



ZINAD DevSecOps: From Foundations to Mastery

How we assess, roadmap, and operate secure delivery at scale

DevSecOps vs. the Classic SDLC

Traditional software development life cycles (SDLC) have long relied on sequential phases: requirements, design, development, testing, deployment, and maintenance. Security in this model has typically been an **afterthought**, introduced late in the cycle through penetration tests, audits, or compliance checks. This "security as a gate" approach often leads to late discovery of vulnerabilities, increased remediation costs, and strained relationships between delivery and security teams.

DevSecOps changes this by weaving security into every stage of delivery. It introduces **continuous security verification**, shifts security left into design and code, and extends it right into production monitoring. Importantly, DevSecOps redistributes responsibility: developers, security architects, and platform engineers **share ownership** of assurance.

This vision is codified in the **OWASP DevSecOps Verification Standard (DSOVS)**, which provides a comprehensive set of control families covering organization, requirements, design, code, build, test, release, and operations. For benchmarking maturity, the **OWASP DevSecOps Maturity Model (DSOMM)** helps assess the degree of adoption across practices. Together, these frameworks give organizations a structured way to evaluate where they are and how to progress.

ZINAD's Four-Level DevSecOps Maturity Model

At ZINAD, we use a **four-level maturity model** to help organizations understand their current posture and define a clear improvement path.

01	02
Level 0 – Foundation Basic controls exist but many activities are missing, manual, or fragmented. Gaps include lack of structured risk assessments, missing security training, limited code scanning, and weak monitoring.	Level 1 – Development Security practices exist but are siloed or manual. SAST and DAST may be used, but not automated. Cloud and IaC security are inconsistent. Secrets may still be stored insecurely.
03	04
Level 2 – Advancement Structured practices spread across teams. Pipelines are hardened, artifact integrity is enforced, IaC scanning is integrated, and incident response starts to connect with delivery pipelines.	Level 3 – Mastery Security is fully embedded and continuously assured. Policy-as-code governs compliance, SBOMs are generated automatically, runtime protections secure live environments, and incident response is automated through playbooks.

Each stage aligns to specific **DSOVS control families** and can be benchmarked against **DSOMM levels**, giving organizations measurable evidence of progress.

How ZINAD Assesses Gaps

Our DevSecOps gap assessment combines:

- DSOVS as the control catalog** – ensuring we verify against internationally recognized controls across design, build, test, release, and operations.
- DSOMM as the maturity benchmark** – helping prioritise improvements by identifying whether practices are ad hoc, repeatable, defined, or optimised.





We assess all lifecycle dimensions:

Organisation & Requirements Risk management, training, champions, user stories.	Design Architecture reviews, threat modelling.
Code & Build Secure environments, SAST, secrets scanning, SCA/SBOM, container security.	Test DAST, IAST, penetration testing, coverage metrics.
Release & Deploy Artifact signing, policy gates, IaC security, compliance scans.	Operate & Monitor Hardening, logging, vulnerability disclosure, certificate management, attack surface management.

The outcome is a structured view of where the organisation sits at each level and where the critical gaps lie.

Roadmap to Closing DevSecOps Gaps

A roadmap is not just a checklist of tools to buy or policies to write. At ZINAD, we design roadmaps as a **strategic journey**—progressing through phases that build resilience across people, process, and technology.

	Early stage (Stabilisation) The focus is on visibility and standardisation . We establish automated security scans in CI/CD (SAST, SCA, secrets detection), formalise risk assessments, and embed security acceptance criteria into user stories. Logging and monitoring baselines are introduced, and issue-tracking workflows ensure vulnerabilities flow into the same backlogs as feature work.
	Intermediate stage (Integration) Security becomes embedded into delivery. We harden build systems, enforce artifact signing and verification, and introduce policy-as-code for IaC and compliance. DAST and IAST testing expand coverage, whilst secrets management moves to centralised vaults. Incident detection starts to connect directly to delivery workflows, reducing time to respond.
	Advanced stage (Optimisation) At this point, security is applied systematically across pipelines and environments . Runtime protections like container scanning and Kubernetes admission controls are deployed, compliance dashboards give executives visibility, and attack surface management integrates with operations. Incident response playbooks become semi-automated, ensuring predictable, repeatable response.
	Mature stage (Continuous Assurance) Finally, security becomes a continuous, business-aligned capability . SBOMs and compliance evidence are generated automatically with every release. Policy-as-code governs every deployment, exceptions are time-bound, and runtime monitoring (e.g., eBPF) detects anomalies in real time. Vulnerability disclosure and bug bounty programmes provide external assurance, whilst metrics like MTTR and first-pass pipeline success rates tie security directly to delivery performance.

This roadmap ensures each milestone reinforces the last. Organisations gain **short-term wins** (visibility, reduced blind spots) whilst also investing in **long-term transformation** (automation, culture, resilience).

ZINAD's DevSecOps Operating Model

To make DevSecOps sustainable, ZINAD defines a full operating model that spans **services, processes, interfaces, KPIs, SLAs, escalation, and RACI**.

Services Secure CI/CD as a service, code and dependency assurance, IaC guardrails, release integrity, runtime security, and threat intelligence integration.	KPIs MTTR, % of builds passing first attempt, IaC policy coverage, vulnerability density per KLOC.
Processes Secure development enablement, pipeline governance, release/change control, operate/monitor with vulnerability disclosure and ASM.	SLAs Critical vulns patched within 48h, high vulns block pre-prod, new pipelines onboarded with secure templates within 2 weeks.
Interfaces Engineering consumes templates; platform provides services; architects define guardrails; executives consume metrics.	Escalation Developer → Security Champion → Platform Team → CISO/Steering Committee.
	RACI Clear accountability for defining checks, implementing templates, adopting practices, and reporting metrics.

Conclusion

DevSecOps is not simply about automating scans or buying tools—it is about **building an operating model where security is continuous, measurable, and scalable**. By aligning with **OWASP DSOVS** for verification and **DSOMM** for maturity benchmarking, and by following ZINAD's four-level journey from Foundation to Mastery, organisations can transform security from a gate into a true enabler of digital delivery.

References

- OWASP DevSecOps Verification Standard (DSOVS): <https://owasp.org/www-project-devsecops-verification-standard/>
- OWASP DevSecOps Maturity Model (DSOMM): <https://owasp.org/www-project-devsecops-maturity-model/>