HACKEN

What We Learned from Hacken's 2025 SSDLC Maturity Survey

Customers & Partners

PROTOFIRE
MaiCoin
MST BLOCKCHAIN
strobe
AeternaFi
CYCLONE
Takadao
ONCHAIN WIN
Lumis
daCAT
AIBTC
Asseto
UFARM DIGITAL
Wristkey Global

hacken.io

What We Learned from

Hacken's 2025 SSDLC Maturity Survey

Customers & Partners

Secure software delivery is no longer a "nice to have" — it's a prerequisite for building reliable products and maintaining user trust. In 2025, Hacken ran a Secure Software Development Life Cycle (SSDLC) maturity survey across our customers and partners to better understand how teams build software today, where security practices are strong, and where organizations still need practical support.

This article summarizes the key signals from the survey and highlights the most actionable improvement areas we observed.

"This survey reflects what I've seen from both sides—as a developer and as a security engineer. Security is always a trade-off: perfect protection vs. time-to-market, risk appetite vs. business value. Not every control fits every context. But the gaps we identified here—regression testing, incident readiness, continuous assurance—these are the ones that hurt when ignored. I've experienced it firsthand: teams without regression tests reintroducing critical bugs, unprepared teams losing precious time while handling security incidents. The data confirms what practitioners already know: maturity isn't about doing everything. It's about knowing which risks you can take, and which ones will break you."

Grzegorz Trawiński

Offensive Security Services Director

LinkedIn

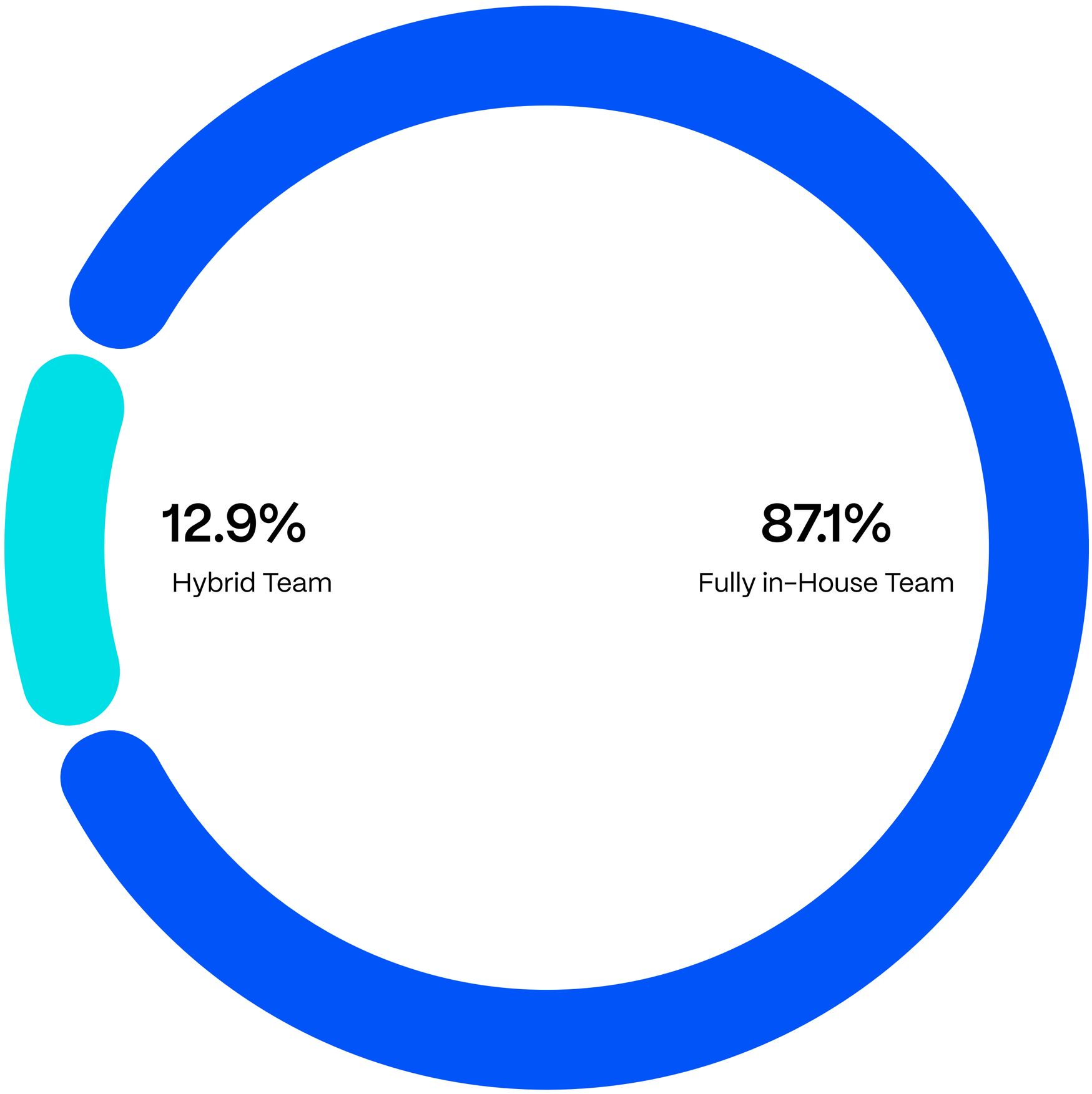
Survey scope and focus

The survey was designed to evaluate maturity across the entire software development journey, covering over 20 targeted questions across several critical domains:

- **Governance & Enablement:** Team composition, Agile adoption, and security training.
- **Security by Design:** Documentation, security requirements, risk analysis, and threat modeling.
- **Engineering Controls:** CI/CD integration, peer reviews, and access management.
- **Verification & Testing:** Manual, automated, and advanced testing methodologies (fuzzing and invariants).
- **Operations & Resilience:** Production monitoring, maintenance, and incident response readiness.

The strong completion rate among participants suggests that the respondents — primarily from the Hacken ecosystem — are deeply engaged with security topics. Furthermore, most respondents expressed interest in being featured as contributors in our public reporting, signaling a strong community commitment to transparency and shared best practices.

Who responded:
teams with ownership and process maturity

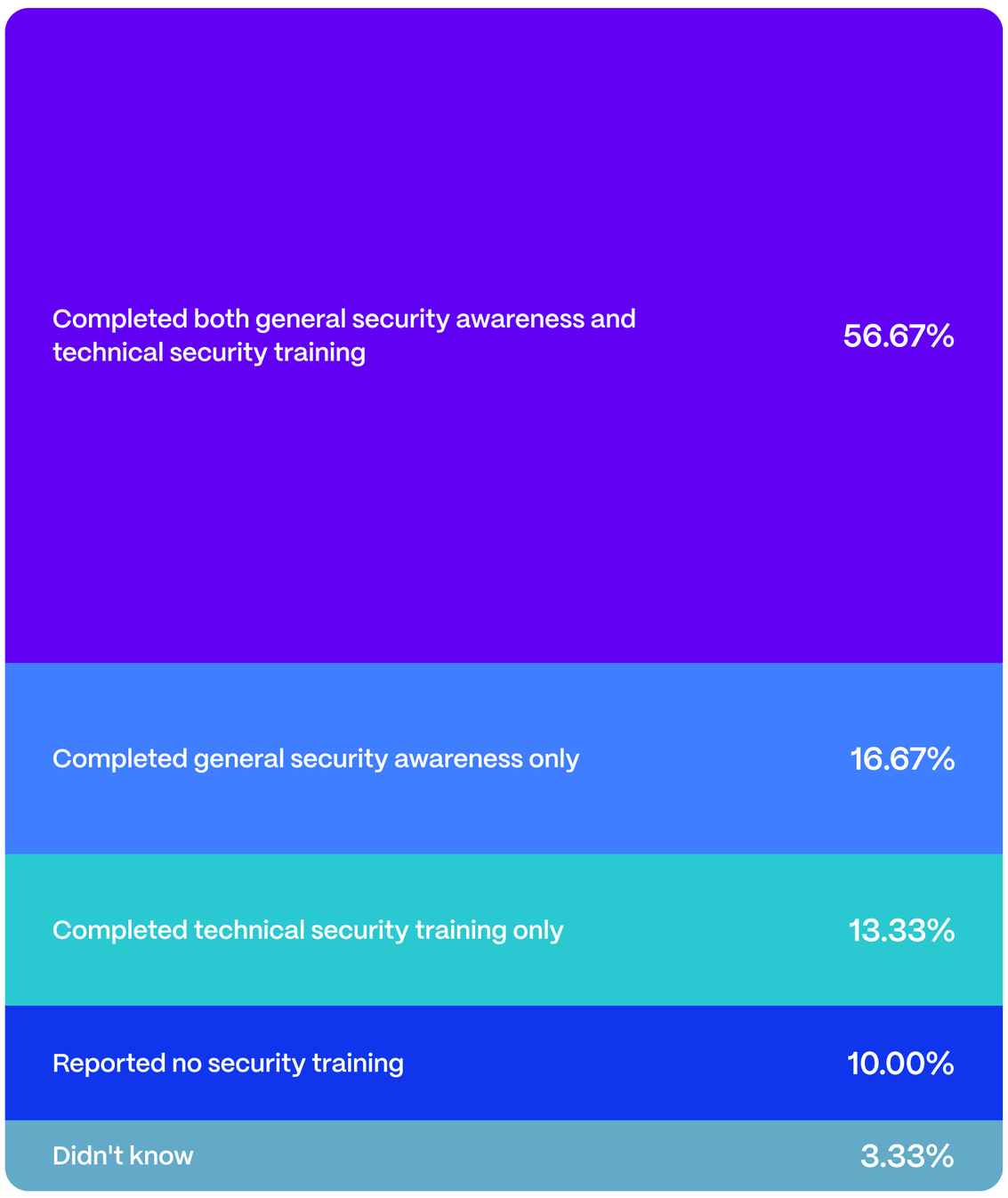


Most respondents described their development teams as **fully in-house (87.10%)**, with **12.90%** operating hybrid models (in-house plus outsourced). This matters because secure delivery is often easier to institutionalize when ownership of architecture, code, and release practices remains internal.

On delivery methodology, **87.10%** reported following Agile principles, suggesting most teams already operate in iterative cycles where security can—and should—be integrated into routine planning, definition of done, and release gates.

Security enablement is present — but not universal

Security training appears broadly adopted, but uneven in depth. Among respondents who answered the training question →



This indicates many teams are investing in security capability, but a meaningful portion still lacks the baseline training needed to reduce avoidable vulnerabilities and operational risk.

Around **30%** of surveyed teams lack formal technical security training — a concerning finding that directly undermines secure development practices. While general awareness training (phishing, social engineering) is valuable for reducing human error, it does not equip developers with the knowledge needed to identify and prevent code-level vulnerabilities — whether in web applications, smart contracts, or infrastructure. Without technical security training, developers cannot effectively implement threat modeling, conduct secure code reviews, or write tests that catch security-relevant edge cases. This gap is particularly acute in Web3 and DeFi, where contract vulnerabilities can result in direct financial loss.

The implication is clear: organizations cannot rely on external audits alone to compensate for a team that lacks foundational security engineering skills. Closing this training gap should be a priority for any organization serious about secure development. Technical security training — covering topics like secure coding patterns, common vulnerability classes, threat modeling, and security testing — is not a luxury; it is a prerequisite for building resilient systems. Teams that invest in this capability see measurable improvements in code quality, faster remediation cycles, and reduced audit findings.

On the positive side, teams report strong documentation habits: **93.33%** confirmed they created functional and/or technical documentation. This is important because good documentation makes it significantly easier to introduce security requirements, threat models, test plans, and evidence-based release decisions.

The shift-left gap: security requirements and threat modeling

Two results stood out as "high leverage" areas for improvement—practices that, when introduced early, reduce downstream cost and prevent repeated classes of issues.

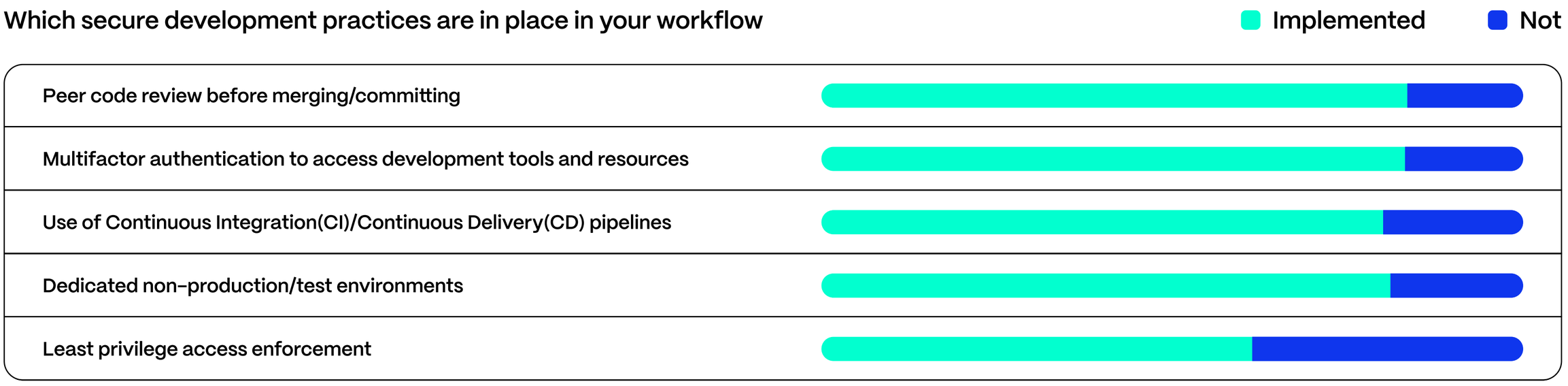
- **Security requirements: 70.00%** said they define and document security requirements, while **26.67%** do not.
- **Threat modeling: 65.52%** reported conducting threat modeling, while **27.59%** do not and **6.90%** were unsure.

These numbers are meaningful because teams can often continue development while still missing a foundational step: defining "what secure means" for their system and proactively analyzing how it can fail.

Threat modeling tends to lag risk analysis (which was higher at 79.31%), but both are essential. Risk analysis answers what matters most; threat modeling answers how it breaks and how we stop it. Increasing adoption here is one of the fastest ways to improve maturity without relying solely on point-in-time external validation.

Strong baseline controls — least privilege remains the hardest

Respondents reported relatively strong adoption of several standard secure development practices:



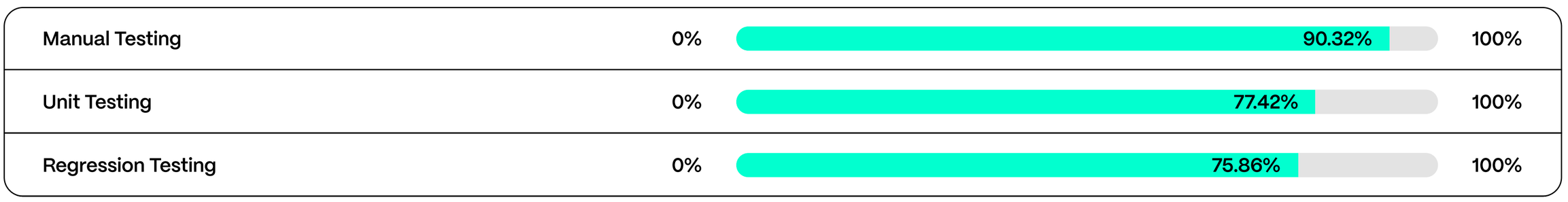
- **Peer code review before merge/commit:** 80.65%
- **Use of CI/CD pipelines:** 77.42%
- **MFA for development tools and resources:** 80.65%
- **Dedicated non-production/test environments:** 77.42%

However, **least privilege access enforcement** was notably lower at **61.29%**, making it the weakest control in this set.

This pattern is common. Teams frequently implement MFA and CI/CD early, but access control maturity is harder: IAM sprawl, shared administrative roles, lack of periodic access reviews, and unclear ownership can all slow development progress. Yet least privilege remains one of the most effective controls for limiting the blast radius of compromised accounts, insider threats, and supply-chain issues.

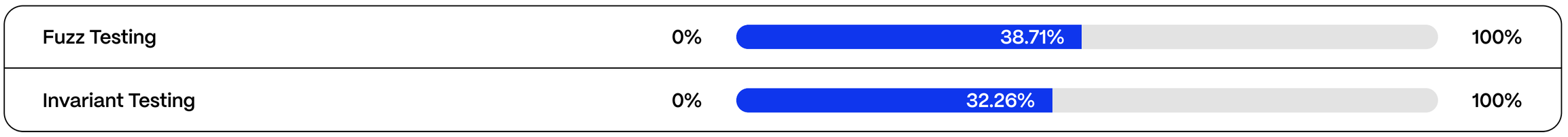
Testing maturity: manual and unit tests are common; advanced methods are not

Regular Testing



Most teams reported performing manual and unit testing. Where maturity drops sharply is in deeper, security-relevant testing approaches:

Advanced Testing



This is one of the clearest technical gaps in the dataset. Manual and unit testing are valuable, but they often miss edge cases and adversarial behaviors that fuzzing and invariant/property-based testing are designed to catch. For teams building complex systems (especially those that manage value or safety-critical flows), these methods can materially reduce exploitability and improve confidence in upgrades.

A significant operational risk identified in the survey is that 20.69% of teams are not running regression tests. In a modern development environment where code is constantly evolving, regression testing is the primary defense against "breaking changes" — the accidental introduction of new bugs into previously stable and secure code. Without a robust regression suite, every update or hotfix carries the risk of silently re-opening old vulnerabilities or breaking critical security logic that was already validated.

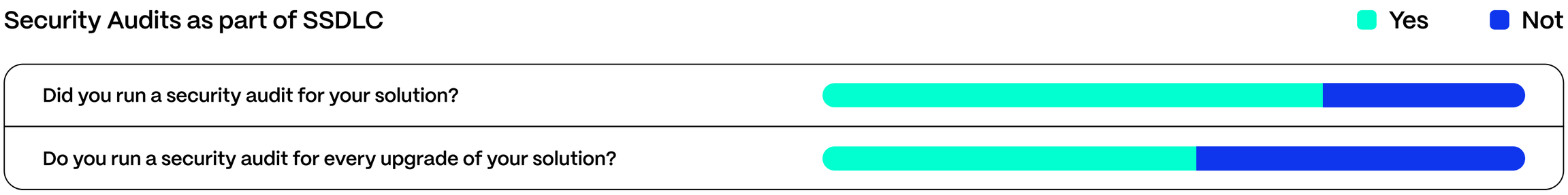
1 in 5

teams does NOT run regression tests

This gap is particularly dangerous when providing updates or patches under pressure. When developers modify existing logic to add a feature or fix a bug, they may inadvertently disrupt dependencies or edge cases they aren't currently focused on. For Web3, where code is often immutable or governs high-value assets, the absence of regression testing means that a single update could compromise the integrity of the entire system. Implementing automated regression suites is a non-negotiable step for teams moving toward a mature, resilient SSDLC.

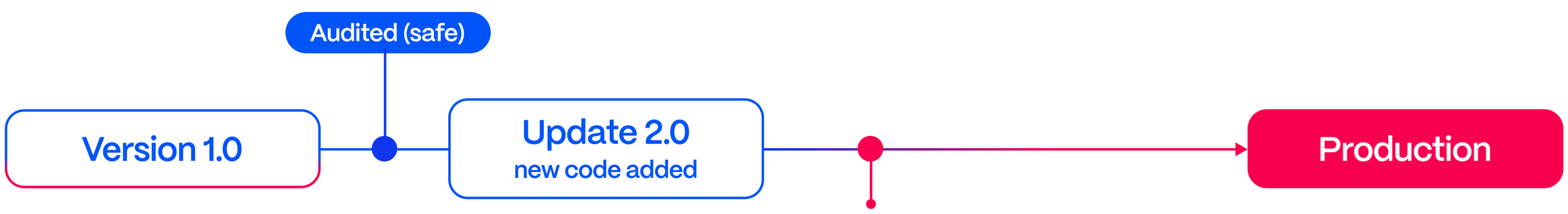
Without automated regression suites, new changes can break existing functionality and introduce new vulnerabilities or reopen previously resolved ones.

Audits are common — but continuous assurance is not



External assurance is widely used:

- **70.97% reported running a security audit**
- **25.81% have not**
- **Regression testing: 7**5.86% (with 20.69% not running regression tests)



29% No Initial Audit

Nearly one-third of projects enter the market without any formal security verification of Version 1.0.

45% Deployed without Audit

Almost half of all surveyed organizations skip security audits for subsequent code deployments.

Critical Risk

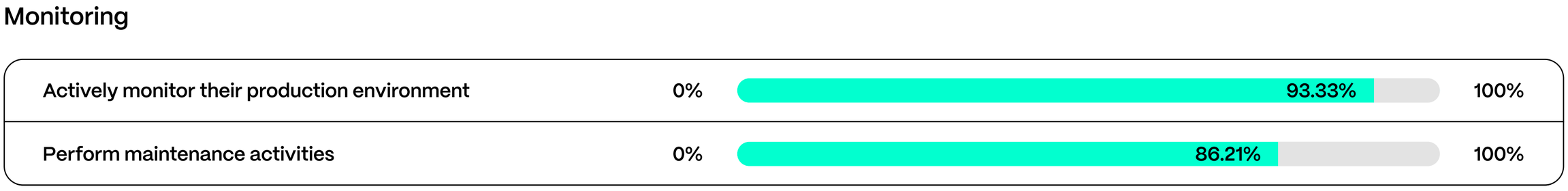
Unverified updates in production create blind spots that can lead to catastrophic smart contract failures.

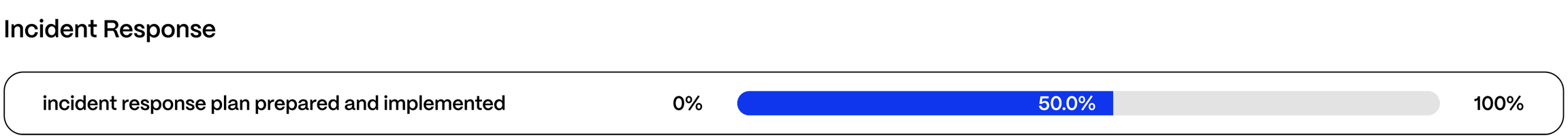
But when asked whether audits occur for **every upgrade**, responses split:

- **51.72% yes**
- **44.83% no**

This suggests many organizations still operate with a "point-in-time" assurance model: audit once, then iterate quickly. In practice, risk concentrates in upgrades—new features, dependency changes, configuration changes, and permission changes. This gap highlights a growing need for tiered, release-driven assurance approaches (lighter reviews for low-risk changes, deeper reviews for high-risk releases), supported by strong automated controls and testing evidence.

Operational maturity: monitoring is high, incident readiness is not



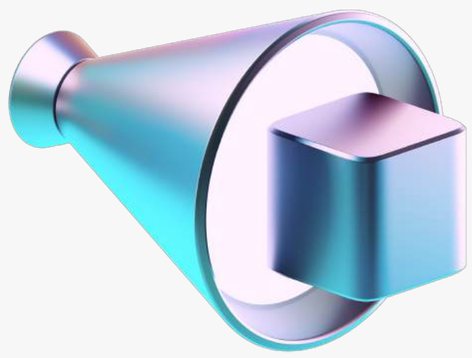


Two operational signals appeared together — and they tell an important story.

On one hand, **93.33%** of respondents actively monitor their production environment, and **86.21%** perform maintenance activities such as hotfixes or upgrades. This indicates teams are operating real systems, watching for issues, and shipping changes over time.

On the other hand, only **50.00%** reported having an **incident response plan prepared and implemented**, while **40.00%** do not and **10.00%** are unsure.

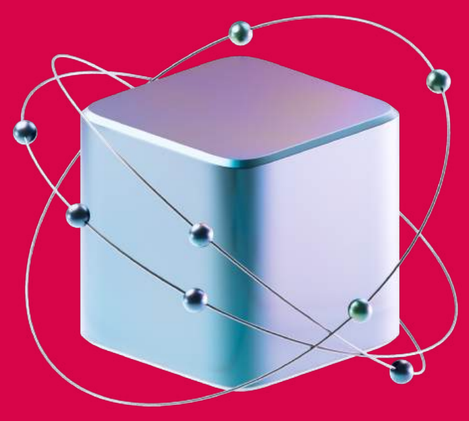
High Alertness



93%

Actively Monitor Production Environment

Low Readiness

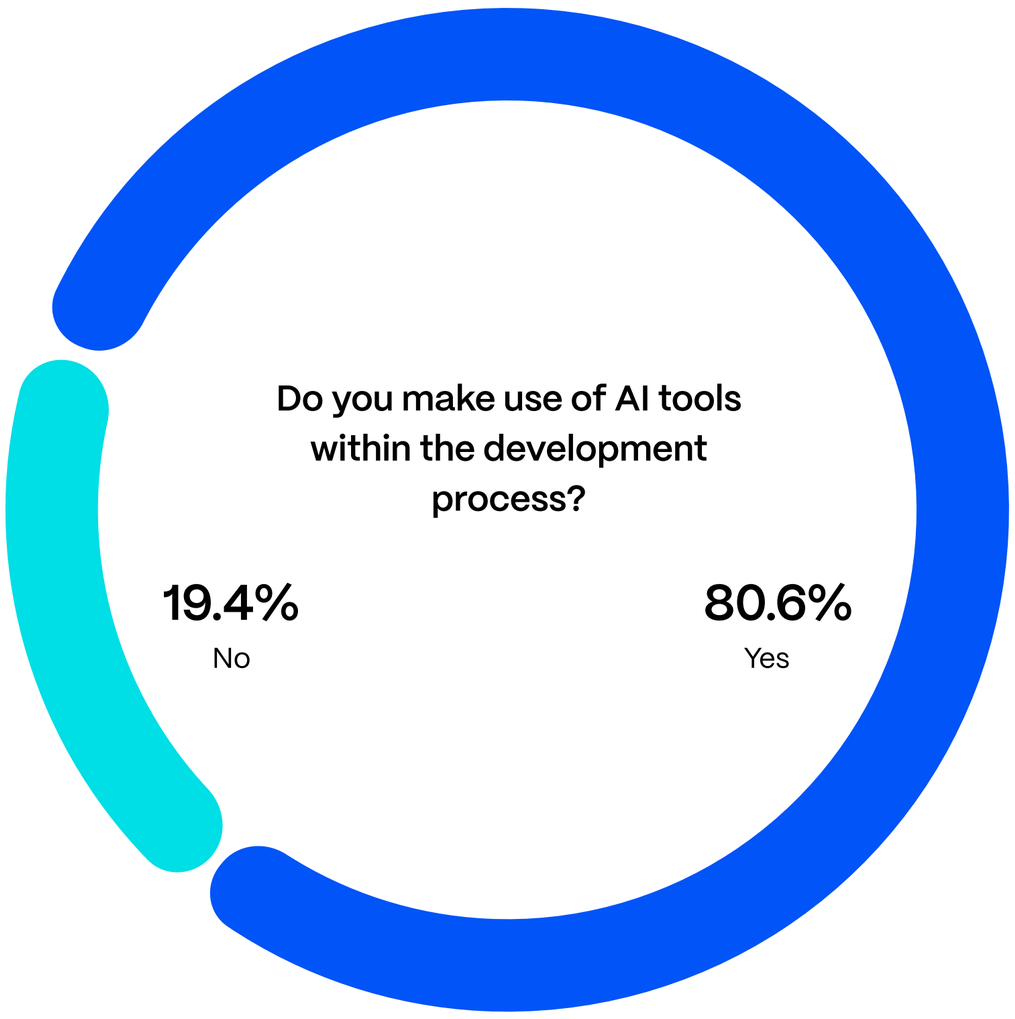


Only 50%

Have an Incident Response Plan

This is arguably the most critical maturity gap in the survey. Monitoring helps detect problems; incident readiness determines whether teams can respond decisively under pressure—contain impact, communicate effectively, and recover safely. In real-world security events, the absence of a practiced plan often translates into delayed decisions, unclear authority, and avoidable escalation.

AI tool adoption: a new and fast-growing risk surface



A large majority (80.65%) reported using AI tools within the development process.

This adoption creates clear demand for updated SSDLC guidance: handling sensitive data in prompts, preventing secret leakage, establishing code provenance expectations, and ensuring AI-assisted coding does not bypass secure review and testing standards. AI can accelerate development, but it also increases the importance of guardrails, traceability, and strong "definition of done" criteria.

AI Writes the Code, You Handle the Aftermath ↗

by Ivan Bondar,
Principal Smart Contract Auditor at Hacken

At first glance, AI-crafted code equals speed, savings, efficiency. But before you know it, your lifeline's at risk and trust collapses.

Maturity is uneven: a prioritization story

High Maturity

Documentation | Agile Adoption | CI/CD | Production Monitoring | Code Reviews

Warning Zone

Least Privilege | Regression Testing | Technical Training | Defining Security Requirements

Fuzzing / Invariant Testing | Incident Response Plans | Upgrade Audits

What these results suggest (and where teams can act next)

Across the survey, teams show strong momentum: documentation, Agile delivery, CI/CD, reviews, MFA, and production monitoring are widely present. The highest-value improvements are concentrated in a set of practices that raise maturity quickly:

01 Incident response planning and tabletop exercises

Moving beyond monitoring to ensure the team can contain, communicate, and recover from exploits under pressure.

02 **Formalizing security requirements and threat modeling**

Shifting security to the design phase to identify "how it breaks" before a single line of code is written.

03 **Mandatory technical security training**

Closing the 30% gap by equipping developers with the specific skills needed to prevent code-level vulnerabilities and smart contract exploits.

04 **Enforcing least privilege and periodic access reviews**

Hardening the development environment by ensuring access is granular, monitored, and revoked when no longer needed.

05 **Automated regression testing for every update**

Ensuring that new features or hotfixes do not silently re-introduce old bugs or break existing security logic in stable code.

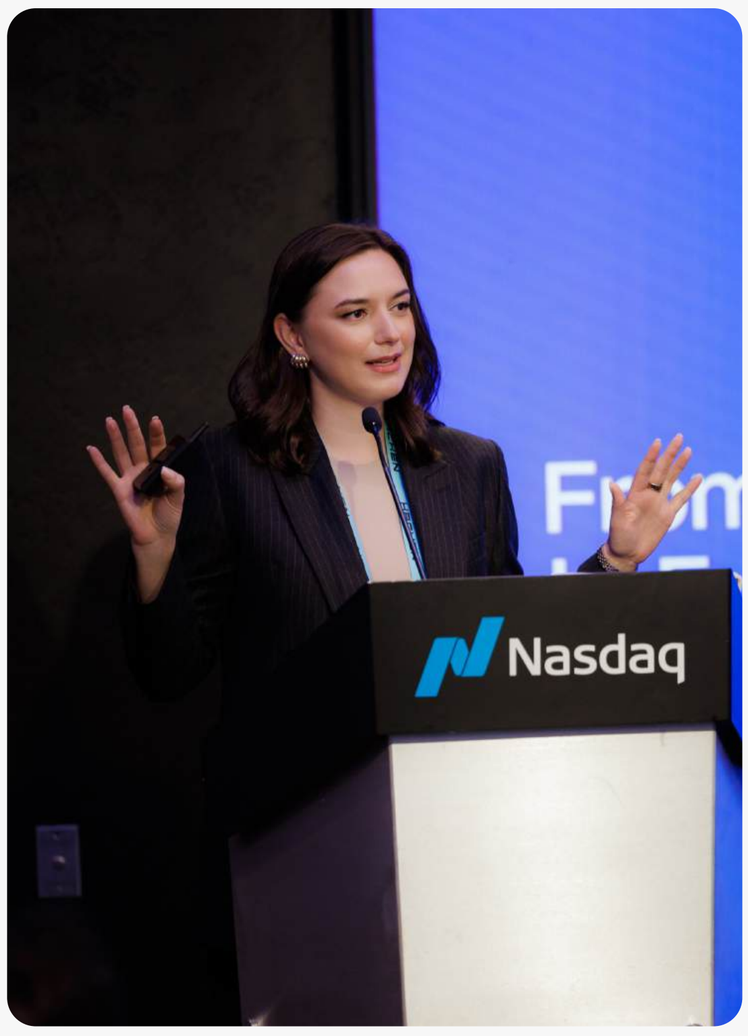
06 **Adopting advanced testing (fuzzing and invariants)**

Moving beyond basic unit tests to catch complex edge cases and adversarial behaviors in critical code paths.

07 **Release-driven assurance for upgrades (continuous assurance)**

Transitioning from "one-off" audits to a model where every significant change is validated before deployment.

Achieving these milestones is not about reaching a state of 'perfect security'—an impossible standard in a shifting threat landscape. Instead, it is about moving from ad-hoc reactions to a model of consistent habits, clear ownership, and repeatable workflows that make security a natural part of the software engineering process.



"Hacken's SSDLC Maturity Survey provides a transparent benchmark for everyone shaping secure software, including development teams, product owners, protocol founders, and investors offering clarity on current market practices and where gaps remain.

For me, the key takeaway confirms Hacken's direction for 2026: moving from limited point-in-time security to continuous assurance, supported by structured training, sufficient testing, and cybersecurity embedded across the entire software lifecycle."

Yev Broshevan

CEO & Co-Founder, Hacken

LinkedIn

Closing

The goal of Hacken's SSDLC maturity survey was simple: understand how teams actually build software today, identify gaps, and use that insight to prioritize practical improvements. Even with a limited sample size, the patterns are consistent with what we see across real engagements: many organizations have built solid foundations, but still need help operationalizing security early and making it sustainable through upgrades and incidents.

Contribute to the 2026 SSDLC Survey

We're mapping how secure software is built in the new digital frontier.

Share your perspective and help define the benchmark. Participants can opt to be featured in the public report.

Participate Now

Who Contributed

The following organizations agreed to be publicly acknowledged as contributors to the 2025 SSDLC Maturity Report.

Protofire · MaiCoin · Takado · Junction · Cipherlabs · Ufarm Digital · FG Wallet

MST Blockchain · Smart Energy Provider Lim · Astra Nova · SKY OPTION · daCAT · Strobe · AeternaFI

Overtake · Cyclone · Sway · Nonco · H4cklabs · Lumis.fi · OnChainWin

Flicker · AIBTC · Asseto · WristKey Global Solutions · Privily

Security is a shared responsibility

The 2025 SSDLC Maturity Survey is an industry-wide research initiative led by Hacken in collaboration with builders, founders, and engineering leaders from across the digital assets ecosystem: from Layer 1 infrastructure and DeFi protocols to payment rails, cross-chain tooling, and community-owned financial tools. This report reflects ground-up insights by the teams actively navigating secure software development in their day-to-day work. By pooling knowledge across organizations of different sizes, structures, and maturity levels, this effort aims to give the industry an honest, data-driven benchmark.

Protofire DAO
https://protofire.io

MaiCoin
https://www.maicoin.com

Takadao
https://takadao.io/

Junction
https://junction.exchange

Cipherlabs
www.cipherlabsx.com

MST Blockchain
https://mstblockchain.com/

Strobe
https://strobe.finance

FG Wallet
https://fgwallet.io

AeternaFi
https://www.aeternafi.com/

Overtake
https://overtake.world/

CycloneChain
https://cyclonechain.com

SKY OPTION
www.skyxso.com

daCAT
https://dacat.fun/

Sway
swaysavings.com

Astra Nova
https://astranova.world/

Lumis.fi
https://lumis.fi/

OnChainWin
https://onchainwin.com/

Flicker
https://flicker.finance/

Nonco
https://nonco.com

UFarm.Digital
https://ufarm.digital/

Asseto
https://asseto.finance/

Smart Energy Provider Limited
https://smartenergychain.org

Privily
https://privily.fi

AIBTC
https://aibtc.com

WristKey Global Solutions
www.wristkeyglobal.com

The contributors listed in this report voluntarily chose to be publicly recognized for their participation. Their openness is what makes cross-industry benchmarking possible, and what helps the entire ecosystem build more securely.

Making Web3 a safer place

Hacken is an end-to-end blockchain security & compliance partner for digital assets

6800+
vulnerabilities found

$430B
on-chain assets verified

1B+
transactions monitored

$15M
paid out in bug bounties

50+
centralized exchanges

30K+
malicious contracts detected

60+
certified security engineers

ISO 27001
certified

Evolving alongside the industry for 8+ years

European Commission · VARA · European Blockchain Sandbox · Enterprise Ethereum Alliance · INATBA · Dubai Blockchain Center · C4 · The Digital Chamber · ERC3643 · Crypto Valley · مركز قطر للمال Qatar Financial Centre · The Millennium Alliance

Trusted by 1500+ digital asset leaders

MetaMask · ethereum foundation · ebsi European Blockchain · ADGM · FORGE Societe Generale Group · Binance · Bybit · crypto.com · OKX · KuCoin · Chainlink · Mawari · nubank · Aurora · Sui

Our Story

Unlike traditional providers, Hacken was born on blockchain, combining deep Web3 expertise with enterprise-grade quality, AI-powered offensive security, and globally recognized certifications. Since 2017, Hacken has been trusted by startups, enterprises, and regulators to secure the new digital frontier.

Learn more — hacken.io

Follow us on social media — linkedIn · X