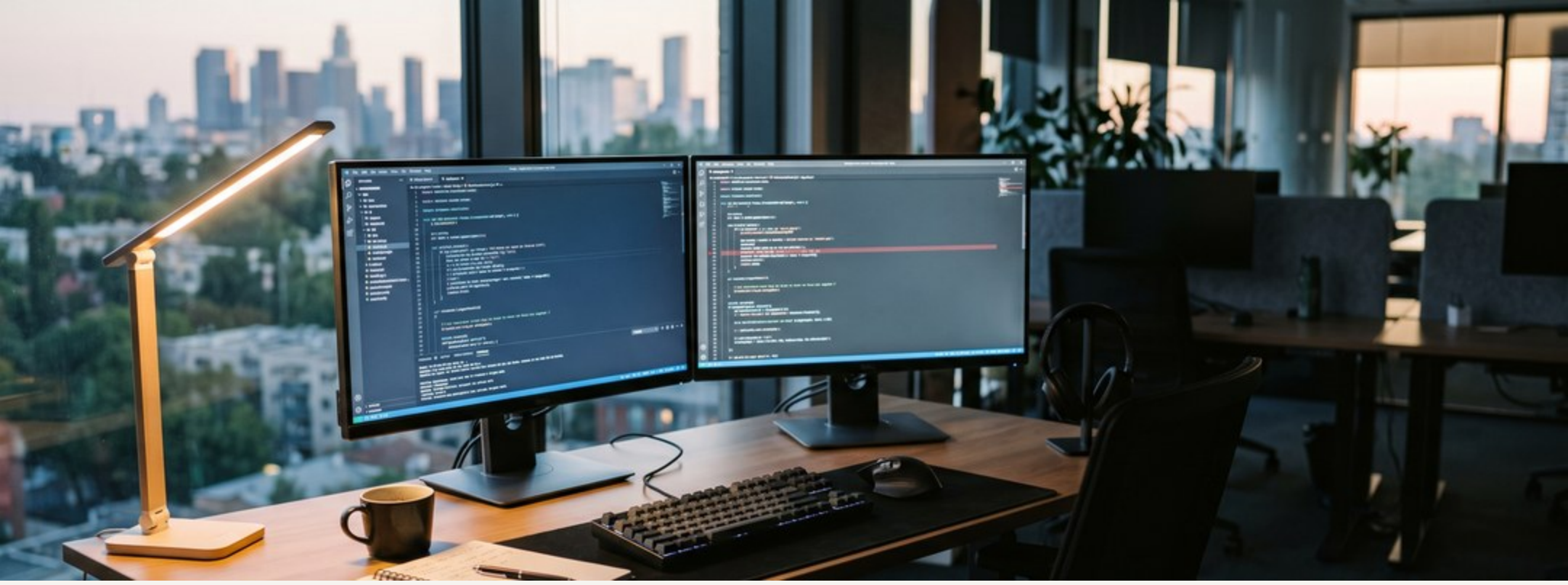
SNOK

Jacek Bugajski • CEO, SNOK

It compiles, so it works

ABAP in the age of the coding assistant

On the barrier that vanished and the signature that lost its backing



01

For thirty years ABAP defended itself

Not through its syntax, which forgives almost everything. It defended itself through the barrier to entry: the data dictionary, the authorization concept, module logic and the ritual of moving transports across three systems.

None of that lived in documentation. It lived in the heads of people who had survived a few implementations and a few failed go-lives. It was passed on at a desk, not in a course.

That barrier is now gone. Not over a decade - in a matter of months.

02

What changed under the hood

SAP released a foundation model trained exclusively on ABAP. SAP-ABAP-1 learned from millions of lines of code: standard objects, customer extensions, BAdI implementations, ABAP Cloud patterns. SAP Community materials put the figure above 250 million lines.

It powers code explanation and generation in Joule for Developers, built into the tooling in Eclipse and in SAP Business Application Studio. In 2026 an ABAP extension for Visual Studio Code was added.

This is not only a story about ABAP developers getting a faster tool. To write something in SAP that compiles and passes the functional test, you no longer have to be an ABAP developer.

A functional consultant will write that report.
An analyst from the data team will write one.
So will a junior three weeks into the project

The code compiles

That proves nothing

The SAP compiler never checked the things
that are actually dangerous in this platform

SNYK

2.74x

more security findings appeared in pull requests co-authored with AI than in those written by humans alone. General issues - logic, performance, maintainability - appeared 1.7 times more often

CodeRabbit, State of AI vs Human Code Generation Report, December 2025: 470 open-source pull requests. A vendor study

Code from an assistant has one property that disarms scrutiny better than any other: it works

SNYK

03

What the compiler never checked

Authorization checks

whether an AUTHORITY-CHECK stands in front of that payroll read

Injection

whether a query assembled from user input can be taken apart

RFC modules

whether the exposed interface verifies the caller's authorizations

Table writes

whether data goes through application logic or straight into the table

Output encoding

whether a field reaching the browser is encoded

The model did not skip these out of malice. It simply does not know them - it has never seen your authorization concept or your segregation-of-duties matrix

04

Three numbers that make this an operational topic

3–4x faster do developers ship code when working with an assistant, while their security findings grew tenfold in six months (Apiiro)

45% of coding tasks produced code with a vulnerability (Veracode, over 100 models)

< 3 h is how long an unsecured SAP application in the cloud survives before it is found and attacked (Onapsis)

The assistant does not write worse than a person. It writes far more, while the number of people reading that code has not gone up by one.

05

Clean Core raised the stakes

For years, custom code sat inside your own four walls. Written carelessly perhaps, but behind a firewall, in a system reachable by a few dozen people.

Clean Core changed the geography. Extensions move out of the core onto BTP and become cloud applications. The direction is right and I recommend it myself. The side effect is that the same code, written with the same care, now hangs on the internet.

The same conclusion ran through the joint Deloitte and Onapsis webinar: Clean Core without moving security controls to the start of the process is a half measure.

06

Who signs

SAP has something most of IT does not: a physical, named moment of accountability. Someone releases the transport, someone else imports it into production. Both names stay in the log.

That mechanism was designed when releasing a transport meant: I have read this, I understand it, I take it on. Today it increasingly means: I approved a suggestion I did not read.

The developer's role is shifting from writing to validating, and that shift is healthy. The trouble is that when teams handed the writing to the machine, nobody added capacity for the reading.

The signature has not changed.
What stands behind it has

07

There is a remedy now

I am not writing this to frighten anyone. I am writing it because the control can be wired into three places SAP cannot bypass.

Code passes through the editor, through the transport and through the production system. At each of those points a red light can come on - before it comes on at your customer's.

In the editor

the flaw shows up in the same minute it appeared, not three days before go-live

In the transport

scanning the package is a precondition for import, so a transport without an authorization check does not move on

In production

real-time detection catches attempts to exploit whatever slipped through the sieve

At SNOK we build these three layers on SecurityBridge. I have followed that product since the company's early days, and it is one of the few security tools for SAP that grows alongside the platform.

Three things for Monday

Find out who actually writes code in your SAP systems. Not in the org chart, in the system - the list of authors of custom objects over the past six months can be surprising.

Put the security scan into the transport, not into the calendar. A control that happens once a quarter in practice never happens.

Decide what your signature on a transport means. One sentence in the policy, so everyone releasing knows whether they confirm code read or function accepted.

The assistant will not stand in front of an auditor

So what does a transport signature mean in your organisation?
Join the discussion below

Jacek Bugajski · jacek.bugajski@snok.ai · snok.ai
Sources: CodeRabbit Dec 2025 (470 pull requests, vendor study) · Veracode 2025 · Apiiro Sep 2025 · Onapsis, June 2026 survey and Threat Research · SAP and SAP Community

SNOK