SHERLOCK

WHY POINT-IN-TIME SECURITY AUDITS FAIL DEFI PROTOCOLS

How Lifecycle Security Solutions Help You Hit Mainnet With Confidence

THE CASE FOR COMPLETE LIFECYCLE SECURITY

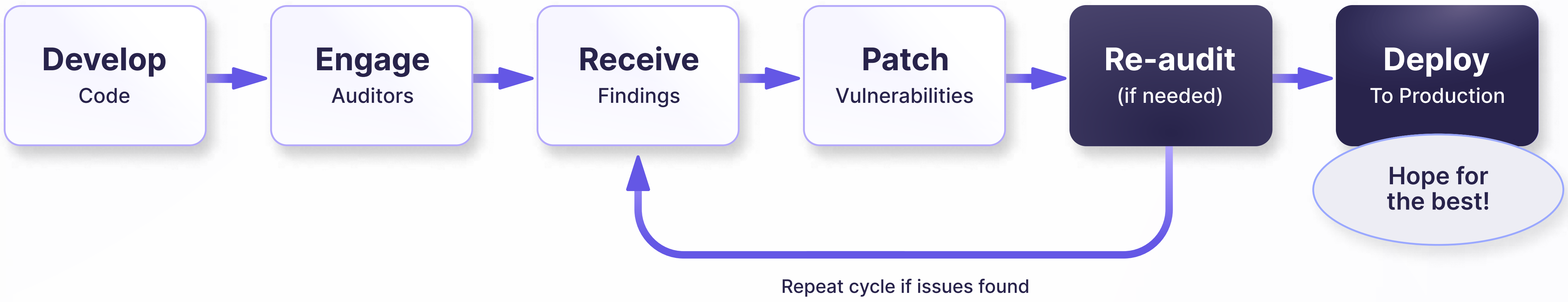
Using Sherlock AI to surface meaningful security signal earlier in multi-chain systems

Traditionally, crypto security followed a linear path: develop code, engage auditors, receive findings, patch vulnerabilities (or sometimes re-write segments of code), re-audit if needed, deploy, and hope for the best.

This point-in-time model treats security as a “scheduled event” is separate from how protocols are designed, built, and monitored. This model is breaking down with an alarming increase in exploits following extensive, point-in-time audits.

TRADITIONAL CRYPTO SECURITY PATH

Traditionally, crypto security followed a linear path: develop code, engage auditors, receive findings, patch vulnerabilities, re-audit if needed, deploy, and hope for the best.



The industry has responded with more audits, bigger bounties, and expanded security budgets. Yet losses continue to climb. Many teams try to compensate for weak internal security by leaning on audits and bounties. These are valuable tools, but real security is enforced internally through posture, process, and discipline. Part of that process is to embrace and implement a lifecycle approach consisting of ongoing security auditing. Lifecycle security fundamentally rethinks when and how security happens.

After over 150 exploits in 2024 totaling $1.48 billion, 2025 was even more devastating with total crypto losses exceeding $3.4 billion. Access control failures accounted for 59% of losses in H1 2025 ($1.83B), while the industry's largest single hack, Bybit's $1.5 billion breach, demonstrated that even well-resourced platforms remain vulnerable."

Many targeted previously audited protocols, exposing a fundamental weakness in the current paradigm, demonstrating that these audits are no longer sufficient in today's DeFi environment.

WHY POINT-IN-TIME SECURITY FAILS

Today's protocols need security that evolves as they do: specifically, we recommend that your security investment should scale as your TVL grows. The question isn't whether your audit was thorough, it's whether your security keeps pace with how your protocol actually operates.

Risk doesn't wait for audits to occur. Code changes, integration shifts, and stress scenarios emerge in real time. A security model built around these so-called point-in-time snapshots can't keep pace.

The Fragmentation Problem

Point in time security is fundamentally inefficient because every code update forces protocol teams to repeat the same high effort audit sourcing, interviews, and contract negotiation from scratch.

As a result, Point-in-time security approaches keep knowledge in “silos” with each auditor, causing valuable information to become fragmented. Different vendors review design, pre-launch, and post-launch—each re-learning the protocol from partial views.

Consider healthcare: you wouldn't see a different doctor for every symptom. You establish continuity with a physician who understands your history, monitors changes, and recognizes patterns. Point-in-time audits examine protocols in isolation, without context or ongoing responsibility. This fails catastrophically when code, money, and coordination occupy the same space.

Four Critical Failure Modes

1. **Assumption Drift**
   Wormhole's $320M hack occurred in thoroughly reviewed code. The vulnerability emerged from logic operating under real-world conditions that differed from audit assumptions. As protocols evolve—adding features, adjusting parameters, integrating new contracts—initial audit assumptions diverge from operational reality.

2. **Post-Audit Code Changes**
   Opyn: Updated code after OpenZeppelin audit, introduced an exploited bug
   Team Finance: Lost $14.5M despite audits from Hacken, Coinspect, CertiK
   Ronin: Failed to call initialization function during protocol update, $12M exploit

Audit reports become obsolete the moment code changes. Without continuous monitoring, these changes create exploitable blind spots.

3. **Scope Limitations**
   Audits examine contracts in isolation but can't model every future integration or market condition. Time constraints force breadth-versus-depth tradeoffs. Traditional audits catch implementation bugs (reentrancy, access control) but miss economic vulnerabilities, composability risks, oracle manipulation, and emergent behaviors.

4. **Post-Deployment Blind Spots**
   After audits conclude, protocols operate without security oversight. New attack vectors emerge, compromised integrations create cascading risks, and market conditions enable previously theoretical attacks. Most projects have months or years of exposure between audits.

WHERE POINT-IN-TIME SECURITY FALLS SHORT

The Evidence

The numbers reveal where traditional audits fall short, and more importantly, why they miss what matters:

Logic errors caused 50 exploits, followed by input validation failures (20), and price manipulation (18). Point-in-time audits examine code correctness but can't anticipate how protocols will behave under real-world conditions, market stress, or adversarial scenarios that only emerge after deployment.

Flash loan attacks account for 83.3% of exploits. They succeed by manipulating protocol economics and financial logic, not exploiting code bugs. Traditional audits focus on whether code functions as written, completely missing vulnerabilities in how the system is designed to handle money. The code works perfectly; the economic model doesn't.

Off-chain attacks resulted in 80.5% of total stolen funds, with compromised accounts responsible for 55.6% of incidents. Private key compromises, social engineering, and infrastructure vulnerabilities emerge entirely after deployment and fall outside the scope of pre-launch code audits.

The pattern is clear: point-in-time audits catch implementation bugs (reentrancy, access control flaws) but systematically miss design vulnerabilities, economic exploits, and post-deployment threats. Traditional security models aren't keeping pace with the threats protocols face once live.

Some Examples Of When Point-In-Time Audits Weren't Enough

Mango Markets (2022)
Audited by Neodyme in September, drained of $112M five weeks later through price oracle manipulation. The audit reviewed the code as written, but couldn't account for adversarial market conditions.

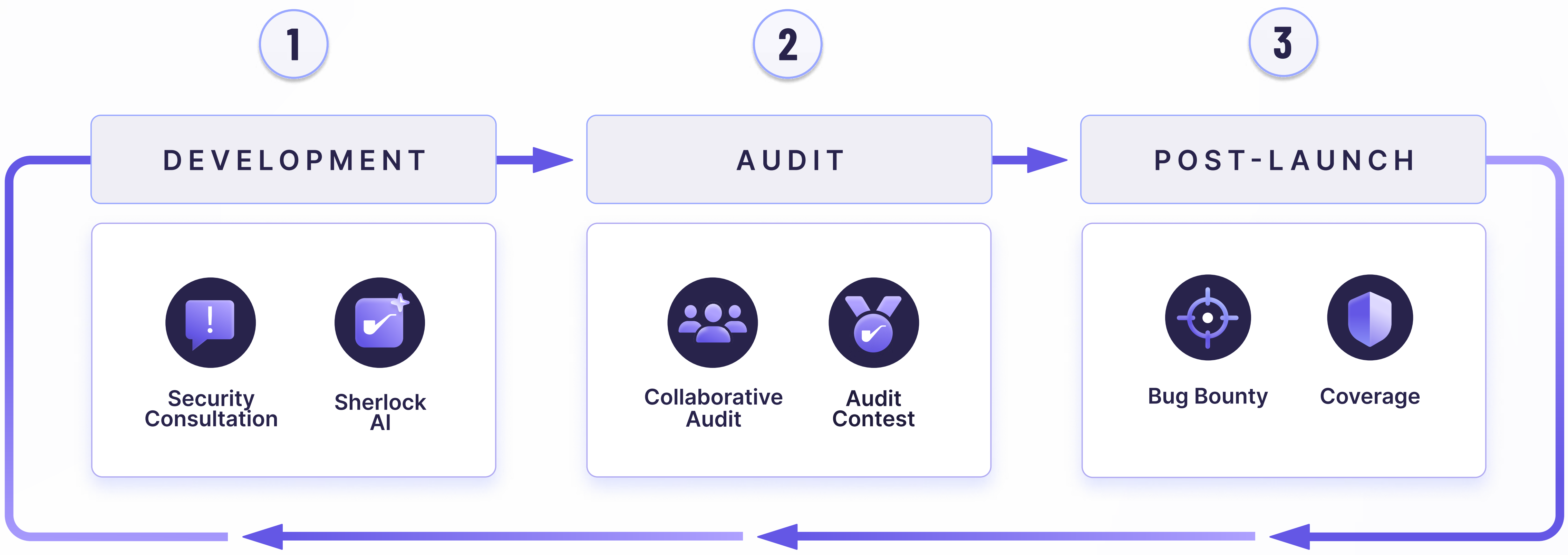
Balancer V2 (2025)
Lost $121M when attackers exploited tiny rounding errors in how the protocol calculated token exchange rates. These mathematical edge cases only appeared under specific market conditions—scenarios a snapshot audit couldn't test for

LIFECYCLE SECURITY IS THE SOLUTION

Sherlock Complete Lifecycle Security is the first unified model that connects security in development, auditing, and post-launch protection.

Lifecycle Security fundamentally changes how protocols approach risk. Instead of treating security as a series of isolated events, it creates a system where every review, every fix, and every insight compounds over time.

The Three-Phase Lifecycle Model



Lifecycle Security connects the three core phases:

1. Expert security consulting and AI-powered security reviews during development.
2. Intensive human review pre-launch with collaborative audits and public audit contests.
3. Active protection post-launch with bounties, coverage, periodic AI reviews, and monitoring.

Our lifecycle model leverages context and data generated in one phase to supercharge the next phase. Sherlock builds each audit team from top security experts in our network, matching specialists to your architecture so you get deeper coverage, faster review, and findings that other teams miss.

MEASURING EFFECTIVENESS OF THE LIFECYCLE

The Compounding Context Advantage

For the first time, context can be shared throughout all stages of a team's security strategy.

Development findings inform pre-launch focus. Audit results shape post-launch monitoring. Production incidents feed back into next version design. Security partners maintain aligned incentives. Instead of separate projects where each vendor re-learns the protocol, you get continuous security where institutional knowledge compounds.

Sherlock takes this another step further by ensuring that the same top auditors are involved in each stage of a protocol's lifecycle all the way from AI implementation and development to post-launch monitoring.

Measuring Effectiveness

- **Issues surface earlier** — Critical vulnerabilities caught during development, not pre-launch
- **Reviews accelerate** — Auditors spend less time learning, more finding sophisticated vulnerabilities
- **Fewer post-launch unknowns** — Monitoring catches identifiable issues, not entirely novel vectors
- **Predictable costs** — No last-minute scope expansions from discovering fundamental flaws

If you're repeating work or rediscovering past risks, your lifecycle isn't connected. If audits start from zero, you're paying the fragmentation tax.

Full lifecycle adoption is the strongest signal you can send to users and investors.

H CONFIRMED HIGH SEVERITY FINDING

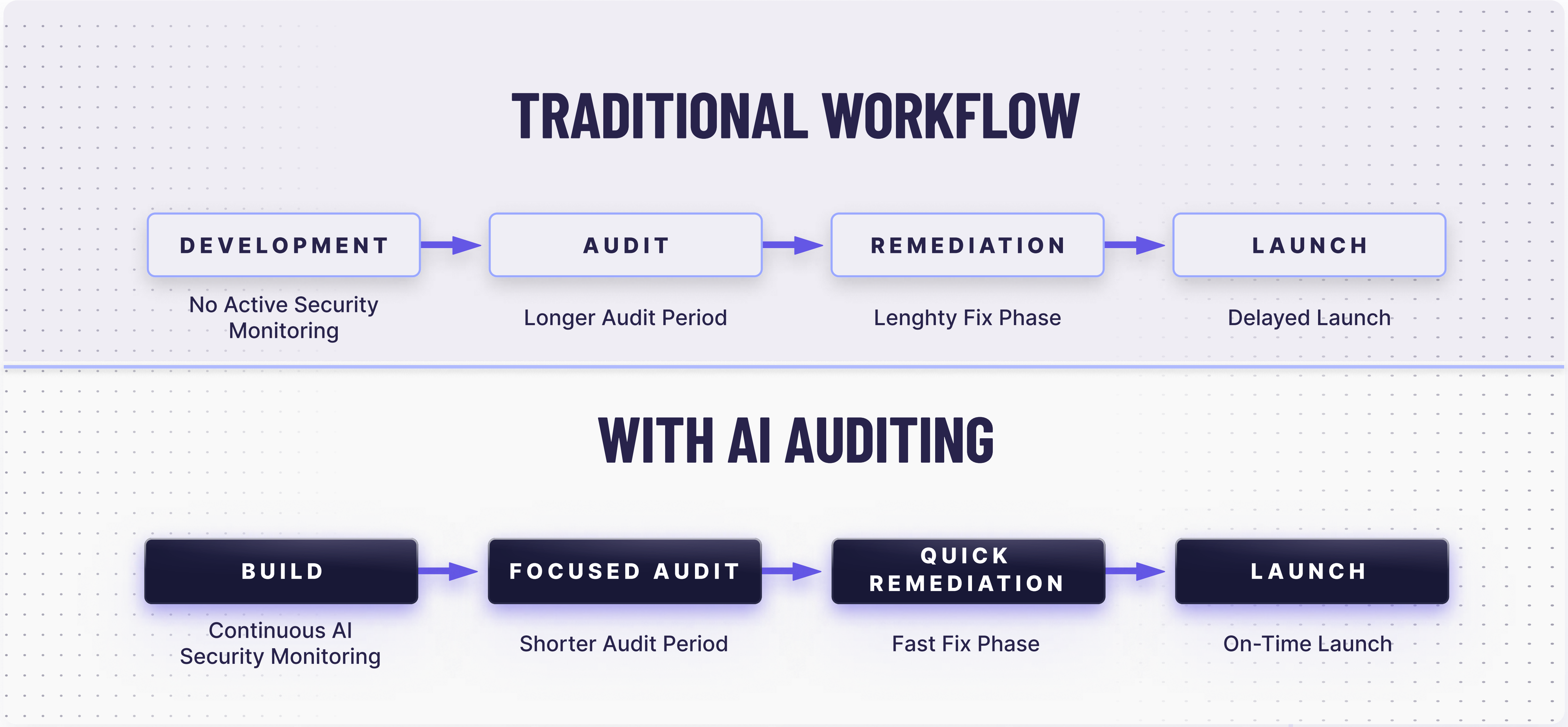
WHY LIFECYCLE SECURITY IS ONLY VIABLE NOW

These stages aren't new. They've been part of blockchain engineering since the very beginning. The core problem is that the Web3 security industry has treated these individual lifecycle stages as separate problems. Development security has one set of vendors. Audits and contests have another. Post-launch protection has a third set of vendors and solutions. Each service has historically operated on its own island, with its own processes, with missed context traveling from one lifecycle stage to the next.

Critical breakthroughs have made lifecycle security feasible:

AI Security Reviews

AI can perform security reviews in 2 hours instead of 2 weeks and scales near-infinitely. AI security reviews are <1% of the price and duration of human audits. While AI security reviews still are imperfect and aren't as reliable as top human auditors, we've seen over and over again that AI auditing is able to find valuable and impressive bugs in sophisticated codebases. A protocol team can have their code reviewed at 100x the frequency by AI during a development cycle, breaking the point-in-time constraints of the past.



TRADITIONAL WORKFLOW

DEVELOPMENT → AUDIT → REMEDIATION → LAUNCH

No Active Security Monitoring
Longer Audit Period
Lenghty Fix Phase
Delayed Launch

WITH AI AUDITING

BUILD → FOCUSED AUDIT → QUICK REMEDIATION → LAUNCH

Continuous AI Security Monitoring
Shorter Audit Period
Fast Fix Phase
On-Time Launch

CLOSING: GIVING TEAMS LEVERAGE THROUGHOUT THE LIFECYCLE

The data is unambiguous: point-in-time audits cannot protect protocols in today's DeFi environment. With $1.48B lost in 2024 despite widespread adoption of traditional audits, the industry needs fundamental change.

Lifecycle security addresses the four critical failure modes: assumption drift through continuous monitoring, post-audit changes via AI scanning, scope limitations with combined AI + human + contest approaches, and post-deployment blind spots through bounties and ongoing protection.

Sherlock's Complete Lifecycle Security has secured over $100 billion in TVL across top protocols, with detection rates that systematically identify issues traditional audits miss. In Web3, where the application is the balance sheet, security must be continuous.

The choice is straightforward: continue with periodic snapshots that miss post-deployment threats, or adopt lifecycle security that protects your protocol as it evolves.

Your protocol changes daily. Your security should too.

> “I’ve tried many different AI audit tools, and none come even close to Sherlock AI. It combines best-in-class AI models with easy-to-use UX and GitHub integration. Other tools are either good at PR runs or manual runs. Sherlock AI is by far the best combination of both.”
>
> **— Jeroen Offerijns, CTO, Centrifuge**

Try Sherlock AI On Your Protocol **sherlock.xyz/contact**

SHERLOCK

SECURE YOUR PROTOCOL WITH SHERLOCK

SHERLOCK.XYZ