Sec3

YEAR-END REPORT: SOLANA SECURITY ECOSYSTEM REVIEW

2025

At a Glance

- **Exploits:** Fewer on-chain smart contract Solana exploits in 2025 (US$ 8mn) down from peak in 2022 (US$ 550mn), even as activity and TVL grew

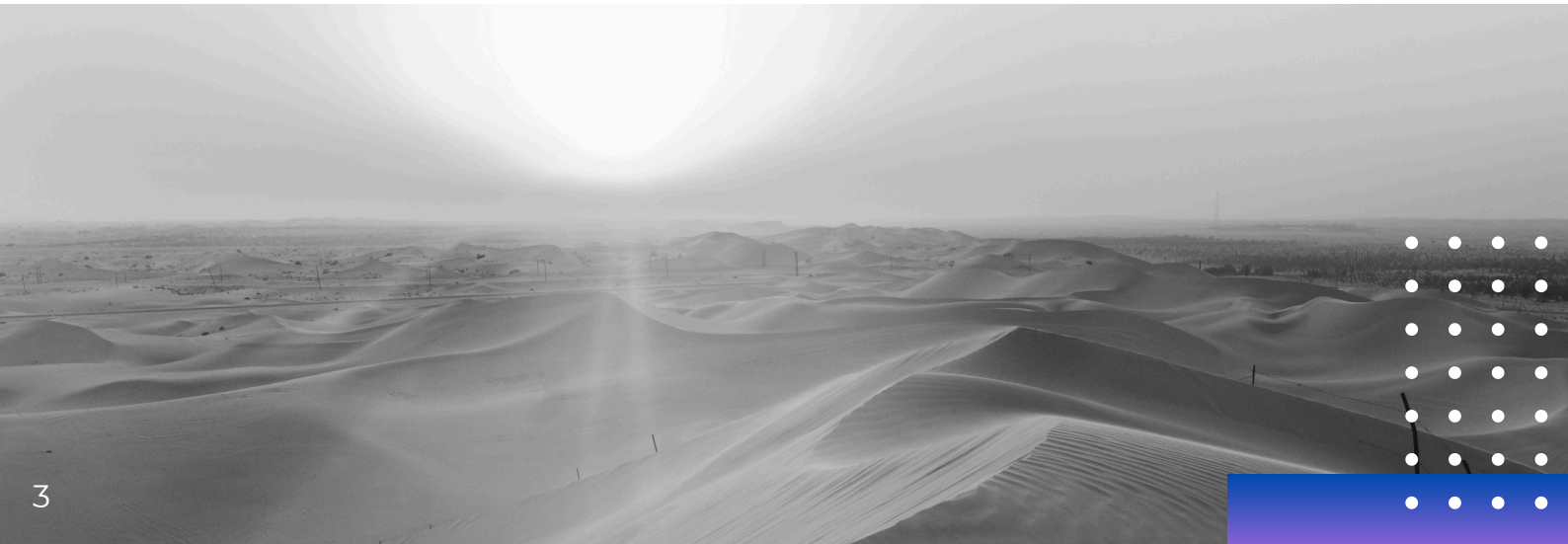
- **Audits Analyzed:** 163 Solana security reviews examined, spanning 1,669 recorded vulnerabilities

- **Finding Density:** Average of 10 issues per audit, with 1.4 High or Critical vulnerabilities in each review

- **Vulnerability Themes:** The most severe issues concentrate in *business logic flaws, access control failures, and protocol design weaknesses*

- **What This Report Delivers:** We translate these audit findings and exploit patterns into actionable guidance for Solana development teams

How To Read This Report



Letter from Sec3

Why we are writing this report

We spend most of the year reviewing Solana programs one code base at a time.

Across that work, patterns emerge - how bugs cluster, which security practices help, and where things can still go wrong even after an audit - but those patterns rarely live in one place. This report is our attempt to put structure around what we are seeing: to quantify what audits actually uncover on Solana, place that evidence next to publicly reported incidents on-chain, and connect both to the design and operational choices development teams were making.

The aim is to give builders, reviewers and ecosystem participants a shared reference point for how Solana security is evolving and where attention is likely to matter most in the next year.

Results:

In 2025, virtually no Solana security reviews came back clean

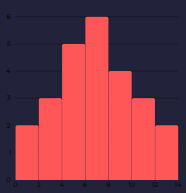
We analyzed **163 Solana security audits** drawn from a mix of publicly released reports and anonymized Sec3 review engagements. Together these reviews produced **1,733** findings, of which **1,669** qualified as vulnerability-level issues.

162 of 163 reviews identified at least one vulnerability*

Statistics:

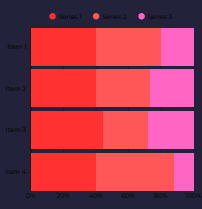
Mean

10.3 Findings Per Review



Median

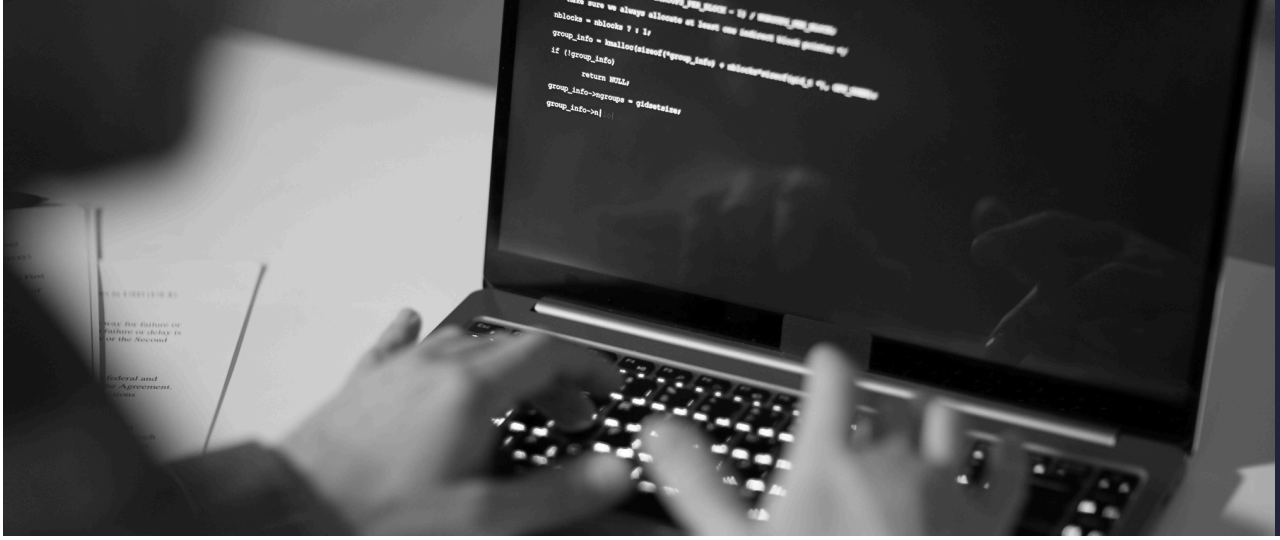
7 Findings Per Review



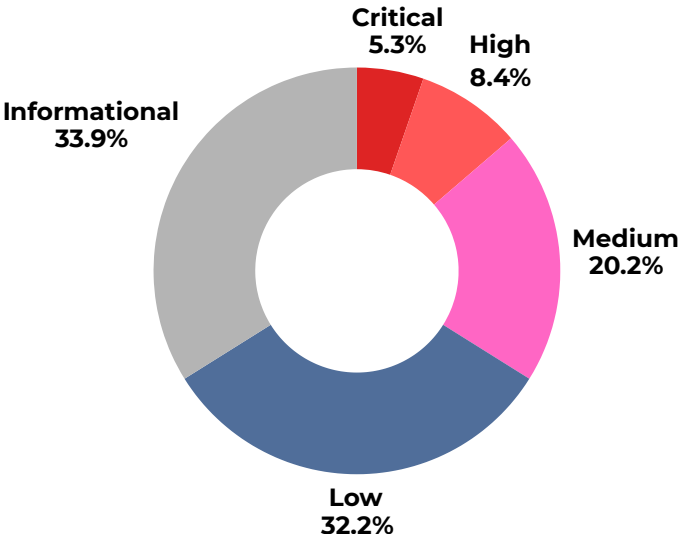
Range

1 To 112 Findings Per Review

[1] The single exception was a small incremental review that surfaced only as a clarification item.



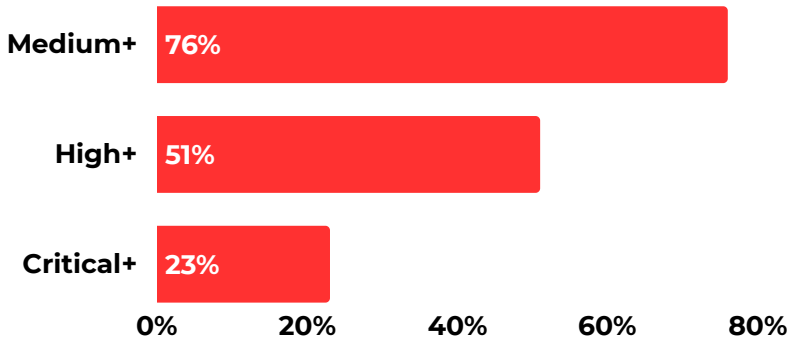
Severity Distribution



Distribution of 1,669 Vulnerabilities

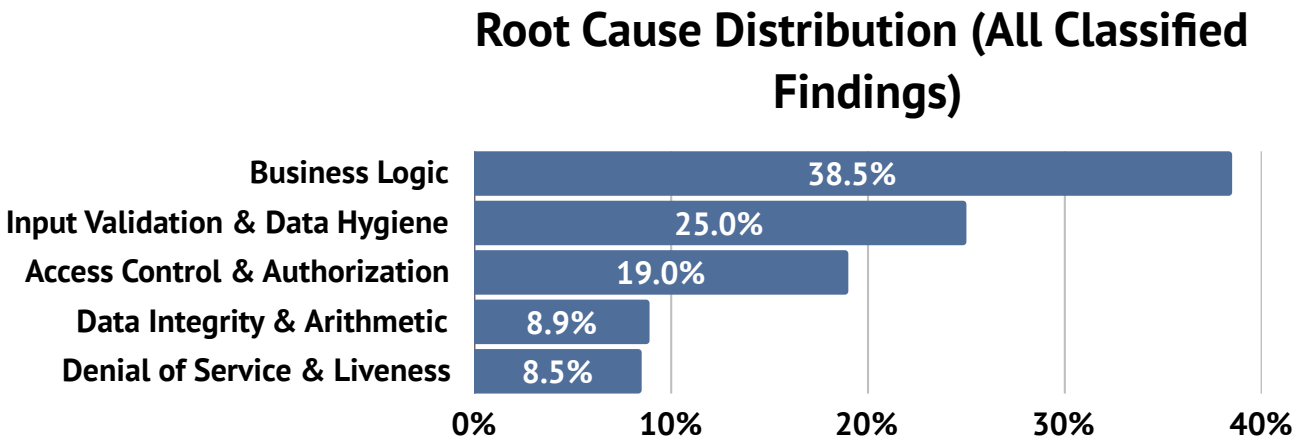
Per Review Finding

- **76% of reviews** contained at least one medium-or-higher issue
- **51% of reviews** contained at least one high-or-critical issue
- **23% of reviews** contained at least one critical issue



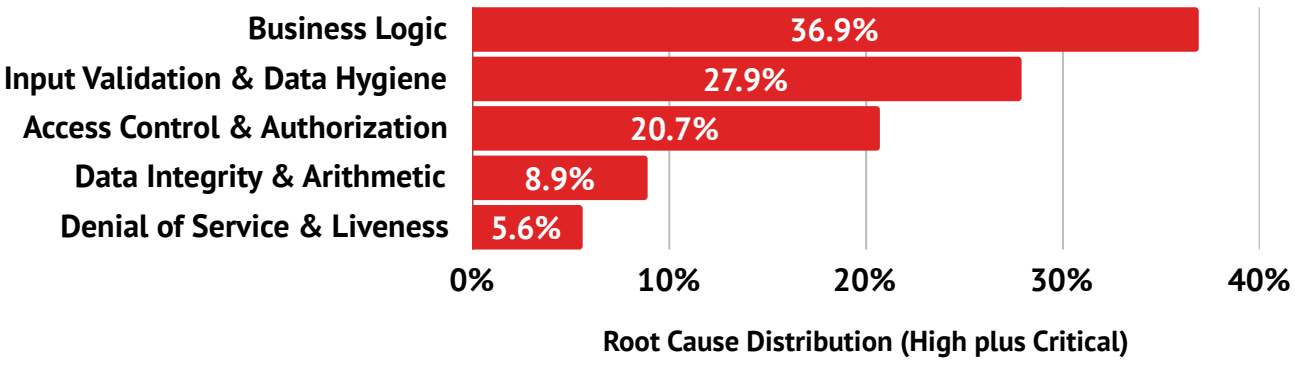
What Vulnerabilities Dominate

Among findings with clear classifications (approximately 70% of the total dataset):



**About 30 percent of findings fell into an "Other Mixed" category representing diverse, lower frequency patterns*

Among High and Critical Vulnerabilities, the distribution is even more concentrated:



Top 3 categories increased from 82.5% → 85.5% of all severe findings

> Serious issues overwhelmingly stem from **Business Logic, Permissions**, and **Validation Errors** rather than low level arithmetic or liveness problems

Top Vulnerability Patterns

We further clustered findings into specific vulnerability types. Excluding the generic "Other Mixed" bucket, the 10 most frequent patterns across all severities:

	Vulnerability	Findings	Details
1	Business Logic or Economic Flaws	413	Incorrect fee/reward accrual, misconfigured liquidation rules, accounting drift over time
2	Input Validation and Sanitation Issues	326	Unbounded user parameters, missing oracle freshness checks, unverified account ownership
3	Arithmetic & Rounding Errors	93	Fixed-point math errors, exploitable rounding at scale, release-build calculation discrepancies
4	Programming Errors	80	Incorrect control flow, module assumption mismatches, incomplete state transitions
5	Denial of Service or Resource Exhaustion	79	Unbounded loops/structures allowing single-account progress blocking
6	Access Control Issues (General)	67	Missing or incomplete permission checks
7	Missing Account Owner Checks	61	Failure to verify account ownership before state modifcations
8	Missing Signer or Authority Checks	36	Operations executable without proper authorization
9	PDA Derivation Errors	32	Incorrect program-derived address computation or validation
10	Lamport and Rent Handling Issues	29	Improper balance management or rent exemption handling

> **Among High-and-Critical issues, Access Control Problems dominate**. Grouping "Access control (general)", "Missing signer/authority check", "Missing account owner check", “PDA derivation issues”, and other access control related patterns together, they account for **over 20% of all high-and-critical findings.**

Methodology & Dataset

Data Sources

Our data sources are 163 Solana security audits drawn from a mix of publicly released reports and anonymized Sec3 review engagements.

Together these reviews produced 1,733 findings, of which 1,669 qualified as vulnerabilities.

Sec3 · Neodyme · Offside Labs · Ottersec · Pashov · Zenith · Zellic · Accretion

Normalize → **Classify** → **Aggregate**

Normalize

Parsed findings into a unified schema with core fields: source file, issue ID, title, severity, vulnerability type, category

Automatic Labeling and Clustering

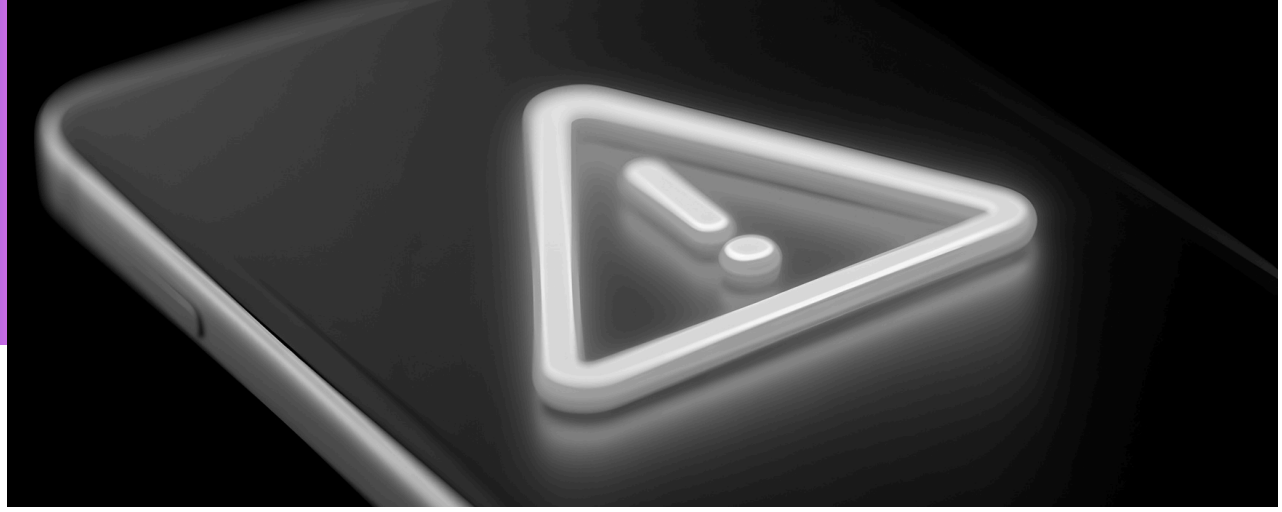
Combined pattern matching and LLM-based classification to assign:

- Vulnerability type
- High-level category

Types were then grouped into canonical labels for cross-report comparison

Sampling and Aggregation

Human reviewers sampled labels to validate groupings. Findings were aggregated to produce the severity distributions, category charts, and frequency tables above



Dataset Scope and Constraints

This dataset is informative but not exhaustive. Key constraints:

1. **Audits do not cover all deployed code**

   Findings represent only projects that engaged auditors and agreed to publish or share results. Unaudited code, informal reviews, and unpublished work fall outside the sample. Read these numbers as "what auditors typically find when they look," not "what exists on-chain overall"

2. **Automated classification is imperfect**

   Despite LLM assistance and manual sampling, some findings inevitably land in the wrong specific type. We maintain a large "Other/Mixed" bucket and avoid over-interpreting rare categories

3. **Severity is defined at report time**

   Severities come from original reviewers. Teams occasionally disagree on edge-case ratings, and some findings shift in importance as protocols evolve or more context emerges. We aggregate severities as reported without re-grading

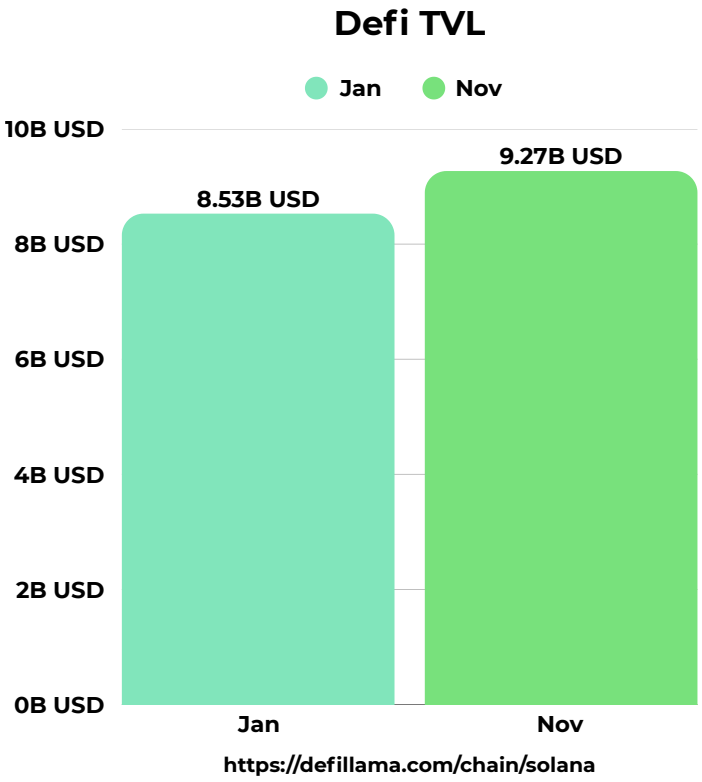
4. **Per-review statistics are averaged**

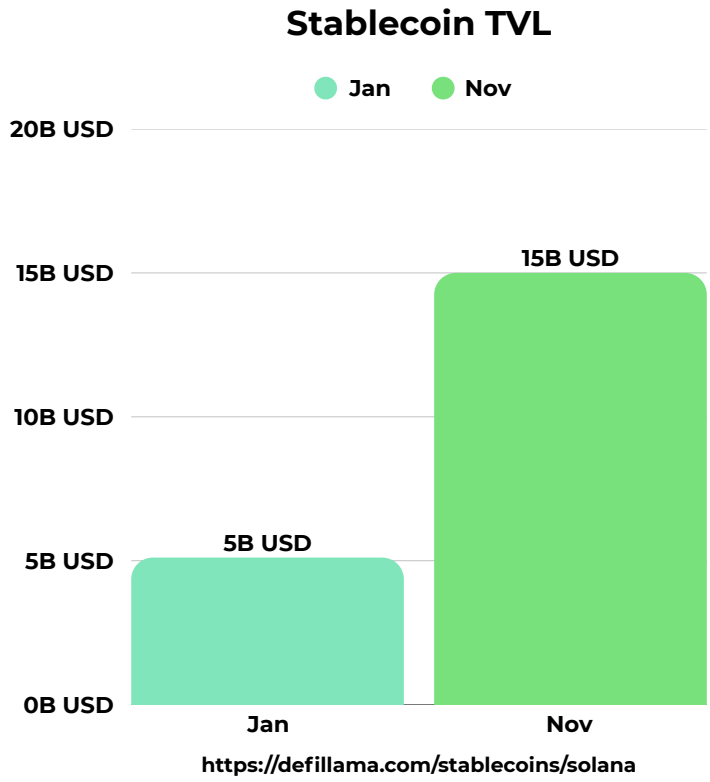
   A small number of very large reviews contribute disproportionately to the total finding count. While the mean is 10.3 findings per review, the median of 7 shows that most audits uncover fewer vulnerabilities. We report both counts and per-review percentages to characterize the full distribution

> Within these constraints, the dataset captures a clear signal: **when independent reviewers examine non-trivial Solana programs, they routinely find Medium, High, and Critical issues** clustering around protocol-specific logic, validation, and access control

Snapshot of Solana in 2025

Higher TVL and accelerating stablecoin growth indicate an ecosystem supporting more applications, more users, and more complex financial activity. As value concentration and protocol complexity increase, the security surface expands





> This broader context makes the decline in large scale exploits particularly notable and strengthens the case that **security practices across the ecosystem have improved**

How Framework Choices Shape Vulnerability Patterns

Most Solana programs use Rust SDK, Anchor, or Pinocchio. Each approach changes where validation happens and which bugs are most common.

	Rust SDK	Anchor	Pinocchio
Guardrail	None by default	Built into the framework	Must be rebuilt manually
Typical Vulnerability Profile	Missing checks, fragile dispatch, incorrect PDA derivations	Protocol logic issues, edge cases, incentive design	Missing owner/signer checks, offset or aliasing errors
Pros	Full control, maximum flexibility	Safe defaults, standardized account validation	Fastest, most compact, lowest compute unit
Cons	Easy to miss basics, lots of boilerplate	More abstraction, larger binaries	Manual validation everywhere, unsafe if rushed
Best Used When	You need total customization	Most new protocols, standard workflows	Highly optimized systems with disciplined engineering

> **The frameworks aren't inherently safe or unsafe.** The difference is simply where validation happens: in the framework (Anchor) or in the team's engineering discipline (Pinocchio).

Before the Audit: How Strong Teams Prepare

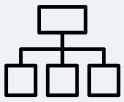
Security outcomes are rarely an accident - strong teams do real work before auditors open the repo

Plan the audit

- Engage auditors early and lock dates
- Define scope: programs that hold value, move balances, govern upgrades, or depend on external assets
- Create dedicated audit branch
- Budget time post review for fixes

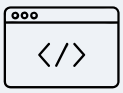
Make the system easy to understand

- Overview: purpose, assets, users
- Roles & permissions table
- Account & PDA map
- Core flow diagrams
- Invariants & dependencies



Ship reviewable code and tests

- Clear structure, no dead code
- Good naming, pinned toolchains
- Simple build/test scripts
- Unit, integration, negative tests
- Invariant and scenario tests



Reduce noise before the audit

- Run manual checks for signers, owners, layouts, PDAs
- Use CI with linting, static analysis, fuzzing
- Audit the high risk modules first
- Clean keys, infra, governance before audit

After Launch: Running A Secure Protocol

An audit is a snapshot, not a guarantee - strong teams treat launch as the beginning of security work

Roll out safely

- Use TVL caps and per vault limits during early launch
- Test with restricted or beta groups before opening access widely
- Gate with pause or withdraw only modes with clear activation criteria
- Constrain impact from unaudited or lightly reviewed modules

Monitor continuously

- On chain: TVL and net flows, liquidation behaviour and errors, oracle deviations, internal invariant alerts
- Off chain: keeper and relayer health, unusual traffic, role or permission changes, failed recurring jobs

Treat changes as new risk

- Run delta or incremental audits for major refactors and new features
- Conduct targeted reviews when you add leverage, new collateral, new yield routes, or new external dependencies
- Setup real bug bounty with clear scope, meaningful payouts, and fast disclosure paths
- Assume a single unaudited commit can invalidate an earlier audit

Be ready for incidents and an evolving stack

- Maintain a written incident-response plan with clear severities, roles, and technical levers (pauses, key rotation, emergency configs)
- Keep a pre-vetted contact list: auditors, SEAL 911, infra and bridge partners, and exchanges
- Review key management, access control, governance safety, and frontend or distribution integrity on a regular cadence
- Run tabletop exercises or game days so teams know how to act under pressure

Where Risk Is Moving

Fewer failures come from simple bugs in isolated programs - more risk now sits in how systems, automation, and people interact

Stacked systems

- Restaking, leverage, looping, multi-protocol strategies
- Shared collateral and infra create correlated failures
- Need stack level stress tests, not just per protocol audits

Yield bearing stablecoins

- Supply has jumped into the low double digit billions
- Risk is in the portfolio behind each dollar, not just the peg
- Problems can ripple through treasuries, treasuries-backed tokens, and structured products

Vibe coded contracts

- Copy paste and AI generated code shipped with thin specs and tests
- Forks tweak logic assuming changes are safe, but inadvertently break critical security guardrails
- Vulnerabilities spread through templates faster than they are fixed

Agents and prompt injection

- Agents act on natural language goals, with keys and x402 payment rails
- Malicious data or prompts can steer them to misuse otherwise safe contracts
- Expect “agent abuse” incidents with no underlying code bug

Wallets and the human layer

- Wallet drains and social engineering still drive most losses
- A single permit or broad approval can empty accounts
- Protocols suffer reputational damage even when their contracts are clean

Raising The Bar Before Complexity Catches Up

CODE EXPLOITS ARE DOWN AND NO ONE FORCES AUDITS YET, WHICH IS EXACTLY WHY WE CAN'T RELAX

In 2025 we saw Solana's security culture continue to grow: fewer catastrophic bugs, deeper audits, and teams starting to design with failure in mind. The next step is simpler and harder at the same time - keep treating security as a constant, not a checkbox, so as complexity grows through stacked protocols and yield-bearing products, the same security discipline still applies

As Solana continues to scale rapidly, security practices must evolve alongside the expanding attack surface

Contact Us

contact@sec3.dev

https://www.sec3.dev

https://x.com/sec3dev

Thank You