

APPLICATION
SECURITY (APPSEC).

Vulnerabilities are cheapest to fix before code ships — **AppSec builds security checks directly into the development pipeline**, catching flaws in code, dependencies, and configuration before they reach production.

- SECURE CODING
- SAST / DAST
- DEPENDENCY SCANNING
- REMIEDIATION WORKFLOW

01 THE MECHANISM



Secure Coding Practices
Building security into the code itself through **standards, training, and peer review.**



Static & Dynamic Testing
Scanning source code and running applications to **surface exploitable flaws before release.**



Remediation & Verification
Prioritizing findings by risk and confirming fixes **close the vulnerability before re-deployment.**

02 THE FLOW

The Development Cycle CONTINUOUS

1

Code is written and scanned for **known vulnerability patterns before commit.**

2

Builds run through automated **SAST, DAST, and dependency checks** in the pipeline.

3

Findings are triaged by severity and **routed back to developers for remediation.**

The Vulnerability Cycle LIVE EVENT

1

A scan or manual test uncovers an exploitable flaw **in code or a dependency.**

2

The finding is validated and scored using **a severity and exploitability framework.**

3

A fix is developed, tested, and verified **closed before the next release ships.**

03 CORE PILLARS & COVERAGE

PILLAR	WHAT IT COVERS
Secure Coding	Establishes standards and training that prevent vulnerabilities at the source.
Static & Dynamic Testing	Scans code and running applications to catch flaws before and after build.
Dependency Scanning	Identifies known vulnerabilities in third-party libraries and packages.
Remediation Workflow	Tracks findings from discovery through verified fix and closure.

04 APPLICATION SECURITY APPROACHES COMPARED

SIGNAL	SUFFICIENT ALONE?	WHAT IT MISSES
Manual Code Review Alone	No	Doesn't scale and misses flaws outside reviewer expertise.
Perimeter Firewall & WAF	No	Filters known attack patterns but not flaws in the application logic.
Penetration Testing Only	No	A point-in-time snapshot that misses flaws introduced after the test.
Production Monitoring Alone	No	Detects exploitation after the fact, not the underlying flaw.
AppSec (Secure Coding + Testing + Remediation)	Yes	Builds security into the pipeline and catches flaws before production.

APPLICATION SECURITY.

CODE · TEST · REMEDIATE

05 PROGRAM OPERATIONS

Security debt accumulates silently. Unaddressed findings pile up until a release is blocked entirely.

Developers aren't security experts by default. Training and tooling have to close that gap, not assume it away.

Dependency risk is inherited, not created. Most vulnerabilities come from third-party code, not custom logic.

Speed and security compete for the same sprint. Without automation, security review becomes the bottleneck.

06 CONTROL MAPPING

OWASP TOP 10

Baseline risk categories guiding what secure coding and testing must address.

NIST SP 800-218

Secure development practices covering the full software lifecycle.

PCI DSS

Application security requirements for software handling cardholder data.

ISO 27001

Secure development lifecycle controls covering system acquisition and engineering.

07 WHAT THIS DOES NOT SOLVE

Zero-day discovery

AppSec catches known and pattern-based flaws; it can't guarantee novel exploits are caught.

Business logic abuse

Automated scanners struggle to see valid-looking requests that abuse legitimate functionality.

Infrastructure security

Secure code doesn't fix a misconfigured server or exposed cloud storage bucket.

Developer buy-in

Tooling doesn't force adoption; the program still needs a culture that treats security as part of the work.

08 MYTHS VS. REALITY

MYTH	REALITY
"A clean scan means the app is secure."	Scanners miss business logic flaws and novel attack chains.
MYTH	REALITY
"Security testing belongs at the end of the cycle."	Late-stage fixes are far more expensive than catching issues early.
MYTH	REALITY
"Open source code is inherently safe."	Dependencies carry the same vulnerabilities as custom code, often unpatched.
MYTH	REALITY
"One scan type covers everything."	SAST, DAST, and dependency scanning each catch different flaw classes.
MYTH	REALITY
"Fixing the finding count is the goal."	Prioritizing by exploitability and impact matters more than volume.

09 SPEAK THE PROGRAM — KEY TERMS

APPSEC

Application Security: practices and tooling that find and fix vulnerabilities in software.

SAST

Static Application Security Testing: scanning source code without executing it.

DAST

Dynamic Application Security Testing: testing a running application from the outside.

DEPENDENCY SCANNING

Checking third-party libraries for known, publicly disclosed vulnerabilities.

CVE

Common Vulnerabilities and Exposures: a standardized identifier for a known flaw.

REMEDIATION

The process of fixing and verifying a closed vulnerability.