Software Assurance Tips

A product of the Software Assurance Tips Team[5]

Jon Hood

Monday 25th May, 2026

1 Software Assurance Tools Belong in Expert Hands

Updated Friday 22nd May, 2026

1.1 Introduction

There is a comforting narrative in modern DevSecOps: install a suite of Software Assurance tools into the developer's pipeline, hand every engineer a dashboard, and the codebase will defend itself. The narrative is comforting because it is convenient. It also misunderstands what these tools actually are. Static analyzers, composition analyzers, fuzzers, and supply chain scanners are precision instruments built by specialists for specialists. Exposing scan results to developers is healthy. Letting developers (or, increasingly, a build server with broad credentials) own the selection, configuration, deployment, and maintenance of those tools is not.

Three news items from the last six weeks make the case better than any opinion piece can. Each of them involves a Software Assurance tool that was either misconfigured by non-experts, misunderstood by its users, or weaponized against the very pipelines it was installed to protect.

CppCheck: "Trusted/Safe Environments" Is Not a Suggestion

On 12 May 2026, Daniel Marjamäki, the long-time maintainer of Cppcheck, opened ticket #14749 against his own project. The defect is small in scope and large in implication:[9]

> We use popen to execute addons. Even though I believe Cppcheck should only be used in trusted/safe environments, we could be careful about the commands that is executed to not leave a door open for some kind of remote execution vulnerabilities.
>
> We should escape or reject certain characters in the command line that the shell could treat in unwanted ways. I.e. a ";" in a filename could be treated as a command separator. Filenames are quoted if they contain spaces but not tabs.

The fix is scheduled for the 2.21 milestone. Read the parenthetical, however: *Cppcheck should only be used in trusted/safe environments.* The maintainer is telling us, in plain English, that the tool is not designed to survive hostile input. Yet a typical "shift-left" rollout drops Cppcheck into a CI runner that processes pull requests from arbitrary contributors (exactly the untrusted environment the maintainer warns against). A cybersecurity expert who has read the documentation knows to sandbox the runner, strip addon execution, and treat filenames from untrusted branches as adversarial input. A developer who simply installed the `cppcheck` package is feeding attacker-controlled filenames into a shell pipeline.

Flawfinder: The Author Himself Tells You What It Cannot Do

Five days later, on 17 May 2026, David A. Wheeler added thirteen lines to the Flawfinder manpage. The commit message is unusually direct: "Flawfinder really isn't a good tool for intentionally *malicious* code; frankly few things are."[11] The expanded warning that now ships in the man page reads:

> Note that flawfinder is primarily designed to find *unintentional* vulnerabilities (common mistakes) by looking for specific patterns. As a result, it is much less likely to find *malicious* code (such as Trojan horses) that has been intentionally inserted. Malicious programmers can easily insert code that is not detected by this kind of tool. This tool should only be used to find unintentional vulnerabilities; other techniques (such as thorough human review of all changes) are needed to find intentionally malicious code. (You can try using AI to analyze for malicious code, but beware, such code may include AI-focused injections that themselves may make the AI ineffective or malicious.)

Unfortunately, the reaction to this kind of language by the wrong stakeholder could be to remove the SwA tools from the compliance pipeline. It should instead be required reading. Wheeler is the author of the tool and a long-time voice in secure development; he is also being precise about its blast radius. Flawfinder finds patterns. It does not find adversaries. An expert running Flawfinder knows to pair it with code review, build provenance attestation, and reproducible builds. A developer running Flawfinder as the last gate before merge believes the green check mark means something it does not.

Checkmarx: When the Scanner Is the Attack Vector

The most expensive lesson comes from Checkmarx, and we are still early in the season of the onslaught of SwA weaknesses. Starting 23 March 2026, the cybercrime group tracked as TeamPCP leveraged credentials stolen during the earlier Trivy compromise to push malicious code into Checkmarx's KICS Docker images, two OpenVSX extensions (`ast-results` v2.53.0 and `cx-dev-assist` v1.7.0), and GitHub Actions workflows.[2, 3] The vulnerability is tracked as CVE-2026-33634 with a CVSS score of 9.4.

The attack was a credential stealer designed to run wherever the scanner ran. CI/CD secrets, cloud credentials, SSH keys, and Kubernetes configurations were exfiltrated from runners on 30 March 2026 and posted to a dark-web leak site on 25 April 2026.[7] Even after Checkmarx revoked credentials and removed malicious artifacts, the attackers retained access. On 22 April 2026 they published a second wave of poisoned artifacts—another KICS Docker image, another GitHub Action, another VS Code extension. The collateral was significant: the Bitwarden CLI npm package (`@bitwarden/cli@2026.4.0`) was briefly compromised during a 93-minute window on the same day, riding the same intrusion outward to roughly ten million Bitwarden users.[1] A third escalation in May saw TeamPCP rename the Checkmarx Jenkins AST Plugin repository to "Checkmarx-Fully-Hacked-by-TeamPCP-and-Their-Customers-Should-Cancel-Now."[8]

The story is not that Checkmarx was breached. Mature vendors are breached. The story is what the scanner was wired into. A typical Checkmarx rollout grants the scanner read access to the entire source tree, write access to merge status, and an injected secret per protected environment. That posture is sensible when an expert chooses the integration boundaries deliberately. It is catastrophic when "the security tool gets all the permissions" is the default DevSecOps onboarding script. The blast radius of any Software Assurance tool is exactly the blast radius the integrating team gave it.

More Tools, More Doors

These three incidents are not isolated. NIST SP 800-53's Supply Chain Risk Management (SR) controls, particularly SR-3 (Supply Chain Controls and Processes) and SR-4 (Provenance), already treat the development toolchain as part of the system's attack surface, not separate from it.[6] CWE-1395 (Dependency on Vulnerable Third-Party Component) applies just as much to a static analyzer wired into a CI pipeline as it does to a logging library shipped in production.[4] Unit 42's continuing telemetry on npm shows the trendline accelerating: "attacks evolving from isolated typosquatting incidents into systematic campaigns by various threat actors to weaponize the trust that powers modern software development."[10]

Every additional tool added to a DevSecOps stack introduces three new attack surfaces: the tool binary, the credentials the tool holds, and the network paths the tool requires. Stacking ten "best-of-breed" scanners without an expert curator produces ten new attack surfaces guarded by whichever team happened to click through the installer. Recall the XZ Utils backdoor of 2024, the eslint-scope npm compromise of 2018, the Codecov bash uploader breach of 2021, and the SolarWinds Orion compromise. The common thread is not that the victim organizations were lax about scanning. Most of them scanned aggressively. The common thread is that the scanning infrastructure itself was treated as a developer convenience rather than a privileged piece of cybersecurity infrastructure under expert care.

What Expert Custody Looks Like

A Software Assurance tool under expert custody has a tool owner who reads the maintainer's documentation (including the parentheticals), tracks the upstream issue tracker, validates the supply chain of every plugin, scopes the credentials the tool can reach, isolates the runner that executes the tool, reviews findings before they reach developers, and tunes rule sets against the program's threat model. Developers still see the results. Developers still fix the findings. Developers do not own the deployment, and the build server does not get an unattended root credential because "the scanner needed it."

This is no different from any other safety-critical instrument. We do not ask the patient to operate the MRI. We do not ask the passenger to torque the engine bolts. We expose the output. We restrict the operation.

Conclusion

The CppCheck ticket reminds us that even simple static analyzers assume a trusted environment that DevSecOps rarely provides. The Flawfinder commit reminds us that the people who write these tools are clear-eyed about what their tools cannot do, and the rest of us should take note. The Checkmarx incident reminds us that a scanner with broad credentials is a credential broker waiting to be subpoenaed by an attacker.

A scalpel does not make a surgeon. A scanner does not make a security engineer. Software Assurance tools belong in the hands of those who know how to sharpen them, sheath them, and put them down when the work is done.

Disclosure: This article was drafted with AI assistance.

References

[1] Bitwarden. Bitwarden Statement on Checkmarx Supply Chain Incident. 2026. URL: https://community.bitwarden.com/t/bitwarden-statement-on-checkmarx-supply-chain-incident/96127.

[2] Checkmarx. Update: Ongoing Checkmarx Supply Chain Security Incident. 2026. URL: https://checkmarx.com/blog/supply-chain-security-incident-update/.

[3] Thomas Claburn. Ongoing supply-chain attack targets security, dev tools. The Register. Apr. 2026. URL: https://www.theregister.com/2026/04/27/supply_chain_campaign_targets_security/.

[4] CWE Content Team. "CWE-1395: Dependency on Vulnerable Third-Party Component". In: (2024). URL: https://cwe.mitre.org/data/definitions/1395.html.

[5] Jon Hood, ed. SwATips. https://www.SwATips.com/.

[6] Joint Task Force. NIST Special Publication 800-53, Revision 5. Security and Privacy Controls for Information S Tech. rep. National Institute of Standards and Technology, 2020. URL: https://doi.org/10.6028/NIST.SP.800-53r5.

[7] Eduard Kovacs. Checkmarx Confirms Data Stolen in Supply Chain Attack. SecurityWeek. 2026. URL: https://www.securityweek.com/checkmarx-confirms-data-stolen-in-supply-chain-attack/.

[8] Ravie Lakshmanan. TeamPCP Compromises Checkmarx Jenkins AST Plugin Weeks After KICS Supply Chain A The Hacker News. May 2026. URL: https://thehackernews.com/2026/05/teampcp-compromises-checkmarx-jenkins.html.

[9] Daniel Marjamäki. Ticket #14749: cmdFilename: handle more bash special characters. Cppcheck Project. May 2026. URL: https://trac.cppcheck.net/ticket/14749.

[10] Unit 42. The npm Threat Landscape: Attack Surface and Mitigations. Palo Alto Networks. 2026. URL: https://unit42.paloaltonetworks.com/monitoring-npm-supply-chain-attacks/.

[11] David A. Wheeler. Commit d6464e7: Emphasize limitations on malicious code. flawfinder. May 2026. URL: https://github.com/david-a-wheeler/flawfinder/commit/d6464e7014f5986fb31acbe53