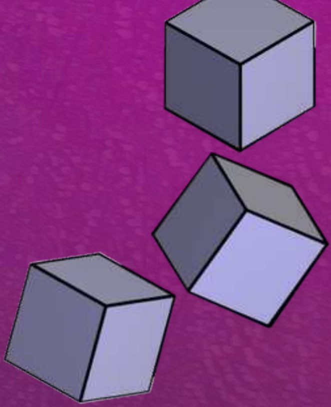
2025
EVM SMART CONTRACT EXPLOIT ANALYSIS

Olympix



CONTENTS

Executive Summary **02**

2025 EVM Smart Contract Exploits **03**

In-Scope Exploits 03

Distribution of Vulnerability Types 07

Top 3 Exploits by Economic Impact 09

Analysis & Risk Implications 13

What 2025's Data Reveals About Security Maturity **17**

2026 Recommendations for Enterprise Teams **18**

Beyond Audits: A Multilayered Security Approach 18

Operationalizing Security Earlier in the SDLC 20

Deterministic vs Probabilistic Security: Why Proof Matters 21

About Olympix **22**

EXECTUTIVE SUMMARY

This report presents a structured review of EVM smart contract exploits that occurred throughout 2025, with the objective of assessing preventability, identifying systemic risk patterns, and evaluating what the data reveals about the current maturity of crypto security practices.

The analysis focuses on exploits arising from vulnerabilities in Solidity-based smart contracts deployed within the EVM ecosystem. These incidents represent failures in logic, accounting, access control, or invariant enforcement, rather than offchain compromise, key theft, or social engineering.

A central theme of this report is preventability.

In this context, an exploit is considered preventable when the vulnerable logic existed in contract code prior to deployment and could have been identified through deterministic security testing during development.

Across the in-scope dataset, the findings are unambiguous.

Of the 50 EVM smart contract exploits reviewed, Olympix would have prevented 49, representing approximately 98 percent of incidents and roughly $240 million in preventable losses. Only a single exploit fell outside the scope of vulnerabilities detectable prior to deployment.

These results underscore a broader conclusion. The majority of losses in 2025 were not the result of novel attack techniques, but of vulnerabilities introduced earlier in development and left undiscovered until exploitation.

This report documents those failures, examines their implications, and outlines how enterprise teams can evolve their security posture heading into 2026.

2025 EVM SMART CONTRACT EXPLOITS

In-Scope Exploits

This review focuses exclusively on exploits arising from vulnerabilities in EVM smart contracts written in Solidity.

Only incidents in which loss resulted from flaws in onchain contract logic were included in the analysis. The objective of this scoping was to isolate failures attributable to smart contract correctness, rather than operational, human, or infrastructure compromise.

Specifically, an exploit was considered in scope if:

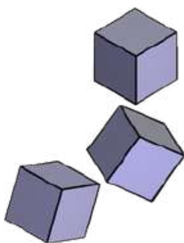
- the affected protocol operated within the EVM ecosystem
- the vulnerable component was a Solidity-based smart contract
- the exploit resulted from logic, accounting, access control, or invariant failures
- the vulnerable code existed prior to deployment
- the exploit was executed through valid onchain transactions

Incidents were considered out of scope if losses resulted from factors unrelated to smart contract correctness, including:

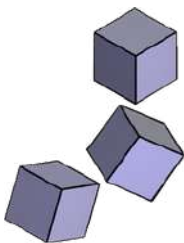
- private key compromise or wallet leakage
- social engineering or phishing attacks
- infrastructure or backend system breaches
- exploits occurring on non-EVM chains
- exit scams or incidents where contract code was unavailable for review

This scoping ensures the analysis reflects only those failures that could reasonably be detected and prevented through deterministic smart contract security testing during development.

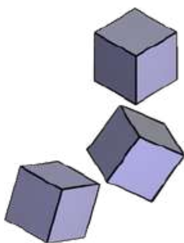
A complete list of the 2025 exploits analyzed can be found below.



Project	Date	Technique	Amount Lost	Could have been prevented?
Sorra Finance	01/04/2025	Logic Vulnerability	$41,000	Yes
Mosca	01/08/2025	Logic Vulnerability	$19,500	Yes
FortuneWheel	01/10/2025	Access Control	$21,000	Yes
UniLend Finance	01/13/2025	Logic Vulnerability	$200,000	Yes
Idols NFT	01/15/2025	Reward Calculation Error	$340,000	Yes
AST Token	01/21/2025	Logic Vulnerability	$65,000	Yes
ODOS Protocol	01/23/2025	Insufficient Input Validation	$50,000	Yes
BankX	02/08/2025	Reentrancy	$43,000	Yes
Hegic Options	02/23/2025	Logic Vulnerability	$80,000	Yes
Venus	02/27/2025	Price Manipulation	$716,000	No
Zoth	03/01/2025	Logic Vulnerability	$285,000	Yes
SIR	03/30/2025	Logic Vulnerability	$355,000	Yes
KiloEx	04/14/2025	Unauthorized Access	$7,400,000	Yes
Numa	04/18/2025	Price Manipulation	$527,000	Yes
Zora	04/24/2025	Access Control	$128,000	Yes
Impermax	04/26/2025	Price Manipulation	$300,000	Yes
MBU	05/11/2025	Precision Loss	$2,200,000	Yes
Nitron	05/16/2025	Price Manipulation	$951,000	Yes
LNDFi	05/16/2025	Unauthorized Access	$1,300,000	Yes
Dexodus	05/26/2025	Signature Replay	$291,000	Yes
Cork Protocol	05/28/2025	Missing Authorization Checks	$12,000,000	Yes
Meta Pool	06/17/2025	Missing Override	$137,000	Yes
ResupplyFi	06/26/2025	Price Manipulation	$9,600,000	Yes
Future Protocol	07/02/2025	Price Manipulation	$4,600,000	Yes



Project	Date	Technique	Amount Lost	Could have been prevented?
Rant	07/05/2025	Price Manipulation	$204,000	Yes
GMX	07/09/2025	Price Manipulation	$42,000,000	Yes
H2O Token	03/14/2025	Logic Vulnerability	$22,000	Yes
WebKey	03/14/2025	Arbitrage	$737,000	Yes
BBX	03/20/2025	Price Manipulation	$12,000	Yes
Alkimiya	03/28/2025	Logic Vulnerability	$96,000	Yes
WETC Token	07/17/2025	Logic Vulnerability	$100,000	Yes
SuperRare	07/28/2025	Access Control	$680,000	Yes
Numa	08/10/2025	Logic Vulnerability	$313,000	Yes
Betterbank	08/27/2025	Logic Vulnerability	$5,000,000	Yes
Bunni	09/02/2025	Rounding Error	$8,400,000	Yes
Kame Aggregator	09/12/2025	Missing Validation Check	$1,300,000	Yes
New Gold Protocol	09/18/2025	Price Manipulation	$2,000,000	Yes
Griffin AI	09/25/2025	Misconfigured Deployment	$3,000,000	Yes
dTrinity	09/28/2025	Access Control	$56,000	Yes
Abracadabra	10/04/2025	Logic Vulnerability	$1,700,000	Yes
Typus Finance	10/15/2025	Missing Authorization Checks	$3,400,000	Yes
Sharwa Finance	10/20/2025	Missing Slippage Protection	$147,000	Yes
Balancer	11/03/2025	Precision Loss	$121,100,000	Yes
Moonwell	11/04/2025	Faulty Price Oracle	$1,000,000	Yes
DRLVaultV3	11/10/2025	Slippage Exploit	$98,000	Yes
Port3	11/23/2025	Logic Vulnerability	$166,000	Yes
Yearn Finance	11/30/2025	Logic Vulnerability	$9,000,000	Yes
DMi Token	12/8/2025	Logic Vulnerability	$124,000	Yes
Prism Protocol	12/11/2025	Access Control & Price Manipulation	$62,000	Yes
1inch	03/05/2025	Logic Vulnerability	$5,000,000	Yes

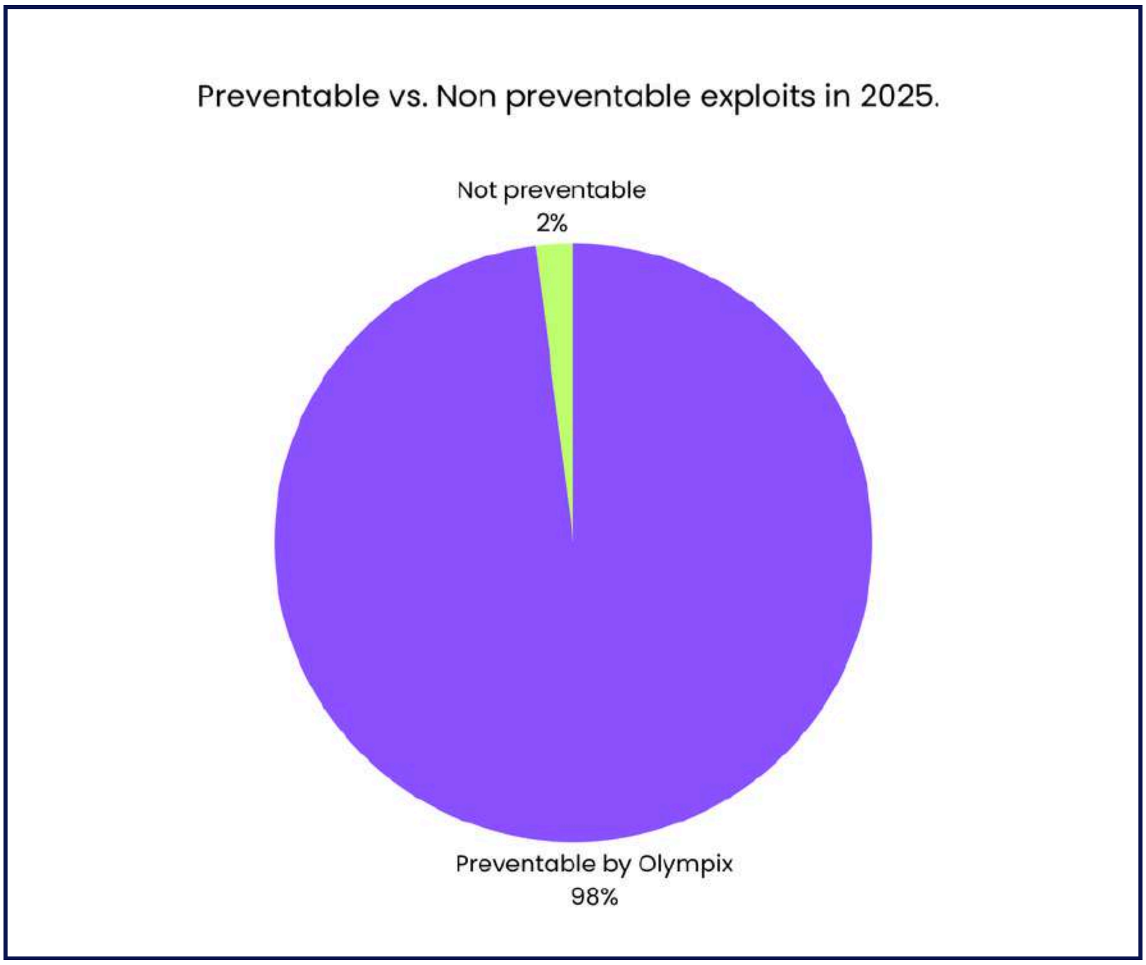


Preventability Summary

Across the 50 in-scope EVM exploits reviewed:

- 49 exploits were preventable using Olympix prior to deployment
- 98 percent of incidents fell within detectable vulnerability classes
- Approximately $240 million in losses were preventable

These findings reflect a consistent pattern across the year. Vulnerable logic was present in deployed contracts well before exploitation occurred, often long before audits or monitoring tools could intervene.



Distribution of Vulnerability Types

A small set of vulnerability categories accounted for the majority of exploit activity.

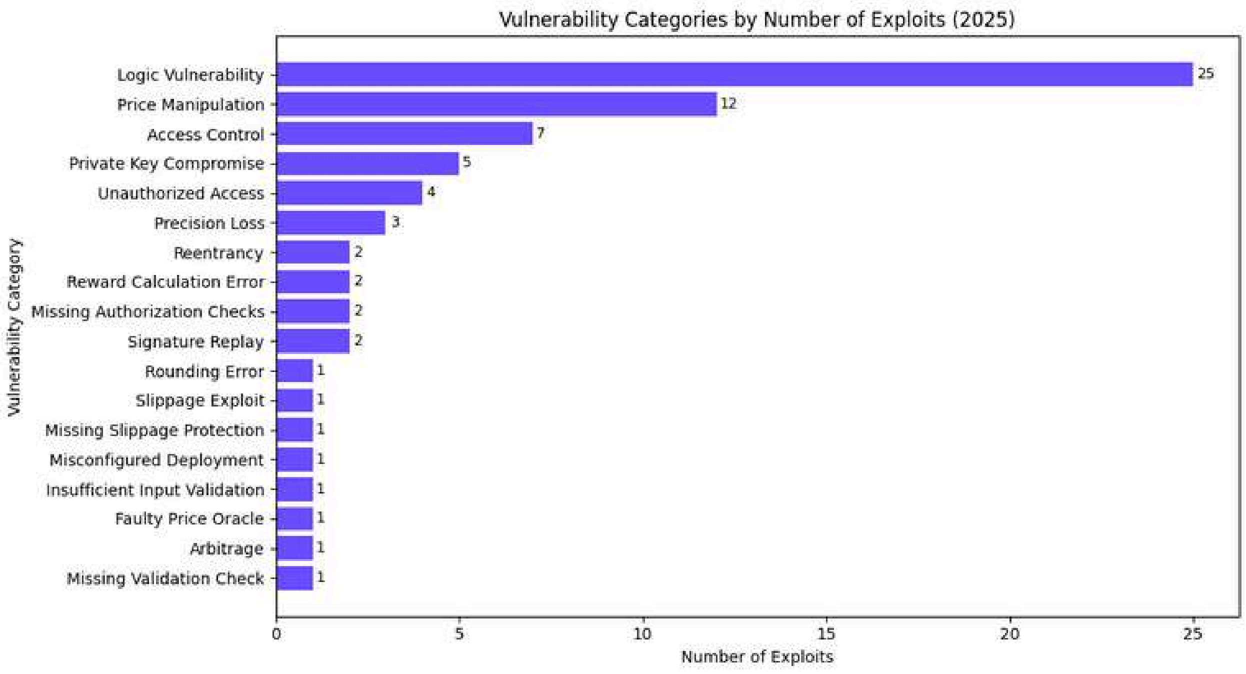
Most Commonly Exploited Vulnerability by Count

Logic vulnerabilities were the most frequently exploited category in 2025.
These failures typically involved:

- incorrect assumptions about state transitions
- unsafe ordering of operations
- multi-step execution paths that violated intended behavior

Attackers did not bypass permissions or exploit infrastructure weaknesses. They used valid function calls.

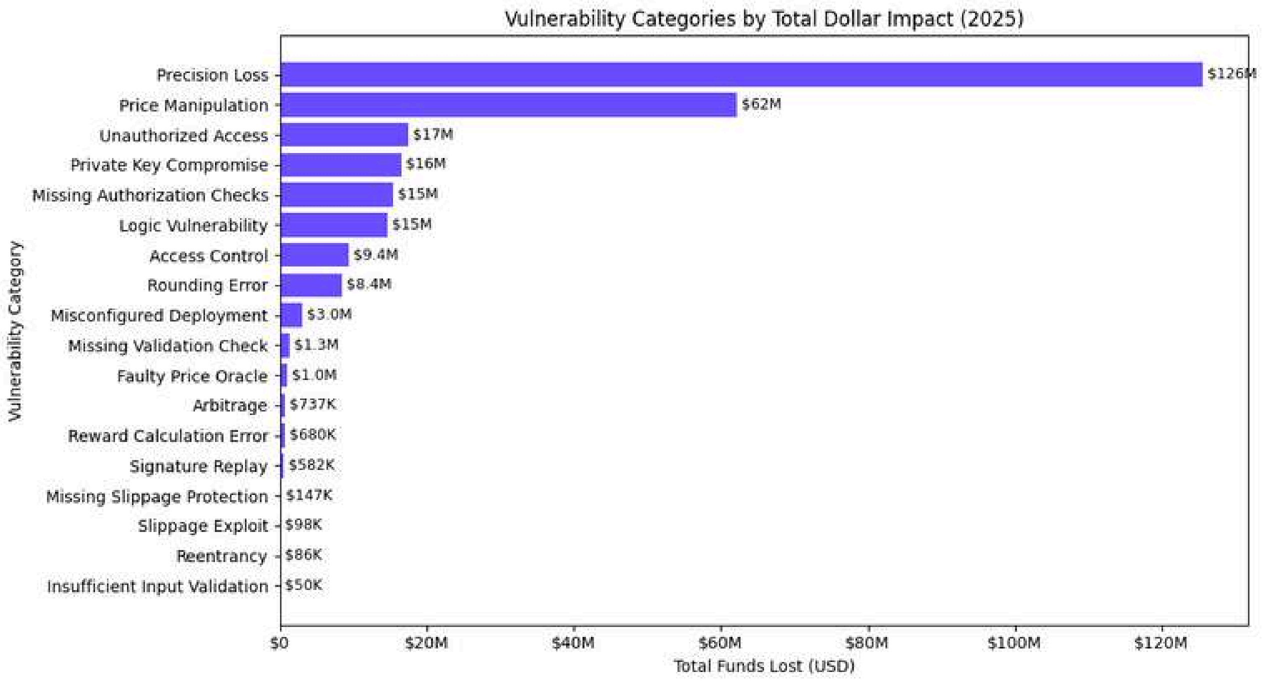
The protocol behaved exactly as written.



Accounting and precision errors were responsible for the highest total dollar losses.

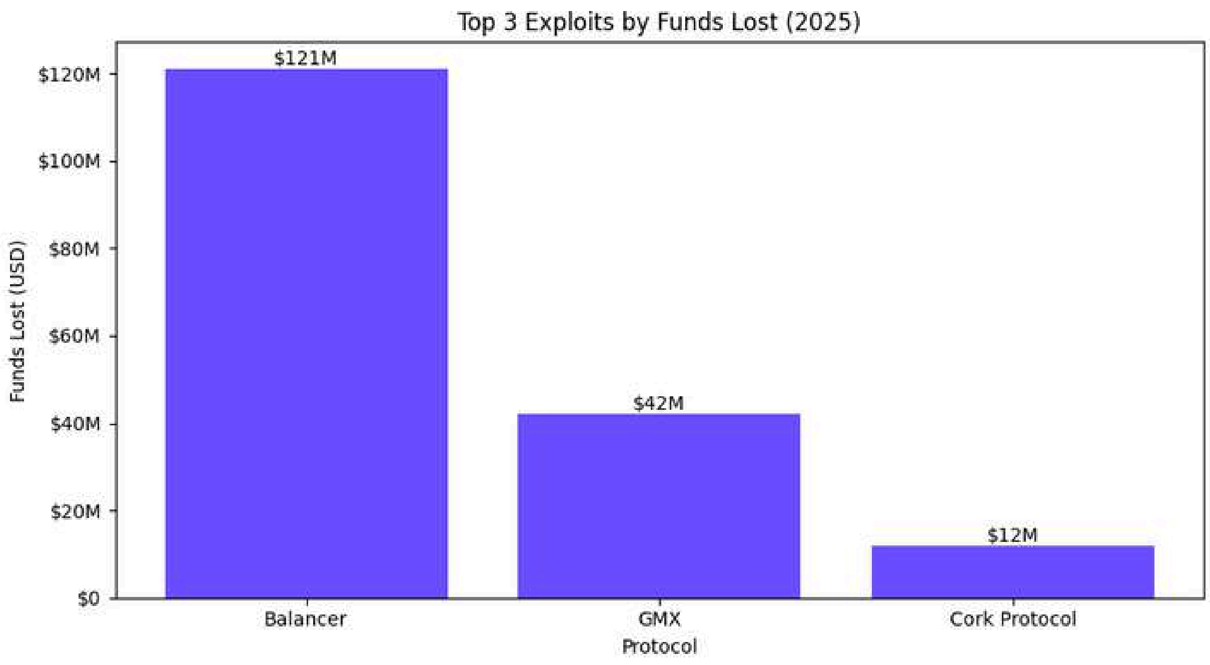
Although fewer in number, these vulnerabilities had severe consequences due to their presence in high liquidity systems. Small numerical errors scaled catastrophically when applied repeatedly.

This imbalance between frequency and impact is one of the most important lessons of 2025.



Top 3 Exploits by Economic Impact

A small subset of incidents accounted for a disproportionate share of annual losses. Each of these exploits stemmed from deterministic logic failures that existed prior to audit and deployment.



Exploit postmortems below.

Balancer

Loss: $121.1M
Date: November 3, 2025

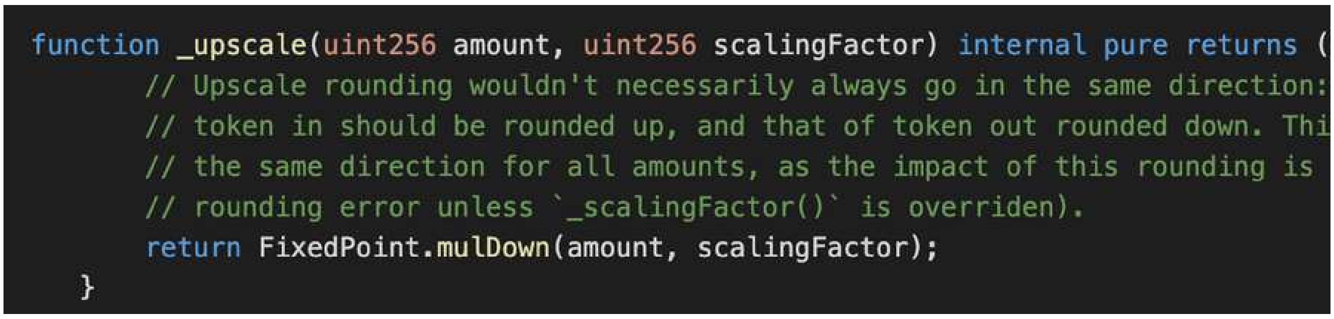
Balancer V2 suffered one of the largest exploits of 2025 after attackers abused a precision and rounding flaw in Composable Stable Pools. Through carefully calibrated micro-swaps, attackers were able to erode pool invariants and extract value across multiple networks.

The exploit was not caused by oracle failure, governance compromise, or privileged access. It originated entirely from deterministic accounting logic.

Root Cause: Asymmetric rounding in rate-augmented scaling logic.

Scaling functions applied consistent downward rounding while embedding dynamic token rates. Under specific pool conditions, this introduced a persistent rounding bias that allowed attackers to underpay during EXACT_OUT swaps.

Repeated over many operations, this bias degraded the pool invariant and deflated BPT value.



```
function _upscale(uint256 amount, uint256 scalingFactor) internal pure returns (
        // Upscale rounding wouldn't necessarily always go in the same direction:
        // token in should be rounded up, and that of token out rounded down. Thi
        // the same direction for all amounts, as the impact of this rounding is 
        // rounding error unless `_scalingFactor()` is overriden).
        return FixedPoint.mulDown(amount, scalingFactor);
    }
```

How Olympix Would Have Prevented This

Olympix's precision analysis identified biased rounding in rate-augmented scaling factors as a High-severity vulnerability.

The system flagged:

- inconsistent rounding direction across scaling operations
- amplification risk when dynamic rates are embedded
- invariant degradation under adversarial pool states

This class of vulnerability is extremely difficult to surface through manual review, but well-suited to automated, deterministic analysis.

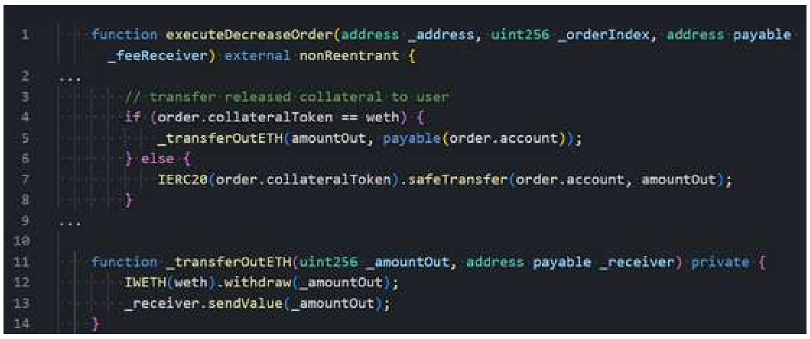
Loss: ~$40M
Date: July 9, 2025

In July 2025, GMX V1 suffered a ~$40M exploit caused by a design flaw in its position execution flow. A user-controlled receiver address enabled a cross-contract reentrancy attack that bypassed routing safeguards and corrupted price calculations used to value GLP.

Root Cause: Unconstrained user-controlled receiver execution.

The decreasePosition flow allowed users to specify an arbitrary receiver address. When this receiver was a contract, execution control was transferred externally before all internal state transitions and invariants were finalized.

This enabled cross-contract reentrancy into sensitive vault functions, bypassing expected routing through GMX's PositionRouter and PositionManager.



```
    function executeDecreaseOrder(address _address, uint256 _orderIndex, address payable
      _feeReceiver) external nonReentrant {
...
        // transfer released collateral to user
        if (order.collateralToken == weth) {
            _transferOutETH(amountOut, payable(order.account));
        } else {
            IERC20(order.collateralToken).safeTransfer(order.account, amountOut);
        }
...

    function _transferOutETH(uint256 _amountOut, address payable _receiver) private {
        IWETH(weth).withdraw(_amountOut);
        _receiver.sendValue(_amountOut);
    }
```

How Olympix Would Have Prevented This

Olympix's source-level analysis would have flagged the dynamic receiver execution path as exploitable.

The system identifies:

- user-controlled external call sites prior to invariant finalization
- cross-contract reentrancy exposure beyond local guards
- sequencing violations where funds or execution control are transferred prematurely

Through mutation testing and invariant validation, Olympix simulates mid-transaction reentry and validates that price and exposure assumptions cannot be violated under adversarial execution.

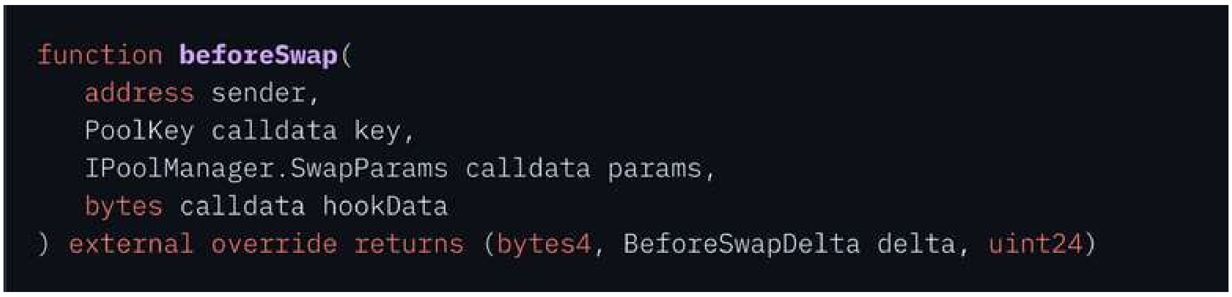
Loss: ~$11M
Date: May 28, 2025

In May 2025, Cork Protocol suffered an ~$11M exploit due to critical weaknesses in its Uniswap V4 hook implementation. Missing access control allowed attackers to invoke hook logic directly, mint derivative tokens without depositing collateral, and redeem them for valuable assets.

Root Cause: Missing access control and insufficient validation in Uniswap V4 hooks.

Critical hook callbacks, including beforeSwap, lacked restrictions ensuring they could only be invoked by the Uniswap V4 PoolManager. This allowed arbitrary contracts to call hook logic directly with attacker-controlled parameters.

Additional weaknesses in token-type validation and market configuration compounded the exploit.



```
function beforeSwap(
   address sender,
   PoolKey calldata key,
   IPoolManager.SwapParams calldata params,
   bytes calldata hookData
) external override returns (bytes4, BeforeSwapDelta delta, uint24)
```

How Olympix Would Have Prevented This

Olympix's source-level analysis would have flagged the hook callbacks as unsafe due to missing PoolManager-only access enforcement.

The system identifies:

- externally callable hook functions lacking trusted caller validation
- execution paths where hook logic assumes correct invocation context
- missing invariants around asset deposits and derivative issuance

Through mutation testing and invariant validation, Olympix simulates adversarial hook invocations and verifies that minting logic cannot be triggered without actual collateral movement.

Analysis & Risk Implications

Smart contract exploits in 2025 followed two clear and contrasting trends.

Exploit activity was distributed relatively evenly across the year, while total financial losses were heavily concentrated in a small number of high-impact incidents.

This divergence between frequency and severity defines the 2025 risk landscape.

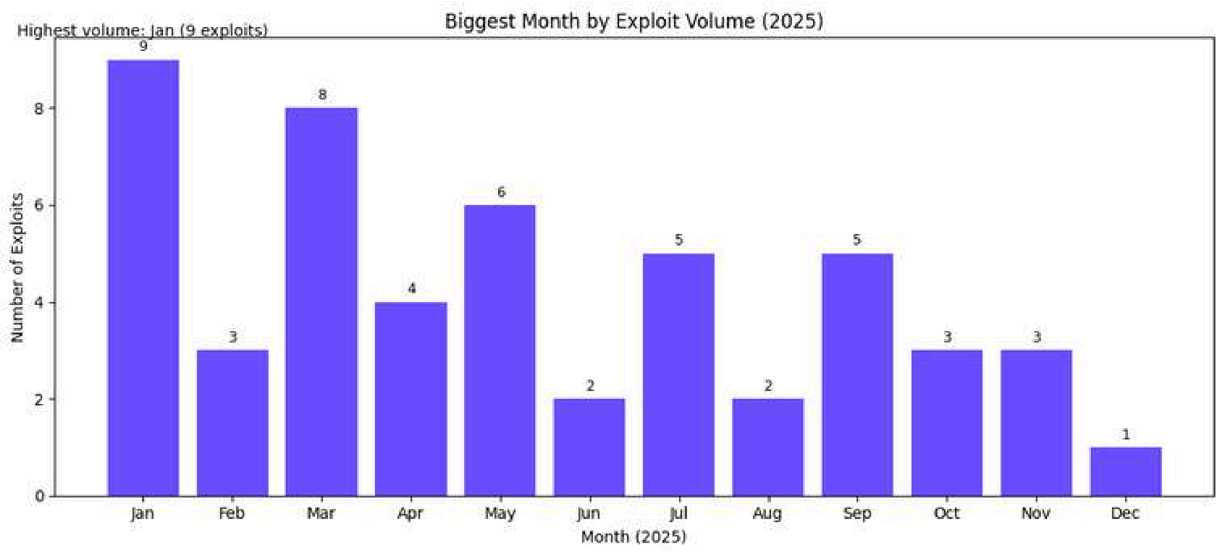
Hacks by Volume vs Hacks by Revenue

Biggest month by exploit volume: January

January recorded the highest number of independent exploit events in 2025, with nine separate incidents.

The majority of these exploits stemmed from logic vulnerabilities and access-control failures. While individual losses were relatively small compared to later incidents, their frequency illustrates how often preventable vulnerabilities continue to ship into production.

This month reflects the most common failure mode of 2025, frequent, low-to-mid impact exploits caused by unsafe assumptions in contract logic.

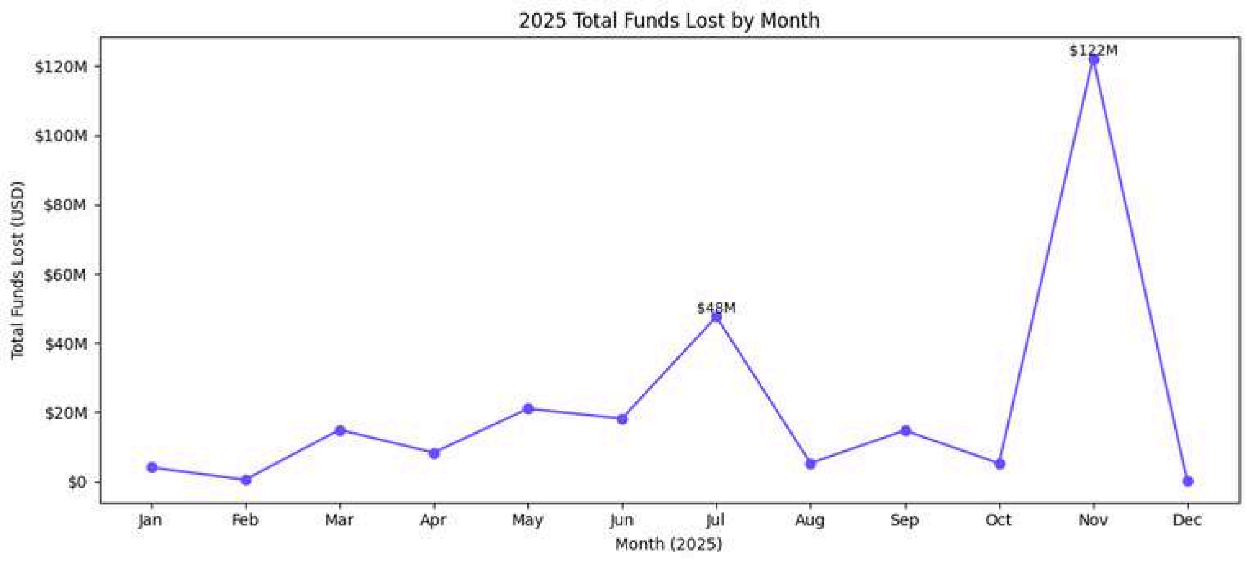


Biggest month by exploit revenue: November

November accounted for the largest share of total funds lost in 2025, despite having far fewer incidents than January.

This spike was driven primarily by a single catastrophic exploit, which outweighed the cumulative losses of multiple earlier months combined.

The data shows that exploit severity does not correlate with exploit frequency. One accounting or precision failure in a high-liquidity system can eclipse dozens of smaller incidents.



Risk Implications

Analytical Finding	Why This Matters
Most Losses Were Caused by Unmeasured Assumptions	The majority of high-impact exploits did not stem from unknown vulnerability classes, but from implicit assumptions about how contracts would be used, composed, or interacted with. When assumptions are not explicitly modeled or tested, risk remains invisible until exploitation occurs.

Risk implications continued on next page.

Analytical Finding	Why This Matters
Most Losses Were Caused by Unmeasured Assumptions	The majority of high-impact exploits did not stem from unknown vulnerability classes, but from implicit assumptions about how contracts would be used, composed, or interacted with. When assumptions are not explicitly modeled or tested, risk remains invisible until exploitation occurs.
Audits Cannot Serve as First Discovery	Audits evaluate code that has already been architecturally finalized. When vulnerabilities are identified at this stage, remediation often requires redesign rather than correction, significantly increasing cost, delay, and residual risk.
High Test Coverage Does Not Equal Adversarial Coverage	Many exploited execution paths were technically covered by tests but never meaningfully exercised under adversarial conditions. Coverage metrics alone fail to measure whether critical invariants actually hold under hostile inputs.
Security Confidence Is Often Assumed — Not Measured	Many teams rely on qualitative confidence derived from audits or test completion rather than measurable, reproducible guarantees. Without provable assurance, residual risk cannot be meaningfully evaluated or compared.
Manual Review Does Not Scale With Deployment Velocity	As deployment frequency and composability increase, reliance on human review introduces coverage gaps that grow with scale. Manual processes cannot match the pace of modern smart contract development.
Security Signals Arrive Too Late in the Development Lifecycle	Most vulnerabilities originated during design or early implementation, yet detection occurred only after integration or deployment. This timing mismatch creates structural blind spots that cannot be closed through downstream controls.

Analytical Finding	Why This Matters
Security Failures Compound Upstream	Vulnerabilities discovered later in the lifecycle are harder to remediate safely. Late discovery increases the likelihood of rushed fixes, partial mitigations, or acceptance of known risk in production systems.
Risk Is Highly Concentrated — Not Evenly Distributed	Losses followed a fat-tailed distribution, with a small number of incidents accounting for a disproportionate share of total economic impact. This makes average exploit size or incident count unreliable indicators of true exposure.
Exploit Frequency and Economic Severity Are Decoupled	The month with the highest number of exploits was not the month with the highest total losses. Incident volume alone fails to capture systemic risk, as fewer but more severe exploits can dominate outcomes.
Economic Impact Is Decoupled From Code Complexity	Contract size or architectural complexity did not reliably correlate with exploit severity. Exposure was more strongly driven by value concentration than by lines of code or perceived technical sophistication.
Business Logic Failures Dominate High-Impact Exploits	The most damaging incidents resulted from violations of intended economic behavior rather than syntactic errors. This reinforces that correctness cannot be inferred from compilation success or pattern-based detection alone.
Post-Deployment Monitoring Is Inherently Reactive	On-chain monitoring detects exploitation after vulnerable logic has already been deployed and funds are at risk. While essential for response and visibility, monitoring alone cannot prevent first-order losses.

WHAT 2025'S DATA REVEALS ABOUT SECURITY MATURITY

Several conclusions emerge clearly from the 2025 exploit data.

1. Most losses did not stem from unknown vulnerabilities.

They resulted from assumptions that were never explicitly tested, measured, or enforced. In many cases, protocol logic behaved exactly as written, but not as intended. When assumptions remain implicit, they become invisible sources of risk.

2. Audits remain essential, but they cannot serve as the first line of discovery.

Audits evaluate systems that are already architecturally defined. By the time code reaches review, foundational design decisions have been made and embedded. Audits can identify flaws, but they cannot substitute for validation during development.

3. High test coverage does not equate to adversarial coverage.

Many exploited execution paths were technically covered by tests, yet never exercised under adversarial conditions. Coverage metrics measure whether code is executed, not whether it behaves safely when inputs are hostile or states are extreme.

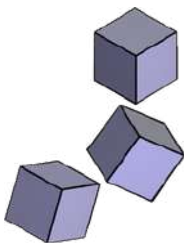
4. Security failures compound as they move downstream.

The later a vulnerability is discovered in the lifecycle, the more costly and risky it becomes to remediate. Issues identified after deployment often require mitigations, compensating controls, or emergency response, rather than clean correction. Prevention upstream consistently delivers the highest risk reduction.

The 2025 data indicates that the industry remains in an early stage of security maturity.

Security practices are largely focused on detection and response, rather than prevention and correctness. Assurance is derived from process completion (audits, test coverage, reviews) rather than from deterministic validation of intended behavior.

Until security outcomes become measurable and reproducible, risk will continue to scale faster than protection.



2026 RECOMMENDATIONS FOR ENTERPRISE TEAMS

Beyond Audits: A Multilayered Security Approach

The 2025 exploit data makes one conclusion unavoidable: no single security control is sufficient to protect complex smart contract systems operating at scale.

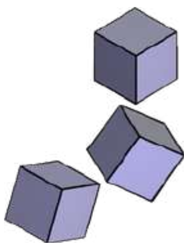
Proactive tooling, audits, monitoring, and incident response each play an important role, but they address different phases of risk. Effective security posture emerges not from relying on one mechanism, but from combining complementary controls across the full lifecycle.

- **Anchor security in development.** The highest leverage security decisions occur during development, before code is deployed or reviewed. Deterministic security tools applied at this stage provide the strongest form of assurance because they evaluate correctness while assumptions are still malleable.

- **Use audits as independent validation, not primary discovery.** Audits remain an essential component of any serious security program, particularly for independent review and external assurance. Enterprise teams should treat audits as a confirmation layer rather than a discovery mechanism, ensuring that the most critical logic has already been validated prior to review. This approach improves audit outcomes, reduces friction, and shifts audit conversations from reactive remediation to risk confirmation.

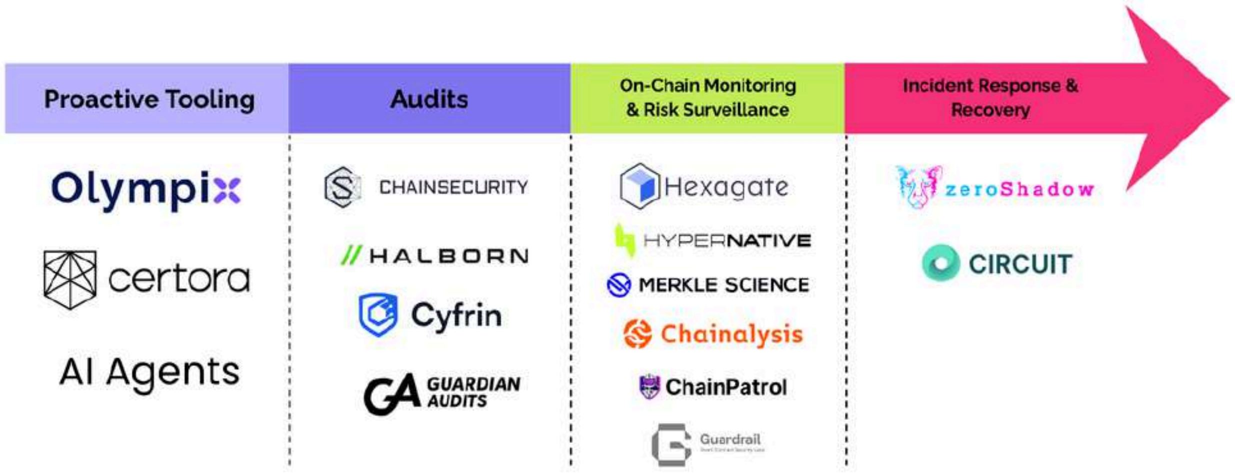
- **Rely on monitoring for detection, not prevention.** On-chain monitoring systems provide visibility once contracts are live. They are indispensable for detecting abnormal behavior, alerting teams, and reducing blast radius during active incidents. However, monitoring is inherently reactive. It observes consequences, not causes. Enterprise teams should view monitoring as a response acceleration mechanism, not a substitute for upstream correctness.

- **Integrate Risk and Intelligence for Contextual Awareness.** These systems enhance situational awareness, support investigations, and assist with compliance and reporting obligations. They complement technical security controls but do not replace them. Their value increases significantly when paired with strong preventive controls upstream.



- **Prepare explicitly for incident response and recovery.** Even mature security programs must assume that incidents can occur. Defined response playbooks, trusted recovery partners, and clear internal ownership are essential for minimizing damage when failures happen. However, response capabilities should be treated as a last line of defense, not a risk strategy. Recovery does not negate loss, reputational damage, or operational disruption. The objective remains reducing the likelihood of incidents reaching this stage at all.

See recommended providers for each stage of the security lifecycle below.



Areta evaluates proactive tooling and audit firms to provide whitelists of providers in key ecosystems such as Uniswap, Polygon, etc.

Operationalizing Security Earlier in the SDLC

As software systems mature and economic risk increases, security consistently shifts earlier in the development lifecycle. Traditional software engineering adopted shift-left practices decades ago, embedding automated validation directly into development workflows. High-assurance industries such as aerospace followed a similar trajectory, prioritizing early verification to prevent costly downstream failures.

The 2025 exploit data indicates blockchain systems have reached a comparable inflection point. As value concentration grows, security controls applied only after development are no longer sufficient. Correctness must be evaluated where logic is written, not after it has been deployed.

Operationalizing shift-left security requires layered validation throughout the SDLC, including:

- **Static analysis** to identify logical flaws and unsafe patterns at development time
- **Unit testing** to validate expected behavior under known conditions
- **Mutation testing** to assess whether tests meaningfully enforce intended invariants
- **Fuzzing** to explore unexpected execution paths and adversarial inputs
- **Formal verification** to mathematically validate critical properties where required

Large language models can play a valuable role in surfacing certain classes of issues and accelerating developer workflows. However, due to their probabilistic nature, LLM-based tools cannot serve as a sole source of security assurance. In high-risk systems, security outcomes must be reproducible, measurable, and verifiable, not inferred from non-deterministic analysis alone.

As markets mature, security spend shifts left.

Deterministic vs Probabilistic Security: Why Proof Matters

As smart contract systems continue to secure increasing amounts of economic value, the standard for security assurance must evolve. Not all security approaches offer the same level of confidence.

Probabilistic security relies on likelihood. These approaches — including pattern detection, heuristics, and many AI-driven tools — surface issues based on probability rather than certainty. While effective for triage and signal generation, their outputs are inherently non-deterministic and difficult to reproduce.

Deterministic security, by contrast, seeks provable correctness. It evaluates whether defined properties, invariants, and constraints hold across all possible execution paths. Outcomes are reproducible, measurable, and verifiable — qualities required when systems carry material financial and reputational risk.

This distinction has become increasingly important as artificial intelligence is applied to security workflows. In a recent open letter, JPMorgan Chase's CISO cautioned against relying on probabilistic AI systems as a primary security control, emphasizing that non-deterministic outputs cannot provide the level of assurance required for high-risk environments.

The implication is not that AI lacks value, but that its role must be appropriately scoped. AI can accelerate discovery, expand surface coverage, and improve developer efficiency. However, it cannot substitute mathematical proof where correctness is required.

As blockchain systems mature, security assurance must shift from confidence inferred to correctness demonstrated. **In environments where failure carries irreversible financial consequences, proof is not a preference; it is a prerequisite.**

ABOUT OLYMPIX

The findings in this report highlight a consistent pattern: **the majority of 2025 smart contract exploits were not caused by unknown vulnerabilities, but by logic flaws and untested assumptions introduced earlier in development**.

Based on analysis of in-scope incidents, approximately 98% of exploits — representing roughly $240 million in losses — could have been prevented had vulnerabilities been identified prior to deployment.

Olympix was built to address this structural gap through a **proactive, automated, and verifiable security** model. Rather than relying on retrospective detection, the platform applies shift-left security principles to evaluate correctness as code is written — when risk can still be removed, not merely monitored.

This approach reflects practices long established in mature industries such as traditional software and aerospace, where early, deterministic validation is used to prevent high-impact failures downstream. By introducing reproducible and measurable security assurance earlier in the SDLC, Olympix supports a security posture centered on prevention, proof, and operational scalability.

Additional information is available at **www.olympix.security**

Trusted by:

catalysis CORK SkyMavis STAKE DAO Hourglass

Covenant Strata SYNDICATE LI FI AGORA

crossmint CIRCLE LUMIA Fire Dev EIGENPIE

UNISWAP UNISWAP FOUNDATION IPOR LUCID

Yield.xyz CLANKER

Olympix