Securing Software Delivery Pipelines in the Age of AI

Introducing OWASP SPVS



Farshad Abasi & Cameron Walters

Co-leads, OWASP Secure Pipeline Verification Standard

LASCON 2026 · Austin · October 30

Two AppSec guys, one frustration

Cameron Walters

Director, AppSec & Security Engineering

Co-lead, OWASP SPVS

Co-host, Coffee & Chaos and the ProdSec Podcast

Leader of Austin Meetup, CYBERLUNCH

Farshad Abasi

Founder & CEO, Forward Security

Co-lead, OWASP SPVS

OWASP Vancouver chapter lead

28+ years in security, incl. senior security architecture at HSBC

“Where’s the ASVS for pipelines?”



Who’s in the room?

DEV

APPSEC

SOC

GRC

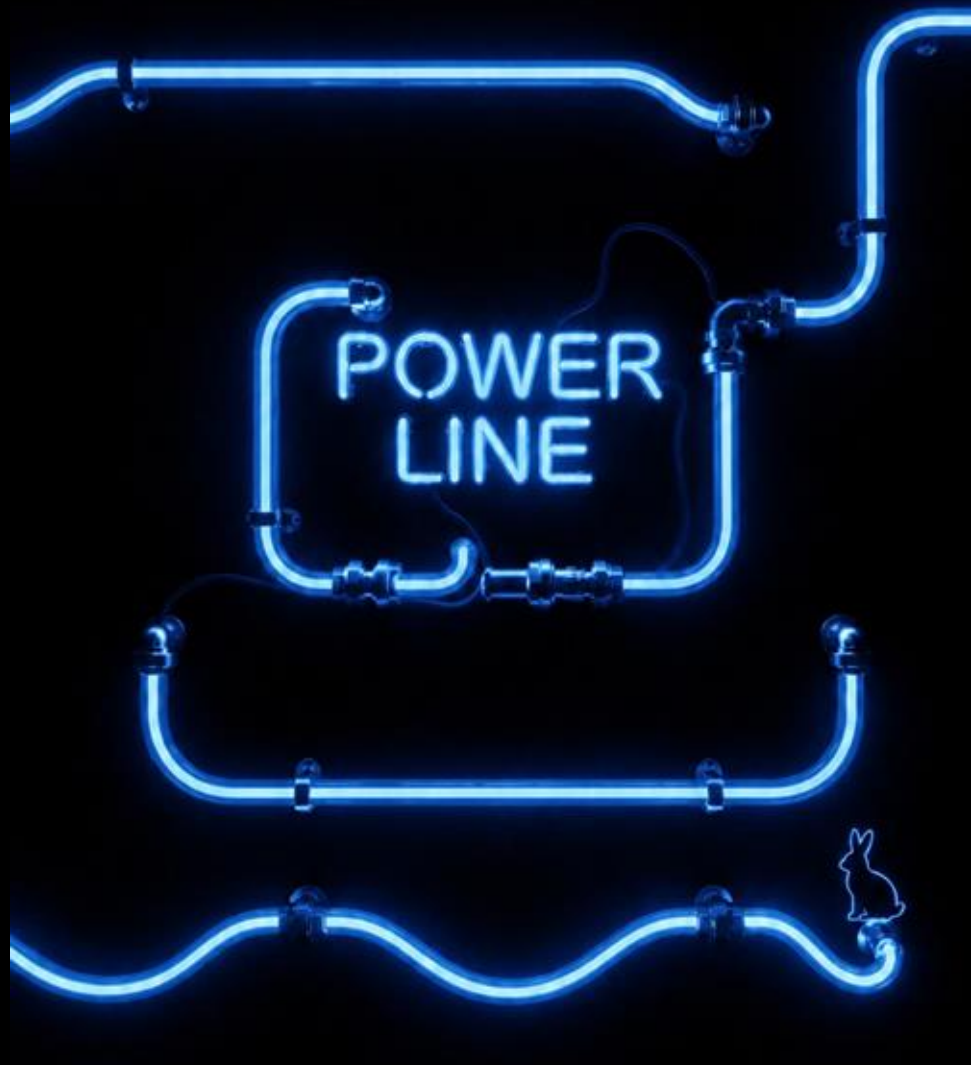
STUDENT

Hands up. It changes how deep we go.

01

What is a pipeline?

Before we secure it, let’s agree on what it is.



The pipeline is not the tool

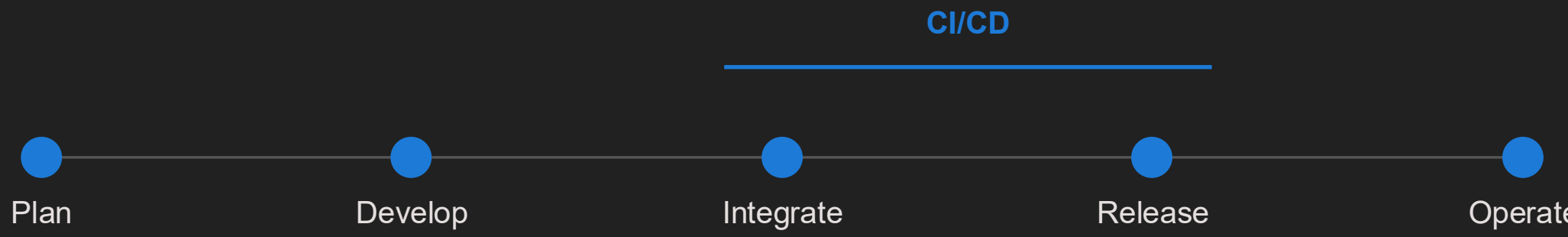
Everything between a developer's first line of code and production.

The pipeline

Developer training	Dev endpoints	Source code	Package managers
Secrets	**CI/CD tool**	Build agents	Artifact registries
Production credentials	Deployment infrastructure	Agents and MCP servers	Runtime operations

Your pipeline has root access to everything that matters.

Source · Secrets · Production · Infrastructure

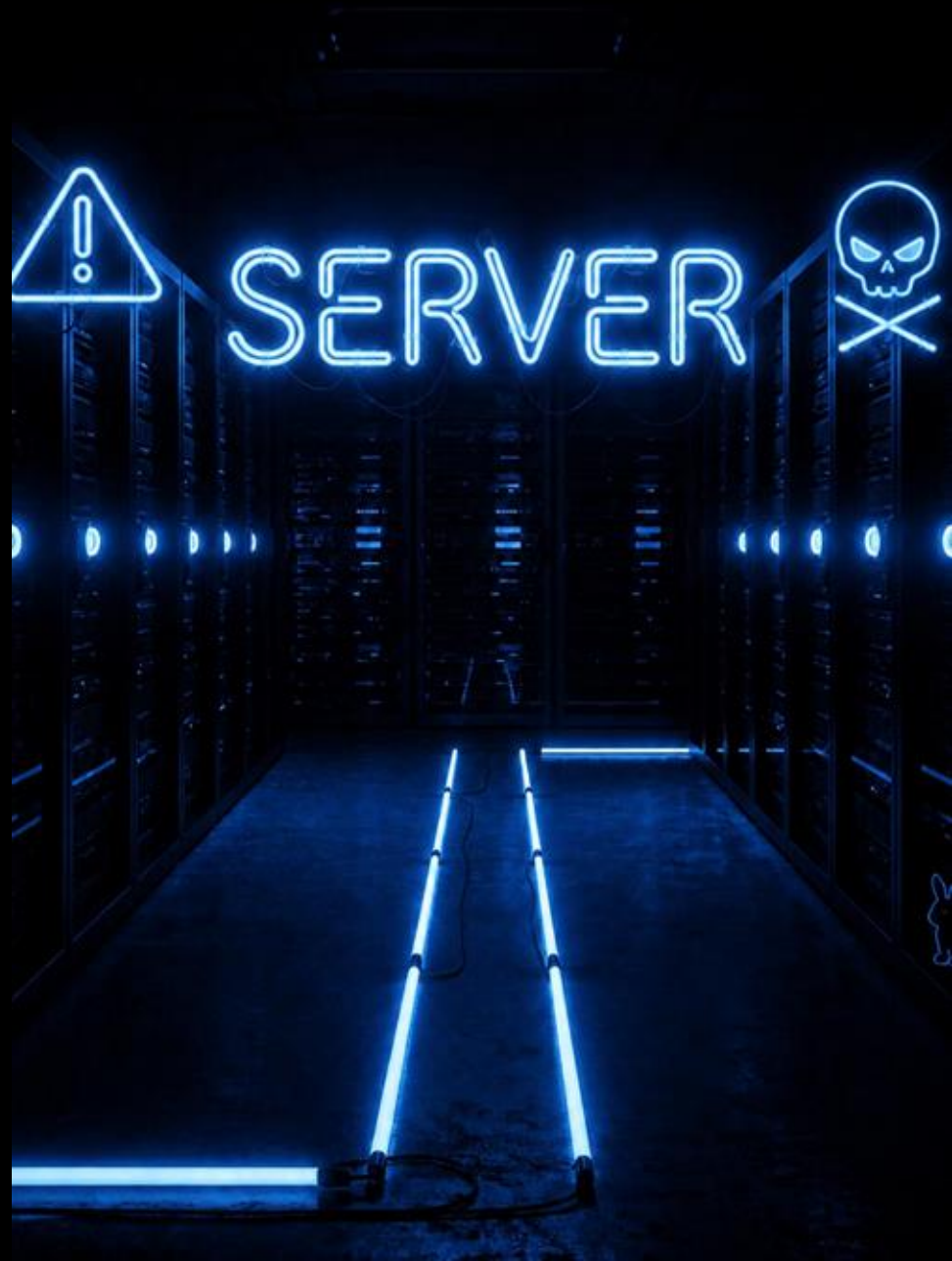


CI/CD is two of the five stages. SPVS covers all five.

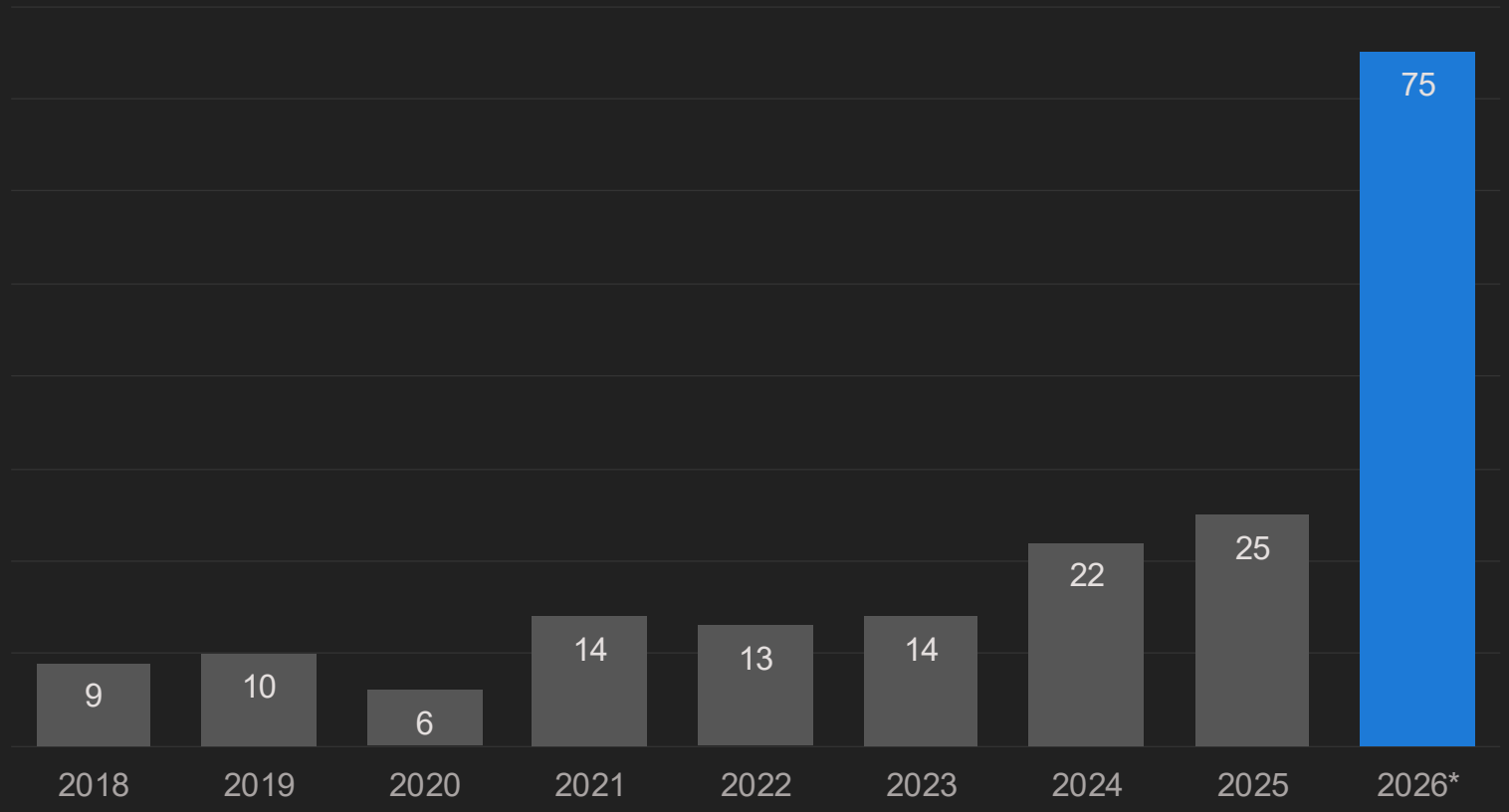
02

What’s happening out there

The threat landscape has shifted, and pipelines are a primary target.



By July, 2026 had already tripled 2025



75

incidents, Jan to Jul 2026

25

in all of 2025

* 2026 through July

Source: Cameron Walters, compiled from public incident reporting, 2018 to July 2026

TeamPCP: one incomplete rotation, then a cascade

March 2026

Trivy

Service account kept access after an incomplete credential rotation

→

trivy-action

Release tags force-pushed to malicious code

→

Checkmarx KICS

Stolen tokens used to pivot days later

→

LiteLLM on PyPI

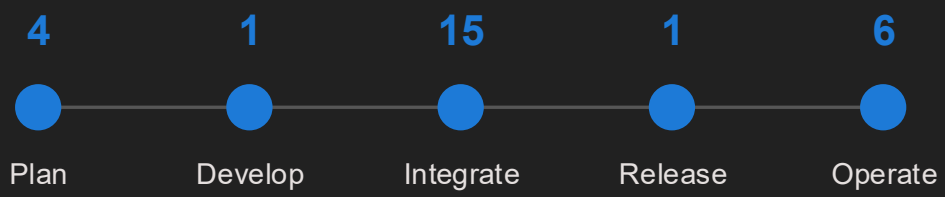
Poisoned versions harvest cloud and CI/CD credentials

LiteLLM itself runs in ~36% of cloud environments (Wiz).

Receipts in SPVS 1.6

V3.3.1	Scanners are integrity-checked before they run
V3.4.7	Release tags are immutable; overwrites rejected and logged
V5.3.1	Compromised credential revoked before its replacement goes live

27 controls in 1.6 trace to TeamPCP



Axios: no code change, just one new dependency

March 31, 2026

Account takeover

Maintainer socially engineered; npm account hijacked

→

Two versions

1.14.1 and 0.30.4 published from the CLI with a long-lived token, no provenance

→

plain-crypto-js

A postinstall script drops a cross-platform RAT

→

~3 hours live

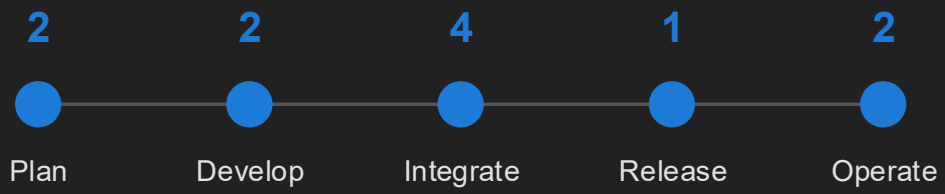
Then removed by npm

Teams installing from a committed lockfile stayed on the version they had tested.

Receipts in SPVS 1.6

V1.1.1	Phishing-resistant MFA on package-publishing accounts
V2.2.1	Install-time scripts off by default
V3.4.2	Signed provenance, validated before use

11 controls in 1.6 trace to Axios



GitHub: trusted extension, poisoned update

May 2026

~3,800

GitHub-internal repos exfiltrated, per TeamPCP's claim

<20 min

a backdoored Nx Console update was live on the VS Code Marketplace

`SPVS 1.0 · V1.4.3 · L1`

Verify developer IDE/code writing tool plugins are routinely updated

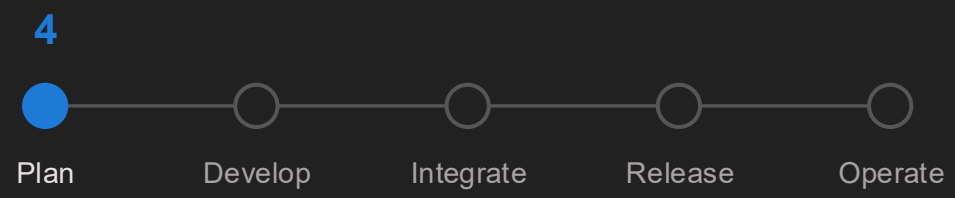
→

`SPVS 1.6 · V1.3.3`

Approved list, pinned version, checksum verified. Auto-update disabled or gated.

1.6 adds monitoring of extension inventory (V1.3.1) and least-privilege repo read access (V1.1.3).

4 controls in 1.6 trace to this breach





03

The standards gap

We have frameworks. We don’t have a verification standard for the pipeline.

Good frameworks, each covering one piece

Framework	Covers	Leaves to you
NIST SSDF	Secure development practices	Implementation and audit criteria
SLSA	Build provenance, attestation	The rest of the lifecycle
SCVS	Component supply chain	The delivery process
S2C2F	Secure dependency consumption	The rest of the pipeline
CIS Benchmarks	Configuration baselines	Stage-level pipeline controls
OWASP CI/CD Top 10	Risk awareness	Verification, maturity, assessment

Missing: one verification standard, by pipeline stage, with testable controls and maturity levels.

The application

ASVS

Testable. Tiered. Widely adopted.

The machine that ships it

?

Nothing equivalent. Until SPVS.

We had ASVS for the application, and nothing for the machine that ships it.

04

Introducing OWASP SPVS

A practical framework for assessing and improving pipeline security across the full lifecycle.



Five stages, plan to operate



Plan	Develop	Integrate	Release	Operate
62	**45**	**100**	**42**	**43**
Threat modeling, security requirements, risk baselines, hardening developer machines	Secure coding standards, code review, automated detection while developers write code	Security checkpoints, artifact integrity, build environment hardening, repeatable scans	Final validation, change control, approvals, secure rollout	Monitoring, incident response, patch management, access control

Controls published per stage across 1.0, 1.5 and 1.6 (292 total). 2.0 consolidates and de-duplicates them.

Three levels, cumulative

L3 — Advanced
Continuous verification, independence and assurance at scale.

L2 — Standard
Adds automation, monitoring and evidence of enforcement.

L1 — Foundational
Baseline pipeline hygiene. Simple, enforceable controls.

Start at Level 1 as your minimum baseline. Each level includes everything below it.

Anatomy of an SPVS control

Integrate · Security of Pipeline Environment

Lockfile-Exact Dependency Installation

Verify that build and CI/CD pipelines install dependencies using a command that installs exactly what's listed in the lockfile, instead of a command that can pull in newer versions on its own.

Level	Origin	NIST 800-53	NIST SSDF	CI/CD Top 10
L1	1.6, V3.1.1	SA-10	PW.4.1	CICD-SEC-3

Every control starts with "Verify", sits in one stage, has a target level, and maps to NIST 800-53.

How to start with SPVS

1

Map

Every component of your pipeline: repos, build, deploy, IaC, monitoring

2

Target

Pick a level. Start at Level 1

3

Assess

Each control: implemented, partial or not. Evidence and an owner

4

Fix

Close gaps at your target level first, by impact and feasibility

5

Make it live

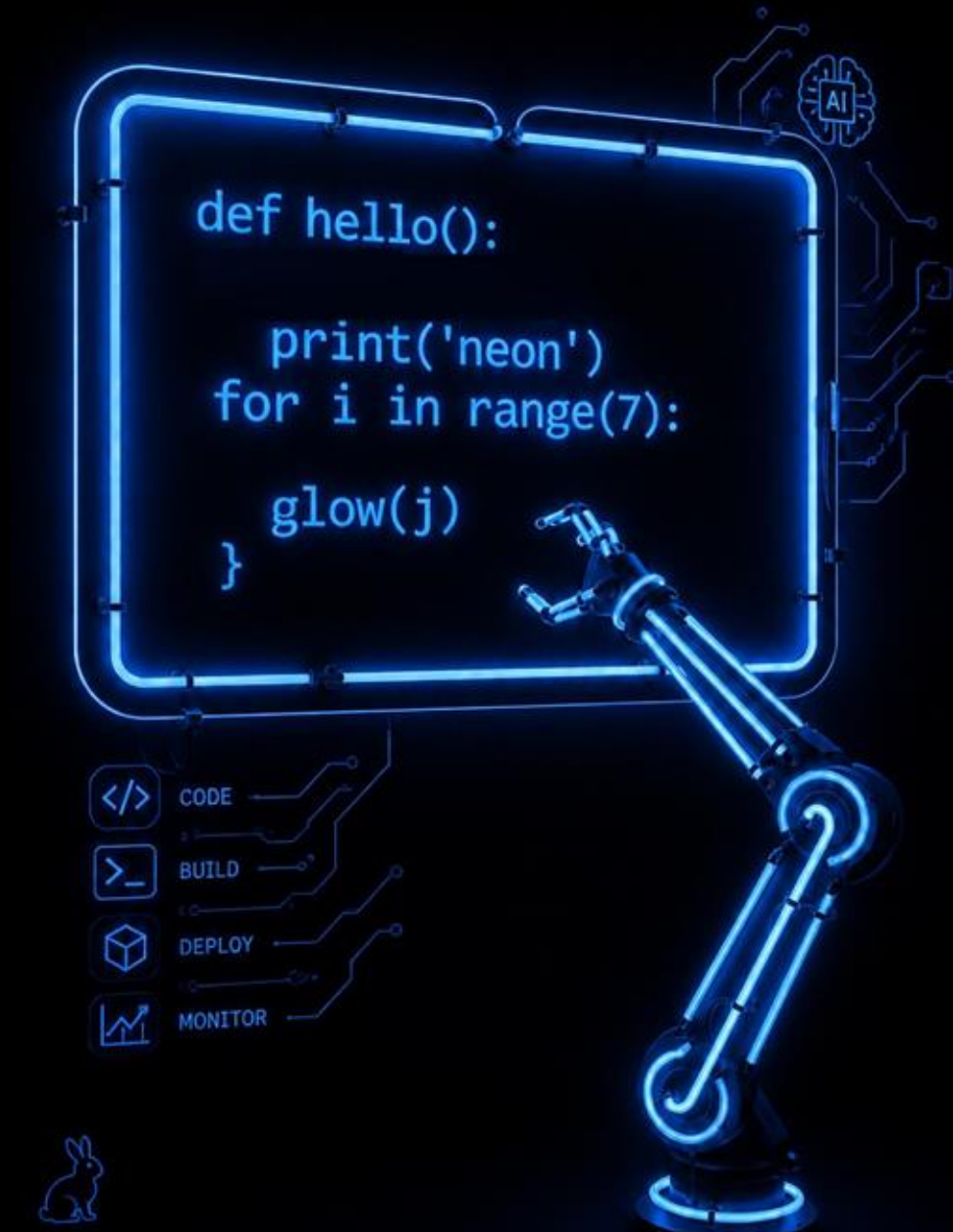
Policy-as-code and pipeline checks. Reassess after major changes, at least yearly

Use it as an internal checklist or an external audit baseline.

05

Pipelines in the age of AI

The pipeline is no longer just CI/CD hardening.



Three assumptions that no longer hold

01

The artifact holds still

An agent rewrites the code many times before review. The final diff hides the process your gate never saw.

02

Humans set the pace

Weekly change boards and quarterly reviews were calibrated to human output, not continuous agent output.

03

You already govern the actor

Agents arrive with no owner, no expiry, a static credential and whatever scope made the task work.

SPVS 1.5 added 132 controls for AI in the pipeline.

Same objectives, different enforcement

Traditional SDLC	Agentic SDLC (our view)
Threat model scheduled, output in a doc	Threat model updated continuously; drift between designed, built and running flagged
One developer writes, another reviews the diff	Coder, review and remediation agents kept separate; review covers diff and reasoning
SBOM generated at build, filed once	AIBOM records the AI tool, model and version, reconciled against what's deployed
Change board approves the release	Machine-checkable evidence assembled automatically; incomplete evidence blocks release
Security watches the app; pipeline trusted	Pipeline is monitored surface; every agent action attributable

What we’re seeing in the wild

01 **More access control issues in AI-generated code**

v1.5-V2.2.2: AI-generated code gets security analysis before merge

02 **Governing bodies requiring more frequent vulnerability identification and remediation**

Annual and quarterly cadences don’t survive continuous delivery

03 **Many teams still don’t have a clear picture of their pipeline security**

SPVS step one: map what your pipeline actually is

SPVS 1.5 AI Examples

Foundation		Enforcement	
Register every agent: owner, scope, expiry	V1.3	Agents can’t approve their own output	V4.4
Log agent actions under that identity	V5.3	Human gate before production deploys	V4.4
Isolate agent runtimes, allowlist egress	V3.8 / V3.9	Gate irreversible actions, rate-limit agents	V5.4
Agent secrets denied by default	V3.7	Bound what agents can delegate	V3.3
System prompts versioned and reviewed	V4.7	Federated identity, no static PATs for agents	V3.6

1.5 sub-category IDs. 2.0 renumbers them.

SPVS 2.0: one standard

1.0

Oct 2025

110 controls

Five-stage baseline

+

1.5

Apr 2026

132 controls

AI pipeline security

+

1.6

Aug 2026

50 controls

Supply chain, incident-mapped

=

2.0

Consolidated, de-duplicated, community-reviewed

Built on community feedback from 1.5 and 1.6.

Questions?

Get involved

github.com/OWASP/www-project-spvs

#owasp-spvs on OWASP Slack

linkedin.com/company/owasp-spvs

Farshad

linkedin.com/in/farshadabasi/

Cameron

linkedin.com/in/cameronww7

Check out the project

Get the slides here



SPVS co-authors: Farshad Abasi, Cameron Walters, Evan Borysko