



QuillAudits  
Web3 Security

# 2023

---

# Annual Web3 Security Report



# TABLE OF CONTENTS

Summary	2
State of Web3 Security: A 2023 Perspective	3
2023 Security Statistics	5
Introduction	8
Major Security Breaches & Observed Hack Trends in 2023	10
Audit Corner: Auditor's Talk on Smart Contract Security & Threat Prevention	34
Recent Developments in Web3 Security	50
Predictions of Web3 Security Landscape in the Coming Years	55
Technical Insights into Secure Smart Contract Coding And Risk Mitigation Steps	59
Web3 Development Resource Hub	69
DeFi Economic Risks	75
QuillAudits' 2024 Security Outlook	85

# SUMMARY

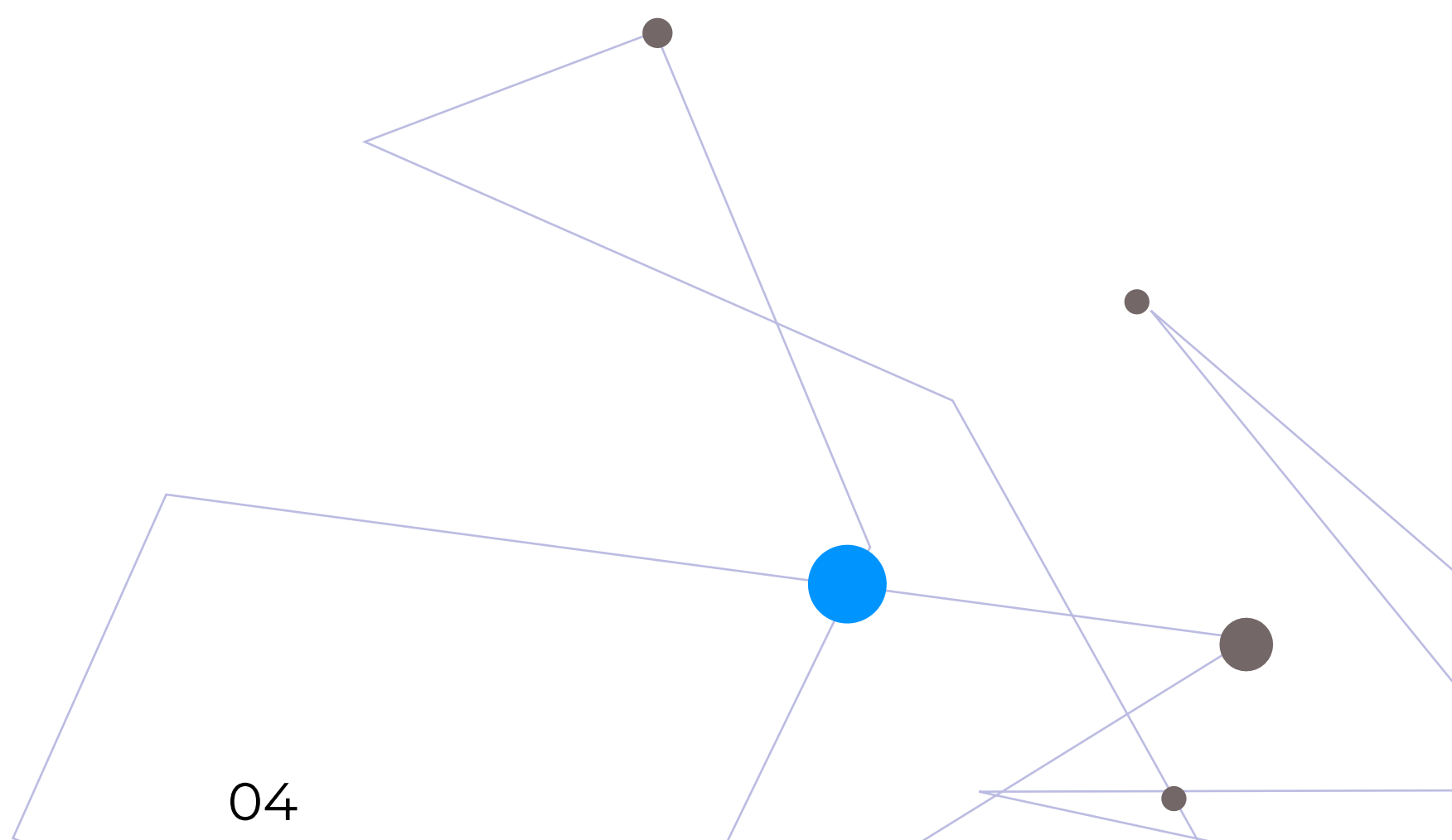
- QuillMonitor, a Web3 vulnerability analytics tool by QuillAudits, has extensively tracked over 1000+ hacks, totalling losses of \$7.5 billion since 2020.
- This report presents a detailed analysis of attacks on various crypto protocols and instances of suspected rug pulls in 2023, as meticulously recorded by QuillMonitor.
- 2023 witnessed a significant total loss of \$1.7 billion across the web3 ecosystem, notably impacted by substantial losses incurred by Mixin Kernel (\$200M) and Euler Finance (\$196M).
- Incident numbers surged to 440 in 2023, surpassing the 376 recorded hacks in 2022, marking an alarming increase in cyber threats and the need for robust security solutions in the crypto & blockchain realm.
- Rug pulls emerged as the primary cause, contributing to the majority of losses, accounting for \$352.27 million across 137 incidents.
- Smart contract vulnerabilities ranked as the second most damaging attack type, resulting in a substantial loss of \$204.55 million.
- The Web3 Security Report 2023 provides comprehensive coverage of major security incidents, Web3's 2023 security landscape, and future predictions, shedding light on the evolving trends shaping the Web3 domain.

# State Of Web3 Security: A 2023 Perspective

From regulatory challenges to the lack of user education, the Web3 security landscape of 2023 is marked by various aspects, such as:

- 1. High-profile Fraud and Financial Misconduct** within the crypto realm, exemplified by the FTX collapse and penalties imposed on Binance, underscored the unregulated nature of the growing industry. As robust regulations are lacking, users must remain cautious while venturing into this domain.
- 2. Emergence of New Attack Vectors:** Unprecedented attack vectors involving compilers and third-party libraries have emerged. These vectors, previously known in traditional security areas, have now been exploited within Web3 environments. For instance, vulnerabilities in the Vyper compiler and the popular ThirdWeb library have affected multiple protocols, revealing a new dimension of vulnerability in Web3.
- 3. Lack of User Awareness and Education** surfaced prominently through instances of Ponzi schemes and rug pulls. These incidents indicated investors' inadequate understanding of blockchain protocols and due diligence processes.

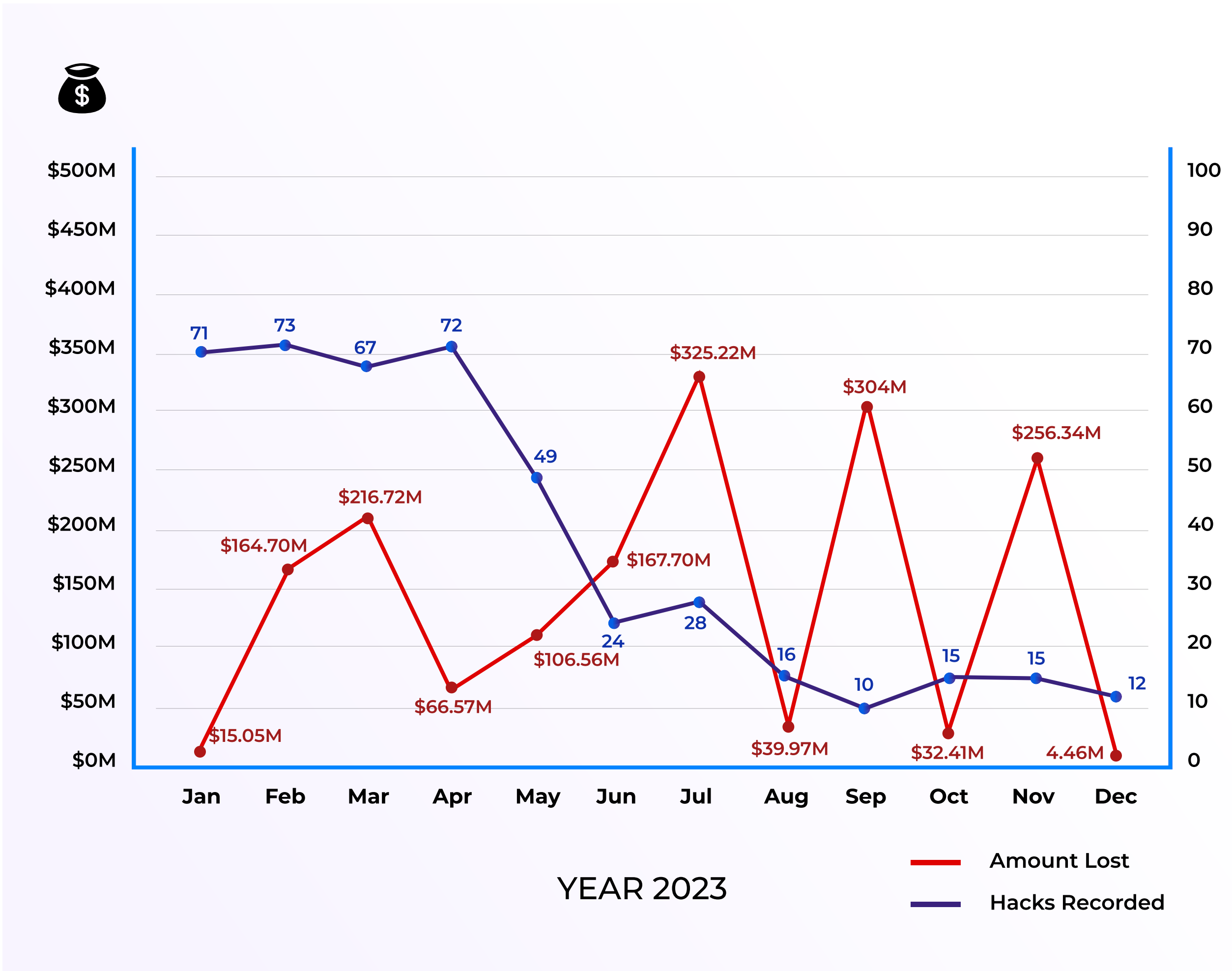
4. **Targeted Attacks on Established Protocols:** Even high-security protocols like Curve and Balancer faced attacks despite undergoing extensive security code audits. This highlights an alarming shift where even well-secured systems are susceptible to breaches.
5. **The rise of Decentralized Finance (DeFi) products in 2023** brought a new wave of accessible financial services built on Web3 capabilities. However, this also exposed persistent security weaknesses, shaking the users' trust in Web3.
6. **Exploits in Complex Web3 mechanisms,** such as bridge exploits, indicate the security weaknesses existential at the protocol level. Taking up thorough and multiple audits of smart contracts is a vital strategy for navigating threats of this kind.
7. **Revival of Old Vulnerabilities with New Impacts:** Previously known vulnerabilities, like precision loss, have resurfaced with unforeseen consequences. Balancer and KyberSwap experienced exploitation due to precision loss, demonstrating the evolving nature of threats and the challenge of evaluating the impacts of common bugs in Web3.





# 1 | 2023 Security Statistics

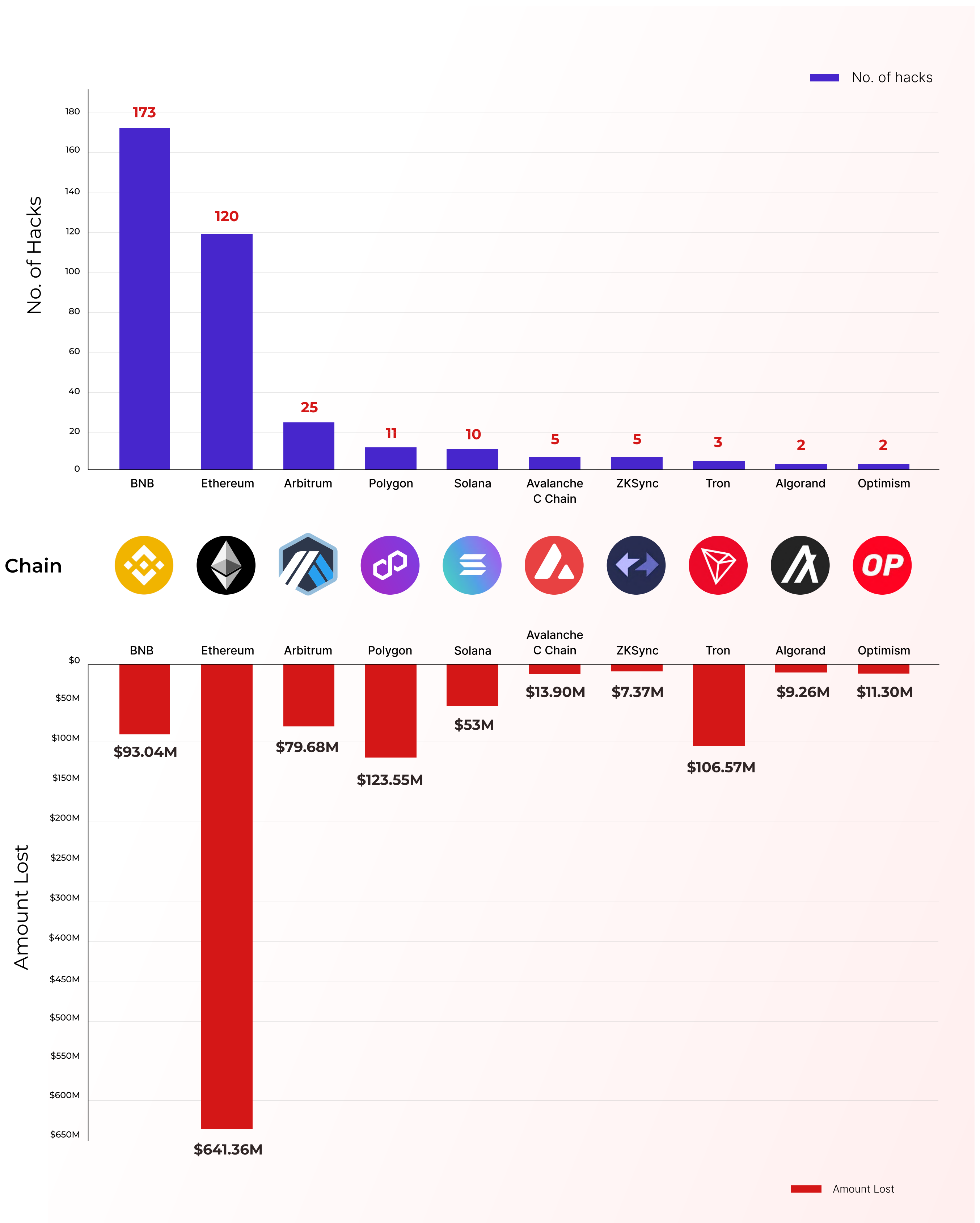
# 1. Incident Impact By Month



# 2. Yearly Losses

Year	Hacks Recorded	Amount lost
2021	67	\$2.25B
2022	376	\$3.32B
2023	434	\$1.70B

# 3. Chain-specific Security Breaches





# Introduction



***"Web3 is all about decentralization and trustlessness. Security is not just a feature; it's the foundation of this new internet. We must prioritize security to ensure the success of Web3."***

**~ Vitalik Buterin, Co-Founder of Ethereum**



2022 hit the crypto world so hard—hacks, scams, collapses—it felt like a relentless rollercoaster of losses. From Web3 breaches to FTX's sudden downfall, the industry felt the tremors.

But 2023 brings a different vibe—a glimmer of hope amidst the chaos.

This year, losses dropped to \$1.7 billion, almost half of the \$3.32 billion lost in 2022. Although it's a sigh of relief, the surge in hacks paints a contrasting picture. Challenges persist, with 440 hacks in 2023, surpassing the 376 recorded hacks in 2022.

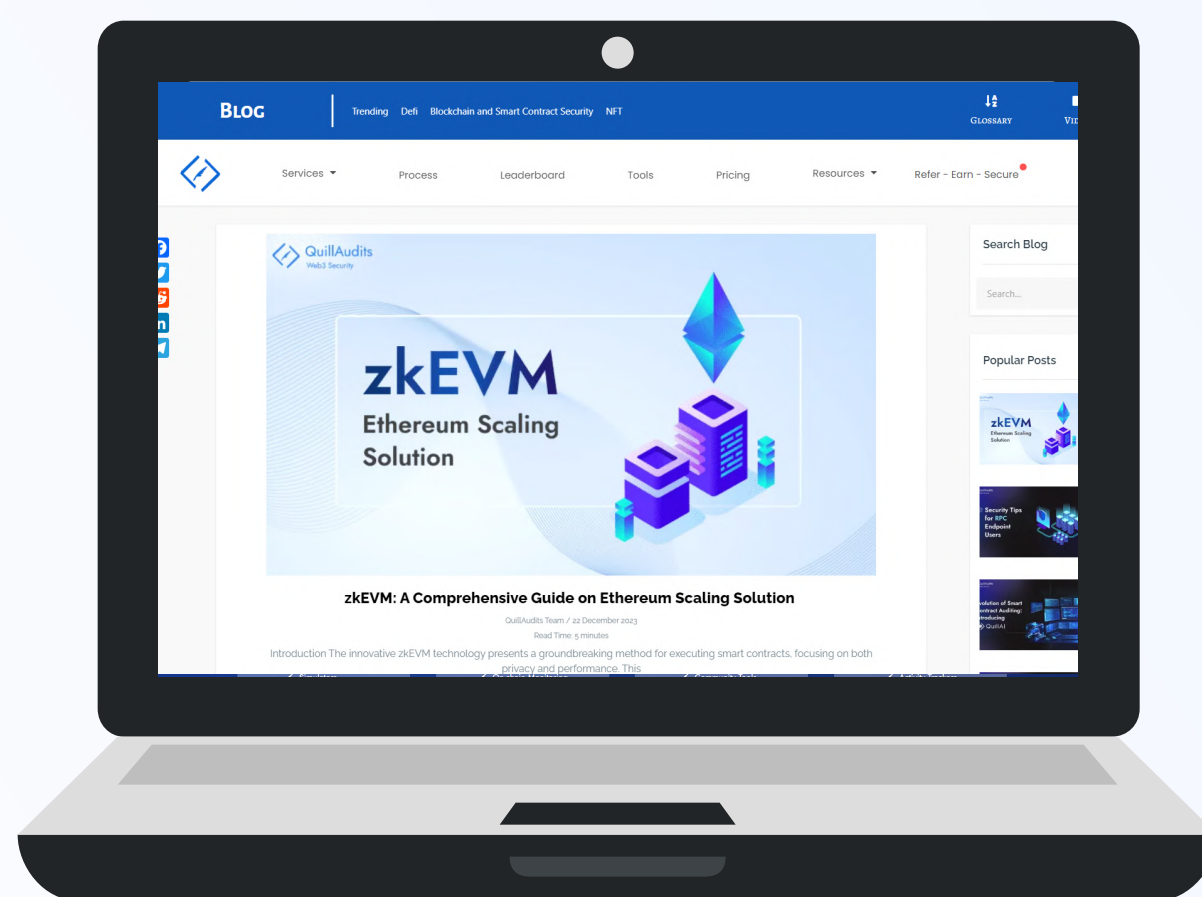
With the total market cap of the Web3 industry skyrocketing from \$927 billion in 2022 to \$1.7 trillion in 2023, the demand for robust solutions intensifies more than ever.

As the stakes rise, so does our commitment to building a more secure Web3 ecosystem. QuillMonitor, a Web3 data analytics tool, assesses the volume of crypto funds lost, painting a vivid picture of the crypto community's losses to hacks and scams in 2023.

Welcome to our Web3 Security Report 2023—a journey through the year's biggest security breakouts, the tales and trends shaping the web3 course, expert insights dissecting the nuances of smart contract security, and a backstage pass to the ever-evolving security landscape where you'll find the mitigation strategies and recommendations to combat the evolving threats.

Also, prepare for an exhilarating ride into exploring the latest advancements and predictions for Web3 security in the coming years. We'll also delve into QuillAudits' unique security approach and how we're crafting a resilient Web3 ecosystem through our cutting-edge product development and initiatives.

## Follow Our Blogs to Stay Current on the Web3 Security at Large!



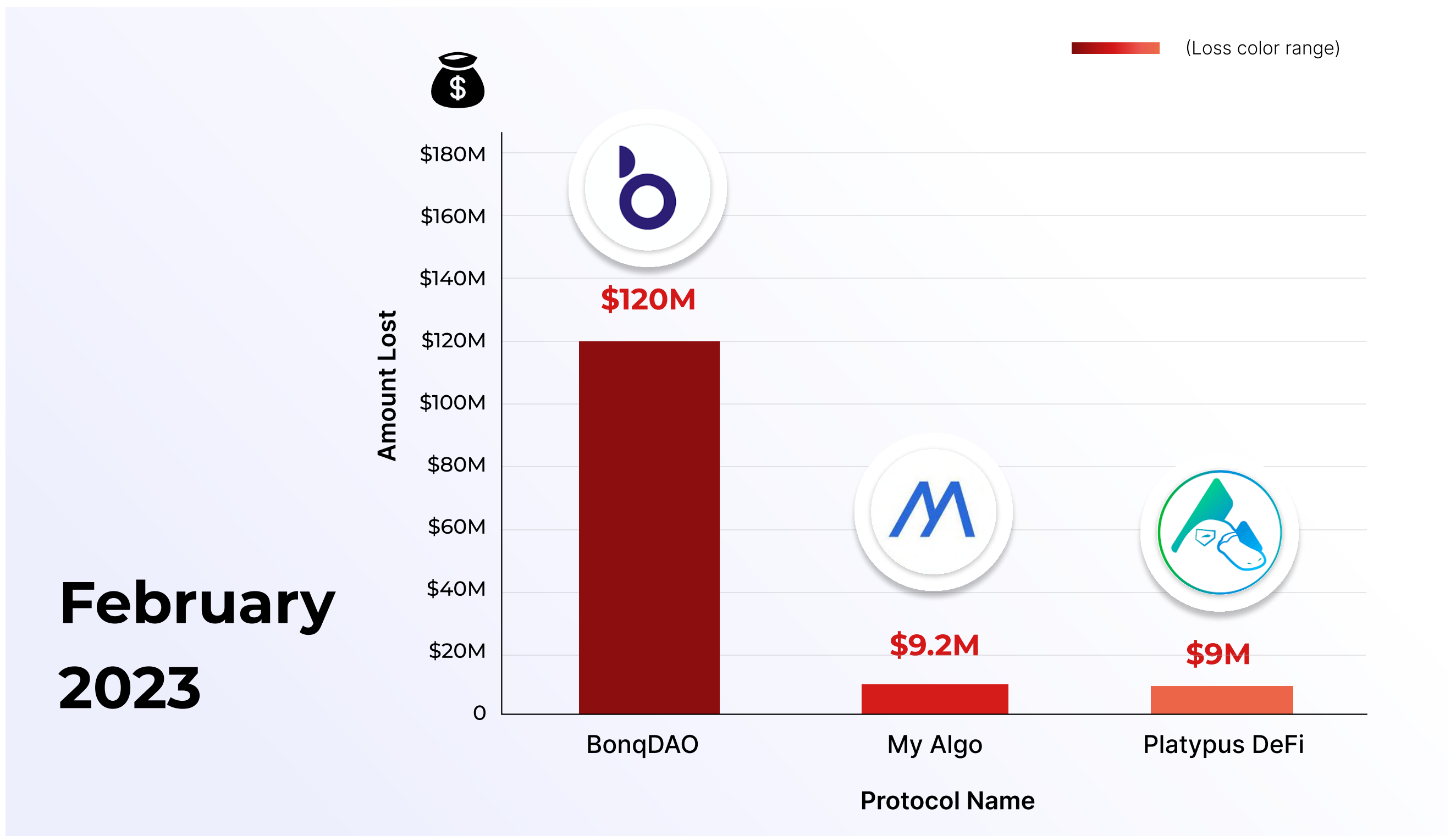
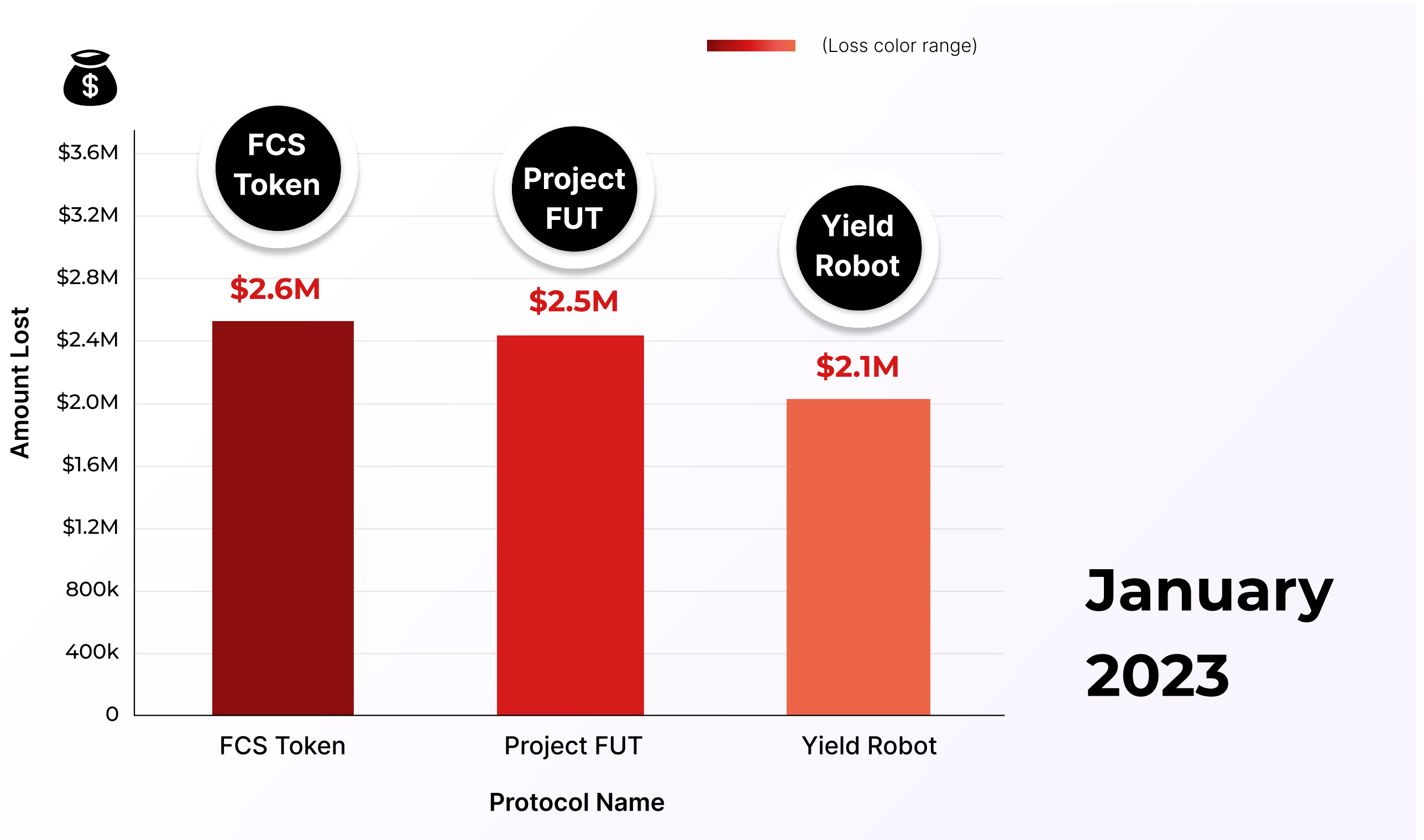
 Visit Our Blog



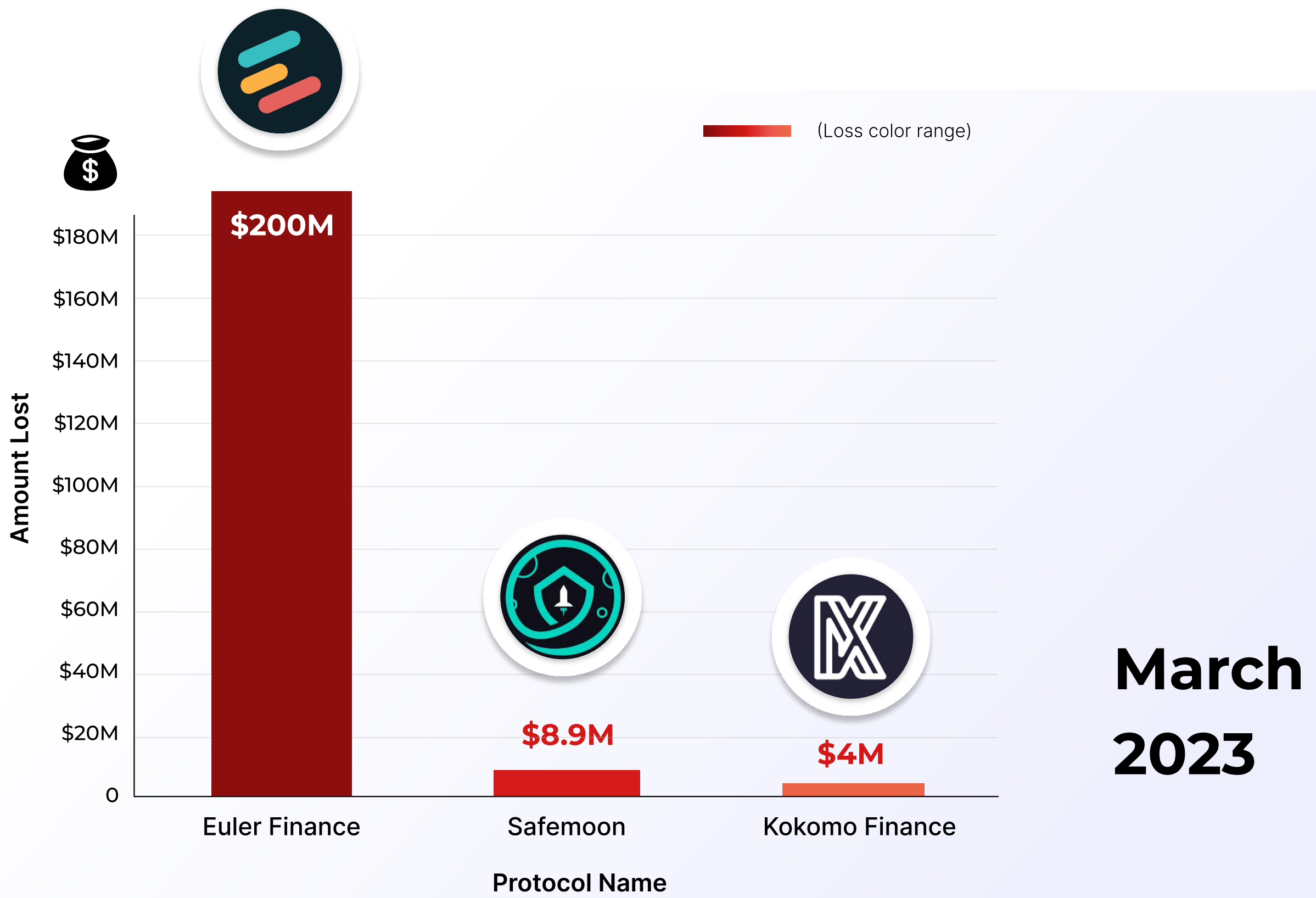
**2** |

**Major Security  
Breaches &  
Observed  
Attack Trends  
In 2023**

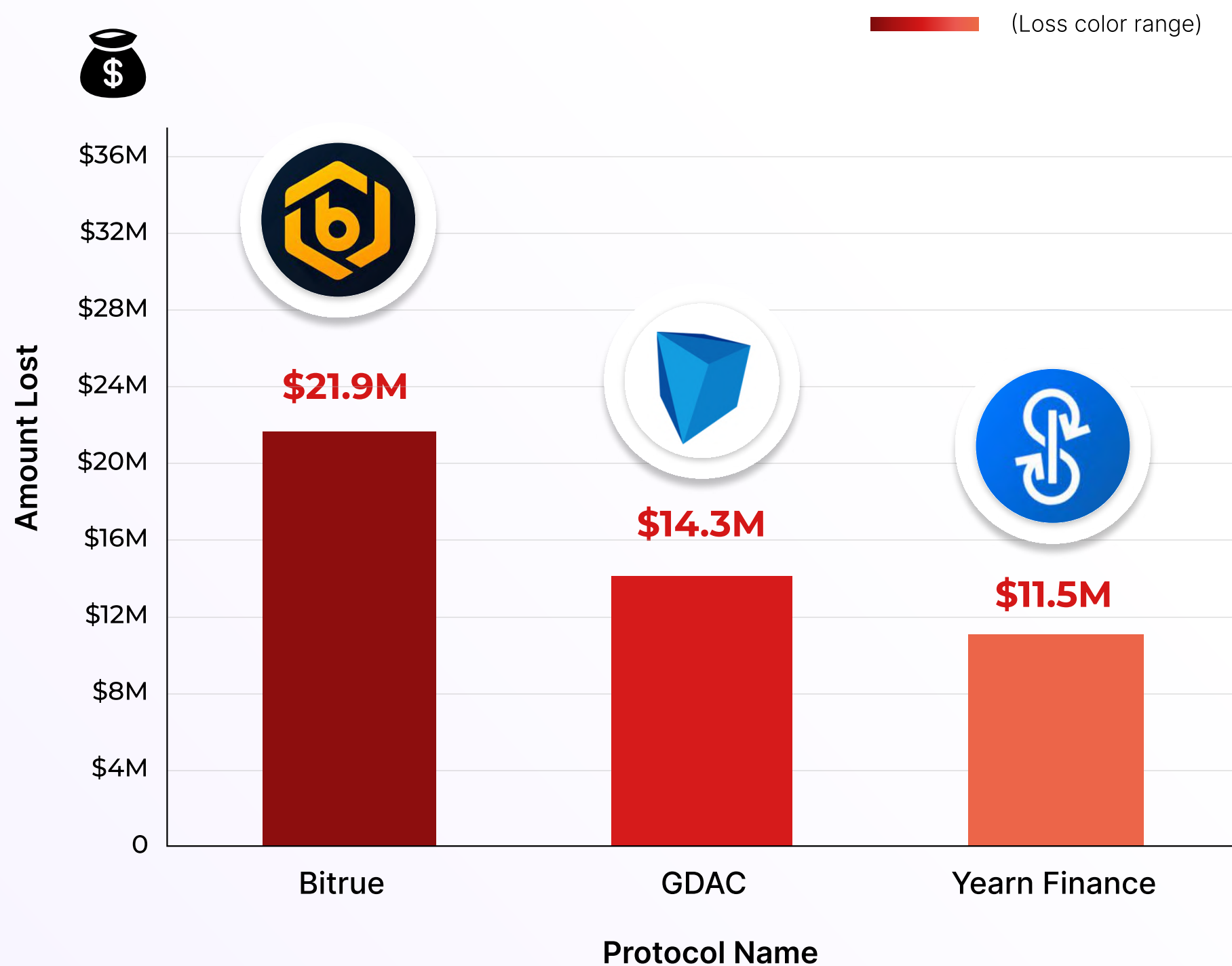
# Top Hacks Of The Month



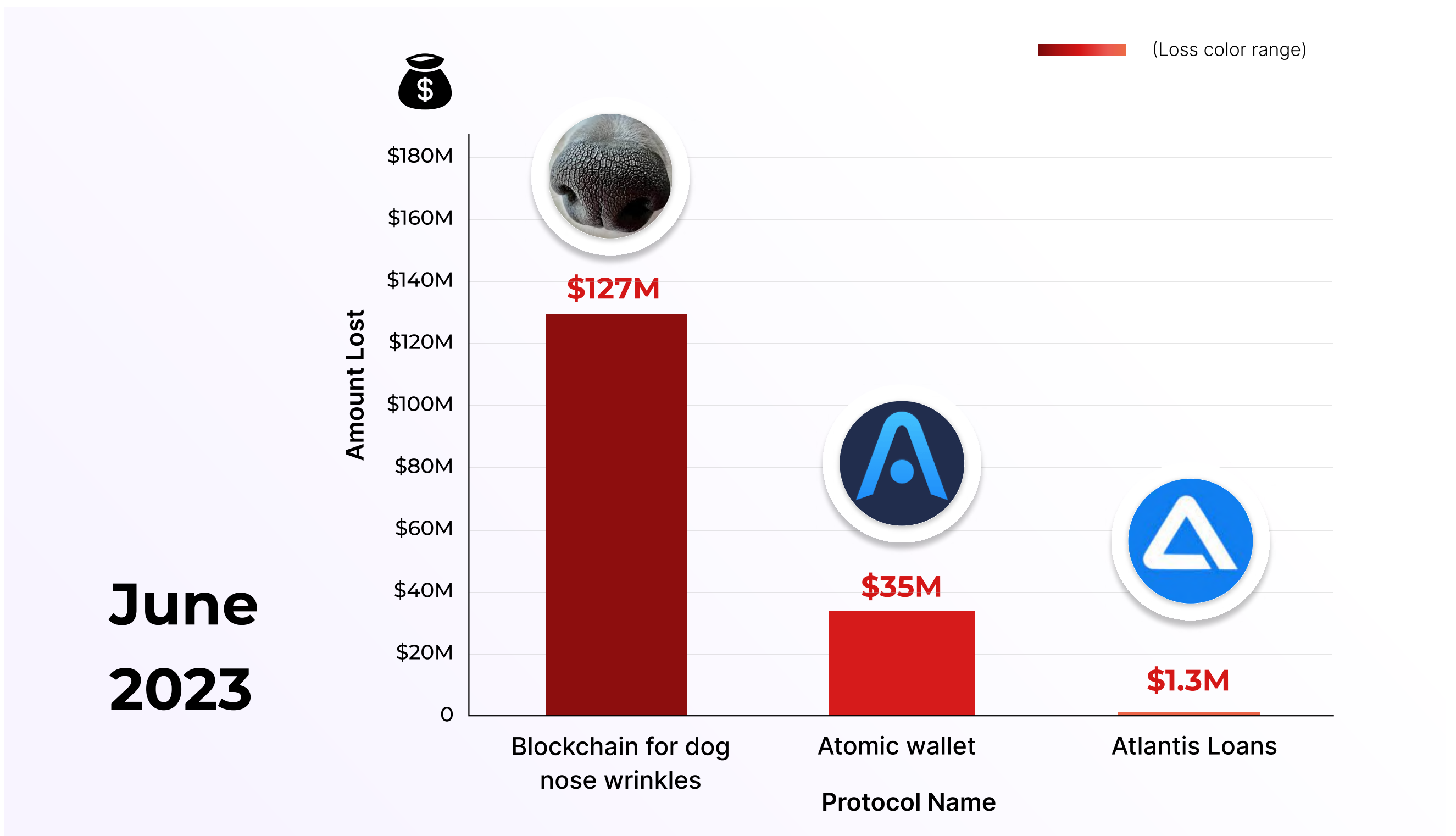
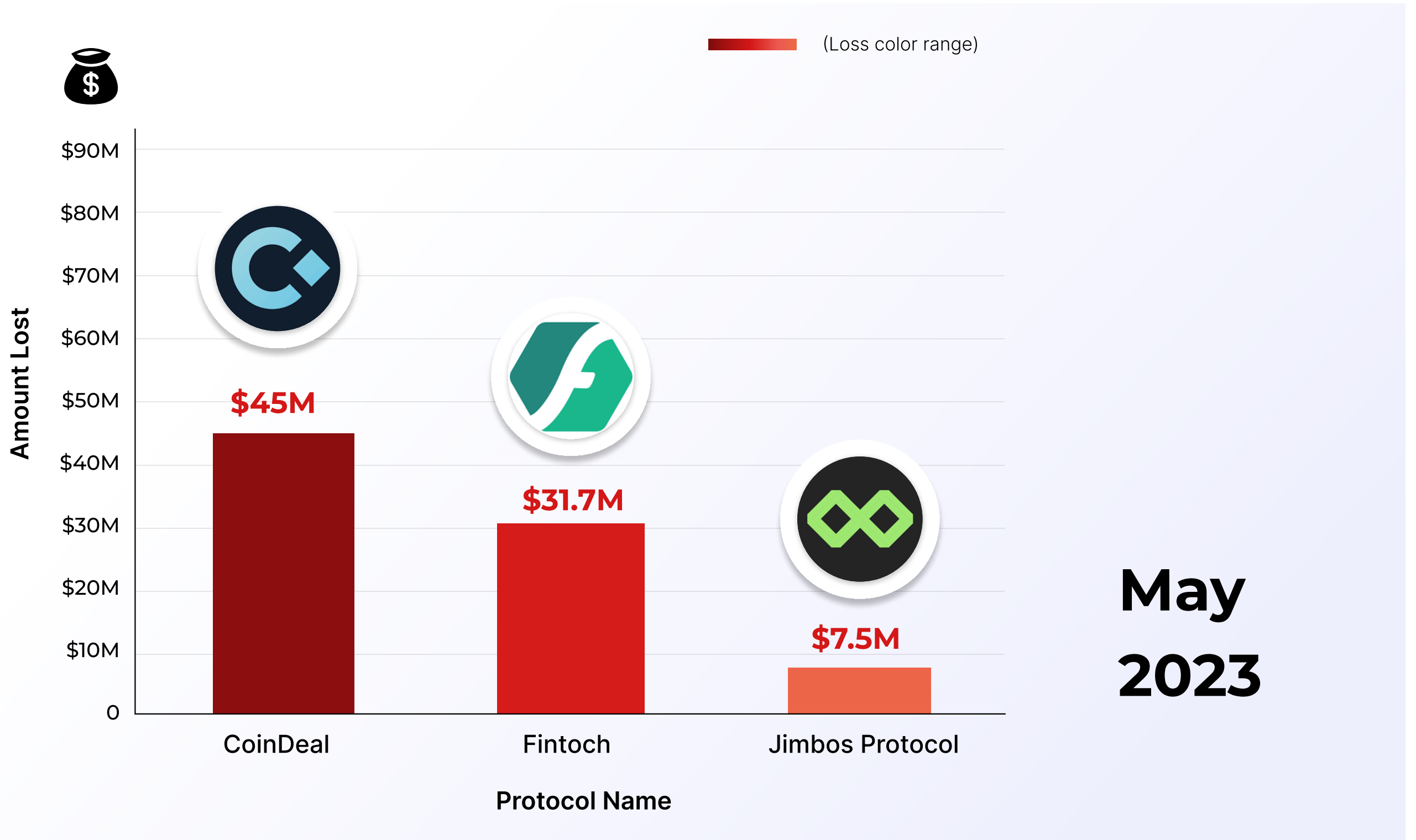
# Top Hacks Of The Month



**April  
2023**

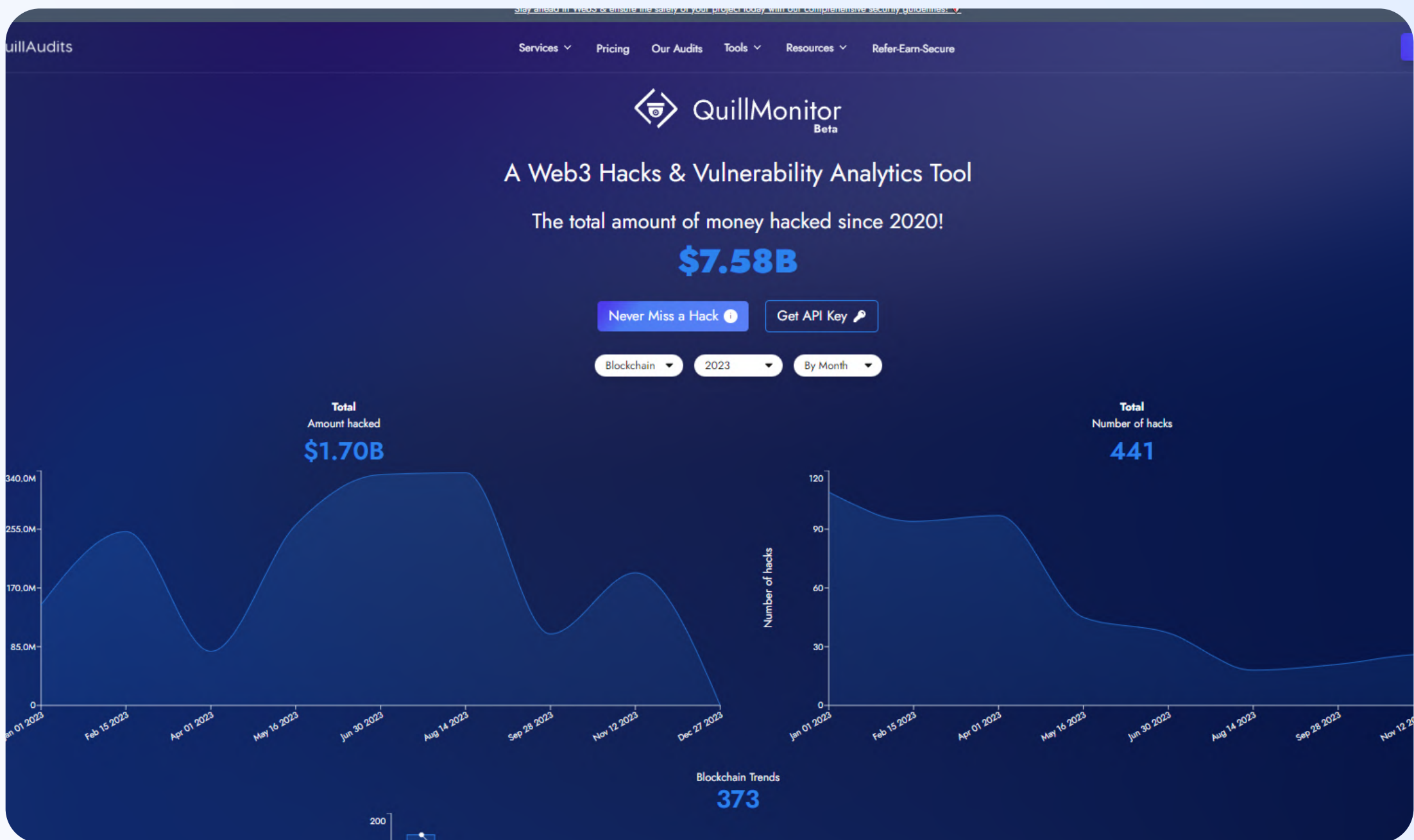


# Top Hacks Of The Month



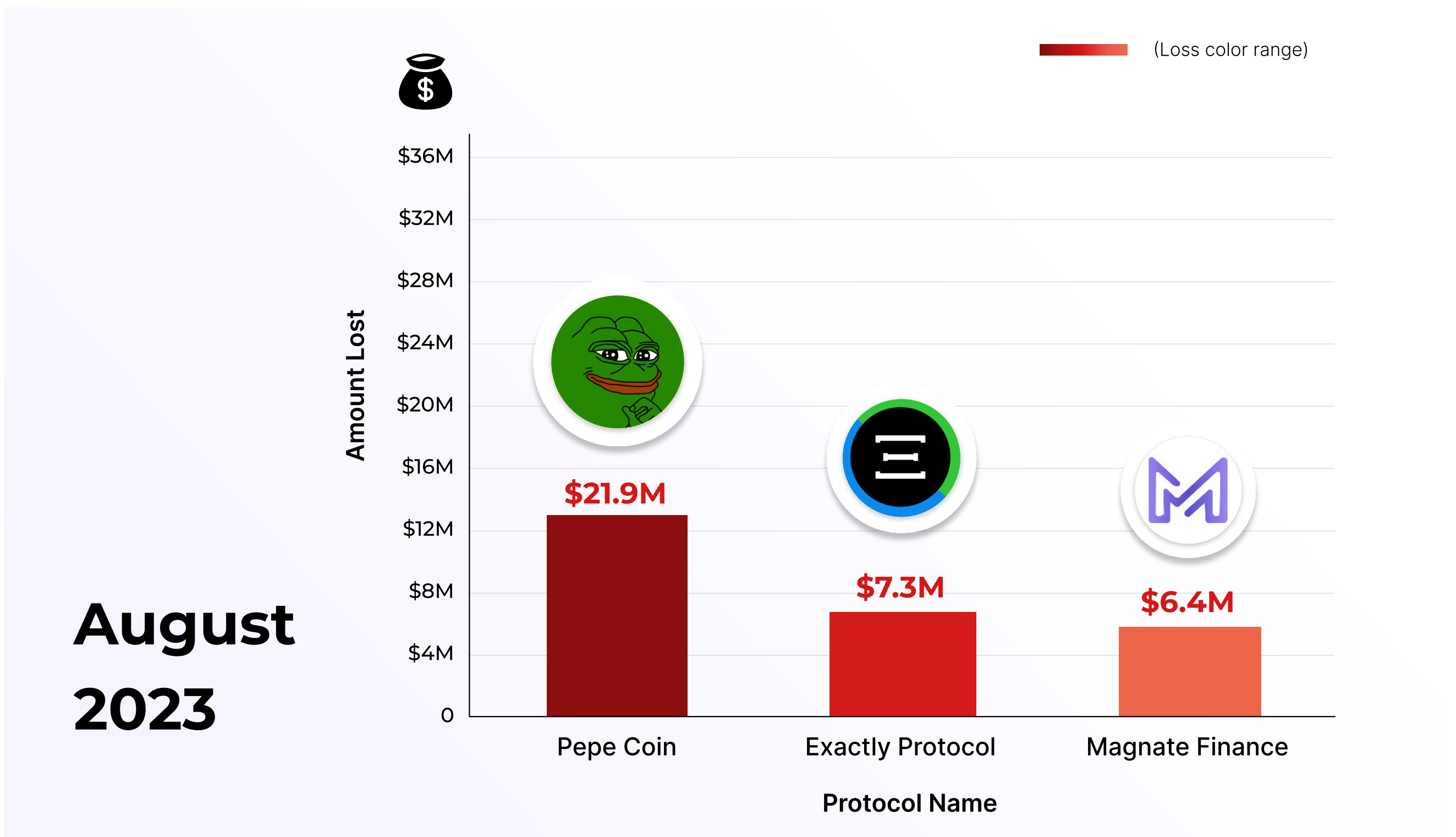
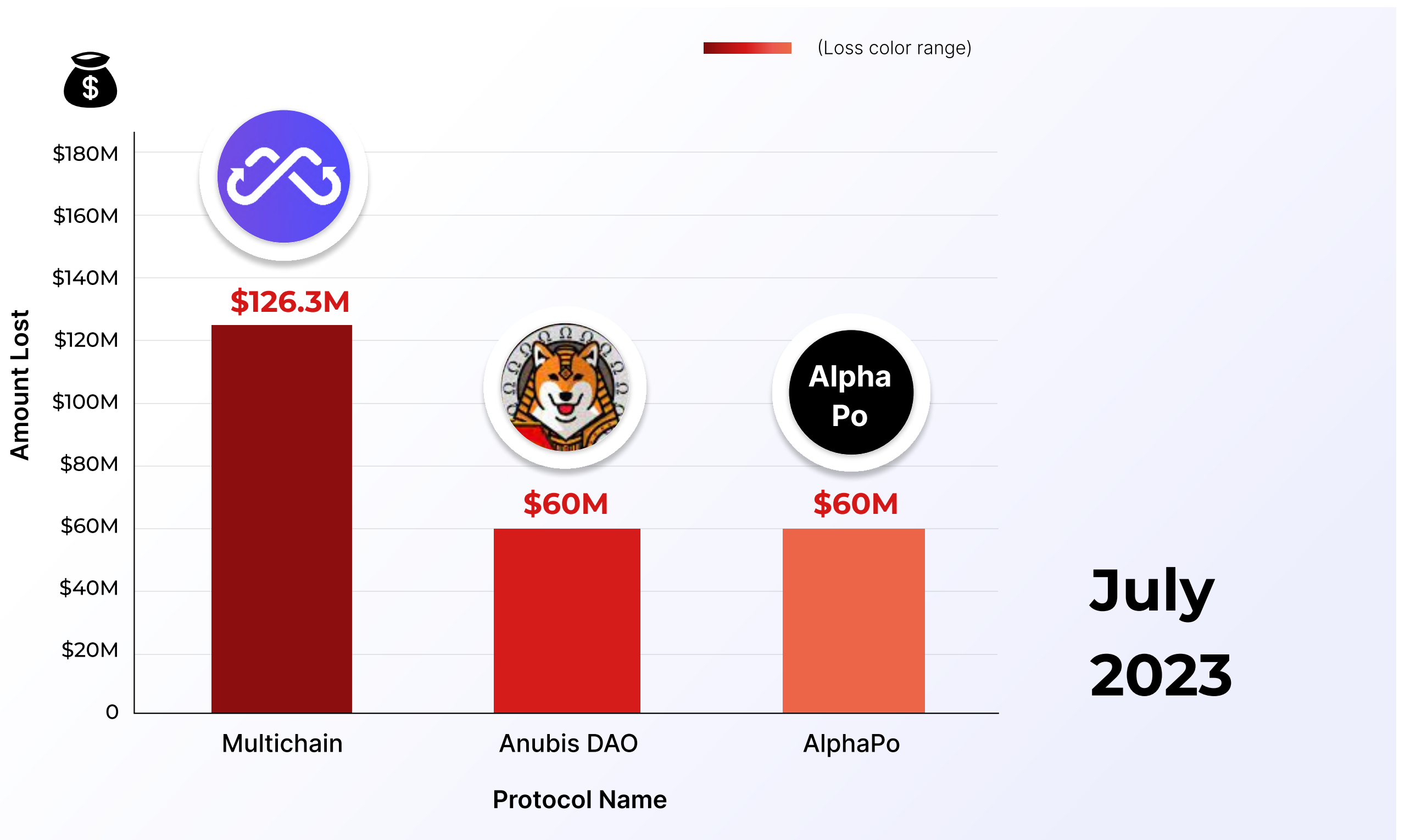


# QuillMonitor



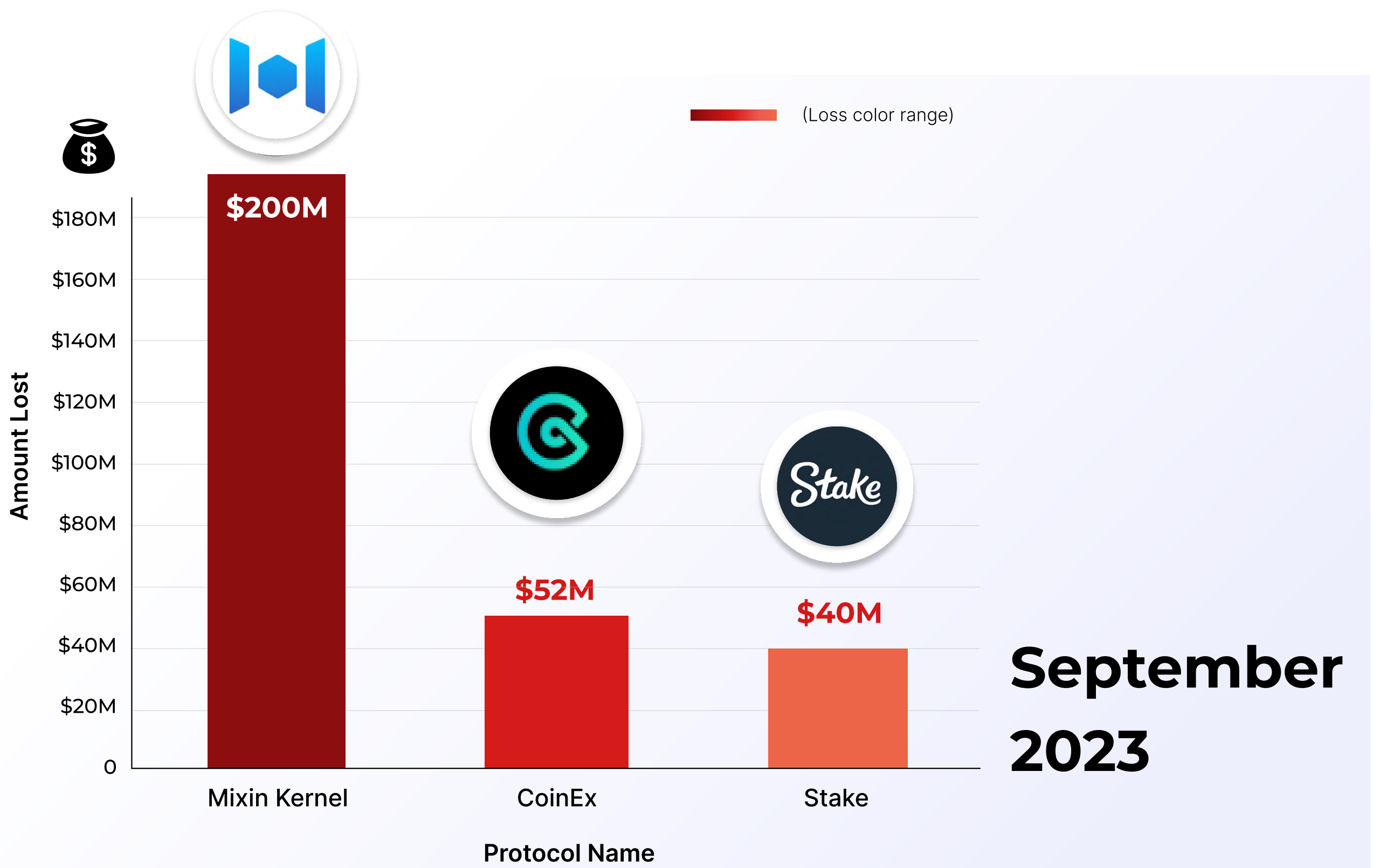
Monitor Web3 Attacks in Real Time

# Top Hacks Of The Month

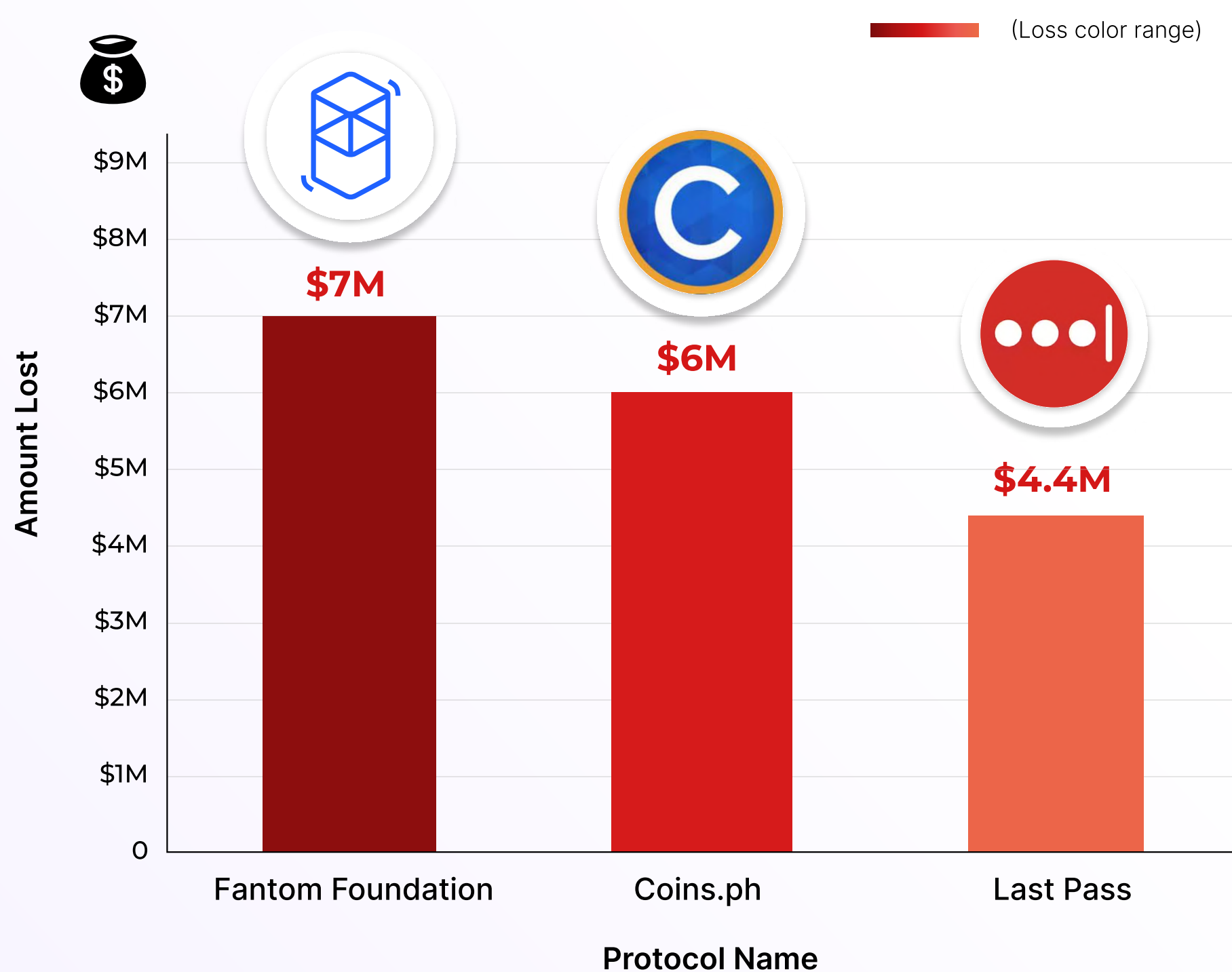




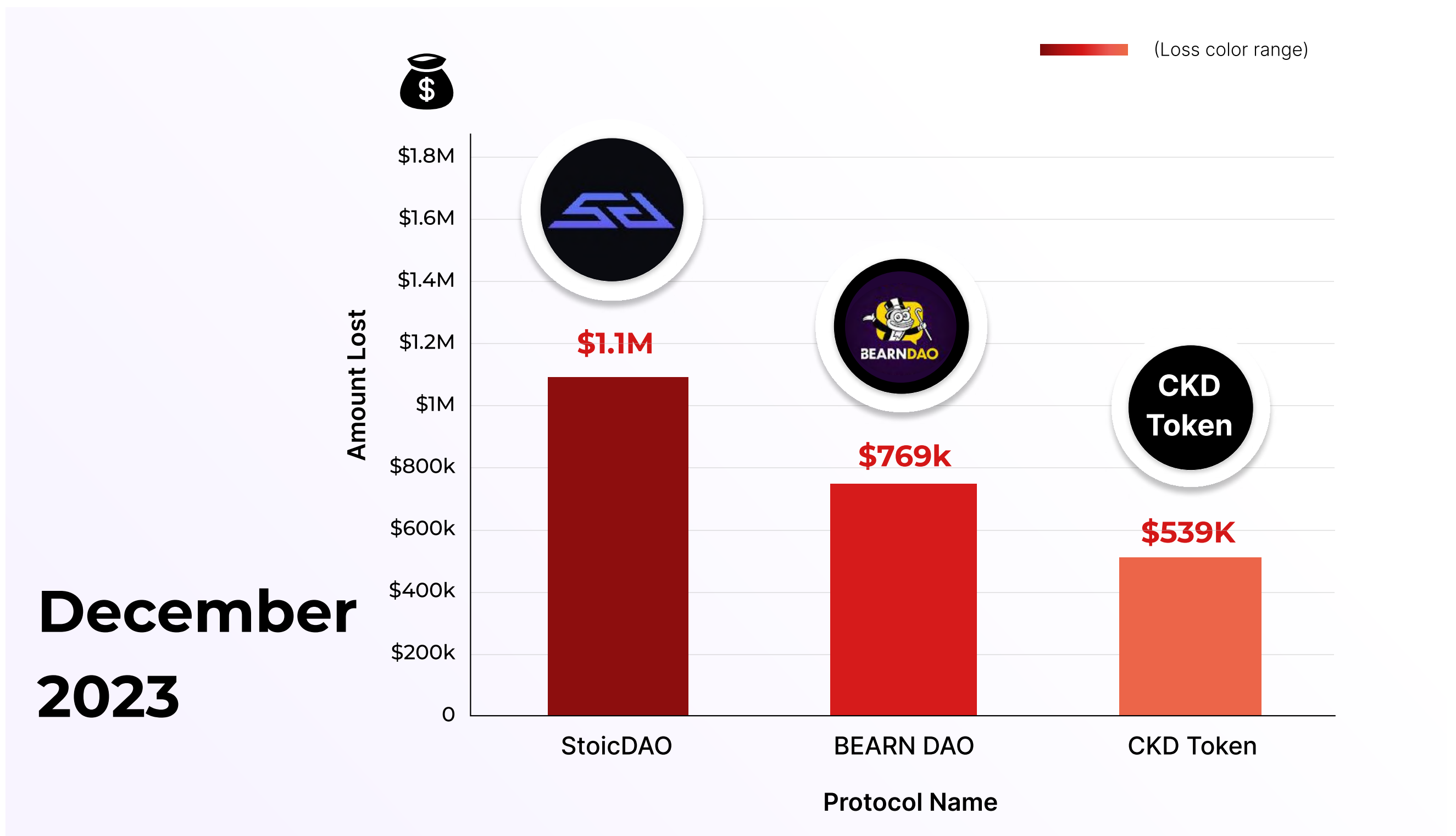
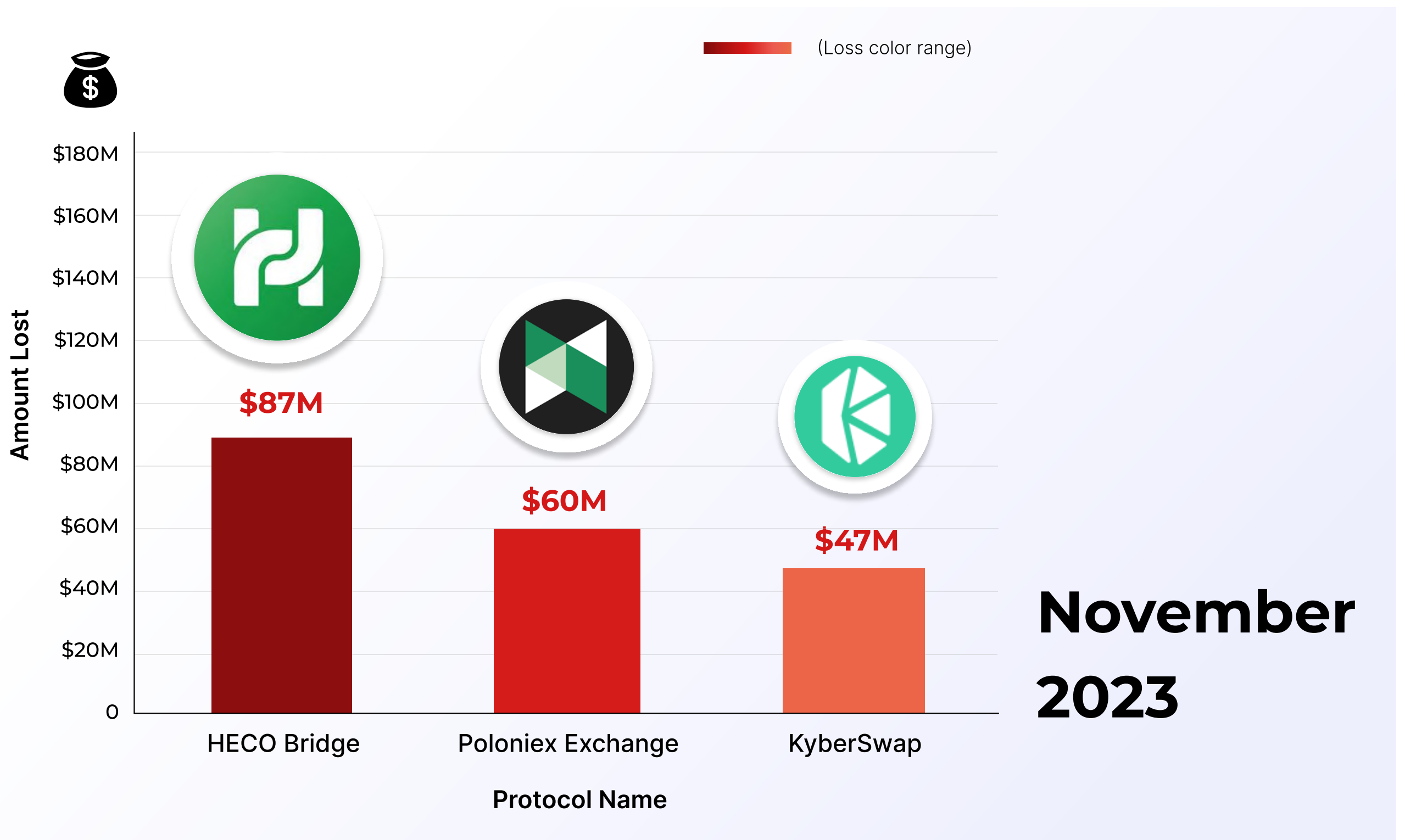
# Top Hacks Of The Month



**October  
2023**



# Top Hacks Of The Month



# Top 10 Security Incident Highlights

1



Mixin Kernel

**\$200M**

**Attack Type:**

**Cloud Service Database Compromise**

## Hack Analysis

The Mixin Network faced a staggering \$200M loss due to a sophisticated attack. It appears the breach occurred through the compromise of Mixin's cloud service provider's database. Hackers likely gained access to users' private keys stored on the cloud, allowing them to conduct seemingly straightforward transfers of assets, including ETH, USDT (later converted to DAI), and BTC. There are suspicions that even Mixin's hot wallets might have been targeted.

QuillAudits recommends the following defense strategies to combat the future occurrences of such security threats:

- ✍ Platforms like Mixin can prioritize decentralized storage solutions for sensitive data.
- ✍ Additionally, can employ robust encryption techniques and multi-layer authentication for accessing critical data
- ✍ Regular security audits and constant vigilance against potential vulnerabilities within third-party services are also essential to avoid such massive losses.



# Top 10 Security Incident Highlights

2



Euler Finance

**\$196M**

**Attack Type:**

**Flash loan Exploit**

## Hack Analysis

The \$196M attack on Euler Finance, a renowned DeFi lending protocol, involved an exploit linked to the donateToReserves function introduced via EIP14. This function lacked a check on user position health, allowing manipulation.

The attacker used two contracts—one to create unbacked debt via donateToReserves and another as a liquidator. Leveraging flash loans and Euler's leverage system, the hacker created a favourable situation, obtained inflated eToken collateral at a discount, and withdrew assets.

QuillAudits recommends the following defense strategies to combat the future occurrences of such security threats:

- ✍ Conduct thorough function checks.
- ✍ Perform meticulous audits, especially for less-used functionalities
- ✍ Employ ongoing monitoring systems to detect irregularities and unusual transactions, especially during contract interactions and token swaps.



# Top 10 Security Incident Highlights

3



## Blockchain for dog nose wrinkles **\$127M**

**Attack Type:**  
**Ponzi Scheme**

### Hack Analysis

A South Korean company's "Dog Nose Wrinkles" scam lured investors with promises of extraordinary returns through a blockchain app supposedly identifying dogs via their nasal folds. Promising up to 150% returns in 100 days, it amassed over \$100M in investments.

However, the app was a complete scam, causing massive financial losses for investors. This fraudulent scheme preyed on people's interest in blockchain technology and their willingness to invest in novel concepts.

QuillAudits recommends the following defense strategies to combat the future occurrences of such security threats:

- Potential investors should conduct thorough due diligence into any blockchain project or investment opportunity.
- Scrutinize the technology behind the protocol along with the background research about the team's credibility.



# Top 10 Security Incident Highlights

4



**Multichain**

**\$126.3M**

**Attack Type:**

**Multichain bridge exploit**

## Hack Analysis

The drain of Multichain addresses led to a loss of \$126M, impacting approximately 50% of the FTM bridge and 80% of Moonriver bridge holdings. The attack history of the project, previously known as Anyswap, added to the gravity of this incident.

Earlier breaches and vulnerabilities hint at persistent security weaknesses, including a significant \$8M hack pre-rebranding and vulnerabilities in multi-token contracts, causing \$3M in losses in early 2022.

In this instance, the assets locked in the Multichain MPC address were illicitly transferred to an unknown destination, triggering the alarm. The root cause remains elusive, prompting concerns about potential insider actions or vulnerabilities exploited by an external attacker.

To avert such exploits, prompt response protocols and transparent communication during crises can help mitigate the fallout and regain user trust.

# Top 10 Security Incident Highlights

5



**BonqDAO**

**\$120M**

**Attack Type:**

**Oracle Manipulation**

## Hack Analysis

The attacker managed to exploit vulnerabilities in the Tellor price feed, manipulating the value of (wrapped) WALBT collateral and resulting in losses of up to \$120M. By staking a small amount of TRB tokens worth around \$175, they inflated the WALBT price drastically. This manipulation allowed them to mint 100M BEUR stablecoins against meagre collateral.

In a subsequent transaction, the attacker reset the WALBT price to an extremely low value, liquidating approximately 113M WALBT tokens and causing massive losses.

The exploit took advantage of flaws in the Bonq Protocol's collateral valuation mechanism, allowing borrowing against falsely inflated collateral values within a single transaction.

QuillAudits recommends setting thresholds or limits on price fluctuations for collateral valuation to mitigate such risks associated with sudden price manipulations.

# Top 10 Security Incident Highlights

6



**HECO Bridge**

**\$87M**

**Attack Type:**

**Private Key Compromise**

## Hack Analysis

HECO Chain's Ethereum bridge suffered a loss of \$86.6M, primarily comprising assets like USDT, ETH, HBTC, SHIB, UNI, USDC, LINK, and TUSD. This breach occurred via a compromised operator account, which illicitly withdrew funds and transferred the majority to an accumulation wallet.

Simultaneously, HTX lost \$12.5M from its hot wallets, involving assets like ETH, USDT, USDC, and LINK. The breach involved direct transfers from compromised hot wallet addresses to separate attacker addresses.

It's likely that vulnerabilities exploited in prior hacks, such as the Poloniex exchange, were reused, given the interconnections between the affected entities.



# Top 10 Security Incident Highlights

7



**Poloniex Exchange**

**\$60M**

**Attack Type:**

**Hot Wallet Compromise**

## Hack Analysis

The Poloniex exchange faced a staggering loss of \$60M from its hot wallets. The incident began when 4900 ETH (approximately \$10M) was siphoned from one of the Poloniex addresses on Etherscan.

The attack expanded across Ethereum, TRON, and BTC chains, resulting in substantial losses across these platforms. The attacker utilized various addresses to disperse stolen tokens, converting some to ETH and dispersing others to new addresses.

This attack seemingly involved phishing or off-chain methods aimed at compromising devices or employees to gain access to private keys.

# Top 10 Security Incident Highlights

8



**Anubis DAO**

**\$60M**

**Attack Type:**

**DeFi Rug Pull**

## Hack Analysis

AnubisDAO promoted as a derivative of OlympusDAO, lured investors by leveraging canine-themed imagery akin to successful meme coins like Dogecoin. Despite lacking a website, the project attracted \$60M in ETH during its token sale.

However, within twenty hours, the project's liquidity vanished into an alternate address, leaving investors distraught. The scheme involved diverting liquidity from the pool to a separate wallet.

To avert such security breaches, it is advisable to use tools like QuillCheck to validate project legitimacy by running through risk ratings along with detailed reports on the project analysis.

# Top 10 Security Incident Highlights

9



AlphaPo

**\$60M**

**Attack Type:**

**Smart Contract Vulnerability**

## Hack Analysis

The AlphaPo hack resulted in a loss of \$60M across Ethereum (ETH), TRON, and Bitcoin (BTC). The breach began by draining AlphaPo's hot wallet, with 2464 ETH (\$4.6M) and various other coins, including over 6M USDT, siphoned off to the attacker's address. The stolen funds on TRON were redirected to centralized exchanges.

The pattern of transactions post-hack strongly aligns with Lazarus, a state-sponsored cybercriminal group known for leaving distinct fingerprints on-chain. The attack's execution involved transferring, swapping, consolidating into secondary accounts, and dispersing the stolen funds.

# Top 10 Security Incident Highlights

10



CoinEx

**\$52M**

**Attack Type:**

**Other Vulnerabilities**

## Hack Analysis

The CoinEx platform encountered a substantial breach, resulting in the loss of \$52M across various cryptocurrencies. The attack began with a transfer of 4950 ETH, worth approximately \$8M. Subsequently, multiple wallets were systematically drained of ETH, TRON, MATIC, XRP, BTC, SOL, XDAG, KDA, ARB, XLM, BCH, MATIC, and OP coins.

Attackers rapidly converted these assets back to native forms and transferred them to new addresses. CoinEx hasn't disclosed the breach's specifics but promised a forthcoming detailed report.



# QuillShield

Our Hackerboard proudly featured on Product Hunt.

QuillAudits Services Pricing Tools Resources Leaderboard Refer-Earn-Secure [Request An Audit](#)

QuillShield

## All-in-one Web3 Security Infrastructure

Enabling enterprises, developers, and communities to monitor and detect threats from development to post deployment.

Enter your business email [Get early access](#)

Launching Soon

COMING FOR YOU

### What you can look forward to



**Audits & Scans**

- ✓ Simulators
- ✓ Optimisers
- ✓ Automated Audits



**Threat Intelligence**

- ✓ On-chain Monitoring
- ✓ Event Alerts
- ✓ Forensics



**Fraud Protection**

- ✓ Community Tools
- ✓ DeFi Security Tools
- ✓ NFT Security



**User Analytics**

- ✓ Activity Trackers
- ✓ Visual Reports
- ✓ Wallet Profiler

**Join the Waitlist**

We are working around the clock to bring our services to the Web3.0 community. If you wish to be a part of our step towards a more secure web3.0 space, drop your email with us and we'll let you know when we are live!

Enter your business email [Get early access](#)

[Mark Your Spot!](#)

# The Most Impactful Attack Types

Hacks Recorded

Attack Type

Amount Lost

137



\$352.27M

Exit scams

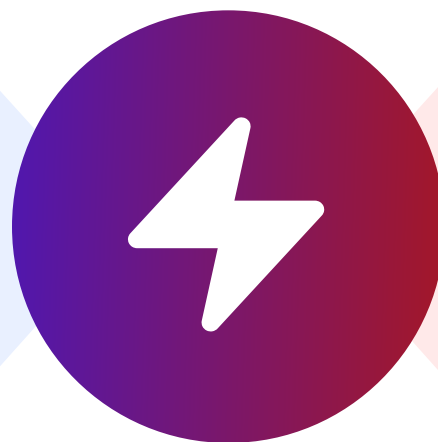
103



\$204.55M

Smart contract vulnerabilities

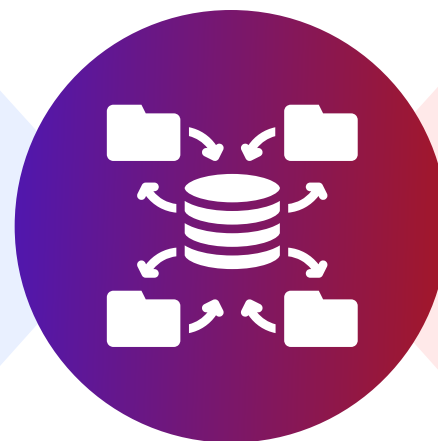
35



\$222.42

Flash Loans

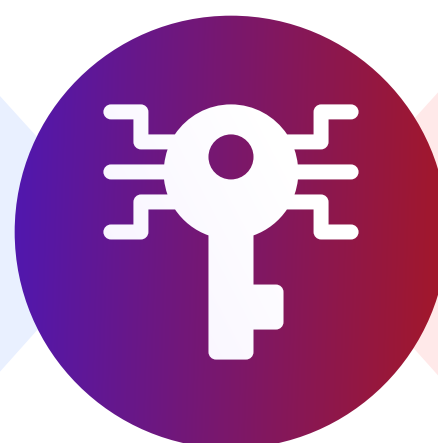
25



\$15.3M

Price/Oracle manipulation

14

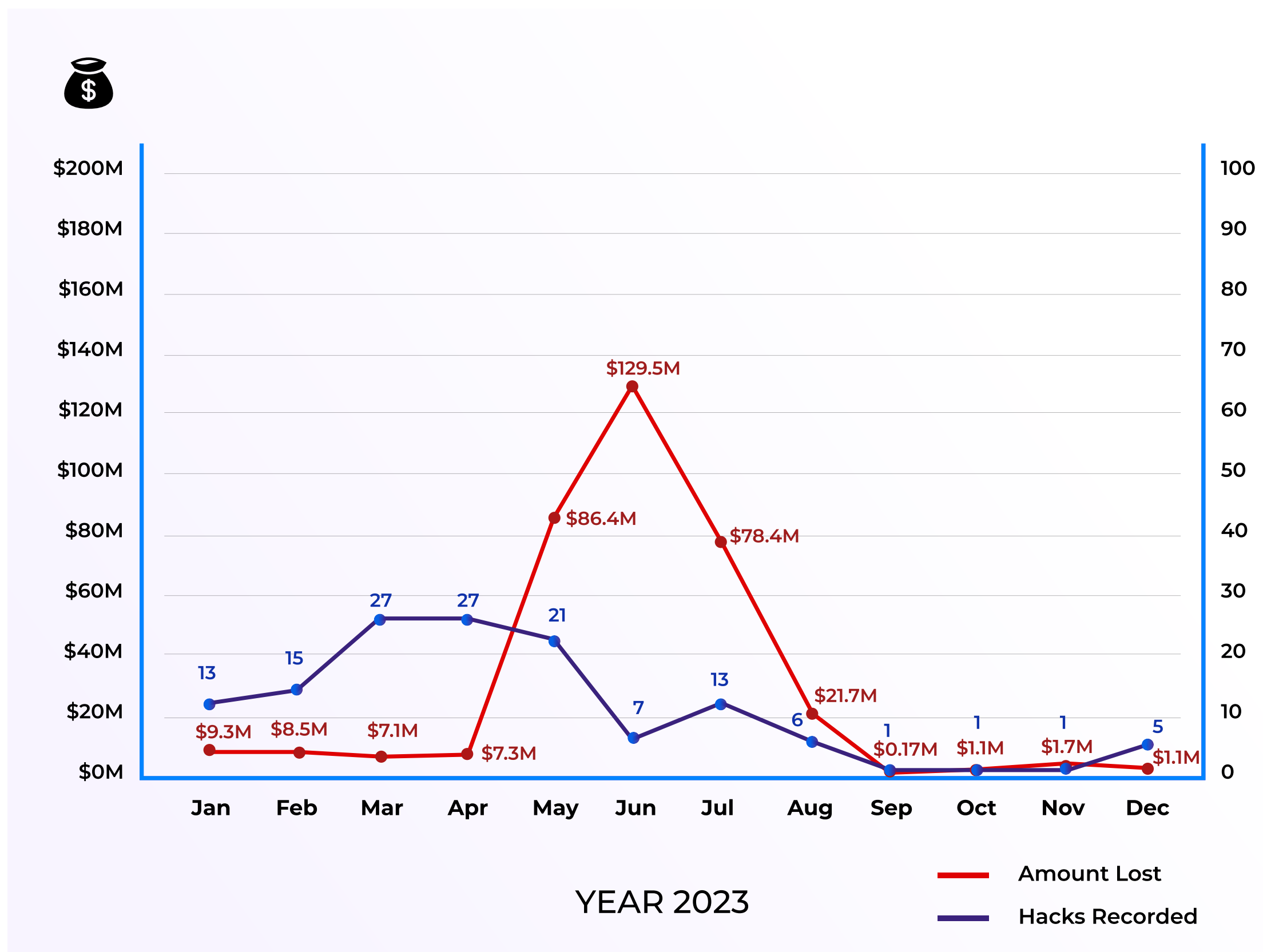


\$174.5M

Private key compromise

- Exit scams saw the highest frequency and most significant financial losses, leading to **\$352.27M** lost across 137 incidents in 2023.
- Smart contract vulnerabilities resulted in the second-highest losses, amounting to **\$204.55M** across 103 incidents.
- Social account takeovers accounted for **107** hacks, contributing to substantial financial impacts
- Flash loans and Oracle manipulations combined for **60 out of 440 recorded** incidents in 2023.
- Observations from incidents highlighted an increased likelihood of **private key compromises**, stressing the critical need for vigilant security practices and thorough assessments before utilizing third-party tools.

## Rug Pulls



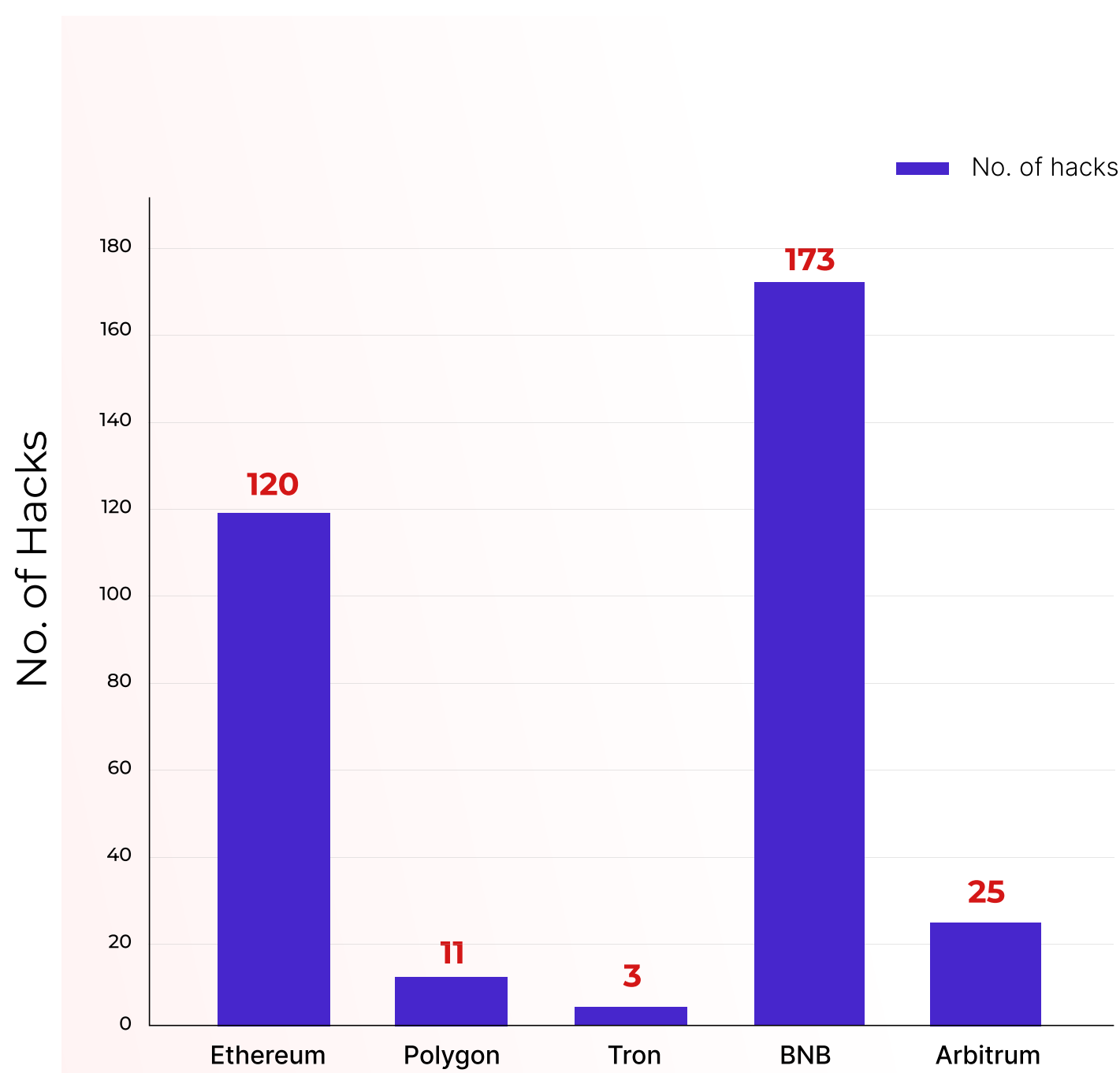
Throughout 2023, rug pull incidents stood out with notable characteristics:

- 1. Frequency and Timeline:** Rug pulls hit a staggering count of 137 incidents throughout the year, with the first half witnessing the most severe impact compared to the latter.
- 2. Financial Impact:** Over 25 projects, including Anubis DAO, Pepe Coin, Fintech, BALD, and Magnate Finance, suffered losses amounting to millions of dollars due to these rug pulls.
- 3. Unaudited Projects:** A significant observation was that most projects involved in rug pulls were unaudited, and some concealed backdoor functionalities within their code, posing challenges for investors while assessing project security.
- 4. Lack of social media presence:** Projects like AnubisDAO and many others managed to attract substantial investments despite lacking a well-developed website and proper social media information.

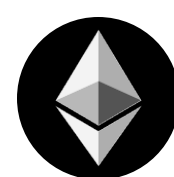
QuillCheck, a crypto due diligence tool, comes to the immediate rescue in addressing these challenges with certainty. It aids investors in comprehending token permissions and market insights of the token, and by generating risk scores with detailed reports, it streamlines the due diligence process for smarter and well-informed decisions.



# Chain-specific loss

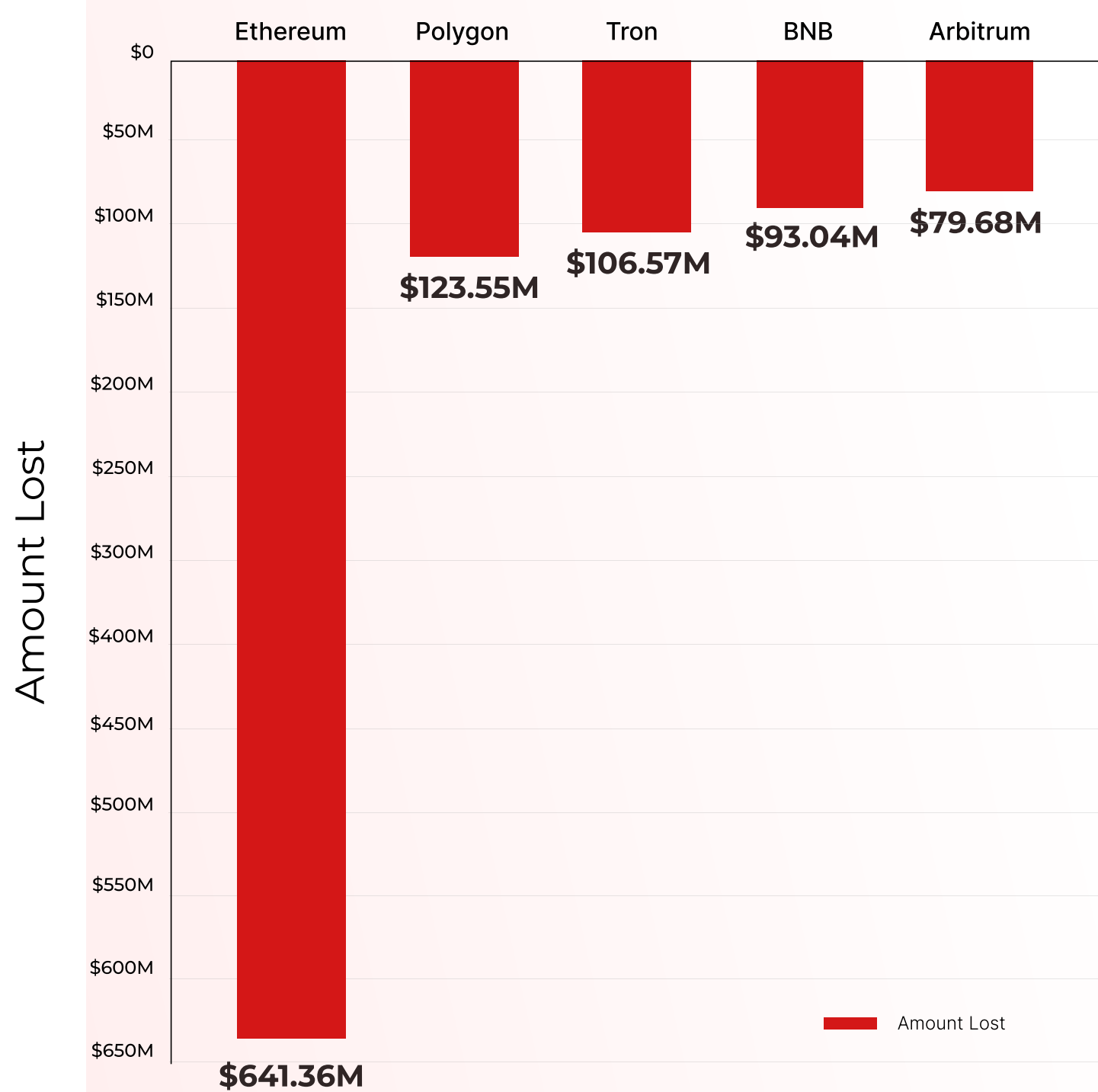


A total of **12 public chains** experienced security breaches in 2023. The most targeted chains, in terms of the number of hacks, were BNB Chain, Ethereum, and Arbitrum. Meanwhile, Ethereum, Polygon, and Tron suffered the most significant losses in value due to these breaches.



**Ethereum** faced over 120 attacks, resulting in losses totaling **\$641.36 million**. This accounts for nearly **38%** of the total losses incurred throughout the year.

Chain



**BNB Chain** encountered **173 attacks**, leading to losses ranging from a few hundred dollars to millions in value



**Polygon** experienced **eleven attacks**, culminating in a total loss of **\$123.55 million**. The **BonqDAO hack** alone contributed to the majority of this figure, accounting for \$120 million of the total losses.



QuillCheck [HoneyPots](#) [Pro](#) [FAQs](#) [API Docs](#) [Get API Key](#)

## Crypto Due Diligence Tool

Safeguard your web3 investments with our advanced detectors. Uncover honeypots, understand token permissions, and get comprehensive market insights. Shield yourself from rugpulls and scam tokens. DYOR here!

**Evaluate Any Token**  
Powered by QuillAudits

[View Sample Report](#)

ETH BSC Polygon

ETH Token Address

**10,000+**  
Honeypots Detected

**50,000+**  
Tokens Scanned

**24x7**  
Scam Token Monitoring

**Get Rugpull Diligence with QuillCheck**

# 3

## **Audit Corner: Auditor's Talk on Smart Contract Security & Threat Prevention**

# Audit Corner: Auditors' Talk On Smart Contract Security & Threat Prevention

“In the world of Web3, security is a big deal because of the Sensitive info and Money flying around in the decentralized space.”



“

*“DeFi represents the democratization of finance, but it's only as strong as its security. We're committed to setting the highest security standards in the industry.”*

~ Stani Kulechov, CEO of Aave

”

To establish safety, developers need to code smart and check for security issues to avoid hacks and cyber problems. It's also crucial to ensure projects can handle large volumes of transactions and users as well. That means smooth coding, clever caching, and a scalable setup.

This section delves into insights and recommendations on smart contract security and risk mitigation strategies gathered from security experts and seasoned auditors at QuillAudits'.

## A. Solidity Best Practices and Readability

- **Correct Use of Pragma:** Avoid using pragma solidity ^0.9.1 when setting the compiler version yourself. This can create ambiguity for verifiers regarding the version of solidity used. This pattern should only be used for library code where you're not the one compiling it. Instead, use pragma solidity 0.9.1.
- **Use of clear and descriptive constants:** Refrain from using unexplained constants in your code. It's more effective to use constant variables that describe the value. For instance, instead of using an arbitrary number like 5000, define it as a constant: public uint256 constant MAXIMUM\_TRANSACTION\_LIMIT = 5000;

- **Clarity of 'Require' statements:** Require statements should always include an explanatory message. Instead of

✗ `require(numTokens < MAXIMUM_TRANSACTION_LIMIT)`

✓ use `require(numTokens < MAXIMUM_TRANSACTION_LIMIT, "Number of tokens exceeds transaction limit")`

- **Avoid Left-to-Right Character:** The Unicode U+202E character, which changes the order in which text is rendered, should be avoided as it's often used for obfuscation purposes.
- **Follow precise naming conventions:** Ensure variable and function names are precise, accurate, and descriptive. For example, instead of naming a variable as 'data', name it as 'transactionData'. Also, flag any inaccurate or outdated comments.
- **Remove unused variables:** Unused variables can hinder clarity. For example, if a variable 'unusedVar' is declared but not used anywhere in the code, it should be removed.
- **Readable keywords:** Exercising proper Use of Readability Keywords (hours, days, ether, 1\_000\_000, etc.) is a must. In Solidity, 1 day will automatically convert to 86400 seconds, which is much more readable. Similarly, use 1\_000\_000 over 1000000 for readability. When dealing with Ethereum quantities, use the ether keyword. For example, instead of writing 10000000000000000000, write 1 ether.

## B. On-Chain vs Off-Chain Computation

- **Optimize gas usage:** Consider using off-chain computation to minimize gas usage and improve performance.

- **Real-time data integration:** Use a suitable off-chain computation mechanism, such as an oracle or external API, to provide real-time data to the smart contract.
- **Criticality of on-chain functions:** Use on-chain computation only when necessary, such as when executing critical functions that require the security and immutability of the blockchain.

## C. Upgradeability

- Implement upgradeability mechanisms to allow for future updates and improvements to the smart contract.
- Use a proxy or delegate contract to separate the contract logic from the contract data, allowing for upgrades without affecting the data.
- Use a versioning system to track changes to the contract and ensure compatibility with existing deployments.
- Implement a governance mechanism to ensure that upgrades and changes are approved by all relevant parties and comply with the requirements of the smart contract.

**Bonus Tip:** Aim for simplicity during development. Always choose the simplest solution that serves your needs. Your solution should be understandable to any member of your team.

## D. Function Composition

- Use function composition to break down complex functionality into smaller, reusable functions.
- Use the **internal** visibility modifier for functions that are intended to be used only within the contract and **public** or **external** functions that can be accessed outside the contract.
- Use **view** or **pure** functions for read-only functionality that does not modify the state of the contract to minimize gas usage.
- Ensure that composed functions are properly validated and tested to avoid potential vulnerabilities.

## E. Inheritance

- Use inheritance to simplify and modularize contract code.
- Use the **is** keyword to indicate that a contract inherits from another contract.
- Implement abstract contracts to provide a template for child contracts to follow.
- Avoid deep inheritance chains, which can make the codebase difficult to manage and increase the risk of bugs or vulnerabilities.
- Ensure the child contract's implementation follows the requirements and constraints specified in the parent contract.

## F. Events:

- Use events to record contract activity and allow for off-chain monitoring and analysis
- Use the **emit** keyword to trigger an event in Solidity.
- Define the event parameters and their types in the contract to provide a clear and standardized event format.
- Use events to provide feedback to users on the status of contract transactions or state changes
- Ensure that events are properly validated and tested to avoid potential vulnerabilities.

## G. Avoid Known Pitfalls

- Use dependencies like external libraries or APIs to simplify and modularize contract code.
- Use the latest version of the Solidity programming language and keep up to date with any security-related updates or bug fixes.
- Ensure that external contracts interacting with your business are compatible with each other, including deflation/inflation tokens and reentrant tokens like ERC-777, ERC-677, and ERC-721. Refer to the [Reentrancy Vulnerability Case for guidance](#).
- Consider the risk of reentrancy in external function calls.
- Avoid excessive loops when reading or assigning contract storage variables.



- Avoid concentrating authority, especially the ability to modify critical contract parameters. Instead, use governance mechanisms, timelock contracts, or multi-signature contracts to manage authority.
- Maintain linear inheritance relationships between contracts and ensure that inherited contracts are necessary for business.
- Avoid using on-chain block data as a seed source for generating random numbers.
- Fully consider the possibility of rollback attacks when acquiring and using random numbers.
- Use Chainlink's VRF to obtain reliable random numbers. Refer to the [Chainlink VRF](#) for more information.
- Avoid calculating LP Token price directly from the token quantity of third-party contracts. See [How to get the price of LP correctly for guidance](#).
- Use at least three price sources when obtaining prices through third-party contracts to avoid relying on a single source.
- Record the status of execution using events in key business processes for data analysis purposes during the project's runtime.
- Have an emergency switch in place for suspending global and core business activities to prevent further losses in case of a black swan event.
- Ensure that dependencies are properly validated and secure before incorporating them into the contract code.

- Use standardized formats like ABI or JSON to ensure compatibility and interoperability with dependencies.
- Regularly update and maintain dependencies to ensure compatibility with Solidity's latest version and address potential vulnerabilities.

## H. Gas optimization

- **Use `uint256` instead of boolean and small units:** A boolean is internally represented as a `uint256` with 255 bits masked out. The extra masking operations incur additional gas costs. It is generally more gas-efficient to use `uint256` for non-storage variables unless there is a specific need to reduce their size.
- **Prefer bit shifting over division or multiplication by two:** Bit shifting operations (`<<` and `>>`) cost three gas, whereas multiplication and division operations cost five gas. If overflow is not a concern, using bitshifts is a more gas-efficient option.
- **Utilize bitmaps for contracts with many boolean variables:** Under the hood, a boolean is actually represented as a `uint8`. This means a `uint256` can hold up to 32 boolean variables. If you have more than 32 boolean variables, it is more efficient to store them as individual bits within a `uint256` variable.
- **Optimize function names for gas-sensitive functions:** Functions with leading zeros in their names can save gas in two ways. First, they are sorted to the top of the bytecode, resulting in fewer checks to determine if that function is called. Second, leading zeros save gas as part of the transaction data. However, this approach can lead to unconventional function names and should only be used when gas savings are significant.

- **Pack function arguments for gas-sensitive functions:** If you need to represent two 128-bit numbers, consider combining them into a single 256-bit number and splitting them into two numbers within the contract. This technique can save gas. However, it may reduce code readability and should only be used for gas-sensitive functions.
- **Avoid using `require(x >= 0, "...")` for units:** Unsigned integers cannot be negative, so checking for `x >= 0` is redundant and wastes gas.
- **Split `require` statements instead of using boolean operators:** Using separate `require` statements for individual conditions is more efficient than combining them with boolean operators. For example:

```
require(a, "message a");
```

```
require(b, "message b");
```

- **Compare boolean literals directly:** When `condition` is a boolean variable, using `if (condition)` is more gas-efficient than `if (condition == true)`.
- **Prefer `x = x + y` over `x += y`:** Although mathematically identical, the compiler treats these operations differently, and `x = x + y` is generally more efficient in terms of gas usage.
- **Prefer `++x` over `x++`:** If a variable is incremented in isolation, prefix incrementing (`++x`) is preferred over postfix incrementing (`x++`) due to compiler optimizations.
- **Use `unchecked { ++x }` when using a loop:** If a loop uses `uint256` (which is recommended) and overflow is not a concern, it is safe to increment the loop variable using `unchecked` to remove overflow checks and save gas.

- **Avoid initializing storage variables to default values:** Storage variables are initialized to their default values (zero, false, etc.) automatically, so explicitly initializing them wastes gas.
- **Swap variables in a single line:** To swap variables without using an explicit temporary variable, you can use the following syntax

`(x, y) = (y, x);`

- **Use `>` or `<` instead of `≤` or `≥` when possible:** Solidity represents `≥` as `not <`, which incurs an extra `not` operation and costs three gas. Using `>` or `<` directly is more gas-efficient.
- **Use `abi.encodePacked` over `abi.encode` for fixed-length arguments:** The `encode` function pads the inputs, which incurs additional gas costs. For fixed-length inputs, using `abi.encodePacked` is more gas-efficient and wrapping in `keccak256` for security.
- **Admin functions can be payable:** If a function is meant for administrative purposes and can receive Ether, making it payable can save gas compared to explicitly checking `msg.value` and reverting. However, for functions accessible to the general public, making them non-payable may still be desirable to prevent unexpected state transitions.
- **Declare the return variable in the function signature:** To save a small amount of gas, declare the return variable directly in the function signature instead of using an explicit variable assignment inside the function body.

# I. Testing and Verification

- Write comprehensive unit tests for each function in the contract.
- Use a testing framework, such as Hardhat, Foundry or Remix, to automate testing and provide standardized test results
- Write integration tests to ensure that the contract works with other components of the system
- Verify the contract code on a blockchain explorer, such as Etherscan or BscScan, to ensure that it is properly deployed and functioning as intended
- Consider audits from **QuillAudits** to identify potential vulnerabilities, improve the security infrastructure of contracts and enhance overall robustness.



# QuillAI

**QuillAI**  
Beta

- Input Contract Address
- Upload Smart Contract
- View Session History

[Request Manual Audit](#)

Join Our Journey

Powered by QuillAudits

## AI Powered Smart Contract Audit Tool

Get real-time audits to protect your Web3 projects from vulnerabilities. Fast results, multi-chain support, actionable insights and more. Audit Now and Audit for Free!

- Precise, automated smart contract audits to ensure security, efficiency, and adherence to best coding practices.
- Find potential vulnerabilities, understand severity of issues, get immediate resolution for your bugs.
- Enter any Solidity based smart contract and get instant audit report to fix your bugs before you commit.
- Deploy with confidence after QuillAI's comprehensive review of your smart contracts.

[View Sample Report](#)

You get **3** FREE Audits every day

All Rights Reserved © Copyright 2023. QuillAudits - LLC

[Click Here to Check QuillAI](#)

# Security Secret: Pro Tips By Experts For Builders In Web3



“

*"Web3 is about empowering individuals, but with great power comes great responsibility. Security is the responsibility of every participant in this decentralized ecosystem."*

~ Joe Lubin, Co-Founder of Ethereum and Founder of ConsenSys

”

## 1. Pre-audit Readiness and Ongoing Audit Practices

- Incorporating multiple external audits, avoiding unaudited updates, and integrating robust security and coding practices during development phases is vital.
- Seek guidance from experts to ensure secure architecture design right from the outset
- Ensure ongoing audit practices by consistently auditing upgrades.

## 2. Education and Skill Advancement

- Developers must continually learn and enhance their skills for secure smart contract coding.

## 3. Do not leave your security issues "acknowledged"

- Vulnerabilities, depending on their type, may not manifest immediately but can be triggered externally. So, it's of utmost importance to fix them if feasible.

## 4. Actively utilize audit firms

- It's necessary to maintain ongoing communication. If there are issues that inevitably need updating in your project or if there are any future developments planned, it's strongly recommended to ask the audit firms, which avoids some potential risks.

## 5. Optimal security practices

- Leveraging AI-based LLMs to uncover vulnerabilities efficiently can be of great help with early-stage security reviews.

## 6. Robust Authentication and Authorization Mechanisms

- Implementing and upholding stringent authentication and authorization protocols for Web3 applications is paramount to mitigating security risks.

## 7. Data Protection and Privacy Measures

- Ensuring the implementation of comprehensive data protection and privacy measures is indispensable for fostering trust and resilience within the Web3 landscape.

## 8. Adapting to Industry shifts

- The Web3 industry is witnessing an early phase of defense-focused products. Notably, there's a transition from Intrusion Detection Systems (IDS) to Intrusion Prevention Systems (IPS), unlike Web2.



Therefore, it is recommended to implement DevSecOps fundamentals across all development stages, which involve rigorous security measures such as smart contract audits pre-release, vigilant post-release monitoring for potential threats, and a well-prepared incident response strategy for optimal protection.

These collective insights illuminate the path toward resilient security practices essential for smart contract developers to handle the evolving landscape of Web3 technologies.

## Effective Threat Response Strategies For Web3 Protocols



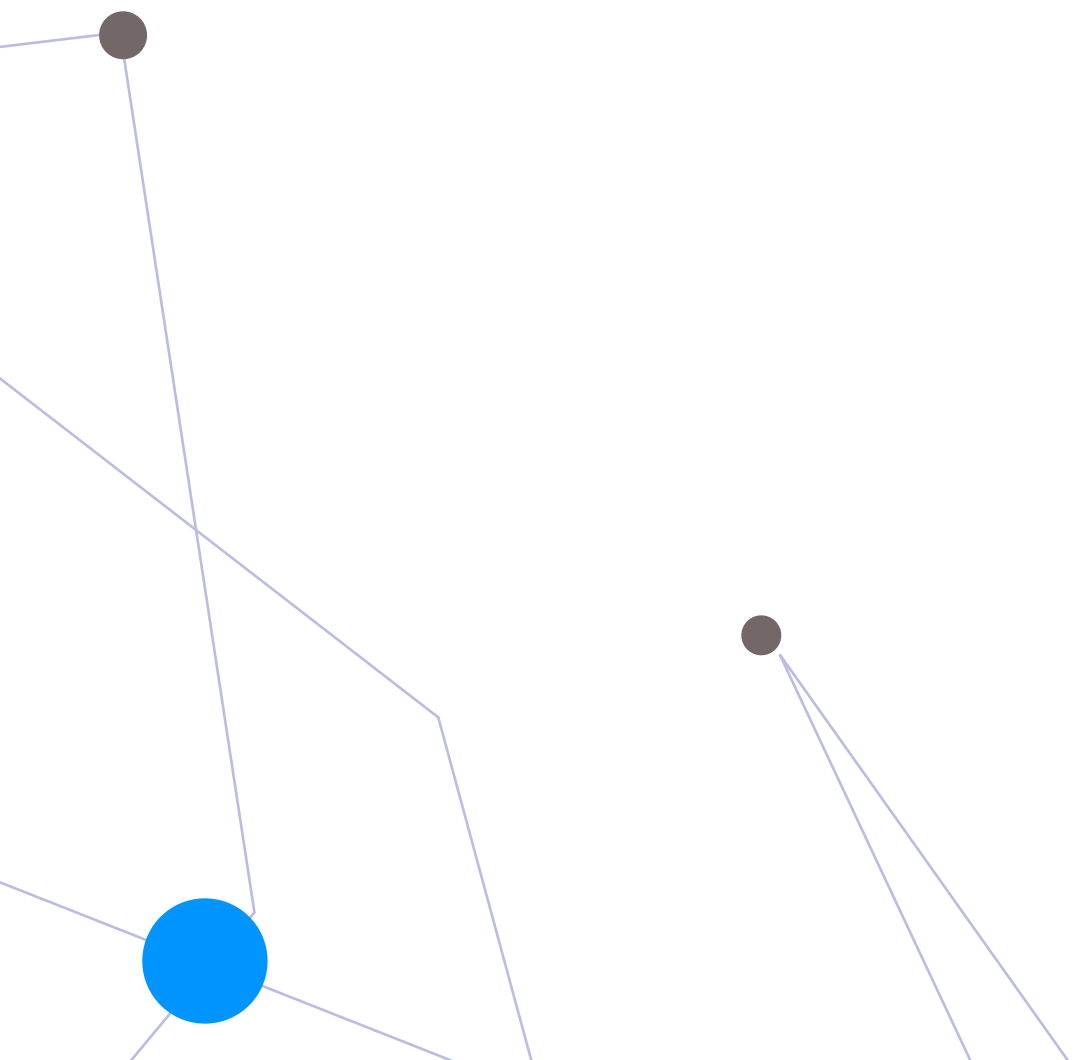
*“To combat threats, we think smart contract code auditing alone is not enough. There is no silver bullet in security, but a continuous and systematic approach is essential. We think there are several techniques that should be adopted by protocols to mitigate threats.”*

~ Andy Zhou, CEO of BlockSec



- **An Effective Incident Response Strategy:** Maintaining awareness of the risk is the first step. Establish a system to monitor transactions that interact with your protocol and evaluate the security impact of these transactions. Though developing such a system by protocol is possible, the security expert’s insights are crucial for their effectiveness.

- **Automated Threat Response Techniques:** Secondly, leveraging techniques that can automatically respond to threats is crucial. Manual responses are often ineffective, as the attack can rapidly drain all the assets in a few transactions. Deploy systems like Falcon Block that can automatically block hacks for this purpose.
- **Streamlining Operational Crisis Management:** Finally, it's crucial to have a solid plan for handling incidents. Identify the main contacts within security teams, law enforcement, and exchanges who can assist during emergencies. Prior to implementation, review and prepare any patches carefully.





**4**

**Recent  
Developments  
in Web3  
Security**

# Recent Developments in Web3 Security

The Web3 landscape is an ever-evolving ecosystem where security remains a paramount concern. With the increasing adoption of blockchain technologies, decentralized applications (DApps), and smart contracts, the need for robust security measures has never been more critical. This report delves into the recent developments in Web3 security, underscoring the innovations and challenges that define this dynamic field.

## **Rise in Use of Artificial Intelligence in Auditing and Tooling**

- The onset of 2023 witnessed a rise in the development of Artificial Intelligence and its role not just in smart contract auditing but in investment as well.
- AI stablecoins became a hotspot for investment during the course of 2023.
- We also saw tools like AuditWizard and ChainGPT making their place in the security community
- AI bots also were the talk of the town in finding low-hanging fruits in audit contests.
- However, the competence of AI in auditing smart contracts is still to be seen with the lens of skepticism since any tool is not enough to make a smart contract completely secure.

## **Monumental adoption of layer2 scaling solutions**

- During 2023, there was a huge increase in the adoption of layer2 scaling solutions for Web3.
- Layer2 scaling solutions provide a comprehensive mitigation of EVM blockchain with faster transaction speeds and low gas fees.
- We saw more and more smart contracts opting for layer2 chains as their mainnet deployment choice.
- Arbitrum came out as the top choice for scaling solutions followed by Optimism as second option.
- For the first time optimistic rollups hit 9.5M unique addresses and more than doubled during the later year.
- Polygon saw a 541% increase in monthly active smart contract accounts.
- Layer2 solutions are expected to continue growing even beyond 2023.

## **Zero Knowledge Proofs**

- 2023 saw the mainstream use of Zero Knowledge proofs in action.
- 2 of the most prominent players, MatterLabs and Polygon came up in the fronts of the battlefield with zkSync and zkEVM, respectively.
- We also saw a huge acceleration in the development of ZKPs in the form of different languages like Starknet and Cairo.

- Zero-Knowledge proofs also made their space in layer2 scaling solutions like Starknet, Scroll, Aztec and many more.
- ZKP's are expected to continue their growth with more and more innovative use cases enhancing the privacy aspect of blockchain.

### **Smart Contract Wallets and Account Abstraction**

- ERC-4337 or Account Abstraction became a buzzword and a game changer in Web3 space. Combined with L2 solutions, account abstraction can really be a game changer and can unlock the full potential of Web3.
- Things like Transaction Bundling, 2FA security model for wallets, account recovery and automated payments will be facilitated by account abstraction and will be a net positive for upcoming years.

### **Increased community initiatives and developer awareness**

- 2023 was a force multiplier in smart contract security activities. We saw a rise in prominent security researchers to secure the Web3 space.
- Solo auditing became a good product market fit and saw a huge attraction from all kinds of people.
- We also saw some huge community initiatives by the security community by providing access to information.
- Tools like Solodit and Defihacklabs became a goto source for beginners.
- 2024 will likely see more activity in onboarding new auditors and huge initiatives from the community.

## **Strengthening defenses against rising threats**

- Project teams are placing greater emphasis on establishing a robust monitoring system for continuous surveillance of external threats.
- With technical monitoring and operational monitoring, platforms like Forta provide technological frameworks that attract security experts to build monitoring bots.
- OpenZeppelin launched Defender 2.0, which allows users to create their own custom monitoring systems.
- Even more notably, companies like BlockSec, after numerous successful white-hat preemptive actions, have introduced product-level solutions Phalcon Block that integrate attack detection and automatic blocking to directly address user issues.
- In the upcoming year, it is foreseeable that as various monitoring infrastructures and services become more sophisticated, a turning point will emerge in the struggle between good and evil, vulnerabilities and safeguards.

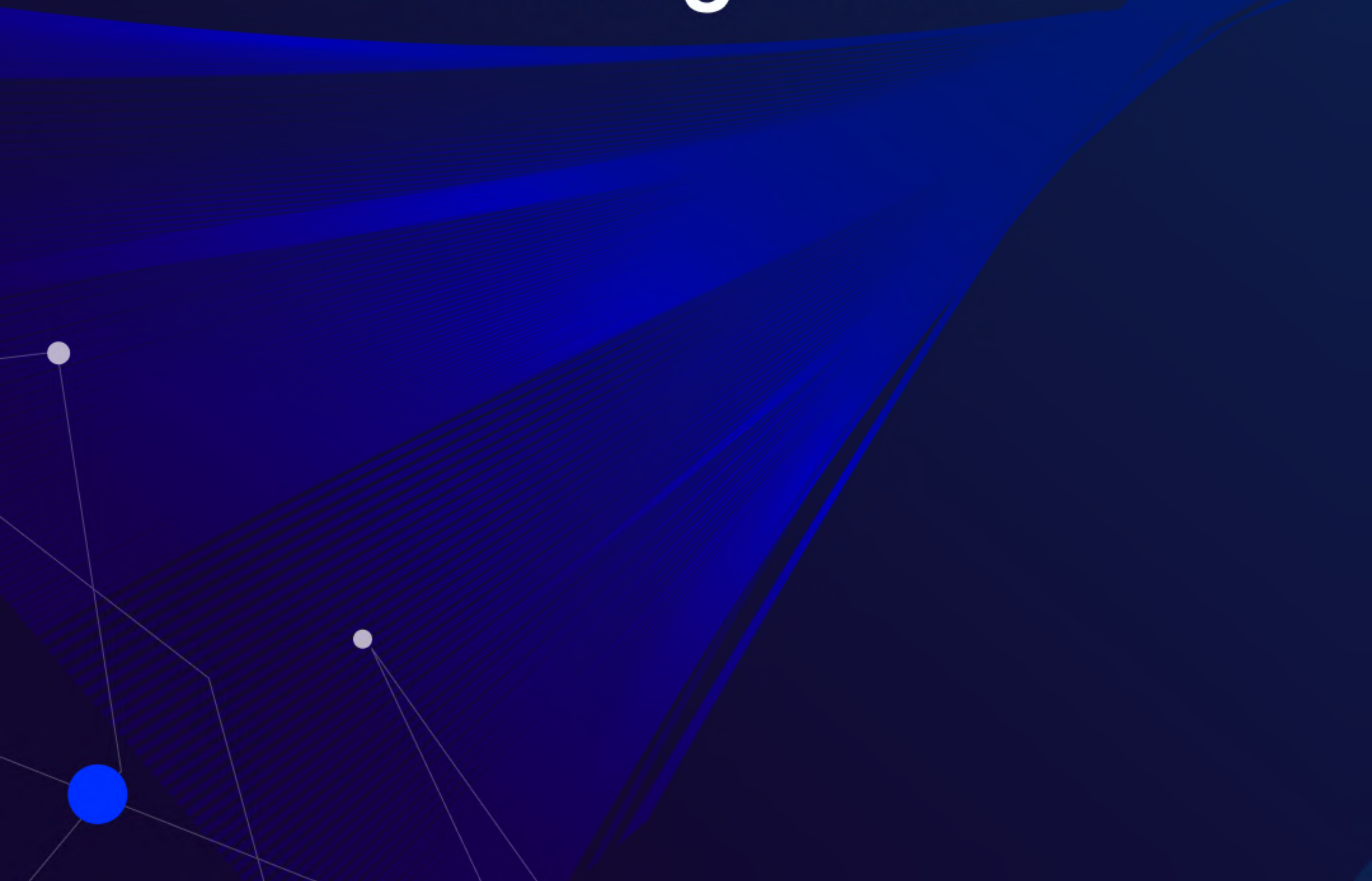

## **Increased community initiatives and developer awareness**

- 2023 was a force multiplier in smart contract security activities. We saw a rise in prominent security researchers to secure the Web3 space.
- Solo auditing became a good product market fit and saw a huge attraction from all kinds of people.
- We also saw some huge community initiatives by the security community by providing access to information.
- Tools like Solodit and Defihacklabs became a goto source for beginners.
- 2024 will likely see more activity in onboarding new auditors and huge initiatives from the community.



**5**

**Predictions of  
Web3 Security  
Landscape in the  
Coming Years**





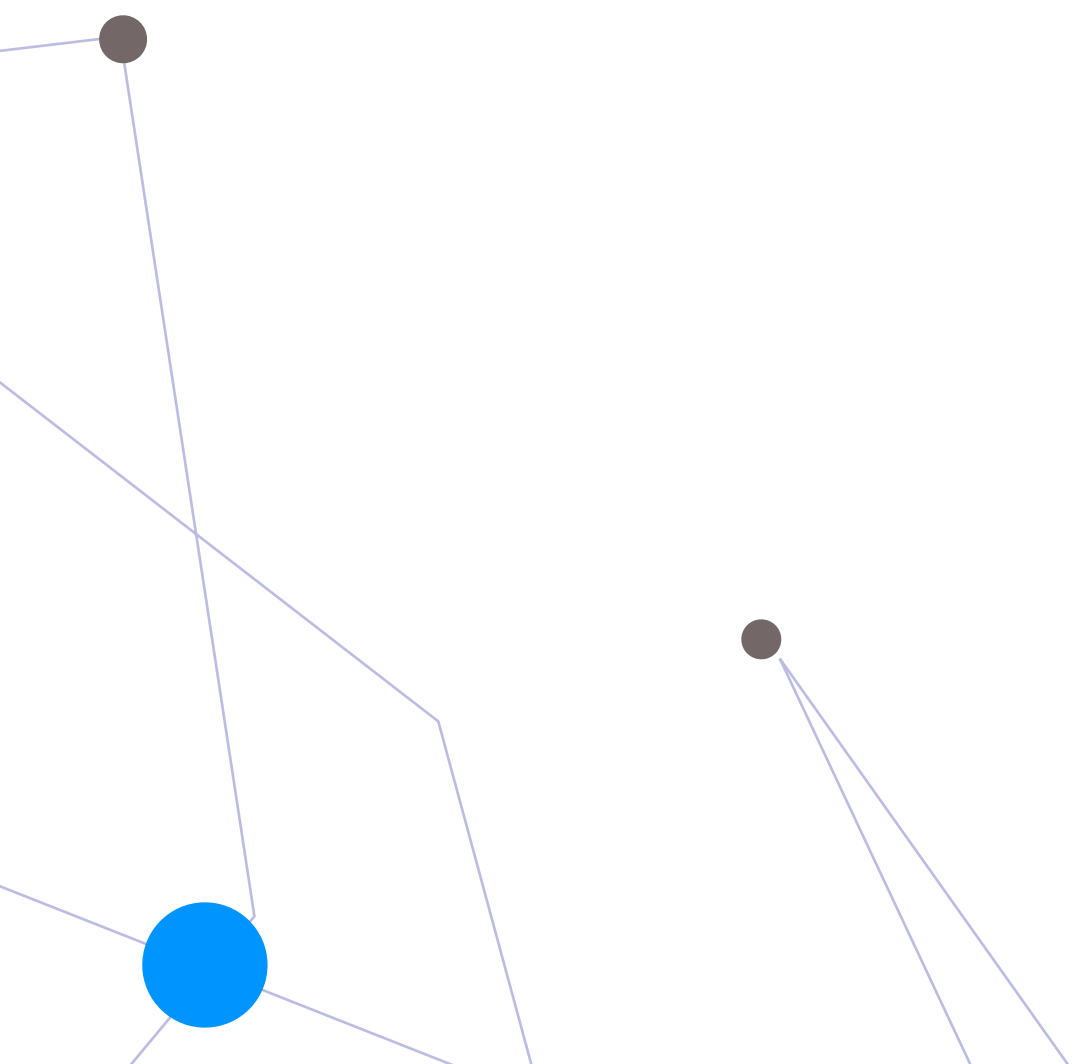
# Predictions of Web3 Security Landscape In The Coming Years

In the coming years, Web3 is poised for significant shifts with a higher degree of data protection and more personal risk for users. Several predictions from security experts outline key transformations:

- 1. Vulnerabilities from Web2.0 Front-Ends:** In the transition from Web2.0 to the decentralized Web3, while advancements have been made, vulnerabilities persist. Despite its decentralized architecture, Web3 remains exposed to familiar risks inherited from Web2.0 front-ends. Threats like user credential theft, cross-site scripting, code injection, bots, and API-based attacks continue to pose significant risks.
- 2. Smart Contract Vulnerabilities:** One critical area of concern resides in smart contracts, the backbone of numerous Web3 applications. Shockingly, they contributed to half of the total hacks recorded in 2022, demanding urgent attention and stringent security measures in the coming years.
- 3. Ransomware Attacks:** The rise of Web3 has attracted the attention of cybercriminals, leading to an 80% year-on-year increase in ransomware attacks targeting Web3 projects and protocols.
- 4. Team Audits and Specializations:** With the increase in security concerns, expect a rise in team-based audits comprising various specializations like testing services, architecture design, economic modelling, and more. Specialized roles focusing on fuzzing campaigns and security reviewing will likely emerge. This evolution could lead to a consolidation in the industry, potentially merging or absorbing existing players.

- 5. Surge in Web3 Security Demand:** With more builders and projects entering the Web3 space, the demand for security measures is set to soar. Similar to the e-commerce boom in the 1990s, significant investments are expected in cybersecurity to mitigate potential threats and instil trust in Web3 technologies.
- 6. Rise of zk Technology:** Zero-knowledge (zk) technology is projected to become a major field post-2024. The complexity of zk security poses challenges for conventional hackers to transition into this specialized field.
- 7. Blockchain and AI Synergy:** More business leaders are recognizing the potential of blockchain beyond cryptocurrency. The understanding of blockchain's capabilities has significantly improved, with 84% now seeing it as a valuable asset. In 2024, it's expected to become a vital support system for AI, enhancing transparency and data security.
- 8. Standard regulatory frameworks for Web3:** The anticipation is for regulatory frameworks to take shape in 2024. These regulations aim to ensure consumer protection, transparency, and fraud prevention within the Web3 space. They'll foster trust, level the playing field for businesses, and encourage broader participation in Web3 technologies.

- 9. Enhanced Data Security and Lineage:** With the increasing reliance on AI and machine learning models for critical decision-making, ensuring the lineage of these models becomes crucial. Blockchain's core technology, rooted in immutability and cryptographic security, is expected to integrate with enterprise AI systems.
  
- 10. Technical Failures:** Alongside these threats, the inherent complexities of blockchain networks are likely to introduce new vulnerabilities such as traffic overloads and technical malfunctions.
  
- 11. Role of AI and ML in Cybersecurity:** However, amidst these challenges, there's promise in the integration of AI and ML technologies in fortifying Web3 security. These advanced technologies are expected to revolutionize threat detection and response capabilities, offering a proactive shield against sophisticated cyber threats.



# 6

## Technical Insights into Secure Smart Contract Coding And Risk Mitigation Steps



*“Web3's potential is immense, but so are the risks. Security should be embedded in the DNA of every project, from the code to the governance model.”*

~ Kathryn Haun, General Partner at Andreessen Horowitz



## Re-entrancy Errors

Reentrancy exploits a contract's logic by allowing reentry into a transaction during execution, typically triggered by an external call. This can lead to unexpected and manipulative behavior, making it a critical vulnerability to address in smart contract coding.

## Types of Reentrancies

### 1. Single Function Reentrancy

A single function reentrancy attack occurs when a vulnerable function is the same function that an attacker is trying to recursively call. Example:

```
1 function withdraw() external {
2     uint256 amount = balances[msg.sender];
3     require(msg.sender.call.value(amount)
4 ());balances[msg.sender] = 0;
5 }
```

- The above example demonstrates a case of single function reentrancy. It involves a withdraw function that sends ETH to msg.sender using call() and then resets the user's balance to 0.

- This function does not follow the Checks Effects Interaction pattern. It first retrieves the balance amount from the balances mapping on the first line, then sends the amount to msg.sender using the call function on the second line, and finally sets the balances mapping for that msg.sender to 0.
- The reentrant call can be made by using a fallback function in the attacker contract. The attacker contract can call the withdraw function of this contract, enabling the attacker contract to react upon receiving funds and subsequently call the withdraw function again.

### *Mitigation Steps*

- Implement Checks-Effects-Interaction pattern: Reorder the sequence of operations to first update the sender's balance and then transfer ETH.
- Use function modifiers like ReentrancyGuard by OpenZeppelin to prevent reentrant calls during function execution.

## 2. Cross Function Reentrancy

Cross-function reentrancy occurs when a vulnerable function shares state with a function that an attacker can exploit.

```
1 function transfer(address to, uint amount) external {
2     if (balances[msg.sender] >= amount) {
3         balances[to] += amount;
4         balances[msg.sender] -= amount;
5     }
6 }
7
8 function withdraw() external {
9     uint256 amount = balances[msg.sender];
10    require(msg.sender.call.value(amount)());
11    balances[msg.sender] = 0;
12 }
```

- In the above example, the attacker can re-enter the system using the fallback function
- The current re-entrancy can be exploited by withdrawing the ETH balance using **withdraw()** and re-entering by using the fallback function to transfer the ETH balance using **transfer()** before the code sets it to 0 on the last line of the 'withdraw' function.

### Mitigation Steps

- Reorder the sequence of setting values to prevent reentrant calls before state updates.
- Apply function modifiers like OpenZeppelin's ReentrancyGuard to halt reentrant calls during execution.

## 3. Read Only Reentrancy

In this form of reentrancy attack, an external caller re-enters a contract that reads the state of another contract.

```
1 // Has a reentrancy guard to prevent reentrancy
2 // but makes state change only after external call to sender
3 function withdraw() external nonReentrant {
4     uint256 amount = balances[msg.sender];
5     (bool success,) = msg.sender.call{value: balances[msg.sender]}("");
6     require(success);
7     balances[msg.sender] = 0;
8 }
9 }
10
11 contract B {
12     // Allows sender to claim equivalent B tokens for A tokens they hold
13     function claim() external nonReentrant {
14         require(!claimed[msg.sender]);
15         balances[msg.sender] = A.balances[msg.sender];
16         claimed[msg.sender] = true;
17     }
18 }
```

- In the above example there are two contracts A and B with **withdraw()** and **claim()** functions respectively.

- Contract B has a `claim()` function that allows claiming B tokens equivalent to A tokens held in contract A.
- Users can withdraw deposited tokens using A: **withdraw()** function. When withdrawing, the function utilizes **call()** to send A tokens (ETH/blockchain native tokens) to the `msg.sender`. Here, the receiver (`msg.sender`) can utilize the `fallback/receive` function to reenter contract B and call **claim()**, allowing them to receive an equivalent amount of B tokens even after withdrawing A tokens (ETH).
- The **claim()** function employs the state of the balances mapping from contract A, which isn't updated immediately after **withdraw()** sends tokens because the state of balances mapping for `msg.sender` is set to 0 after sending tokens with `call()`.
- Although both functions use `nonReentrant` modifiers, these modifiers belong to different contracts and employ different states. While they prevent reentrant calls for functions within the same contract, they do not synchronize states across different contracts.
- This sample example highlights the potential risks of read-only reentrancy in smart contracts, which can be dangerous based on the code implementation.

### *Mitigation Steps*

- **Checks-Effects-Interaction Pattern:** Follow the pattern to ensure state changes before interacting with other contracts. For example, Rearrange the sequence to first update state variables, then proceed with external calls.
- **Synchronization of State:** Ensure consistent and synchronized state between contracts. Validate state updates and access across contracts to prevent discrepancies.



# Denial Of Service

DoS in smart contracts refers to disrupting the flow of the contract, rendering it unworkable or consuming excessive gas, which freezes the contract execution. This attack can temporarily or permanently incapacitate the contract.

```
1 for (uint256 j = 0; j < _sellHistories.length; j ++) {
2
3     if (_sellHistories[j].time >= maxStartTimeForHistories) {
4
5         _sellHistories[i].time = _sellHistories[j].time;
6         _sellHistories[i].bnbAmount = _sellHistories[j].bnbAmount;
7
8         i = i + 1;
9     }
10 }
11 uint256 removedCnt = _sellHistories.length - i;
12
13 for (uint256 j = 0; j < removedCnt; j ++) {
14
15     _sellHistories.pop();
16 }
```

- In this example above, the `\_sellHistories[]` array is utilized in a for loop within the `\_removeOldSellHistories()` function.
- Whenever tokens are transferred to the UniswapV2Pair address, elements are added to this array. It is possible for any user to perform multiple transfers, thereby increasing the array size.
- Whenever `\_removeOldSellHistories()` function is called, the array is traversed, and certain elements are modified. If the array size becomes very large, the transaction's gas consumption will surpass the block gas limit, resulting in transaction failure.

## Mitigation Steps

- **Bound Loops:** Unbounded loops, especially when user or market data is involved, are susceptible to DoS attacks. Bounding loops can prevent this issue by limiting the size and complexity.

- **Handle External Calls:** Unforeseen behavior in external calls can disrupt contract execution. Implement robust mechanisms to handle unexpected outcomes from external calls to prevent contract failures.

## Voting/Governance Attacks

Governance vulnerabilities lead to scenarios like voting amplification and governance takeover. One common flaw involves the manipulation of voting mechanisms to amplify the number of votes a user can cast, allowing multiple votes using the same token amount.

Voting Amplification allows an attacker to leverage the same tokens for multiple votes, inflating the voting impact beyond the token count.

Consider a smart contract where users deposit funds and utilize these deposited tokens for voting. Withdrawals are restricted until the proposal's voting period concludes, aimed at preventing vote amplification.

The voting mechanism employed was to allow voting only when "the current timestamp is greater than or equal to the start time of the proposal and the current timestamp is less than or equal to the end time of the proposal," otherwise, it reverts.

```
1 function _vote() internal virtual {
2     // required code
3     if (
4         block.timestamp > proposal.endTimeOfVoting ||
5         block.timestamp < proposal.startTime ||
6         proposal.votingStatus != VotingStatus.CREATE
7     ) {
8         revert DAOImpossibleVote();
9     }
10 // required code and events
11 }
```

- Here, because it allows voting at `endTimeOfVoting`, that is on the last `block.timestamp`, a user can vote at the last timestamp and then withdraw the deposited amount 'n' from the contract. As shown in the `withdraw` function's code, it allows withdrawals when `proposal.endTimeOfVoting` is less than or equal to `block.timestamp`.
- In this case, `block.timestamp` and `proposal.endTimeOfVoting` are equal, so the user withdraws the deposited funds. Now, as `block.timestamp` and `proposal.endTimeOfVoting` are equal, and the user has retrieved their tokens, they can vote again with those tokens, as users are allowed to vote when the end time and timestamp are the same.
- In this way, the user utilized 'n' number of tokens to vote 'n\*2' times.
- The mentioned flaw can also lead to governance takeover in cases where a user can use a large number of tokens, for example, using flashloan to vote, and then using `withdraw()`, the user can withdraw the deposited amount and repay the flashloan and fee to the lending pool in only one transaction.

### ***Mitigation Strategies:***

- To address this vulnerability, disallow voting at ``endTimeOfVoting``. By doing so, users can only vote before the end time while retaining the ability to withdraw on or after the end time, thwarting the exploitation opportunity.

## **Access Control Vulnerabilities**

It allows unauthorized parties to gain privileges, potentially leading to unauthorized access, privilege escalation, or manipulation of contract state or funds.

Common Access Control Vulnerabilities include,

- Missing or weak access controls

```
1 //An example of missing access controls: attacks can burn any user's Tokens
2
3 Function burnToken(address _from,uint256 _assestId,uint256 _amount) external override {
4     _burn (_from, _assestId,_amount);
5 }
```

- **Improper Function Modifiers:** Function modifiers control access to functions in a smart contract. If modifiers are implemented incorrectly or omitted, unauthorized users may execute sensitive functions.
- **Incorrectly Scoped Access Controls:** Improper scoping of access controls can result in unintended access or manipulation of contract state. This issue arises when contract variables or functions are not explicitly marked as private or internal.
- **Dependency on External Contracts:** Smart contracts might depend on external contracts for specific functionality or access controls. Inadequately securing or auditing these dependencies can introduce vulnerabilities.

### ***Mitigation Steps***

- Apply the principle of least privilege, ensuring that each user or contract is granted the minimum privileges necessary to perform their intended actions.
- **Implement Role-Based Access Controls (RBAC):** Define and enforce roles and permissions within the contract using RBAC to help restrict access based on predefined roles and responsibilities.
- Utilize function modifiers to encapsulate access control logic and apply it consistently across relevant functions.
- **Be Mindful of Public Functions:** Exercise caution when exposing functions that can modify critical contract state or funds.
- **Conduct Comprehensive Security Audits** of smart contracts, including reviewing access control mechanisms, to identify and address vulnerabilities before deployment.

# Centralization in Smart Contracts

Despite the decentralized promise, certain dApps grant excessive control to a few entities, risking fund theft or censorship. Centralization stems from contract owner privileges, poor private key management, and founders' malicious tactics.

In one audited contract by QuillAudits, the contract owner had control over users' deposit, reinvestment, and fund withdrawal permissions. Additionally, the owner could expel users who had already deposited funds. Using the OpenZeppelin Ownable and Pausable contracts, the owner governed most user activities. Subsequently, investors in this contract fell victim to a rug pull. Below is an example code snippet depicting some of the owner's privileged functions.

```
1  function leaveProject(address[] calldata account, bool[] calldata _left) external onlyOwner {
2      require(account.length == _left.length, "Non-matching length");
3      for (uint256 i; i < account.length; i++) {
4          left[account[i]] = _left[i];
5      }
6  }
7
8  function deinitialize() external onlyOwner {
9      _pause();
10 }
11
12 function initialize() external onlyOwner {
13     _unpause();
14 }
15
16 function ProjectPool(uint256 amount) external onlyOwner {
17     payable(_msgSender()).transfer(amount);
18 }
```

## ***Mitigation Strategies:***

- Instituting decision-making processes scrutinized by DAO members.
- Employing multi-sig wallets to strengthen private key management despite occasional attacks due to poorly managed hot addresses.
- Deploying timelock contracts to delay transactions enabling users to withdraw funds in case of potential malicious activities



# **7** | **Web3 Development Resource Hub**

# Web3 Development Resource Hub



*"Web3 is for the people, and people deserve user-friendly, secure solutions. Security is the key to unlocking mass adoption of blockchain technology."*

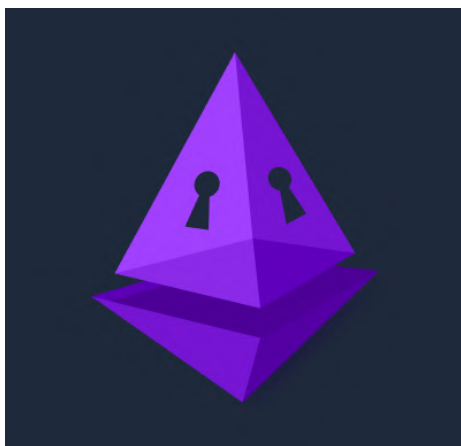
~ Yaron Velner, Co-Founder of Kyber Network



## We have collected the Top developments in Web3 security in 2023

Name:

**Solodit**



*Project Description:*

List of all disclosed bugs at one place

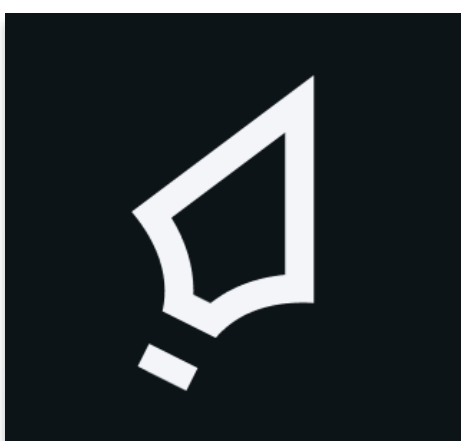
*Project type:* Platform

*Link:*

<https://solodit.xyz/>

Name:

**Spearbit Armory**



*Project Description:*

All available education material from Spearbit

*Project type:* Repo

*Link:*

<https://github.com/spearbit/armory>

Name:

## Audit Wizard



*Project Description:*

Online ide with inegrated tools essential for auditing

*Project type:* Platform

*Link:*

<https://www.auditwizard.io/>

Name:

## Awesome on chain Investigation - By OfficerCIA



*Project Description:*

List of all tool for on chain investigation

*Project type:* Repo

*Link:*

<https://github.com/OffcierCia/On-Chain-Investigations-Tools-List>

Name:

## HackedWalletRecovery



*Project Description:*

Free service that specializes in recovering funds from wallets that have fallen into the hands of hackers.

*Project type:* Tool

*Link:*

<https://hackedwalletrecovery.com/>

[View All Developments](#)



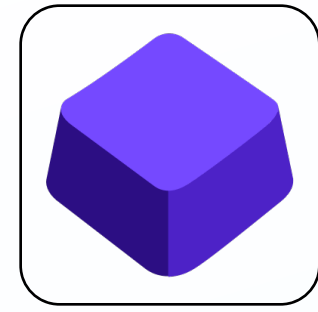
# Our Partners



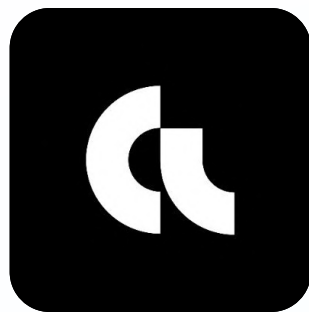
RareSkills



Blocksec



c4



Chainlight



ChainRisk



Curta



Moledao

# Our Collaborators



**Johnnytime**  
Founder-  
GingerSec



**Shanefan**  
Founder -  
Curta



**Smarty**  
Web3  
Security  
Researcher



**Sun Huang**  
CISO - XREX,  
Founder -  
DeFiHackLabs



**Chrisdior**  
Co-Founder -  
CDSecurity\_



**gmhacker**  
Smart Contract  
Security at  
Immunefi



**Chirag Agrawal**  
Co-Founder  
Web3Sec.News



**Wayne**  
Technical writer at  
Chainlight



**Juno**  
ChainLight lead



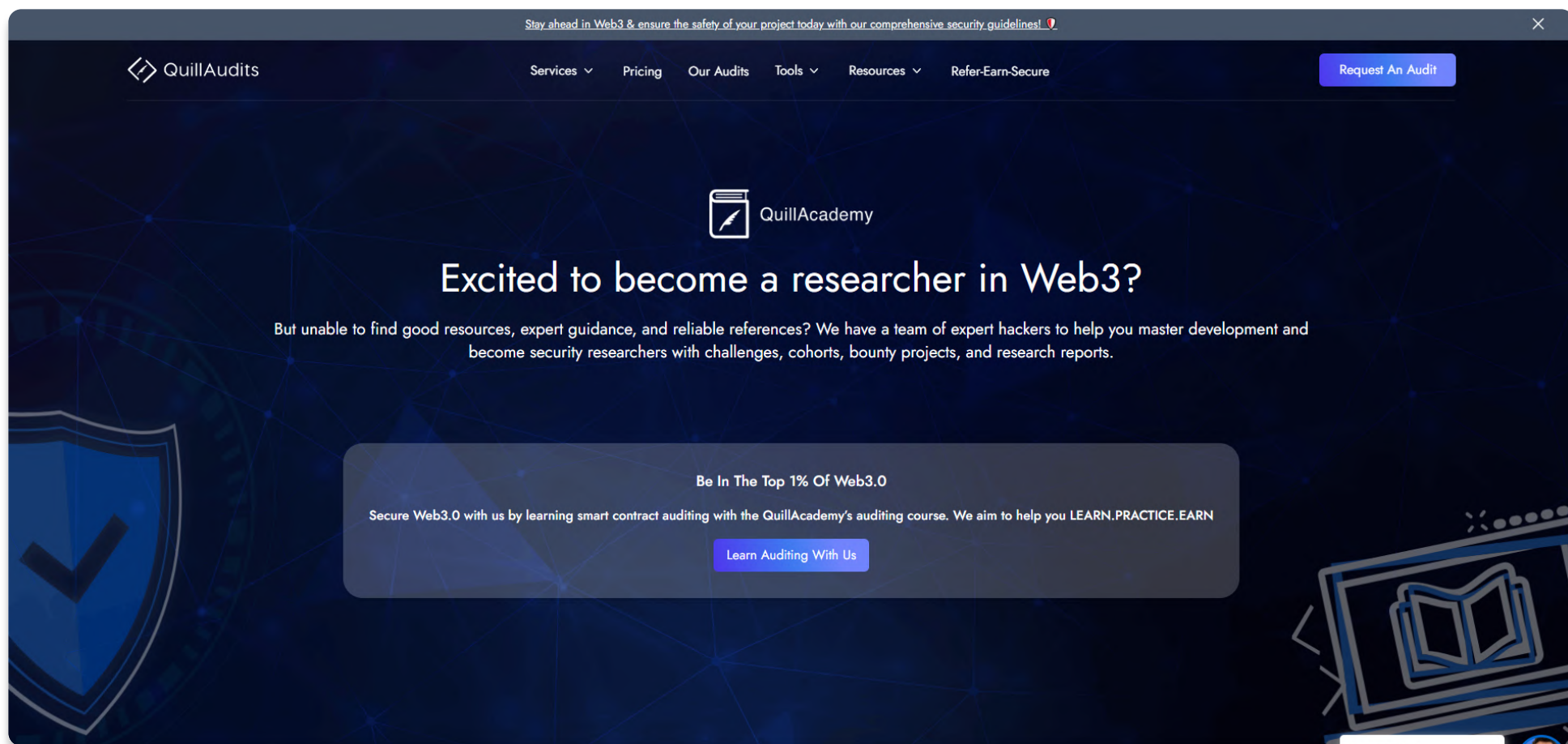
**c4lvin**  
Web3 Research  
Analyst at Chainlight



**Brian**  
Product Marketing  
Manager

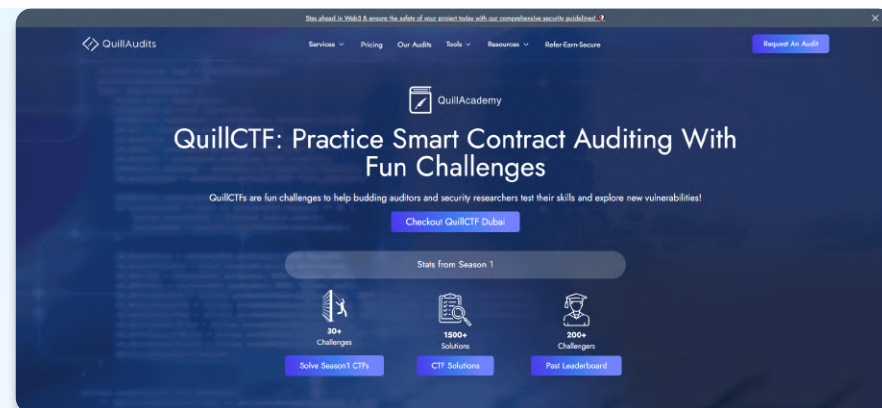


# Explore Our QuillAcademy

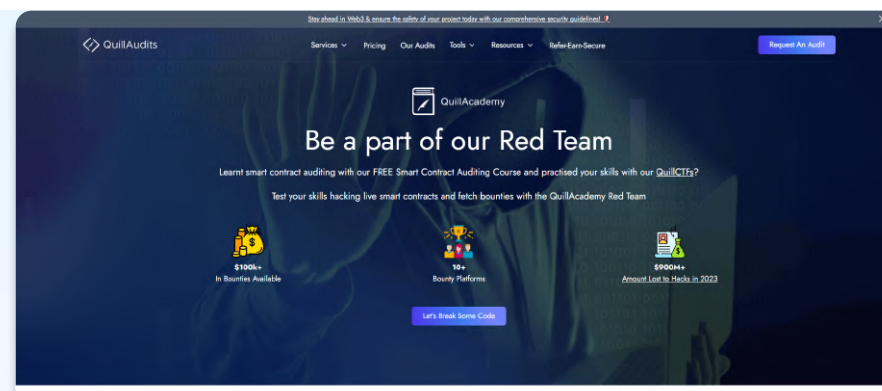


Explore

## Checkout Initiatives under QuillAcademy



Practice your Auditing Skills



Join our Team of Hackers



Build the best Web3 Security tools



**8**

**DeFi Economic  
Risks**

# DeFi Economic Risks

DeFi, that wild west of finance where your money can moon rocket to Mars or vanish into a black hole faster than a greased whistle, is booming. With over \$40 billion riding the rollercoaster, no surprise risks abound. We've all heard the horror stories of rug pulls, hacks galore, and even flash loan fiascos, but there's a lurking shadow in the alleyway we tend to forget: Economic Attacks.

Economic Risks in DeFi are not centered around faulty code or stolen keys; rather, they involve manipulating the intricate workings of the DeFi ecosystem for personal gain. Economic risks in DeFi arise from exploiting the economic design of protocols. Attackers mess with the economic design of DeFi protocols to create unfair imbalances and make profits, often without directly breaking the code.

In contrast to technical risks, which can be mitigated through coding and system enhancements, economic risks revolve around manipulating markets or incentive structures over time, potentially resulting in financial losses for the protocol and its users.

## Technical Risk vs Economic Risks

In decentralized finance (DeFi), risks fall into two main categories: Technical Risks involve exploiting code vulnerabilities, such as smart contract bugs, while Economic Risks focus on manipulating the economic design of protocols. For a comprehensive analysis, refer to the table below.

# Technical Risk vs Economic Risks



## Technical Risks

Exploits vulnerabilities in code or infrastructure

Smart contract bugs, private key lapses, etc.

Atomic exploits: Single action guarantees profit

Instantaneous, no time gap involved

Reentrancy attacks, rug pulls

Code audits, secure smart contracts, safeguards

## Aspect

**Nature of Exploits**

**Origins**

**Exploit Type**

**Temporal Nature**

**Examples**

**Mitigation Solutions**



## Economic Risks

Manipulates the economic design of DeFi protocols

Exploits in economic models, market structures, and incentives

Non-atomic exploits: Multiple actions, no direct control over steps

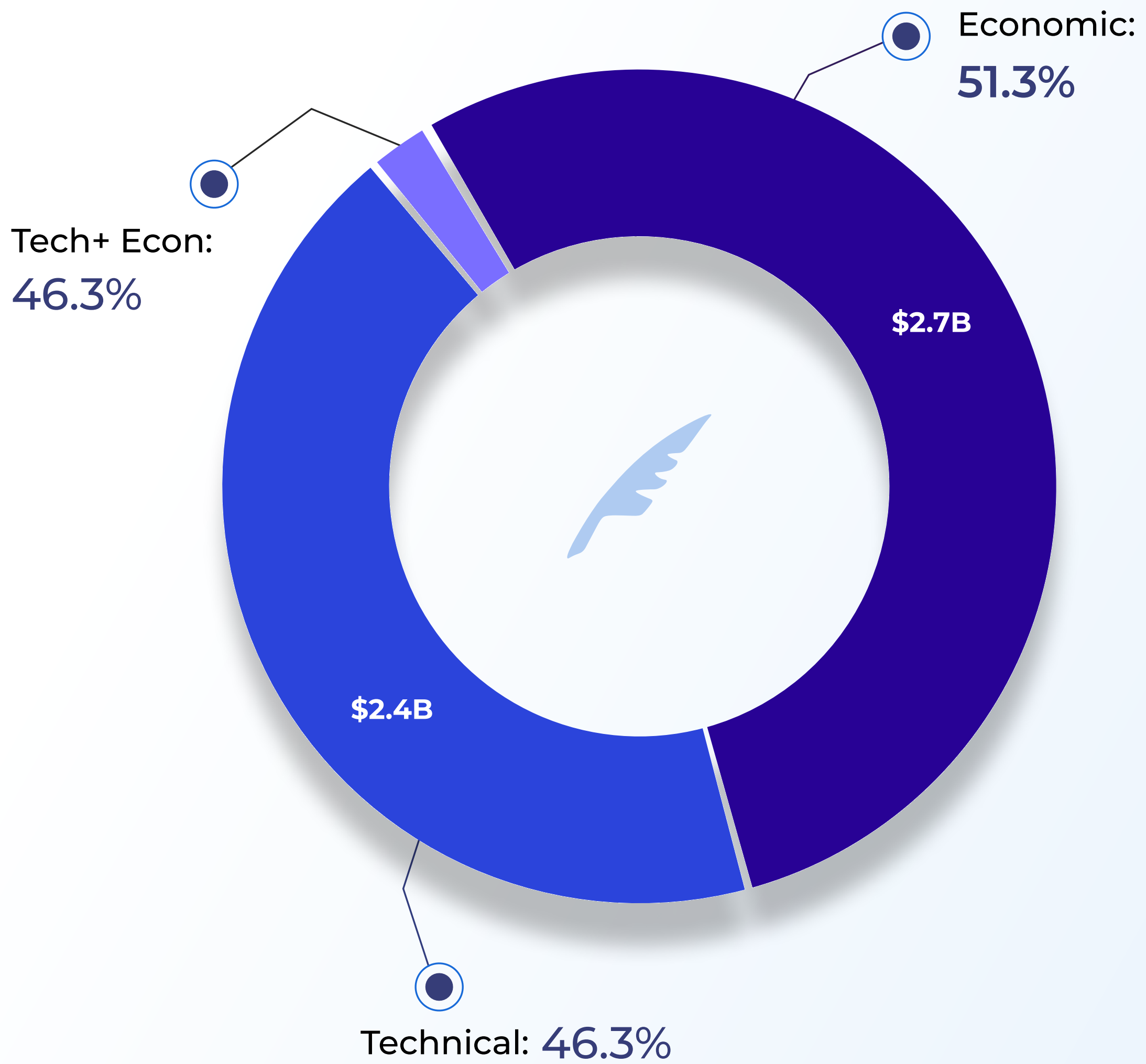
Gradual, occurs over time

Market manipulation, exploiting economic models

Economic models, improved protocol incentives

# DeFi Loss Distribution by Risk

While technical factors are often the root cause of most DeFi attacks, it's noteworthy that a larger amount of money has been lost due to economic risks.



Source: *A Primer on DeFi Risks*

# Why is Economic Security Important?

While technical security is a fundamental requirement, economic security gains significance when supported by robust technical measures. Economic security focuses on the economic design of DeFi protocols, requiring the development of models for market equilibria and resilient incentive structures, rigorously tested against potential exploits. This comprehensive process enables developers to gain a deeper understanding of how decisions related to security, governance, and consensus mechanisms impact network activity and asset value.

Prioritizing economic security empowers DeFi protocols to achieve optimal capital efficiency while minimizing risk exposure for users. This proactive approach sets the stage for a more resilient and sustainable DeFi ecosystem, fostering increased confidence and adoption.

## Case Studies:



### KyberSwap Exploit

- **Date:** November 23, 2023
- **Amount Lost:** **\$54.7 Million**
- **Chain:** Ethereum, Optimism, Polygon, Arbitrum, Avalanche & Base
- **Attack Technique:** Price Manipulation due to miscalculation of Tick Value



## Attack Steps:

- **Flash Loan Acquisition:** The attacker borrowed a large sum of wstETH using a flash loan.
- **Strategic Price Manipulation:** A portion of the loan was strategically swapped into the pool, causing a significant decrease in the wstETH/ETH price. This targeted an area with zero existing liquidity to exploit a specific curve within the system.
- **Creation of Fake Liquidity:** At the manipulated price point, the attacker created artificial liquidity using the remaining loan amount.
- **Deceptive Swaps:** Through a series of carefully executed swaps within the manipulated price range, the attacker tricked the pool into believing they possessed more liquidity than they actually did.
- **Profit Extraction and Maximization:** By capitalizing on the artificially inflated liquidity, the attacker was able to withdraw more wstETH than initially deposited, generating substantial profit.

The attacker repeated this exploit across multiple blockchain networks, maximizing their overall gains totaling approximately **\$46 million**.

# Case Studies:



## Jimbo Protocol Exploit

- **Date:** May 28, 2023
- **Amount Lost:** **\$7.5 million in ETH**
- **Chain:** Arbitrum
- **Attack Technique:** Protocol-specific price manipulation

### Attack Steps:

- **Flash Loan:** The attacker borrowed a large sum of ETH (around 10,000) using a flash loan.
- **Price Manipulation:** This borrowed ETH was strategically used to purchase JIMBO tokens at significantly inflated prices, setting them above the actual market value.
- **Rebalance Mechanism Trigger:** By controlling the price of JIMBO, the attacker triggered a rebalancing mechanism within the protocol. This mechanism automatically adjusts the distribution of assets within the liquidity pool.
- **Exploiting the Rebalance:** The attacker strategically interacted with the rebalance mechanism, causing it to move ETH into price bins below the artificially inflated JIMBO price. This created an opportunity to purchase JIMBO at a significantly lower price, repeating the process multiple times.
- **Profit Extraction:** The attacker accumulated a substantial amount of JIMBO tokens through this repeated manipulation. These tokens were then swapped back for ETH, allowing the attacker to repay the flash loan and retain the remaining ETH as profit, totaling approximately 7.5 million.

# Economic Risk Mitigation Strategies?

Economic security is a critical aspect of DeFi, ensuring protocols are financially resilient against exploits and user losses. Unlike traditional security threats, economic attacks are often non-atomic, meaning they unfold over time, making them harder to detect and mitigate.

Therefore, achieving economic security requires a multi-pronged approach:

- **Robust Economic Modeling:** Rigorously assessing the viability of collateral types, liquidity pools, and incentive structures is essential. This minimizes vulnerabilities that attackers could exploit for financial gain.
- **Comprehensive Risk Assessment Protocols:** Establishing clear metrics and frameworks for monitoring key economic indicators, such as loan-to-value (LTV) ratios and interest rates, allows for early detection and mitigation of potential risks.
- **Continuous Monitoring and Adjustment:** Economic environments are dynamic, so constant monitoring and adaptation of the protocol's financial model is crucial. This could involve adjusting parameters like interest rates or collateral requirements based on market conditions.

Economic risk is a growing concern in the DeFi space, one that requires a different kind of vigilance and defense. It's not just about securing code but about understanding and safeguarding the very fabric of the financial system we're building. By acknowledging this threat, embracing sound economic modeling, and fostering a vigilant community, we can ensure that DeFi remains a path to financial innovation, not a playground for manipulators.

## QuillAudits— Leading Web3 Security Firm

Owning something on the internet has emerged as a core new concept. We see Web3 as the next internet iteration with decentralisation at its core.

But businesses are skeptical about BUILDING in Web3 due to underlying security concerns.

We believe businesses can BUILD and GROW in Web3 with a product that people want— and that can be done "SECURELY". And we make it possible with our extensive security practices, an ecosystem of Web3 security tools, and a community of security professionals.

### Smart Contract Auditing



- Ethereum Audit
- Polygon(Matic) Audit
- Binance Smart Chain Audit
- Solana Audit
- NEAR Audit
- Algorand Audit

### Blockchain Penetration Testing



Penetration testing is an important step following security audits for web3 projects as it provides a safe and comprehensive attack simulation to expose the most intricate flaws in crypto exchanges, wallets, and DApps.

### Due Diligence



When using DeFi protocols, users put their funds at risk of being rugged(stolen) and should take appropriate precautions to mitigate protocol risks.

- DeFi Due Diligence
- NFT Due Diligence
- Rug Pull Due Diligence

### KYC Verification



Web3's decentralisation and anonymity provided many advantages, but they also made it easier for those with bad intentions to use these capabilities for their own gain. However, by using KYC as a factor in building confidence between projects and users, we can address this issue.

### Most Promising Use-Cases of Web3

DAOs

Decentralized Finance (DeFi)

Privacy and Digital Infrastructure

Blockchain Games

Metaverse

Creator Economy

### Top Web3 Threats

Cryptojacking

Smart contract logic hacks

Ice phishing

Data manipulation in Dapps

Data confidentiality

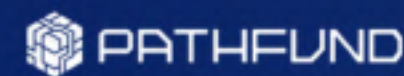
NFT exploits

#### Clients




And 1000 more

#### Partners





And 40 more

### Follow us on Social Media

 /QuillAudits

 t.me/quillaudits\_official

 /quillaudits

 /quillaudits



9



**QuillAudit's  
2024 Security  
Outlook**

# QuillAudits Security Solutions

## QuillAI



In 2023, the Web3 industry faced immense losses due to major hacks and security breaches, totalling over \$1.7 billion. The increasing complexity and scale of DeFi projects heightened the demand for more advanced smart contract security tools.

QuillAI operates on a sophisticated infrastructure powered by machine learning algorithms specifically crafted for web3 security. It employs an intricate analysis mechanism that scrutinizes smart contracts, swiftly detecting vulnerabilities and potential security gaps within the code.

Moreover, its user-friendly interface ensures seamless integration into developers' workflows and auditing processes to strengthen the security posture of smart contracts within the Web3 ecosystem.

### ***Solution QuillAI offers Web3 Security***

- **For Developers:** Integrated into the DevSecOps toolkit, QuillAI assists developers in identifying and mitigating security vulnerabilities during the coding phase. This integration ensures the creation of robust and secure smart contracts from the outset, elevating code quality and reducing post-deployment risks.

- **For Auditors:** QuillAI complements auditors' skills by providing advanced analytical capabilities. It accelerates the identification of potential vulnerabilities, offering comprehensive insights that optimize the auditing process and enhance the thoroughness and accuracy of audits.
- **For Non-technical stakeholders:** By providing clear explanations and highlighting potential risks, QuillAI makes complex smart contracts more accessible and understandable to non-technical users.

**Shield Your Web3 Security Efforts with QuillAI!**

## QuillCheck



*“In Web3, transparency and security go hand in hand. Users and investors need assurance that their assets are safe, and that starts with rigorous security measures.”*

~ Ryan Selkis, CEO of Messari



The decentralized finance (DeFi) space faces rising vulnerability to scams like rugpulls and honeypot schemes, impacting investors financially. These scams have surged from 5,000 rugpull incidents in 2020 to over 117,629 by 2022, adversely affecting nearly two million investors. The financial losses incurred have severely impacted investor confidence in the DeFi market.

### ***How is QuillCheck quick in addressing challenges?***

QuillCheck tackles this alarming trend by providing comprehensive due diligence and risk assessment of tokens within the DeFi market. With a focus on restoring and maintaining trust, it aims to shield investors from fraudulent activities prevalent in the market.

The specialized due diligence tool is designed for Ethereum, BSC, and Polygon blockchains. It evaluates token code and market conditions, generating a user-friendly risk score for a broad range of users.

With over 50,000 tokens scanned, QuillCheck has detected 10,000+ honeypots in real-time, employing features like Scam Token Detector and Transaction Simulator. Over 100+ subscribers have experienced its value, acknowledging its efficacy in safeguarding users from fraudulent activities.



**Check out QuillCheck for swift due diligence  
and real-time risk assessment of threats!**



# QuillAudits



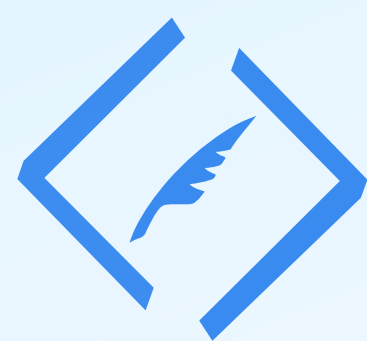
“

*“The third iteration of the internet is here to fulfil the diversifying consumer needs and technological advances. Innovations in Web3 are accelerating. But these solutions are accessible to investors and fraudsters alike.*

*We seek to secure & protect emerging Web3 ventures with our security solutions and professional expertise.”*

~ Preetam Rao, CEO of QuillAudits

”

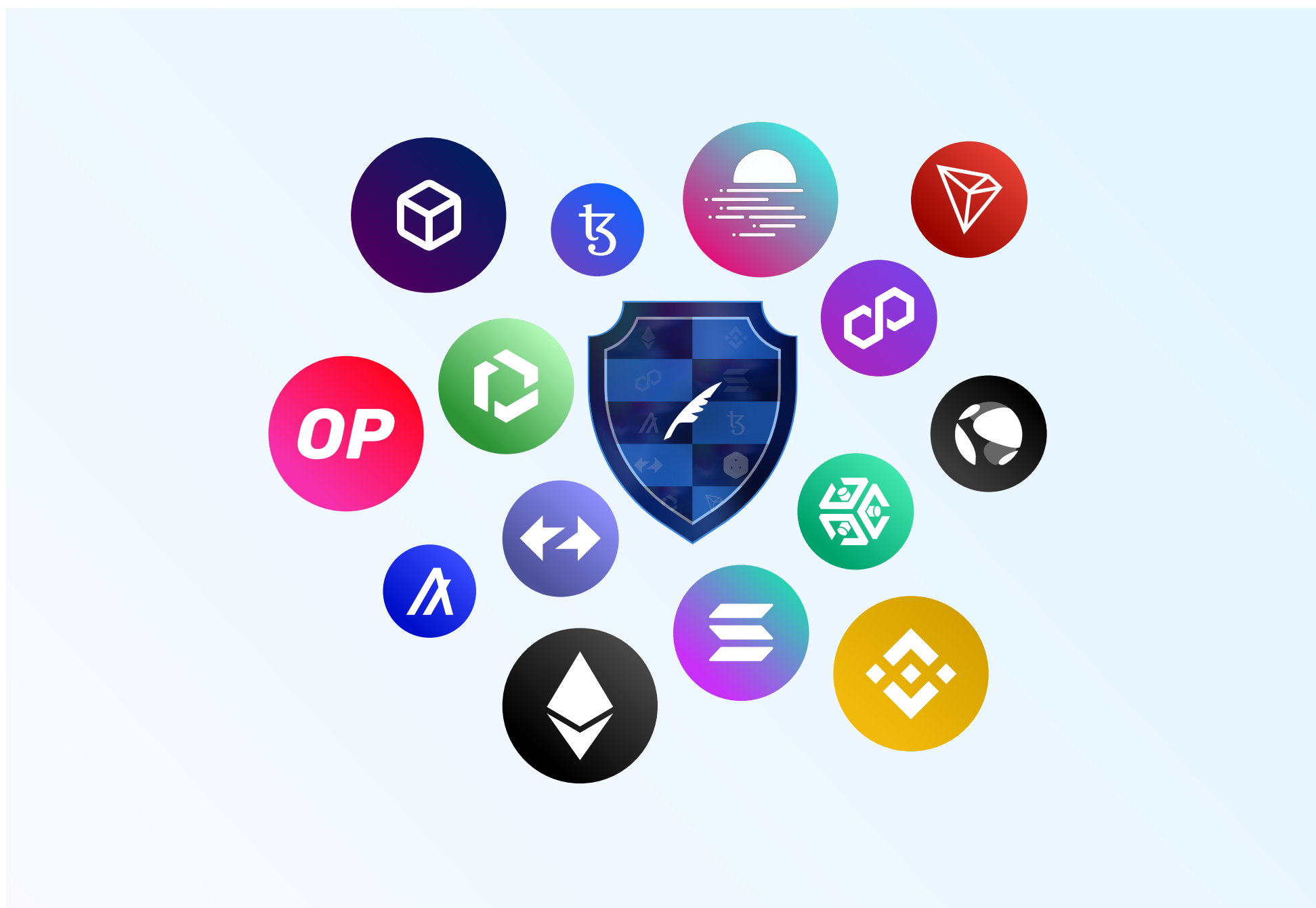


# QuillAudits

**QuillAudits** has safeguarded over 850 projects and a staggering \$35 billion in assets from potential loopholes and vulnerabilities lurking within smart contract codes so far.

We've achieved significant milestones in 2023 too, securing over 300 projects and auditing over 400,000 lines of code. As a result, we safeguarded over \$10 billion in digital assets, solidifying our commitment to raising secure Web3.

Our expertise in meticulous code auditing ensures the projects are well-guarded against any potential hacking attempts or vulnerabilities. From Ethereum to BSC, Polygon to dApp, QuillAudits employs a dedicated team that scrutinizes and secures smart contracts, ensuring they are resilient, safe and sound. Our services span across a multitude of blockchains, such as:



- Ethereum Audit
- Polygon Audit
- BSC Audit
- Solana Audit
- NEAR Audit
- Algorand Audit
- Wallet Audit
- ZkSync Audit
- Starknet Audit
- dApp Audit
- Sui Audit
- Tezos Audit
- Polkadot Audit
- L1 Audit
- Fantom Audit
- Hyperledger Fabric Audit

*Secure your Web3 journey today with insights from our expert team! Get in touch for any Web3 Security Assistance.*

# QuillRedTeam (QRT)



*"Blockchain and Web3 technologies promise a new era of trust. But trust without security is meaningless. Security is the bedrock upon which the decentralized future is built."*

~ Charles Hoskinson, CEO of IOHK



The inception of QuillRedTeam (QRT) stemmed from the pressing need to address the subpar security standards prevalent in the industry. Accessibility to such critical services was limited due to cost concerns, leading to a misconception that audits guaranteed safety without meeting necessary security standards.

QRT emerged as a proactive solution to elevate web3 project security while ensuring affordability.



## ***How does QRT solve the challenge of poor audit standards?***

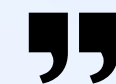
QRT comprises a team of 100+ white hat hackers, independent security experts, and specialized researchers dedicated to amplifying project security. Having secured over 50+ projects, the team acts as vigilant guardians, offering continuous support at every step of the way to protect projects from potential security threats.

QRT seamlessly integrates with QuillAudits, directing client audits through the RedTeam as a preemptive measure. This strategic integration ensures thorough security evaluations for projects, leveraging additional expertise to enhance overall security.



***"Security breaches can erode trust in Web3 projects and the entire blockchain industry. As stewards of this technology, we must prioritize security at every level."***

**~ Dan Morehead, CEO of Pantera Capital**



In the forthcoming year, QuillRedTeam aims to expand its influence within the industry. Its commitment to proactive security assessments and integration within QuillAudits reflects its dedication to rectifying prevailing security gaps and setting a new standard for Web3 project security.

***Get tailor-made audits suitable for your project's complexity and budget allocations through our QRT Services!***

**Check our QRT Services!**

## QuillMonitor

Data scarcity, user awareness, trust, and the necessity to learn from past mistakes pose significant challenges in safeguarding digital assets from hacks and vulnerabilities.

QuillMonitor aims to exactly make this information more accessible and affordable for all stakeholders while effectively addressing these industry-wide challenges.

### ***How QuillMonitor effectively tackle the challenge?***

Harnessing QuillAudits' extensive experience in project auditing, the tool provides real-time insights, historical data, and security alerts, effectively countering the growing challenges within the Web3 domain.

Housing an extensive database of past hacks, the tool aids users in identifying patterns and trends. It includes over 25,000 detailed post-mortems of major hacks, providing insights for effective awareness and remediation strategies.



**QuillMonitor**

The continuously updated database spans hacks since 2020, ensuring users are informed about the evolving threat landscape

Users can subscribe to QuillMonitor and receive daily hack updates or set project-specific monitoring, getting alerts via email for relevant incidents.

It has monitored and reviewed over 1000+ hacks, totalling \$7.5B in losses to date. It showcases 25+ post-mortems of major exploits that amount to \$59.3 million. For these reasons, it has attracted over 1000+ subscribed users who find this tool indispensable in navigating the complex Web3 domain.

***Stay Ahead of Threats! Explore QuillMonitor's Dashboard for Real-Time Insights and Historical Data on Hacks Since 2020!***



[Check QuillMonitor](#)

# Academic Nexus

## QuillAcademy

QuillAcademy is a learning initiative established by QuillAudits to help Web 3.0 developers acquire skills in smart contract auditing and other Web 3.0 cybersecurity practices through fun and engaging activities.

Under the academy hood, we have several initiatives to heighten the knowledge of web3 among developers and Web3 enthusiasts. The list of such initiatives includes



## QuillAcademy

### ***QuillAcademy Audit Fellowship***

Every year, we award an auditing fellowship to 10 Web 3.0 developers. These developers are selected after multiple rounds of interviews and problem-solving. They then undergo rigorous training sessions conducted by top-notch auditors, ending the fellowship with immense skills and knowledge of different auditing techniques. Our first fellowship had 2000+ applicants, where we chose 10 fellows and trained them to be competent auditors.

### ***QuillCTF***

We publish a new Capture the Flag challenge every week. All participants must audit the code we share to identify and rectify vulnerabilities. We have launched 30 challenges to date for the first season and received over 2000+ submissions, with 200+ contestants making it to the leaderboard.

# Hashing Bits

It is one of the highest-rated Web 3.0 security newsletters, sharing weekly insights into different hacks and the latest information about events, research, and tools. We help the community stay up-to-date with all the latest cybersecurity techniques and the best tips for avoiding exploitation.

*Subscribe to Hashing Bits for Weekly Updates on Web 3.0 Security, Hacks, Events, Research, and Expert Tips!*



Hashing Bits is a Web3.0 Cybersecurity focussed weekly newsletter. We share the latest hacks, research, and tools, as well as career opportunities. Subscribe now and don't fall behind!

<https://quillaudits.substack.com/>

Subscribe





SCAN



### Contact Us

[quillaudits.com](https://quillaudits.com)

[audits@quillhash.com](mailto:audits@quillhash.com)

Follow Us:

