Action**: Chose to implement the following tools:

 **Semgrep** for lightweight, customizable SAST.

 **Gitleaks** for secret detection.

 **OWASP Dependency-Check** for open-source vulnerability scanning (depending on languages used)

 Write initial YAML templates to plug into GitHub Actions (reusable workflows).

* **Reason:** Tested rule severity levels, filter thresholds, and false positive tuning with developers. Off-the-shelf tools are great, but I had to tune the rules to avoid overwhelming devs with noisy results. I made sure only critical/blocker-level issues failed builds initially.

**STEP 3: Providing Training to developers and collaboration. Also, providing them with easy-to-access resources regarding the usage of newly implemented tools, if needed.**

**[2 weeks]**

* **Action:** Run a 30-minute async recorded session called “Why CI Security Matters” and shared it on Slack.
* Created a Slack channel #secure-pipelines to:

1. Log pipeline alerts
2. Collect developer feedback
3. Drop cheat sheets for Semgrep rules and custom suppressions

* Created a Confluence page with:

1. Few sample alerts + fixes
2. Guidance on “How to handle a false positive”
3. Easy-to-Access CI configs, if encountered problem with any configuration-related issues with software and repositories.

* **Reason**: Security adoption thrives when developers feel empowered, rather than policed ownership comes through visibility and collaboration.

**Step 4: Taking regular feedback from all the teams, implementing integration checks and making sure the integration works smoothly. So that we can move forward to integrating other required tools.**

* **Action:**
* After 2 weeks of pilot feedback, expand integration to all actively maintained repos.
* Paired with devs to do 20-minute “pull request review” parties where we add scanning to their repos.
* Monitor build failures and issues caused by the integration and fix them quickly.
* **Reason:** Security is all about trust. If you cause downtime or frustration, your program will be resisted. Gradual rollout + fast support = high adoption

**Step 5: Implemented dashboards that will show scan results for vulnerabilities found, and we can measure the success and growth of our implemented security integration technologies.**

* **Action:** Created a dashboard (Grafana + Prometheus + webhook logs or GitHub API) tracking to show results for metrics like:
* projects with scans enabled
* high/critical issues caught per week
* False positive/true positive ratio
* **Reason:** As a security professional, I would need statistics and metrics to measure the work done in order to see how beneficial it is and what should be improved. The metrics help us to understand what things and parts of software are being scanned through and what are left without scanning.