

ChainLight

WEB3 HACK POSTMORTEM 2024

Ver 1.0



Website
<https://chainlight.io>



X/Twitter
[@ChainLight_io](https://twitter.com/ChainLight_io)

TABLE OF CONTENTS

DISCLAIMER.....	7
WRITER'S COMMENT.....	8
RESEARCH GOALS.....	9
RESEARCH SCOPE.....	9
1. SECURITY INCIDENTS.....	10
Scale.....	10
Root Cause of Security Incidents.....	10
Categorization of Security Incidents.....	10
Damage per Incident Type.....	12
Underlying Chains of Security Incidents.....	14
2. SECURITY AUDITS.....	16
3. POST-INCIDENT RESPONSES.....	18
Postmortem Publishments.....	18
User Compensation.....	19
Post-Incident Security Audits.....	20
4. STOLEN FUNDS.....	21
Bounty Negotiations.....	21
The Path of the Exploited Assets.....	22
Bridges.....	22
Destination.....	23
Duration.....	24
5. STANDARDIZATION OF SECURITY.....	25
6. INDIVIDUAL CASE ANALYSIS.....	27
1. Orbit Bridge (Jan 1).....	28
2. Radiant Capital (Jan 2).....	30
3. Gamma Strategies (Jan 4).....	33
4. CoinsPaid (Jan 6).....	35



5. Narwhal (Jan 6).....	36
6. MangoFarmSOL (Jan 6).....	37
7. Wise Lending (Jan 12).....	38
8. Hector Network (Jan 15).....	40
9. Socket.Tech (Jan 16).....	41
10. Manta Network (Jan 18).....	43
11. Concentric Finance (Jan 22).....	44
12. Ondo Finance (Jan 22).....	46
13. Gamee (Jan 23).....	47
14. Nebula Revelation (Jan 25).....	48
15. Somesing (Jan 27).....	49
16. Goledo Finance (Jan 29).....	51
17. Abracadabra.money (Jan 30).....	53
18. Chris Larsen, Ripple Co-founder (Jan 30).....	56
19. PlayDapp (Feb 9).....	57
20. Miner ERCX (Feb 14).....	59
21. Duelbits (Feb 14).....	61
22. Particle Trade (Feb 15).....	62
23. xPet (Feb 16).....	63
24. FixedFloat (Feb 18).....	65
25. DeezNuts (Feb 21).....	66
26. Jeffery Zirlin, Ronin Co-founder (Feb 23).....	67
27. Bitforex (Feb 23).....	68
28. Blueberry Protocol (Feb 23).....	69
29. RiskOnBlast (Feb 25).....	71
30. SenecaUSD (Feb 29).....	72
31. Shido Network (Feb 29).....	74
32. WOOFi (Mar 5).....	75
33. OrdiZK (Mar 5).....	77
34. TGBS Token (Mar 6).....	78
35. Humanized AI (Mar 6).....	79
36. Unizen (Mar 9).....	80
37. Juice Bot (Mar 9).....	82
38. Blastoff (Mar 11).....	83
39. Polyhedra Network (Mar 13).....	85
40. Cloud AI (Mar 13).....	86
41. MOBOX (Mar 14).....	87
42. Mozaic Finance (Mar 14).....	88
43. NFP (Mar 15).....	90
44. Wilder World (Mar 16).....	92
45. Charlotte Fang, Milady Co-founder (Mar 17).....	93
46. TICKER Token (Mar 18).....	94



47. Paraswap (Mar 20).....	95
48. Dolomite Exchange (Mar 21).....	97
49. AirDAO (Mar 22).....	99
50. Super Sushi Samurai (SSS) (Mar 22).....	100
51. Ark Token (Mar 24).....	102
52. Curio Ecosystem (Mar 24).....	104
53. ZongZi Token (Mar 25).....	105
54. Munchables (Mar 27).....	106
55. LENX Finance (Mar 27).....	108
56. Prisma Finance (Mar 28).....	109
57. Lava Lending (Mar 29).....	111
58. Solareum (Mar 30).....	113
59. Nibiru (Mar 31).....	114
60. OpenLeverage (Apr 1).....	116
61. FixedFloat(2) (Apr 1).....	118
62. ATM Token (Apr 2).....	119
63. AvoLend Finance (Apr 3).....	120
64. Condom Token (Apr 4).....	122
65. Arkham (Apr 5).....	123
66. xBlast (Apr 9).....	124
67. Pac Finance (Apr 11).....	126
68. Zest Protocol (Apr 12).....	128
69. Sumer.Money (Apr 12).....	131
70. Grand Base (Apr 15).....	133
71. Mars Token (Apr 16).....	135
72. Hedgey Finance (Apr 19).....	136
73. ZKasino (Apr 20).....	138
74. Z123 Token (Apr 22).....	141
75. Magpie Protocol (Apr 23).....	142
76. YIEDL (Apr 24).....	144
77. SaitaChain xBridge (Apr 24).....	145
78. IO.NET (Apr 25).....	146
79. FENGSHOU Token (Apr 26).....	147
80. Pike Finance (Apr 26, Apr 30).....	148
81. Ember Sword NFT (Apr 28).....	150
82. Rain Exchange (Apr 29).....	151
83. Perpy Finance (May 6).....	152
84. OSN Token (May 6).....	154
85. GNUS (May 6).....	155
86. Bloom (May 9).....	157
87. Galaxy Fox Token (May 10).....	159
88. Tsuru (May 10).....	160



89. Predy Finance (May 14).....	162
90. ALEX & XLink Bridge (May 15).....	164
91. BlockTower Capital (May 15).....	166
92. Sonne Finance (May 15).....	167
93. Pump.fun (May 17).....	169
94. TCH Token (May 17).....	170
95. Gala (May 20).....	171
96. YESorNO Token (May 22).....	172
97. TonUP (May 23).....	173
98. Normie Token (May 26).....	174
99. Orion Protocol (May 28).....	176
100. MetaDragon (May 29).....	177
101. DMM Bitcoin (May 31).....	178
102. Velocore (Jun 2).....	179
103. Lykke (Jun 4).....	181
104. Gemholic (Jun 8).....	182
105. YYS Token (Jun 9).....	184
106. Loopring (Jun 9).....	185
107. UwU Lend (Jun 10, Jun 13).....	186
108. JokInTheBox (Jun 11).....	187
109. Bazaar (Jun 11).....	188
110. NFTPerp (Jun 14).....	190
111. Holograph (Jun 14).....	191
112. Dyson (Jun 17).....	193
113. Farcana (Jun 19).....	194
114. CertiK <> Kraken (Jun 20).....	196
115. BtcTurk (Jun 22).....	198
116. CoinStats (Jun 22).....	200
117. Sportsbet (Jun 23).....	202
118. Shentu OpenBounty (Jun 26).....	204
119. MintRisesPrices Token (Jul 2).....	205
120. Bittensor (Jul 2).....	206
121. DefiPlaza (Jul 5).....	208
122. LW Token (Jul 8).....	210
123. Wasabi Wallet (Jul 10).....	211
124. Dough Finance (Jul 12).....	212
125. Minterest (Jul 15).....	213
126. LI.FI (Jul 16).....	215
127. WazirX (Jul 18).....	217
128. RHO Markets (Jul 19).....	219
129. DeltaPrime (Jul 23).....	221
130. Spectra (Jul 23).....	223



131. MonoSwap (Jul 25).....	224
132. Casper Network (Jul 28).....	226
133. Compound Governance (Jul 28).....	227
134. Anzen Finance (Jul 30).....	228
135. Terra (Jul 31).....	229
136. Convergence Finance (Aug 1).....	231
137. Satoshi Token (Aug 3).....	233
138. OCF Token (Aug 5).....	235
139. Ronin Bridge (Aug 6).....	236
140. OMPX Token (Aug 7).....	237
141. Nexera (Aug 7).....	238
142. iVest DAO (Aug 12).....	240
143. Vow (Aug 14).....	241
144. HFLH Token (Aug 23).....	243
145. Aave Peripheral Contract (Aug 28).....	244
146. Penpie (Sep 3).....	245
147. Pythia (Sep 3).....	247
148. DAI (Sep 4).....	249
149. CUT Token (Sep 10).....	250
150. Indodax (Sep 11).....	251
151. BaseBros Finance (Sep 13).....	253
152. OTSea (Sep 13).....	255
153. MintStakeShare (Sep 15).....	257
154. WXETA Token (Sep 16).....	258
155. DeltaPrime(2) (Sep 16).....	259
156. BingX (Sep 20).....	261
157. BananaGun (Sep 20).....	262
158. Shezmu (Sep 20).....	263
159. Bankroll Network (Sep 22).....	264
160. Inferno (Sep 24).....	266
161. ReHold (Sep 26).....	268
162. Onyx DAO (Sep 26).....	270
163. Truflation (Sep 26).....	272
164. Bedrock (Sep 26).....	273
165. FIRE Token (Oct 1).....	275
166. EGA Token (Oct 5).....	277
167. P719 (Oct 11).....	278
168. Radiant Capital(2) (Oct 16).....	279
169. IBX (Oct 18).....	281
170. Tapioca DAO (Oct 19).....	282
171. Cryptobottle (Oct 22).....	284
172. Ramses Exchange (Oct 24).....	286



173. U.S. Government (Oct 24).....	287
174. Unverified Contract (Oct 25).....	288
175. Essence Finance (Oct 26).....	289
176. M2 (Oct 31).....	290
177. Sunray Finance (Oct 31).....	291
178. Metawin (Nov 4).....	292
179. CowSwap (Nov 7).....	294
180. CoinPoker (Nov 8).....	295
181. DeltaPrime(3) (Nov 11).....	297
182. vETH Token (Nov 14).....	299
183. Thala (Nov 15).....	300
184. Polter Finance (Nov 18).....	302
185. BTB Token (Nov 18).....	304
186. DCF Token (Nov 25).....	305
187. Pump.Science (Nov 26).....	307
188. XT Exchange (Nov 28).....	308
189. Spectral (Dec 1).....	309
190. Clipper DEX (Dec 1).....	311
191. GAGAW Token (Dec 2).....	313
192. Vestra DAO (Dec 4).....	314
193. Clober (Dec 10).....	315
194. Haven Protocol (Dec 11).....	317
195. DCF Token (Dec 15).....	318
196. bnbs Token (Dec 15).....	319
197. LABUBU Token (Dec 17).....	321
198. GemPad (Dec 17).....	322
199. Moonwell Third-Party Vault (Dec 23).....	324
200. Nani Finance (Dec 23).....	325
201. BIZNESS Token (Dec 28).....	327
202. FEG (Dec 29).....	328



DISCLAIMER

This report is presented for informational purposes only and is not intended as legal, financial, or investment advice. The analysis and findings are based on publicly available data and our understanding of the events as of December 31, 2024, the last working day of 2024. We have made every effort to ensure the accuracy and reliability of the information provided, but we do not guarantee its accuracy, completeness, or relevance.

The report covers various web3 security incidents that occurred in 2024, including analysis of the underlying chains, exploited assets, attack vectors, summaries of the attacks, user compensation processes, audit histories, post-mortem reports, and the current status of the exploiters and exploited assets. The content is intended to provide a comprehensive overview of these incidents for educational and research purposes.

We recognize that the field of web3 security is constantly evolving, and new information may arise after this report's publication. Therefore, we encourage readers to conduct their own research and seek professional advice as needed.

This report does not intend to defame any organization, company, or individual. The mention of specific audit firms or projects serves only illustrative purposes and does not imply endorsement or criticism. We prioritize the confidentiality of sensitive information and have refrained from disclosing any proprietary or confidential data.

We disclaim any liability for errors, inaccuracies, or omissions in this report, as well as for any actions taken based on the information provided herein. The views and opinions expressed in this report are those of the authors and do not necessarily reflect the official policy or position of any associated organization or entity.

When we refer to "Audit History" in this report, it does not imply that the audit scope includes the attack vector of the exploit. It is intended to alert the community that partial audits do not guarantee the protocol's security.

We do not guarantee the security of all the links included in this report.

All consequences arising from the use of this report—including adoption, modification, or redistribution of its content—are the sole responsibility of the individual or entity utilizing it. We disclaim any and all liability for damages or losses arising from such usage.

Any reproduction, redistribution, or use of this report's content for commercial purposes is strictly prohibited without the express written consent of the author.



WRITER'S COMMENT



Thank you for taking interest and opening this report.

2024 was a year when projects that had been building through the long crypto winter finally got to taste the fruits of their labor. Plenty of projects that had long delayed their TGE announced their token launches, and the market was full of vitality. Typically, the Web3 market tends to show lower retail maturity during bull markets, but this year I felt that retail maturity has significantly improved compared to previous cycles. However, what's somewhat sad is that in the area of security, the knowledge and awareness level of the general public still hasn't caught up.

As I felt while writing last year's report, when security incidents occur, there's still a tendency to focus heavily on the scale of the incident, while quickly losing interest in the project's follow-up measures or the background of the incident. This tendency can only lead to negative feedback. This is likely why we still see bad cases such as losing large capital due to simple code mistakes, projects going dark and declaring sunset despite being potentially recoverable, or emphasizing project safety despite having audits with very low coverage.

This report is not just statistical data about incidents that occurred in 2024.

I've investigated over 200 cases to make it as easy as possible to understand from a general user's perspective: what criteria can be used to judge a "safe project," what positions projects take when incidents occur, how hackers' funds move, and what causes security incidents.

For builders/developers, we've included as many code references as possible to easily grasp real-world exploits. Many cases occur due to simple mistakes that are easily understood, and I believe if entry-level developers study this thoroughly, we can definitely reduce the frequency of low-level security incidents in 2025.

The one thing I hoped for while investigating over 200 cases for this report was that next year, we would have fewer incidents to investigate. Consistent attention to security incidents will lead more quickly to a future where everyone can "Thrive without Fear" in the Web3 ecosystem.

Thanks again for your interest in this report. There might be small mistakes, but I hope this research proves helpful.

2025. 01. 17.

[c4lvin](#), Research Analyst of ChainLight



RESEARCH GOALS

The purpose of this report is not just to introduce security incidents but to suggest how to handle future incidents. To combat the current rampant Web3 security incidents, the following efforts are necessary:

- Reviewing vulnerabilities to prevent the recurrence of vulnerabilities with similar patterns.
- Tracking the flow of funds from past incidents to secure financial control in future incidents.
- Conducting additional security audits and bug bounties to eliminate remaining vulnerabilities.

We aim to emphasize the importance of not only the immediate response to security incidents but also tracking the perpetrators' actions and the projects' own efforts afterward to prevent further occurrences.

Plus, ChainLight is currently operating [LUMOS](#); a historical security incident dashboard focused on the root cause analysis, non-security related details, user compensation and fund flow. We aimed to provide more detailed information in the report than LUMOS currently offers, including exploit transactions, the entire fund flow (not just the destination), updated code-level details, and the current state of each case as of January 1, 2025. These updated details will be incorporated into LUMOS soon.

RESEARCH SCOPE

In 2024, there were over 200 security incidents, including minor ones. Please note that simple phishing exploits were excluded due to the challenges associated with tracking criminal funds. Only the phishing incidents (suspectedly) related to project hot wallets are included. Additionally, rugpull cases due to simple market dumps are also excluded.

1. Exploits
2. Security Audits
3. Post-Incident Responses
4. Stolen Funds
5. Standardized Process on Security Incidents
6. Individual Case Analysis

1. SECURITY INCIDENTS

Scale

In 2024, we identified **204** security incidents with various root causes, occurring at an average interval of **1.78** days. The total damage amounted to at least **\$1.88 Billion**, with each project suffering an average loss of **\$9.57 Million**.

Root Cause of Security Incidents

The attack vectors of security incidents were predominantly hacking due to **contract vulnerabilities**, accounting for **120** of the incidents. This was followed by **57** incidents resulting from **control hijacking**, which means the control of project wallet or contracts had been compromised. There were also **15** cases of **rugpulls** (that were not just developer's market dumps), **4** cases of unfair movement of the **circulating supply**, and **1** case of **malicious governance proposal**. The others are the cases that are still under investigation or their root causes are not publicly disclosed.

Categorization of Security Incidents

In this report, we define the (sub)categories of the exploit as follows:

1. Control Hijacking Attack¹

The cases where the control of critical components, such as project wallets/API/critical contracts get compromised. These attacks can be categorized again as follows:

- **Social Engineering:** Compromise one's computing environment or social accounts by inducing the target to install/execute malicious software or links.
- **Private Key Leakage:** Unencrypted private key or mnemonic gets exposed to the exploiter. For example, the case where the developer mistakenly commits the source code with a plaintext private key in GitHub commits.
- **Authorization Management Failure:** The case where an ex-employee abuses the authorized access to the project that has not been removed.
- **Supply Chain Attack:** Exploits due to the vulnerabilities of third-party service or source code libraries.
- **Web2 Exploit:** The case where the exploits on project's web2 components lead to the loss of control for major components.

¹ There might be cases of using a zero-day vulnerability (browser, PDF viewer, etc.), but it is difficult to identify from a third-party perspective, so it has been excluded.



2. Contract Vulnerability

While the types of vulnerabilities found in actual smart contracts are too diverse to enumerate here, we have abstracted and categorized them as follows for the reader's understanding.

- **Lack of Access Control:** When some critical functions that should only be executed by privileged entities/contracts are set to public due to the absence or misimplementation of access control.
- **Improper Input Validation:** When the data given as user input allows unintended behavior due to the lack of validation. In most cases, this vulnerability allows exploiters to abuse the authorization that has been set to the target contract.
- **Misconfiguration:** When critical variables of the victim contract are misconfigured. This includes the cases when the contract is not initialized or decimals of certain tokens are mistakenly set to a wrong value.
- **Price Dependency:** When a price calculation of the token has dependency on a single/few pools, allowing the exploiter to manipulate the token price via a massive swap / flashloan.
- **Misimplemented Transfer Function:** When the transfer function of the ERC20 token has critical vulnerabilities, such as doubling balances via self-transfers.
- **Precision Loss:** When the rounding errors of some variables are inappropriately managed. This includes the price manipulation vulnerability of CompoundV2/AaveV2 forked projects.
- **Reentrancy:** When the reentrance of functions leads to an unintended behavior, due to the misimplemented sequence of function calls and updates on variables.
- **Storage Collision:** When upgraded contracts are misimplemented to overwrite & collide the previous contract's storage mappings. In these cases, variables of an upgraded contract may refer to the value that had been set in the deprecated contract.
- **Logic Bug:** When the business logic of the contract is misimplemented and allows unintended behavior. This includes the misimplementation of price calculation, the control flow of the contract itself, etc.
- **Backdoor:** When a backdoor function that has been implemented by insiders is executed. Since these cases can be significantly identified in the audit phase, we classified this case in contract vulnerabilities. In 2024, this category only includes one case; Munchables.

3. Rugpull

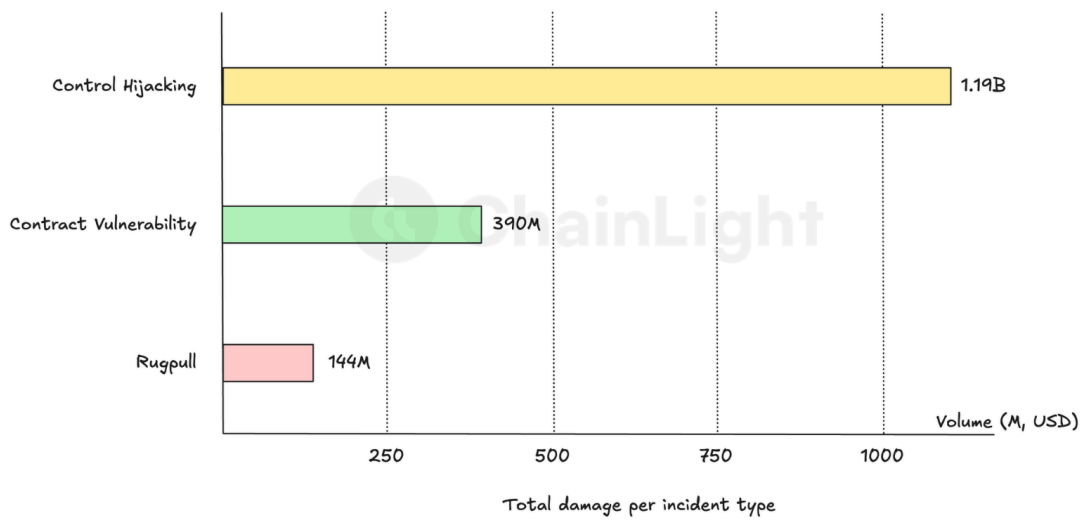
This year, we excluded the cases where a developer naively dumps his own liquidity to the market. As the memecoin market surged in 2024, too many dev dumps happened to be included in the report (imagine sorting all the cases of dev rugpulls in pump.fun). For this reason, only major / notable cases are included in the report.



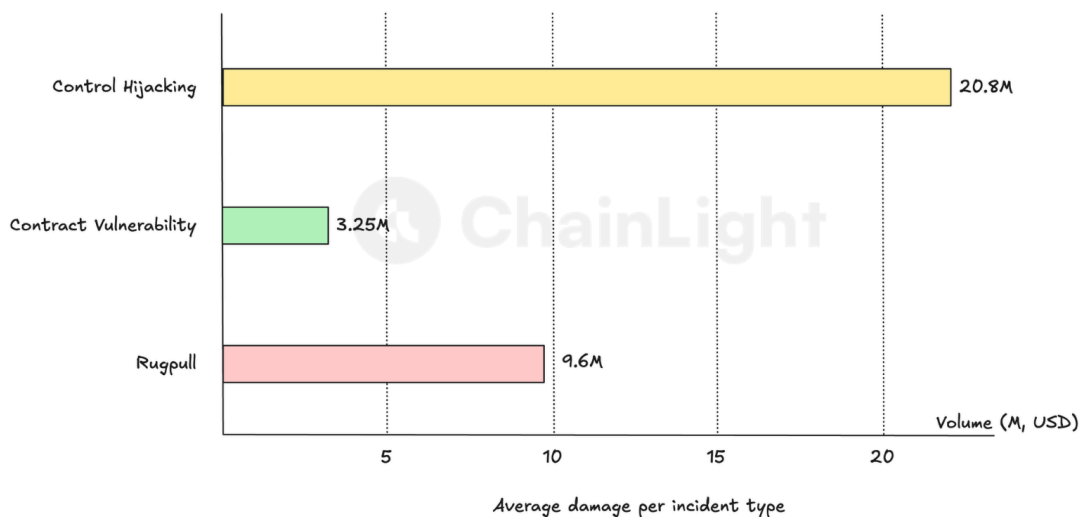
4. Circulating Supply

The cases where the project/foundation moves the substantial amount of tokens that have not been announced on their distribution plan. This is a pretty sensitive topic, and as far as we know the case that only our report provides. We would like to include these cases as there's still a lack of public awareness about the centralized behavior of projects/foundations, which is directly related to market/price manipulation and customer protection.

Damage per Incident Type

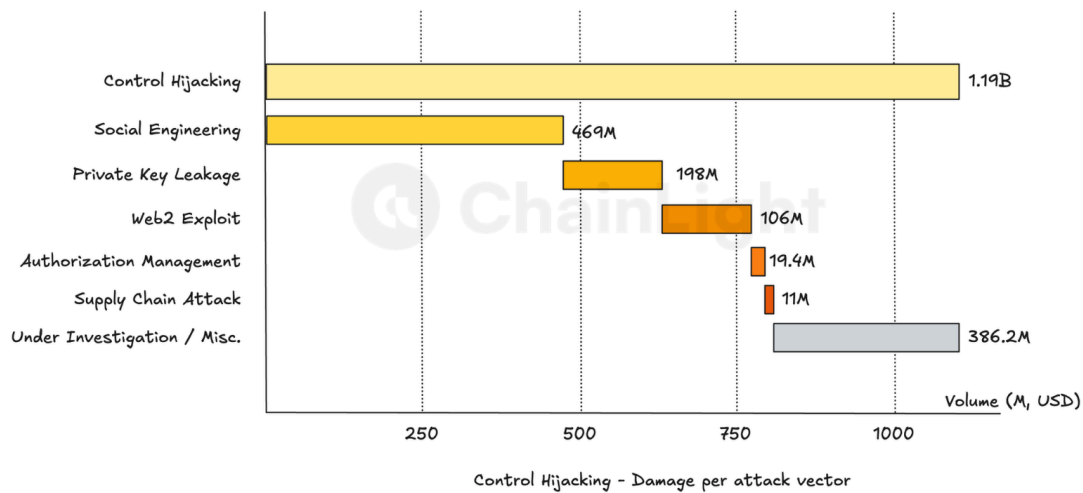


Control hijacking attacks had the highest total damage of \$1.19 billion. Contract vulnerability and rugpulls followed by \$390 million and \$144 million.



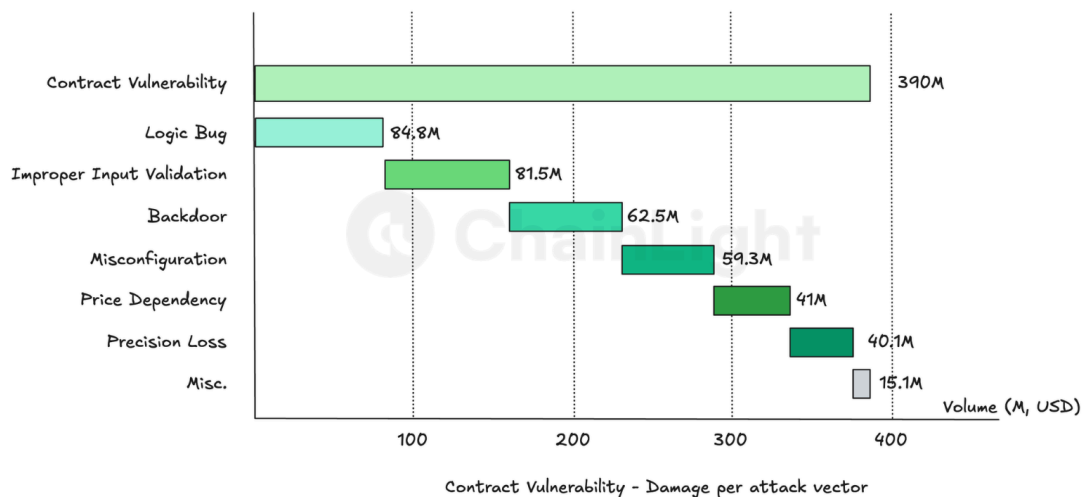
Notably, control hijacking incidents also had the largest average volume of \$20.8 million, while the rugpull incidents and contract exploits followed for \$9.6 million and \$3.25 million.

Control Hijacking



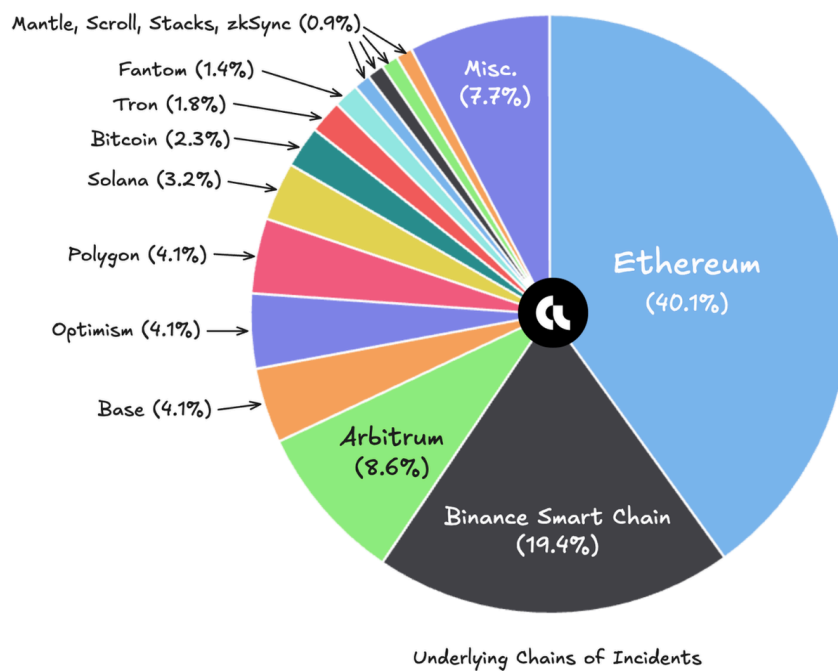
Among various attack vectors of control hijacking attacks, social engineering resulted in the most damage of \$469 million. Private key leakage, web2 Exploit, authorization management failure, supply chain attack followed for \$198 million, \$106 million, \$19.4 million, and \$11 million each. For some cases, such as Orbit Bridge and WazirX incidents, are categorized as `under investigation`.

Contract Vulnerability - 390M



Attack vectors of contract exploits were quite fairly distributed than control hijacking attacks. Logic bugs, improper input validation, misconfiguration, price dependency and precision loss accounted for \$84.8 million, \$81.5 million, \$59.3 million, \$41 million and \$40.1 million each. Notably, the backdoor incident filled \$62.5M for only one incident, which happened to Munchables on Mar 27 2024.

Underlying Chains of Security Incidents

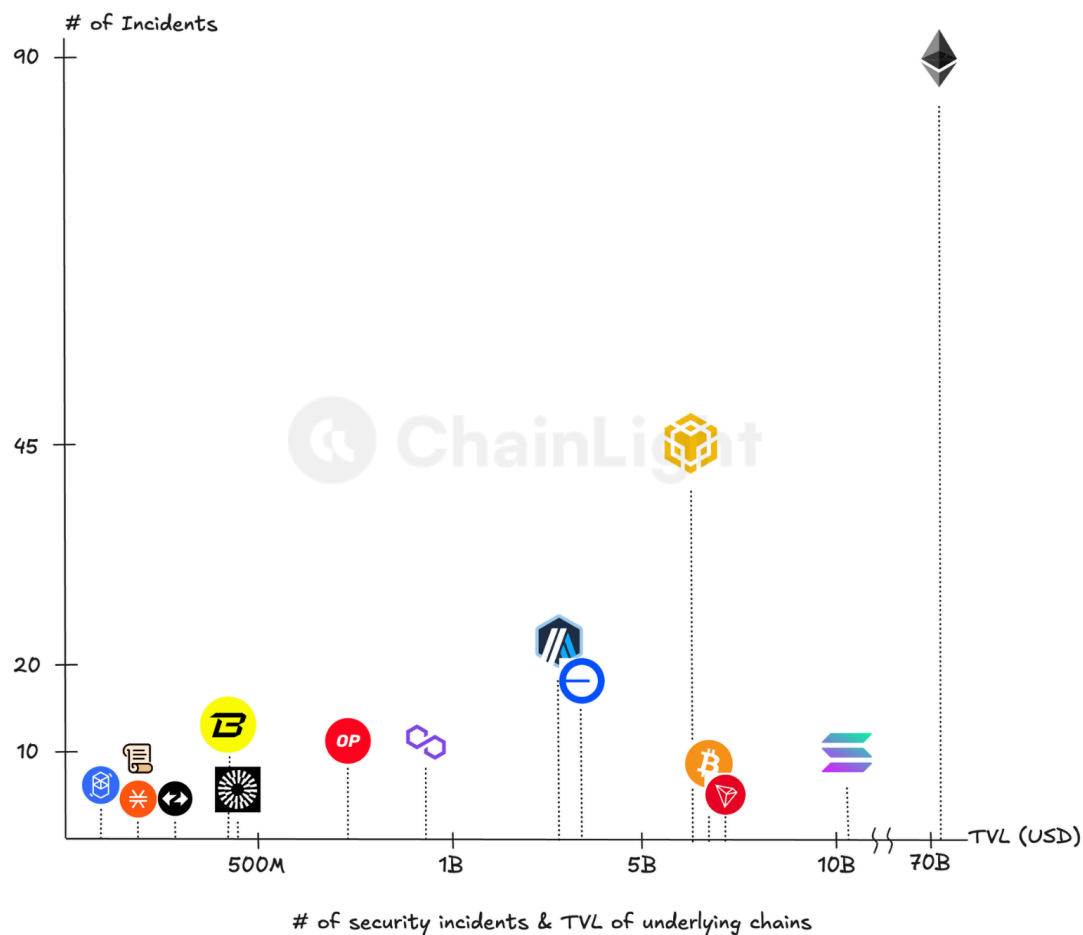


The frequency of the underlying chain for each incident was as follows:

- Ethereum: 89
- Binance Smart Chain (BSC): 43
- Arbitrum: 19
- Base: 9
- Optimism: 9
- Polygon: 9
- Solana: 7
- Bitcoin: 5
- Tron: 4
- Fantom: 3
- Mantle, Scroll, Stacks, zkSync: 2
- Aptos, Avalanche, Bitcoin Cash, Bittensor, Casper Network, Conflux, Klaytn, Linea, Litecoin, Manta, Monero, Nibiru, Ripple, Ronin, Terra, Ton, unknown: 1

The count is based on the chain where the exploited funds were initially located, not the chain that the project is based on. For example, when the project that operates on Solana but their vault on Ethereum takes an exploit, this counts for Ethereum.

The chart below illustrates the relationship between the Total Value Locked (TVL) and the number of incidents occurred at each underlying chain.



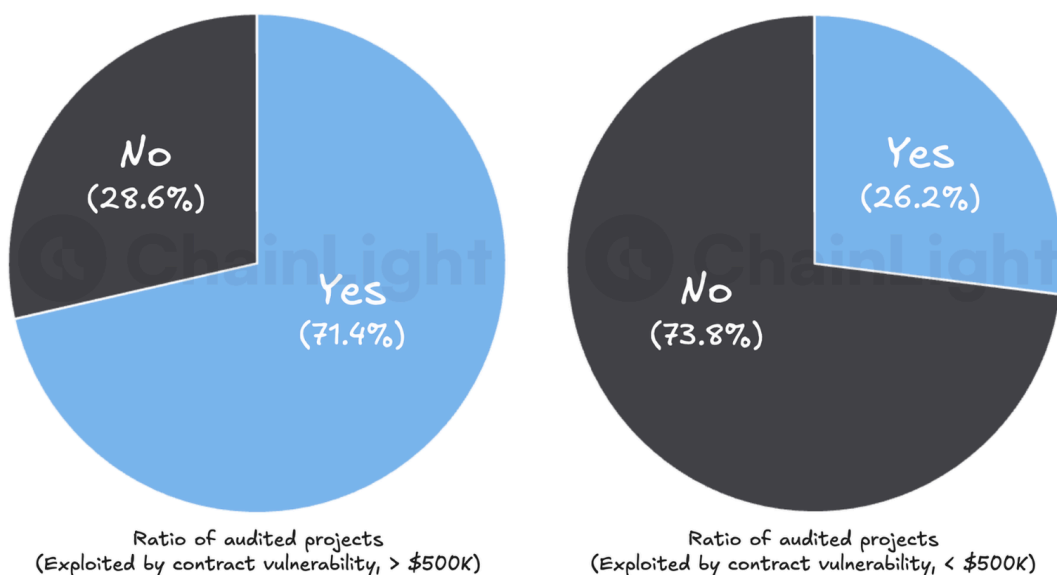
As shown in the chart, the number of security incidents generally showed a proportional trend with the TVL of chains (TVL provided by Defillama). Notably, Solana, Bitcoin, and Tron showed figures that deviated from this trend, which can be interpreted in various ways. Due to the characteristics of non-EVM chains, they might be less exposed to exploits from simple attack vectors, or conversely, it could be because their ecosystems are less active, resulting in a lower frequency of new projects/contracts being deployed. I will leave this interpretation up to the reader's judgment.

2. SECURITY AUDITS

Security audits have become a crucial criterion for Web3 community participants when investigating projects.

However, do you believe that putting your money in a project that has undergone audits equals "your money is safe"?

To answer this, let's take a look at the exploited projects due to the contract vulnerabilities if they had undergone any security audits.



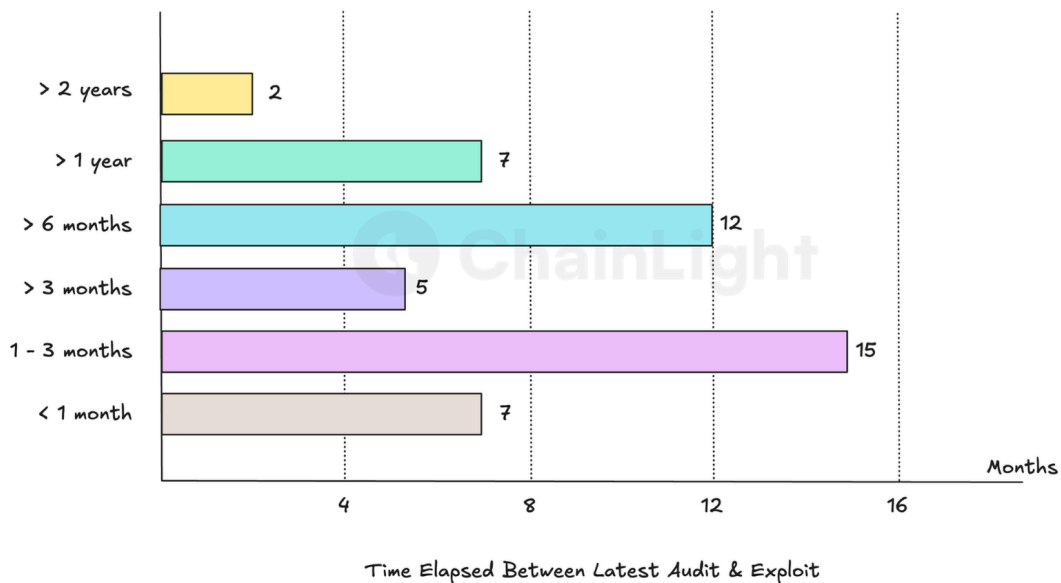
The results varied greatly depending on the scale of the incident. Generally, security incidents under \$500K mostly occurred in small-scale projects with very low TVL or less-known token contracts, as these typically haven't undergone security audits. In contrast, larger security incidents over \$500K mainly occurred in projects with an established user pool, and in these cases, over 70% of the projects were found to have undergone at least one security audit previously.

So why do contract hacks occur even when security audits have been conducted? While security researchers would likely know the reasons, let me explain for general readers:

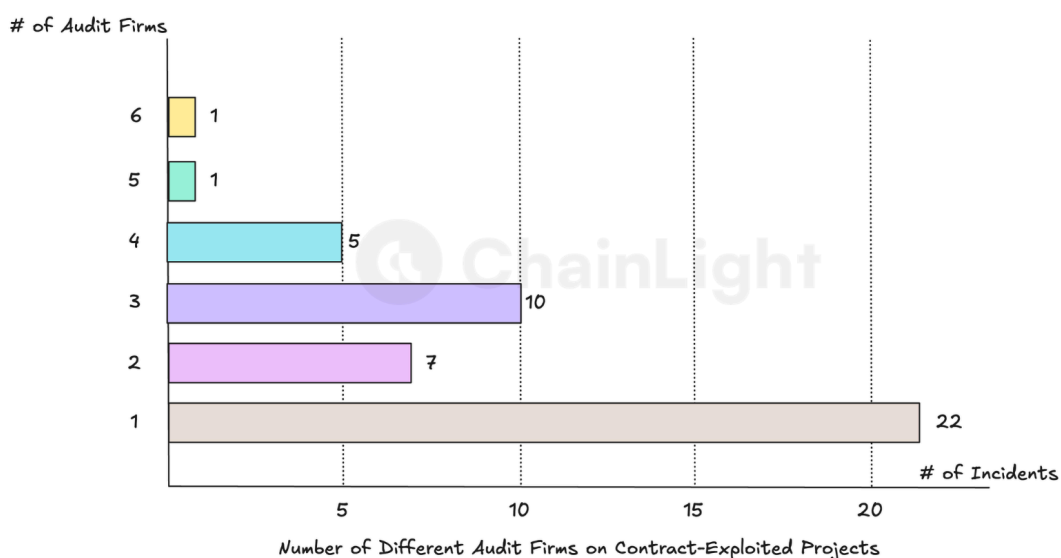
- **Low audit coverage:** Audits are performed on only a very small portion of the project.
- **Inconsistent security audits:** Despite upgrading major project contracts, no audits are performed on these changes.
- **Low-quality audits:** Exploiting retail users' low awareness of security firms, projects hire cheap and quick audit services and promote these to consumers.



We observed i) how many days had elapsed between the latest audit and the contract exploits, and ii) how many different audit firms conducted audits for the contract-exploited projects. The result was as follows. (The total number of two are different; we could not find the exact date of the audit for some cases)



About half of the exploited projects conducted audits for at least 6 months ago, which implies that the security audits had not been consistently conducted. Notably, 7 projects were exploited less than 1 month after the audit, which implies that the audit had not identified the existing vulnerabilities or the projects made an update for contracts that were out of scope of the audits.



Also, we have identified that more than half of the projects relied on a single audit firm. This actually is a problem, as this implies that every contract may have conducted an audit by few (2~3) researchers, only once in their lifetime.



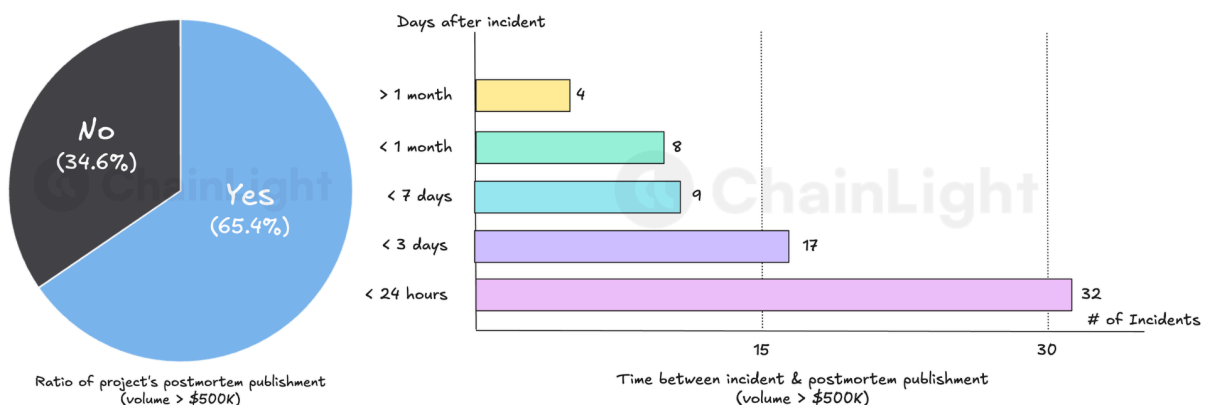
3. POST-INCIDENT RESPONSES

Post-incident responses are a good indicator of the maturity of the Web3 ecosystem as a whole. This is because there is generally little interest in a project's own post-incident responses. Possible post-incident measures by a project include:

- Publication of detailed post-mortem reports on the attacks.
- Direct financial compensation to users or restoration of project capital through the treasury.
- Additional security audits after the incident.

We tracked whether such measures were implemented in projects that experienced security incidents and compiled the statistics. Rugpull projects were excluded from this study as post-measures are impossible in their cases.

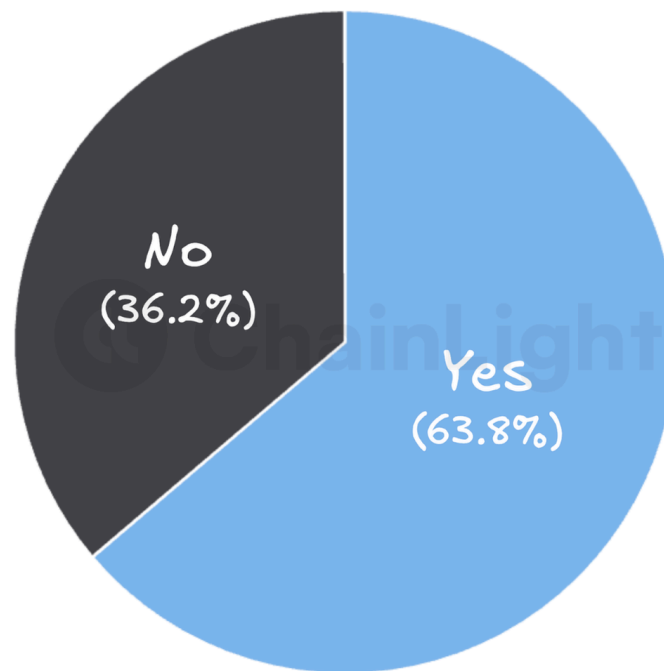
Postmortem Publications



Notably, 65.4% of the projects published their postmortem, which significantly increased from 51.8% of 2023. In addition, most of the projects have published the postmortem less than a week from the incident. Writing a post-mortem is a way to inform the community about the severity of the incident from the project's perspective and contribute to preventing similar incidents in the ecosystem.



User Compensation



Ratio of projects that reimbursed after the exploit
(volume > \$500K)

User compensation is one of the post-incident measures that projects should definitely implement. However, unfortunately, only 67 out of 105 cases (63.8%) clearly communicated and implemented user compensation, which is similar to the ratio of 2023 (61.4%). Some projects were completely bankrupt after the security incident and could not afford compensation, but efforts to maximize compensation were also observed.

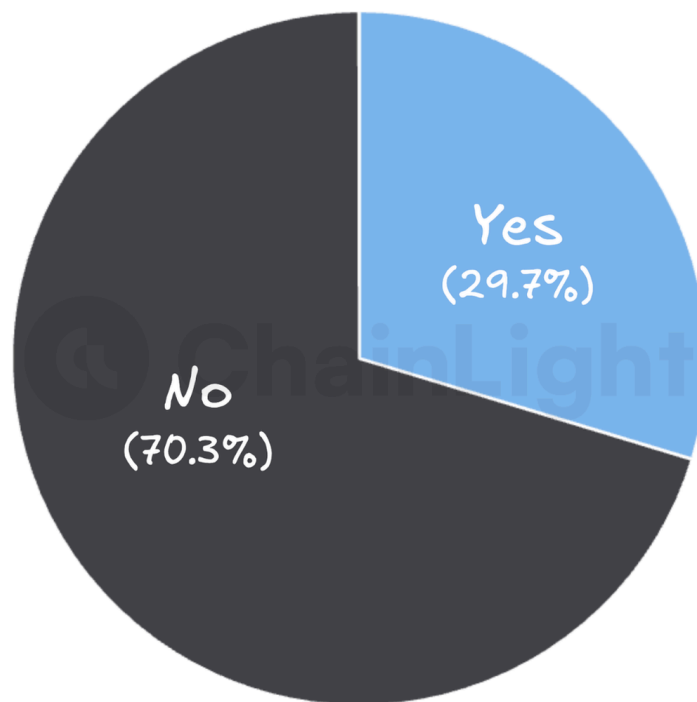
For example, ALEX on Stacks suffered a control hijacking attack of \$27.4 million worth of STX in May 2024. After the massive amount of exploit, the team has been continuously communicating with the community and preparing several treasury grant programs and reopening plans to compensate its users. The team announced an asset recovery plan and treasury grant proposal and decided the direction of the user compensation through their governance. The decision was well decentralized, since the plan to fairly distribute the remaining STX in the project vault to cover LP loss and user compensation was selected with 19% of voting power (among the other four options, to mint synthetic STX were voted for 10%, and over half amount which is suspected to be foundation's voted for abstain), which was quite fair. ([Ref.](#)) In Jun 2024, project's remaining assets were used to recover affected STX-paired pools, and the community decided to utilize 100% of project's future revenue for user compensation through the governance. Plus, the team developed a recovery dashboard so that users can track the revenue distribution, which was done on Sep 29 2024 and Dec 18 2024. ([Ref.](#)) The case of ALEX



shows that projects can continue to operate despite significant damage from incidents as long as there is willpower from the project and faith from the community.

Post-Incident Security Audits

If a project intends to continue operations after an incident, conducting additional security audits is also a necessary procedure. This is because the original security audit may not have included the range of the vulnerability or audited the modified codes. So, how many projects conducted security audits after incidents?



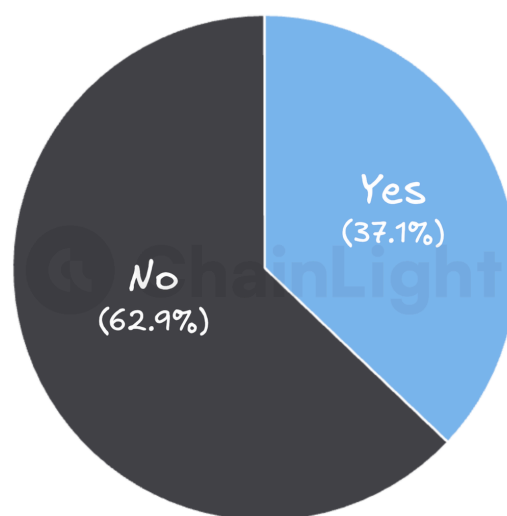
Ratio of projects that conducted audits after the exploit
(volume > \$500K)

Only 19 out of 64 cases received security audits after the incident, which is a bit similar ratio to that of 2023 (34.0%). Projects that did not undergo additional security audits either released new versions with only the affected parts fixed or completely halted the project.

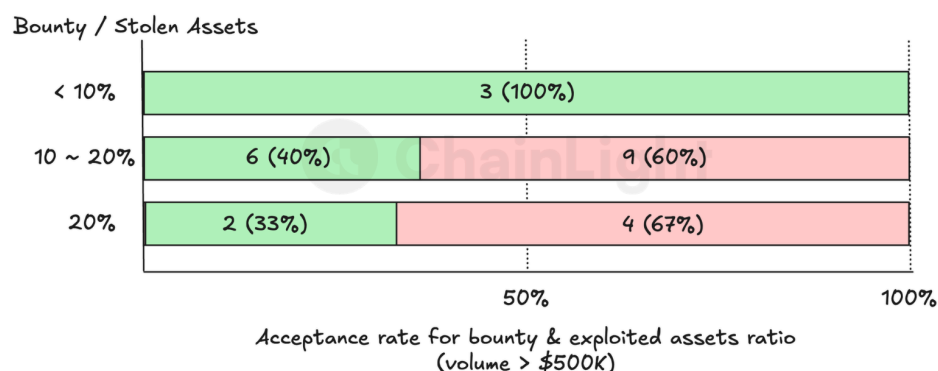
4. STOLEN FUNDS

Bounty Negotiations

In cases of contract hacking or control hijacking attacks, projects often offer a certain amount of bounty to exploiters. Most offer about 10% of the damaged amount, and there have been almost no cases of the bounty amount being negotiated. Exploiters who reject the bounty typically move the funds to their addresses using mixers. The statistics on bounty negotiation are as follows.



Ratio of project that offered a bounty to the exploiter
(volume > \$500K)



Acceptance rate for bounty & exploited assets ratio
(volume > \$500K)

Only 33 out of 89 projects tried to contact the exploiter for negotiation through public announcements or on-chain messages. Ironically, the exploiters refused the bounty offers of larger percentage to the exploited assets (~20%), which implies the confidence of the exploiters today to launder the exploited assets into their trusted destinations.

* There were some cases where the exact amount of bounty offers were unavailable.

The Path of the Exploited Assets

Bridges

Exploiters typically use bridges to move funds to Bitcoin-based chains or to Ethereum/BSC where Tornado Cash is located, as these facilitate money laundering. Bridges are also used to create complex routes that make transaction tracking difficult, making it harder for CEXs to control deposits of criminal funds. Below are the bridges used by attackers of the projects we investigated for moving funds.

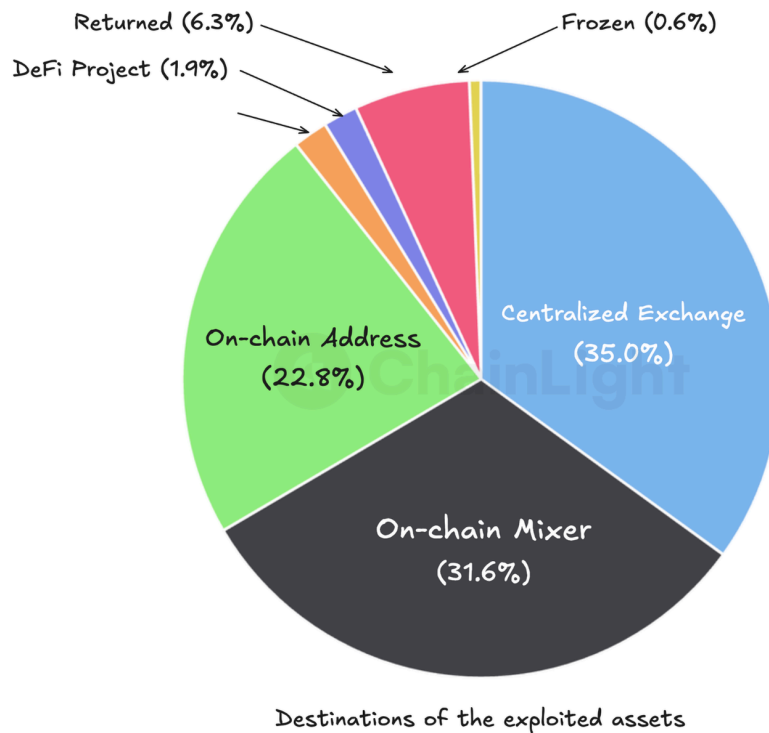
- LayerZero Stargate - 25
- Across Protocol - 16
- THORChain - 13
- Native Bridge - 11
 - Arbitrum 2, Blast 3, Optimism 1, Polygon 1, Ronin 1, Scroll 1, zkSync 1, IBC 1)
- Synapse - 9
- Axelar Squid - 7
- Orbiter Finance - 6
- deBridge - 4
- Hop Protocol - 4
- Rhino.Fi - 4
- XY Finance - 4
- Wormhole - 3
- Allbridge - 2
- Celer, Connex, DODO, Jumper Exchange, MaestroBots, Maya Protocol, OKX Web3 Proxy, Owlto, RangoBridge, Reservoir, SmartBridge, Socket.Tech - 1

THORChain was primarily used as a route between Bitcoin/Litecoin and EVM chains, while Allbridge, deBridge, and Wormhole were mainly used for transfers between EVM chains and Solana/Aptos. An interesting point was that exploiters predominantly used common bridging protocols where transactions are easier to interpret, while some bridging protocols that are more difficult to track were rarely used. It's also noteworthy that native bridges provided by Ethereum L2 chains (mostly L1 → L2 deposits) were relatively frequently used for fund transfers.



Destination

Readers may think that most of the exploited assets from security incidents are immediately laundered through Tornado Cash or other mixers. However, in real-world cases, the exploited assets have various destinations with different timelines. We observed the destination of every (traceable) security incident and the duration of the required time.



Notably, the number of cases where the exploited assets headed to centralized exchanges and on-chain mixers counted similarly (112 & 101). Plenty of cases left the exploited assets on on-chain accounts, waiting for the future laundry. The detailed destinations of the exploited assets are as follows:

Centralized Exchange

* Red → Non-KYC exchanges

- Binance: 25
- ChangeNOW: 13
- **eXch**: 12
- **FixedFloat**: 10
- Kucoin, Gate.io, MEXC, OKX: 5
- Bybit, HitBTC, HTX: 4
- (Routed via) Bridgers, Chainflip: 3
- Bitget, Defiway, **N.exchange**, SideShift: 2
- **Aeroswap**, Coinbase, LBank, Tap, **TradeOgre**, WhiteBIT: 1

On-chain Mixer

- Tornado Cash: 83
- Railgun: 17
- Jambler: 1

On-chain Address

- Ethereum: 36
- Binance Smart Chain: 8
- Bitcoin: 6
- Litecoin: 5
- Solana: 5
- Arbitrum: 4
- Base, Optimism, Polygon: 2
- Atom, Avalanche, Conflux, Kava: 1

DeFi Protocols

- Aave, Ether.fi, ListaDAO, Trader Joe, Stargate, UniswapV3: 1

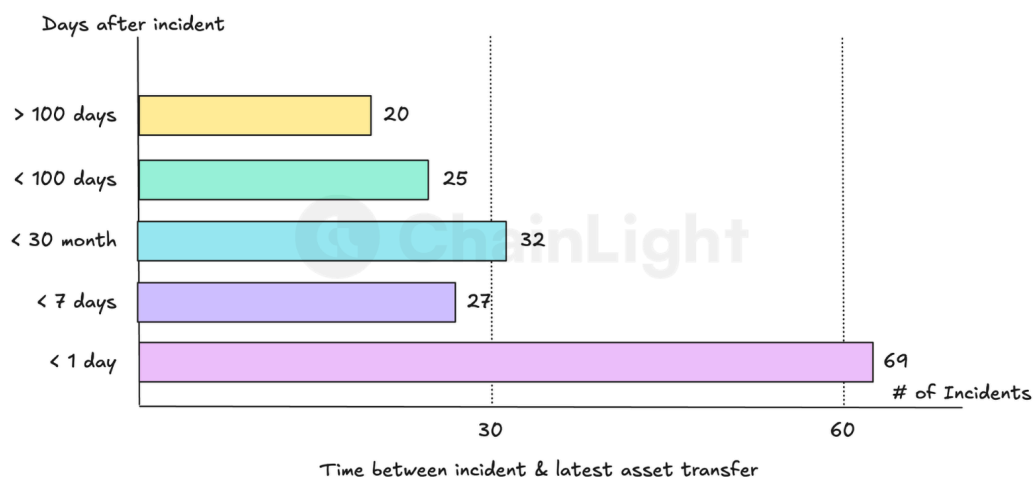
Wallet / Web2 Services

- CoinsPaid, Bitget Wallet, Huione Guarantee, Revolut, Stake.com, zkBOB: 1

Returned: 20, Frozen: 2

Duration

Each case of the exploit has a distinct time of the asset transfers, and most of the readers haven't met detailed statistics of it. Researching each exploit case, I have measured the period between the exploit and latest movement of the assets, and the result is as follows.



5. STANDARDIZATION OF SECURITY

Writing the report, we've thought it would be helpful for both projects and users to know how the web3 projects can maximize their security and promptly respond to future exploits.

These are principles that security researchers & [SEAL](#) (Security Alliance) have continuously emphasized, but I will take this opportunity to explain them once again through this report. We could classify them into two; before and after the incident.

While in Normal Operation

- **Bug Bounty Operation:** Security researchers often observe cases where bug bounty programs haven't been established even for projects that are more than 3 years old. However, bug bounty programs are one of the most important channels for receiving reports about vulnerabilities that weren't detected through audits and other means, and they are a mechanism that can prevent millions of dollars in damages with just tens of thousands of dollars. If you don't have the capacity to operate a bug bounty program internally, outsource it to bug bounty platforms like Immunefi. Also, provide prompt responses to vulnerability reports. Exploits are strictly a matter of time.
- **Continuous and Complete Audits:** Most contract exploits occur in contracts that haven't undergone security audits. While there are various reasons for this, ultimately, it stems from a low awareness of security. We sometimes see cases where projects focus on promoting their security by having a single audit from a famous audit firm right before launch, but then fail to conduct additional audits for subsequent updates. Furthermore, even when audits are performed, if they are very narrow in scope, the project can still become a target for attacks. This can actually end up diluting the effectiveness of the audit as it may reveal vulnerable code locations to attackers. The point we want to emphasize is that contracts can become targets of attacks due to even a single line of error, which is why all code must be audited. Additionally, to prevent potential mistakes in audits, multiple audit firms should review the same source code to ensure a relatively high level of security. In this context, competitive audits where tens of (often hundreds of) individual security researchers participate would be a great solution. Code4rena, Cantina, Sherlock, CodeHawks are representative ones.
- **SEAL Wargame:** SEAL Wargames helps protocol teams strengthen their security posture through tailored wargaming and incident response simulations. Over a 4-6 week process, they design a customized tabletop wargame followed by a live simulation, enabling teams to practice incident response, uncover vulnerabilities, and identify improvements to mitigate future risks. SEAL has previously conducted drills with Compound, Yearn, Aave, Optimism, Uniswap and many more protocols. Public resources are available on the [github](#). Teams can apply for a wargame through the form



on the Security Alliance website or can reach out directly to [Shield3](#), the wargames initiative lead.

- Some may ask the difference between the normal audit and wargame. The main difference is that wargames are not intended to identify new bugs in smart contracts, but to push the protocol in extreme situations of various conditions. Wargame covers an exploit scenario in whole perspective; including devs, legal, finance, governance and related entities.
- Currently, SEAL wargame prioritizes protocols with large impact on the wider ecosystem. Thus, teams that are relatively recently building should register themselves in a waiting list and wait for their turn. For these cases, SEAL provides the wargame framework on GitHub as public resources. (e.g. [drills toolkit](#)) Or the projects can work with SEAL's partner Shield3 directly.

When Exploit Happens

- **Contact SEAL 911:** SEAL 911 is an emergency hotline operated by Security Alliance, which helps the projects to get connected with the group of highly-trusted security researchers and investigators. There are several incidents that were resolved this year through SEAL 911's help; including Magpie, Radiant Capital, Thala, etc. ([Ref.](#)) SEAL also offers a whitehat safe harbor, the legal agreement between projects and whitehats to rescue the project assets during an active exploit.
- **Postmortem Publication:** Postmortem is a crucial means of providing transparency to users who were directly affected by an incident, allowing them to understand the root causes while also being informed of the follow-up action plans. Furthermore, by properly disclosing the causes of the incident, it serves to prevent other projects from experiencing similar issues in the future. After an incident occurs, the project team should publish on their public website as quickly as possible, including the technical root causes of the incident, its precise impact, and future plans for user compensation.
- **Security Enhancement:** For projects that can continue operations after an incident, there is a need to strengthen security to a higher level. The existing audit should be renewed when possible, and to ensure safety, the audit should be performed with the widest possible coverage. If feasible, implementing monitoring services such as Hypernative could be one potential solution.



6. INDIVIDUAL CASE ANALYSIS

In this section, we would like to share our analysis of 202 selected security incidents that happened in 2024. Not only the brief summary of the attack vector and scenario, we also investigated the compensation process, audit history, postmortems, and the detailed path of the exploited assets to their final destinations. We hope this in-depth analysis will help the crypto community and security researchers remember past incidents, and study real-world cases that led to the massive loss of numerous protocols.

Note1

The “pre-incident audit history” **does not** mean that the audit included the victim contracts and failed to find the vulnerabilities. This is rather to emphasize to non-security expert users that “the audit itself does not guarantee the security of the project.”

Audited projects often suffer security incidents due to contract vulnerabilities. As we mentioned earlier, there are several reasons, including **low audit coverage** and **Inconsistent security audits**. To prevent this, projects should be audited by multiple audit firms / get competitive audits (code4rena, cantina, sherlock, codehawks, ...) / ensure that entire production-level codes are audited.

Note2

Due to the lack of resources (research & writing is done by 1 person), the report does not include the trace of the assets after they were transferred to the crypto mixers / CEXs. For some cases, on-chain investigators may be able to provide more specific flow of the exploited assets.



1. Orbit Bridge (Jan 1)



Underlying Chain: Ethereum

Amount: \$81,000,000

Attack Vector: Control Hijacking - Misc.

Brief Summary

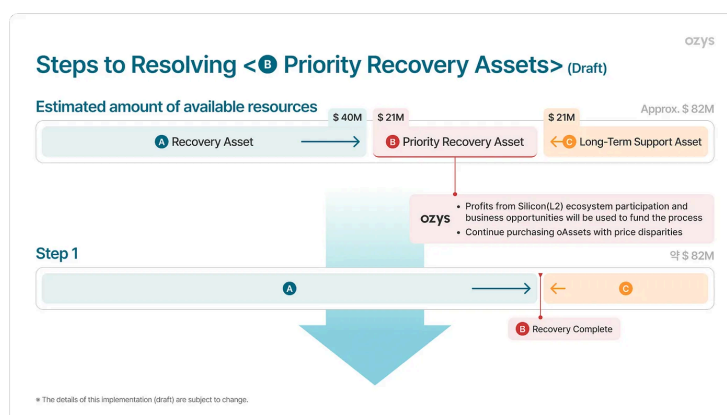
On January 1, 2024, an unidentified exploiter stole the entire assets from the Orbit Bridge's Ethereum vault. As the attack pattern resembles that of Lazarus, the Korean National Intelligence Service and police agency proceeded to investigate the incident.

During the investigation, it was discovered that Ozys' former CISO had arbitrarily changed the firewall policies on Nov. 22, 2023. Ozys took legal action to the prior CISO in charge of the incident.

Regarding the incident, Ozys clarified that the exploit did not result from a vulnerability in the Orbit Bridge's smart contract or the theft of a validator key. The detailed process of the attack will be opened after the full investigation of national services.

Compensation Process

Orbit Chain plans to launch another chain [Silicon](#), an Ethereum L2. On Feb 28, 2024, Orbit team published a service resumption strategy, with a plan to compensate affected users through revenues from Silicon and continuous buy-backs of depegged assets.



Pre-Incident Audit History

Theori/ChainLight (2022.04.08) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.01.25

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#), [Tx5](#), [Tx6](#)

Fund Flow

Exploited assets are distributed into multiple addresses. They are partially laundered through Tornado Cash, while a substantial amount of the exploited assets still remains in various EOAs.

[0x84bf](#) (~12931 ETH, Laundered through Tornado Cash), [0x5892](#) (4320 ETH)

[0x009b](#) (9500 ETH), [0x3a88](#) (10M DAI)

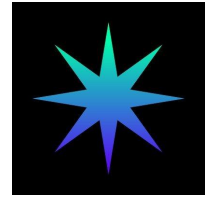
[0x157a](#) (5M DAI), [0x817b](#) (5M DAI)

Note

As mentioned above, Orbit is building another Ethereum Layer 2 blockchain, Silicon. Through the team's X account, it seems Silicon is currently in the process of building its ecosystem.



2. Radiant Capital (Jan 2)



Underlying Chain: Arbitrum

Amount: \$4,500,000

Attack Vector: Contract Vulnerability - Precision Loss (Aave V2 Fork)

Brief Summary

Radiant Capital is a multichain lending protocol, which forked Aave V2. Similar to CompoundV2 forks, AaveV2 forked projects often have a price manipulation vulnerability when the pools are deployed with empty liquidity. It is due to the combination of a rounding error and totalSupply value of 0. As a Aave V2 fork, Radiant capital also experienced the exploit due to the vulnerability. The exploiter attacked the new market pool contract 5 seconds after its activation, generating a bad debt of \$4.5 million.

In depth, the totalSupply of newly deployed aToken (LP token) contracts was uninitialized and set as 0. If the exploiter operates as a first-time liquidity provider, the liquidityIndex which manages user's AToken balance could be inflated. Furthermore, a rounding error existed in the rayDiv function, and the precision loss could be inflated even more when aToken was minted and burned as the rayDiv function was used for scaling token amounts.

```
/**
 * @dev Divides two ray, rounding half up to the nearest ray
 * @param a Ray
 * @param b Ray
 * @return The result of a/b, in ray
 */
ftrace | funcSig
function rayDiv(uint256 a↑, uint256 b↑) internal pure returns (uint256) {
    require(b↑ != 0, Errors.MATH_DIVISION_BY_ZERO);
    uint256 halfB = b↑ / 2;

    require(a↑ <= (type(uint256).max - halfB) / RAY, Errors.MATH_MULTIPLICATION_OVERFLOW);

    return (a↑ * RAY + halfB) / b↑;
}
```



```

/**
 * @dev Mints `amount` aTokens to `user`
 * - Only callable by the LendingPool, as extra state updates there need to be managed
 * @param user The address receiving the minted tokens
 * @param amount The amount of tokens getting minted
 * @param index The new liquidity index of the reserve
 * @return `true` if the the previous balance of the user was 0
 */
fttrace | funcSig
function mint(address user↑, uint256 amount↑, uint256 index↑) external override onlyLendingPool returns (bool) {
    uint256 previousBalance = super.balanceOf(user↑);

    uint256 amountScaled = amount↑.rayDiv(index↑);
    require(amountScaled != 0, Errors.CT_INVALID_MINT_AMOUNT);
    _mint(user↑, amountScaled);

    emit Transfer(address(0), user↑, amount↑);
    emit Mint(user↑, amount↑, index↑);

    return previousBalance == 0;
}

```

Right after the incident, Radiant DAO Council paused all markets by invoking the emergency admin controls.

Compensation Process

The project repaid its bad debt through the treasury fund after the DAO proposal. ([Ref.](#))

Pre-Incident Audit History

Zokyo (2022.05.06) - [Report](#), SourceHat (2022.05.12) - [Report](#),

Peckshield (2022.07.28) - [Report](#), BlockSec (2023.03.22) - [Report](#),

Peckshield (2023.03.05) - [Report](#), OpenZeppelin (2023.10.18) - [Report](#)

Post-Incident Audit History

BlockSec (2024.06.28) - [Report](#) , Pashov (2024.07.26) - [Report](#),

OpenZeppelin (2024.07.23) - [Report](#)

Postmortem

[Link](#) - 2024.01.20

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#)



Fund Flow

Exploited assets are initially distributed to over 50 EOAs with various amounts of ETH, which seems to be the exploiter's strategy to make the trace harder.

[0x4D60](#) (537 ETH)

→ Moved to various EOAs

→ Bridged via Hop Protocol, Synapse, LayerZero Stargate (Arbitrum → Ethereum) & Laundered through Tornado Cash ([0xe837](#), [0xa478](#), [0x28a4](#), [0x78E9](#), [0x4836](#), [0xaCb0](#), [0x10ae](#), [0xaCe1](#), [0xE3e4](#), [0x8E10](#), [0xa8A6](#), [0x2a79](#), [0x8DF7](#), [0x1159](#), [0x7c71](#), [0x3255](#), [0x6Ef9](#), [0x1350](#), [0xe6BE](#), [0xA6C6](#), [0x4137](#), [0xb16d](#), [0x9CD5](#), [0x5486](#), [0x27BC](#), [0xc84c](#), [0xdCCc](#), [0xfAE6](#), [0xBCEb](#), [0xc3d2](#), [0xc3d2](#), [0x9A37](#), [0xD83E](#), [0x4959](#) - 1870.2 ETH) / Leftover sent to [0x2793](#), [0x8DF7](#), [0x326A](#) / Transferred to eXch ([0x418f](#), [0xD3a8](#) - 0.23 ETH) / Transferred to ChangeNOW ([0xA45F](#), [0x73F3](#), [0x9068](#), [0x8ed0](#), [0x92bb](#) - 12.17 ETH)

→ Bridged via THORChain (Ethereum → Bitcoin) ([0x8E10](#) - 0.47 ETH) → Bitcoin address ([bc1q33](#)) / Transferred to eXch ([0xC93B](#) - 0.1 ETH)

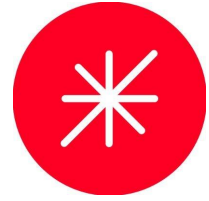
The whole laundry took about 20 days. One notable point is that the address leftover assets headed to, [0x2793](#), is an active address, holding \$3.8M worth USDT and USDC in Ethereum.

Note

After the remediation, Radiant Capital has operated normally. However, Radiant Capital suffered another exploit in October 2024. Due to the severe damage (~\$58M) of the second exploit, Radiant Capital had lost most of its remaining assets. On Dec 30 2024, the team updated the remediation plan stating a long-term recovery. ([Ref.](#)) The project is not currently operating.



3. Gamma Strategies (Jan 4)



Underlying Chain: Arbitrum

Amount: \$6,500,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

The main cause of the attack was a misconfigured price change threshold. Gamma strategies had implemented the price change threshold inside its logic, but was not set as intended. At the time of the exploit, the price threshold was set to allow 100% increase & 50% decrease of the token price, while the team intended a maximum 2% range of the price fluctuation. Exploiting the misconfigured threshold, the attacker performed a classic flashloan attack and drained the pool.

Compensation Process

After the incident, the project released their remediation plan with postmortem. Since the attacker withdrew the stolen funds through Tornado.Cash, the project announced gradual repayment of its bad debt, with a goal of 8 months. The team shared a detailed remediation plan with the community feedback taken. ([Ref.](#))

Pre-Incident Audit History

CertiK (2021.07.07) - [Report](#), Arbitrary Execution (2022.03.09) - [Report](#)

Consensys Diligence (2022.03.28) - [Report](#)

Post-Incident Audit History

OpenZeppelin (2024.01.23) - [Report](#)

Postmortem

[Link](#)

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#), [Tx5](#), [Tx6](#), [Tx7](#), [Tx8](#), [Tx9](#), [Tx10](#), [Tx11](#), [Tx12](#), [Tx13](#)



Fund Flow

All the exploited assets are laundered through Tornado Cash, 27 days after the exploit.

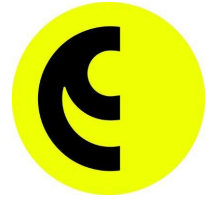
[0x5351](#) (~2827 ETH)

→ Bridged via Stargate (Arbitrum → Ethereum)

→ Laundered through Tornado Cash (1000 ETH) / Transferred to [0x5C6d](#) & Laundered through Tornado Cash (~1827 ETH)



4. CoinsPaid (Jan 6)



Underlying Chain: Ethereum

Amount: \$7,500,000

Attack Vector: Control Hijacking - Private Key Leakage, Backdoor

Brief Summary

After six months from its first security breach in 2023, CoinsPaid experienced another exploit of \$7.5 million. It is suspected to be the impact of the first exploit performed by Lazarus. CoinsPaid did not make any response regarding the incident.

Compensation Process - N/A

Postmortem - N/A

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#)

Fund Flow

Distributed to multiple EOAs ([0x20f4](#), [0x6c41](#), [0x943b](#), [0x3cc1](#), [0x8f84](#))

→ EOA, Binance, HitBTC, ChangeNOW, N.exchange, Aeroswap (Ref: Arkham [\[1,2,3,4,5\]](#))



5. Narwhal (Jan 6)



Underlying Chain: Ethereum

Amount: \$1,000,000

Attack Vector: Control Hijacking - Private Key Leakage / Rugpull

Brief Summary

Through the owner-authorized withdrawal function, \$NRW tokens were withdrawn and dumped to the market which caused a 99% price drop of the token. However, CertiK's in-depth [investigation](#) suggests the possibility of an exit scam, as the involved EOAs that previously funded the Narwhal deployer wallet.

Compensation Process

According to the team, they redeployed the liquidity pool contract, along with \$500,000 of liquidity from their reserves. ([Ref.](#))

Postmortem

[Link](#) - 2024.01.08

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

[0x9481](#)

→ Bridged via Axelar Squid (BSC → Ethereum) → Laundered through Tornado Cash ([0x9481](#), 375 ETH)

Laundry took 2 days. The exploiter deployed [\\$SolarCity](#) and [\\$FreeMintToken](#) on Ethereum.

Note

Narwhal launched a new DApp on Feb 22 2024 with a new domain narwhal.name and nrw.ac, but none of them are working at current state. No update has been made since Mar 20 2024.



6. MangoFarmSOL (Jan 6)

Underlying Chain: Solana

Amount: \$1,267,000

Attack Vector: Rugpull

Brief Summary

MangoFarmSOL, an yield farming protocol on Solana, rugged after its token presale of \$1.29M. Notably, The rugpull was done by the drainer injected in its frontend, not the smart contract side. ([Ref.](#))



foobar | Clusters @0xfoobar · Jan 7, 2024

mangofarm frontend compromised, don't use

141

258

583

168K

Pre-Incident Audit History

0xfoobar - Unavailable

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

Most of the exploited assets were bridged to Ethereum via Wormhole, and laundered through Railgun or deposited to eXch. Laundry was done immediately after the rugpull.

Exploiter ([8ggv](#))

→ Bridged via Wormhole (Solana → Ethereum) ([0x8816](#), [0xc504](#), [0x09E3](#), [0x6898](#), [0xC3Dd](#) - 1.26M USDC)

→ Swapped to ETH & Deposited to Railgun ([0x7b47](#), [0xe6c9](#), [0x6fBa](#) - 262.7 ETH) / eXch ([0xe4b4](#), [0x72e8](#), [0x7d67](#) - 286 ETH)



7. Wise Lending (Jan 12)



Underlying Chain: Ethereum

Amount: \$460,000

Attack Vector: Contract Vulnerability - Precision Loss

Brief Summary

The root cause of the exploit was a precision loss existed in the calculation of the share distribution. As the protocol used a rounding up mechanism when calculating share withdrawals to mitigate the rounding issues, these logics could be combined to inflate the share price by repeated call of the withdrawal function.

```
46     function _calculateShares(  
47         uint256 _product,  
48         uint256 _pseudo,  
49         bool _maxSharePrice  
50     )  
51     {  
52         private  
53         pure  
54         returns (uint256)  
55     {  
56         return _maxSharePrice == true  
57             ? _product % _pseudo == 0  
58               ? _product / _pseudo  
59               : _product / _pseudo + 1  
60             : _product / _pseudo;  
61     }  
62 }
```

²

By manipulating the share price through donation, the exploiter could profit \$460,000 from the attack.

Compensation Process - N/A

Pre-Incident Audit History

OxGuard (2022.09.26) - [Report](#), OxGuard (2022.04.02) - [Report](#)

CoinFabrik (2020.09) - [Report](#), Internal Audit (2023.10.18) - [Report](#)

² [Source](#)



Post-Incident Audit History

Hats Finance (2024.02.20) - [Report](#)

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

All the exploited assets are laundered through Tornado Cash, 6 days after the exploit.

Exploiter ([0xB90C](#), 179 ETH) → [0x5928](#) → Laundered through Tornado Cash ([0xCfbB](#), 179 ETH)

Note

Even though most of the assets were stolen due to the exploit, Wise Lending recovered most of its TVL in December 2024. It is normally operating, with a TVL of \$377K.



8. Hector Network (Jan 15)



Underlying Chain: Ethereum, Fantom

Amount: \$2,700,000

Attack Vector: Control Hijacking - Private Key Leakage / Rugpull

Brief Summary

In Hector Network's liquidation process, a privileged role named "moderator" distributes the treasury token from Fantom to Ethereum. The moderator can call an authorized function "addEligibleWallet" with the amount of token, which allows the wallet to claim the token from its treasury. The attacker somehow exploited the moderator role and drained the token from the protocol. After the incident, the project published a post-mortem on its homepage, claiming further actions after the exploit. However, the website is currently inaccessible and the social accounts are deleted, implying a potential rugpull. Notably, CertiK has already stated the centralization-related risk in the audit report prior to the incident.

Compensation Process - N/A

Pre-Incident Audit History

CertiK (2022.01.25, 2023.12.19) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.01.19 (By @lilbagscientist, not by the team)

Exploit Transactions - [Tx](#)

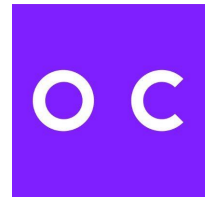
Fund Flow - [Ref.](#)

Note

It seems the project has sunsetted, even though they claimed further actions promptly after the exploit. All the social accounts and websites are closed.



9. Socket.Tech (Jan 16)



Underlying Chain: Ethereum

Amount: \$3,300,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

Socket.Tech is a cross-chain messaging protocol that supports cross-chain token swap. Three days before the incident, the team added a new module under the aggregator system, which swaps the tokens on behalf of the users.

According to the team, an operational mixup happened between the pre-review version and the reviewed version of WrappedTokenSwapperImpl, resulting in the vulnerable module attached to the Gateway.

Unfortunately, the vulnerability in the newly deployed contract allowed arbitrary calls, which led the exploiter to execute transferFrom functions for the previously approved tokens. About 200 users of Socket API and Bungee Exchange have been reportedly affected by the exploit.

File 68 of 70 : swapWrappedImpl.sol

```
57         );
58
59         // Send weth to user
60         ERC20(toToken).transfer(receiverAddress, amount);
61     } else {
62         _initialBalanceTokenOut = address(socketGateway).balance;
63
64         // Swap Wrapped Token To Native Token
65         ERC20(fromToken).safeTransferFrom(
66             msg.sender,
67             socketGateway,
68             amount
69         );
70
71         (bool success, ) = fromToken.call(swapExtraData);
72
73         if (!success) {
74             revert SwapFailed();
75         }
76
77         _finalBalanceTokenOut = address(socketGateway).balance;
78
79         require(
80             (_finalBalanceTokenOut - _initialBalanceTokenOut) == amount,
81             "Invalid wrapper contract"
```



Compensation Process

As 80% of the exploited funds were returned from the exploiter, Socket was able to reimburse 100% of the stolen funds to its users. The team promptly announced a reimbursement plan and a website to check the asset amount allocated to each user. ([Ref.](#))

Pre-Incident Audit History

Peckshield (2021.09.26) - [Report](#), Consensys Diligence (2023.03.03) - [Report](#),

Zellic (2023.07.27) - [Report](#), Hexens (2023.08.04) - [Report](#),

Hexens (2023.08.16) - [Report](#)

Post-Incident Audit History

Hexens (2024.01.23) - [Report](#), Hexens (2024.03.22) - [Report](#),

Decurity (2024.04.16) - [Report](#), Decurity (2024.06.04) - [Report](#),

Decurity (2024.11.26) - [Report](#)

Postmortem

[Link](#) - 2024.01.16

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

90% of the exploited assets are returned to the Project ([Tx1](#), [Tx2](#), 1027 ETH), 7 days after the exploit. The exploiter laundered 10% through Tornado Cash ([0xC378](#), 110 ETH), 15 days after the exploit.

Note

Since the significant amount of assets were returned from the exploiter, Bungee Exchange (cross-chain swap protocol operated by Socket.Tech) has been operating normally.



10. Manta Network (Jan 18)



Underlying Chain: Manta

Amount: \$5,700,000

Attack Vector: Circulating Supply

Brief Summary

At the day of its token launch, Manta foundation reserved 3 million \$MANTA from the ecosystem fund and sent them to the Korean CEX, Bithumb. The entire tokens were immediately sold, and had been taken out as ETH.

The problem is:

- 1) This was not a community-noticed action.
- 2) The tokens were transferred to personal wallet of Manta's Korean BD
(0x9670359ce856771369ad04A67DD6cF276b9fE8aF)
- 3) The transferred token amount was more than 75% of Bithumb's total circulation volume.

After the incident, Manta foundation claimed that the funds were intended to be used for "Investment and growth of the Manta community in Korea". ([Ref.](#))

Postmortem

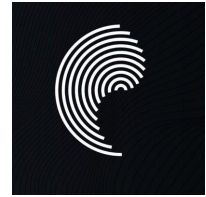
[Link](#) - 2024.01.20

Exploit Transactions

[Tx](#)



11. Concentric Finance (Jan 22)



Underlying Chain: Arbitrum

Amount: \$1,650,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

The attacker, posing as a recruiter on a professional networking platform, performed a social engineering attack on the team member and let the victim install the malware under the guise of a routine skill assessment. This compromised the private keys of the deployer wallet.

According to the team's postmortem, the exploiter first shifted the ConeCamelopFactory's ownership, upgraded the existing vault's implementation contract, and minted new LP tokens allegedly to drain the assets in the liquidity pool.

Compensation Process

As the funds were not recovered after the attack, the protocol's TVL went 0 with \$25K assets in the treasury. Despite the difficulty, the team decided to build new products and promised its users the remediation process through the future income. ([Ref.](#)) In Feb 2024, the team proceeded with the reimbursement process for the \$CONE and \$xCONE holders with the remaining funds. ([Ref.](#))

Pre-Incident Audit History

Cantina (2023.10.24) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.01.23

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#)



Fund Flow

Exploited assets are initially distributed to 3 EOAs, bridged to BSC via Stargate and consequently laundered through Defiway, a centralized cross-chain bridge.

Exploiter ([0x105f](#), 715.7 ETH) → EOAs ([0x1786](#), [0x1F14](#), [0xFD68](#) - 715.7 ETH)

→ Bridged via LayerZero Stargate (Arbitrum → BSC) ([0x429E](#), [0xeb69](#) - 1.69M USDT)

→ ... → Laundered via Defiway ([0xa0bB](#))

Note

Despite the team's announcement of launching new products for remediation, there has been no update since May 2024.



12. Ondo Finance (Jan 22)



Underlying Chain: Ethereum

Amount: \$6,900,000

Attack Vector: Circulating Supply

Brief Summary

On-chain analyst @ai_9684xtpa reported that one of the Ondo finance team addresses had deposited \$3.2 million worth \$ONDO to Bybit and \$3.7 million worth \$ONDO to Coinbase. According to the analysis, the address had received 138 million tokens from the project's multi-sig wallet in January 2024, which implies the relation of the account with the team. ([Ref.](#))

Postmortem - N/A

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#)

Fund Flow

20M ONDO were directly deposited to Kucoin and Gate.io, and 25M ONDO headed to Coinbase after 58 days. Notably, 93M ONDO were returned to the team's multisig wallet on Mar 21 2024.

Team's Multisig ([0x677F](#)) → [0x4159](#) (137M ONDO)

→ Deposited to Kucoin (10M ONDO), Gate.io (10M ONDO) / Coinbase ([0xE9B5](#), [0xf719](#) - 25M ONDO) / Returned to Team's Multisig ([0x677F](#), 93M ONDO)



13. Gamee (Jan 23)



Underlying Chain: Polygon

Amount: \$16,350,000

Attack Vector: Control Hijacking - Private Key Leakage, Key Exposure

Brief Summary

An unauthorized access to the protocol's GitLab compromised the private key of the contract deployer. The attacker utilized the `recoverERC721s()` function and drained 600 million GME tokens in the contract.

Compensation Process

As the exploited assets only include the team token reserves, user assets were not drained through the attack. However, to prevent the second impact through price fluctuation, the team decided to isolate the tokens in Polygon, where the exploit happened, and migrate the token to Ethereum. ([Ref.](#))

Post-mortem

[Link](#) - 2024.01.25

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#)

Fund Flow

Exploiter ([0x16af](#))

→ Bridged via Socket.Tech (Polygon → Ethereum)

→ Laundered through Tornado Cash ([0x16af](#))



14. Nebula Revelation (Jan 25)



Underlying Chain: Optimism

Amount: \$180,000

Attack Vector: Contract Vulnerability - Reentrancy

Brief Summary

Nebula Revelation, a gaming platform on Optimism, had a simple reentrancy vulnerability in the withdrawal function of the staking contract. Leveraging the vulnerability, the exploiter drained the entire ETH staked in the contract with a self-destructing contract. Notably, the team displayed its audit report which was for a token minting contract with 19 lines of code. ([Ref.](#))

Compensation Process

Through the team's own resources, the team compensated users in \$NBL at a fixed price, which was worth \$160K in total. ([Ref.](#))

Pre-Incident Audit History

CertiK (2024.01.19) - [Report](#)

Post-Incident Audit History

CertiK (2024.02.07, 2024.05.05, 2024.07.05) - Reports not open for public

Postmortem

[Link](#) - 2024.01.26

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0x1FD0](#)) → Bridged via Hop Protocol (Optimism → Ethereum)

→ Laundered through Tornado Cash ([0x1FD0](#))



15. Somesing (Jan 27)



Underlying Chain: Klaytn (Now Kaia)

Amount: \$11,500,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

Somesing experienced a security breach of its wallet, resulting in the leakage of 730 million SSX tokens. Among the stolen tokens, 504 million SSX were undistributed tokens that were planned to be circulated. The team claimed that the incident is not related to their team members. After the incident, the project reported the incident to the national police agency and suspended the token flows in CEXs.

On Feb 27 2024, Somesing team published an official statement regarding the delisting of \$SSX token from Korean exchanges with the details of the exploit. According to the statement, the exchange response staff of the team executed the malware attached in the phishing mail. After the exploit, the team requested a digital forensic to the compromised server, and it was confirmed that the North Korean hacker group 'Kimsuky' carried out the exploit.

To prevent the exploited assets supplied to the exchanges, the team migrated \$SSX into a new token contract, \$SSG on Mar 2024.

Compensation Process

Since a substantial amount of tokens were over-supplied due to the exploit, the team promised the burning of the team token in Q1 2024 and the exploited tokens if returned. However, no announcement of a token burning event has been announced till now.

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A



Postmortem

[Link](#) - 2024.02.27

Fund Flow

Huobi ([0x8fd0](#), [0x4c97](#))

Note

Due to the exploit, \$SSX token was delisted from major Korean CEXs due to the decision of Digital Asset Exchange Association (DAXA). In Mar 2024, Somesing team submitted the application of interim injunction of trading support termination, but this was dismissed by the court a week after the submission. The project itself is currently operating as normal.



16. Goledo Finance (Jan 29)



Underlying Chain: Conflux

Amount: \$1,700,000

Attack Vector: Contract Vulnerability - Precision Loss (Aave V2 Fork)

Brief Summary

Goledo is an AaveV2 fork on the Conflux Network. The detailed methodology of the exploit is ambiguous whether it is a classic price manipulation attack or the attack leveraging Aave V2 fork's precision loss vulnerability, while the team's post-mortem describes the attack as a classic flash loan attack. According to the post-mortem, the attacker profited \$1.7 million by manipulating the exchange rate through flashloan.

After the incident, the project claimed that some of the stolen assets were frozen by CEX. Shortly after the incident, we could find the attacker and project team communicating through on-chain messages, and they proceeded on negotiation. ([Ref.](#))

In Jul 2024, the team announced that the lost funds were partially retrieved with the support of Conflux Network. ([Ref.](#))

Compensation Process

In Mar 2024, the team announced the reimbursement for users who had the assets in the lending pool less than \$20K, based on the snapshot before the exploit. ([Ref.](#))

After the partial retrieval of the exploited assets, the team promptly opened the reimbursement process for the users with assets more than \$20K. ([Ref.](#))

Pre-Incident Audit History

Peckshield (2022.10.12) - [Report](#)

Post-Incident Audit History - N/A



Post-mortem

[Link](#) - 2024.02.01

Exploit Transactions - [Tx](#)

Fund Flow

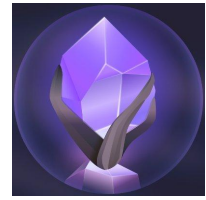
[Ref.](#)

Note

By looking at the [website](#), it seems the project is not operating in its current state.



17. Abracadabra.money (Jan 30)



Underlying Chain: Ethereum

Amount: \$6,400,000

Attack Vector: Contract Vulnerability - Precision Loss

Brief Summary

Abracadabra.money is also known as MIM_spell, the project with \$MIM and \$SPELL.

The root cause of the exploit was a rounding issue in the CauldronV4 contract. This resulted in the miscalculation of user debt, enabling the exploiter to steal \$MIM from magicAPE and yvCrv3Crypto cauldrons.

According to the in-depth analysis [report](#) of Three Sigma, the exploiter flashloaned \$MIM from the project's DegenBox, deposited it into BentoBox, and triggered the `repayForAll` function that repays the debt of others. repayForAll function set the totalBorrow.elastic to zero, but due to the rounding error, totalBorrow.base did not drop to 0 but remained as a certain integer, leading to the miscalculation of the share price.

```
/// @notice Used to auto repay everyone liabilities'.
/// Transfer MIM deposit to DegenBox for this Cauldron and increase the totalBorrow base or skim
/// all mim inside this contract
function repayForAll(uint128 amount, bool skim) public returns (uint128) {
    accrue();

    if (skim) {
        // ignore amount and take every mim in this contract since it could be taken by anyone, the next block.
        amount = uint128(magicInternetMoney.balanceOf(address(this)));
        bentoBox.deposit(magicInternetMoney, address(this), address(this), amount, 0);
    } else {
        bentoBox.transfer(magicInternetMoney, msg.sender, address(this), bentoBox.toShare(magicInternetMoney, amount, true));
    }

    uint128 previousElastic = totalBorrow.elastic;

    require(previousElastic - amount > 1000 * 1e18, "Total Elastic too small");

    totalBorrow.elastic = previousElastic - amount;

    emit LogRepayForAll(amount, previousElastic, totalBorrow.elastic);
    return amount;
}
```



```

/// @notice Calculates the base value in relationship to `elastic` and `total`.
function toBase(Rebase memory total, uint256 elastic, bool roundUp) internal pure returns (uint256 base) {
    if (total.elastic == 0) {
        base = elastic;
    } else {
        base = (elastic * total.base) / total.elastic;
        if (roundUp && (base * total.elastic) / total.base < elastic) {
            base++;
        }
    }
}

/// @notice Calculates the elastic value in relationship to `base` and `total`.
function toElastic(Rebase memory total, uint256 base, bool roundUp) internal pure returns (uint256 elastic) {
    if (total.base == 0) {
        elastic = base;
    } else {
        elastic = (base * total.elastic) / total.base;
        if (roundUp && (elastic * total.base) / total.elastic < base) {
            elastic++;
        }
    }
}

```

Exploiting the bug, the exploiter repeatedly borrowed and repaid a small amount of assets and inflated the share price. By borrowing a massive amount of MIM token with inflated shares, the exploiter took profit and left a \$6.4 million bad debt to the project.

After the incident, the project announced that no user assets are at risk as the exploited contracts are not in their latest products, and the issues were patched. They claimed the cooperation with Chainalysis regarding the trace of stolen assets, but it seems that the exploiter had already laundered the assets through Tornado.Cash.

Compensation Process

As user funds were not exposed during the exploit, the project announced further plans of instituting a depeg contingency fund governed by the DAO multisigners. ([Ref.](#))

Pre-Incident Audit History

Guardian Audits (2023.11.14) - [Report](#)

Post-Incident Audit History

Guardian Audits (2024.02.06) - [Report](#)



Postmortem

The team promised the publication of a detailed postmortem regarding the exploit, but it has not been done yet. ([Ref.](#))

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

All the exploited assets are laundered through Tornado Cash, 50 days after the exploit.

Exploiter ([0x87F5](#), 2737.5 ETH) → Laundered through Tornado Cash ([0x3528](#), [0x516a](#))



18. Chris Larsen, Ripple Co-founder (Jan 30)



Underlying Chain: Ripple

Amount: \$11,300,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

The incident was initially reported by the on-chain detective ZachXBT ([Ref.](#)), with the abnormal movement of 213 million XRP. Shortly after, Ripple Co-founder Chris Larsen claimed that the exploited wallets were his personal wallets, not the foundation wallet. However, the exploited account, [rJNLz3A](#), was tagged in XRPScan and Bithomp as the entity Ripple.

The stolen assets were transferred to various CEXs.

Exploit Transactions

Tay of Metamask(@tayvano_) provided a bunch of exploit transactions on X. ([Ref.](#))

Fund Flow

Web3 audit firm Hacken and ZachXBT provided an in-depth analysis on the fund flow of the incident. ([Ref.](#), [Ref.](#), [Ref.](#))

Exploiter ([rJNL](#)) → Distributed to 8 addresses ([rGhR](#), [rHQV](#), [rLsU](#), [rKPE](#), [rpjs](#), [rLRh](#), [rnCy](#), [rHVj](#))

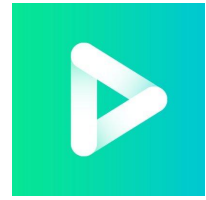
→ Deposited to Binance

→ Distributed to HTX, Gate.io, Kraken

→ Withdrawn as BTC, partially deposited to Jambler, a Bitcoin mixer. ([Ref.](#))



19. PlayDapp (Feb 9)



Underlying Chain: Ethereum

Amount: \$32,500,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

The exploiter compromised the minting authority of token contract, and the deployer lost its control by the exploit. The exploiter minted additional 200 million \$PLA and dumped the tokens through Gate.io and Binance. Consequently, the exploiter drained all the assets of the deployer and also transferred the tokens to Binance. Regarding the incident, the project offered \$1 million, but no response from the exploiter could be found. On Feb 12, 2024, 1.59 billion \$PLA was additionally minted, and distributed to EOAs. ([Ref.](#)) We identified that some of the additional allegedly minted \$PLA were moved to HTX.

According to the postmortem of the team, the root cause was the domain spoofing email from the exploiter, resembling an information request email from a major exchange. The exchange response staff opened the malware attached in the email, and this gave the remote control of admin access to the team server that stored the admin private key.

Compensation Process

The team migrated the token contract from \$PLA to \$PDA. In Apr 2024, the team announced the launch of its mainnet and airdrop to the users. ([Ref.](#))

Pre-Incident Audit History

CertiK (2019.02.26) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.04.01



Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#)

Fund Flow

Exploiter ([0xd150](#), 300M \$PLA) → [0x1cae](#) (100M \$PLA), [0x21eb](#) (200M \$PLA)

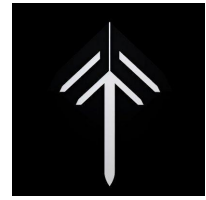
→ Bridged via Polygon official bridge (Ethereum → Polygon) ([0x1cae](#), 60M \$PLA) / Deposited to
Binance ([0x1cae](#), 35K \$PLA), HTX ([0x9529](#), [0xcb66](#), [0x657A](#), [0xE15f](#), [0xe84d](#), 37M \$PLA),
Gate.io ([0x25E5](#), 47.5M \$PLA), Bitget ([0x10b6](#), [0xA176](#), 30M \$PLA)

Note

In Mar 2024, \$PLA was delisted from Korean major exchanges. In Apr 2024, PlayDapp launched its mainnet, as a subnet on Avalanche, and migrated the token to \$PDA. The project seems to operate normally, at \$28M market cap. Currently, \$PDA is listed at international exchanges, such as Binance, Kraken, and HTX.



20. Miner ERCX (Feb 14)



Underlying Chain: Ethereum

Amount: \$435,000

Attack Vector: Contract Vulnerability - Misimplemented Transfer Function

Brief Summary

\$MINER, a token minted with an experimental token scheme named "ERC404", crashed over 99% by an attack exploiting the bug in the token contract. Due to the cached balance in the `_update` function, the attacker could double his balance when the tokens were sent to himself.

```
function _update(address from, address to, uint256 value, bool mint) internal virtual {
    uint256 fromBalance = _balances[from];
    uint256 toBalance = _balances[to];
    if (fromBalance < value) {
        revert ERC20InsufficientBalance(from, fromBalance, value);
    }

    unchecked {
        // Overflow not possible: value <= fromBalance <= totalSupply.
        _balances[from] = fromBalance - value;

        // Overflow not possible: balance + value is at most totalSupply, which we know fits into a uint256.
        _balances[to] = toBalance + value;
    }
}
```

The team offered 30% of the exploited funds, \$120K, as a bounty, but the exploited assets were consequently laundered through Tornado Cash.

After the incident, the bug was promptly fixed and the team announced a post-incident audit with Cyberscope, also with the community bug bounty program of 2ETH.

Compensation Process

The project relaunched 5 days after the exploit, with an airdrop based on the snapshot before the exploit. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History

Cyberscope (2024.02.21) - [Report](#)



Postmortem

[Link](#) - 2024.02.15

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0xea75](#)) → Laundered via Tornado Cash ([0xC181](#), 156 ETH)

Note

The project seems not operating in its current state, and no updates have been made since May 2024.



21. Duelbits (Feb 14)



Underlying Chain: Ethereum, BSC

Amount: \$4,500,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

Duelbits, a crypto gambling platform, took a hit of \$4.5 million due to the private key leakage. Regarding the exploit, no official announcements could be found.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

All the exploited assets are laundered through Tornado Cash, after being bridged from BSC to Ethereum. Laundry took 105 days.

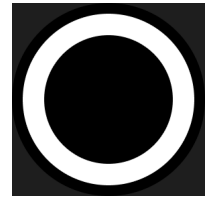
Exploiter ([0x3933](#), 1780 BNB)

→ Bridged via Wormhole (BSC → Ethereum)

→ Laundered through Tornado Cash ([0x07a0](#), 1760 ETH)



22. Particle Trade (Feb 15)



Underlying Chain: Ethereum

Amount: \$141,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

In the NFT trading contract of Particle Trade, a vulnerability that allows the exploiter to overwrite user data and manipulate the account balance existed. Leveraging the vulnerability, the exploiter drained assets in the contract, profiting \$141K. Regarding the incident, the team claimed that the targeted contract was deprecated, while promising additional rounds of audits for all the contracts. ([Ref.](#))

Compensation Process - N/A, No user funds affected

Pre-Incident Audit History

Code4rena (2024.01.26) - [Report](#)

Post-Incident Audit History

Cantina (2024.03.28) - [Report](#)

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0x2c90](#))

→ Laundered through Tornado Cash (23 ETH) / Deposited to Binance ([0x7ca6](#), 0.5 ETH)

Laundry was done immediately after the exploit.



23. xPet (Feb 16)



Underlying Chain: Arbitrum

Amount: \$254,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

xPet is an Arbitrum-based P2E game. The token swap functionality in the game contract allowed the exploiter to withdraw excessive amounts of \$BPET tokens after staking his tokens. As the exploiter was suspected to be an active user of xPet, the team could identify the identity of the exploiter (@beeyondthe22849) and contacted through X direct messages. Fortunately, the exploiter returned the whole amount of exploited assets to the project.

Compensation Process - N/A, User fund not affected

Pre-Incident Audit History

Secure3 (2023.12.20) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.02.17

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

Exploiter ([0x3493](#), 91.5 ETH)

→ Bridged via Across Protocol (Arbitrum → Ethereum) ([0x9a0e](#), 91.5 ETH)

→ Bridged via Orbiter Finance (Ethereum → Arbitrum) ([0x0533](#), 91.5 ETH)

→ Returned to the Project ([Tx](#))



Note

The project seems not operating in its current state, as no updates have been made since Aug 2024.



24. FixedFloat (Feb 18)



Underlying Chain: Ethereum

Amount: \$26,120,000

Attack Vector: Control Hijacking

Brief Summary

FixedFloat lost 1700 ETH and 409 BTC, which were \$26M worth in total. The root cause of the exploit is suspected to be private key leakage, but no details are published by the team. The team initially claimed “minor technical problems” regarding the transaction failure of the service ([Ref.](#)), but later confirmed the hack and theft of funds. ([Ref.](#))

Compensation Process - N/A, No user funds affected

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#)

Fund Flow

Exploiter ([0x85c4](#))

→ Distributed into various EOAs

→ eXch ([0xDE28](#), [0xd349](#), [0x1Acf](#), [0xb2aF](#), [0x4a00](#), [0x4A91](#), [0xE729](#), [0xF80C](#), [0x44aA](#), [0x5889](#), [0x16b6](#), [0x2627](#), [0x10f9](#), [0x9C6f](#), [0x7780](#), [0x4897](#), [0xeFc0](#), [0x7433](#), [0xd13a](#), [0x18E9](#), [0xfd43](#), [0xE41E](#), [0xCcBd](#), [0x8d96](#), [0x92A7](#), ~1381 ETH), Coinbase ([0xc76E](#), ~12.9 ETH)

Note

Since no user assets are affected, FixedFloat has been operating normally. Unfortunately, FixedFloat suffered another exploit from the same exploiter, which happened in Apr 2024.



25. DeezNuts (Feb 21)

Underlying Chain: Ethereum

Amount: \$170,000

Attack Vector: Contract Vulnerability - Misimplemented Transfer Function

Brief Summary

An ERC404 token named DeezNuts faced an exploit due to the same vulnerability to Miner. The vulnerability in the token contract allowed the attacker to mint infinite amounts of token by self-transfers, and the tokens were directly dumped to the market.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

All the exploited assets are bridged to Bitcoin via THORChain, 83 days after the exploit.

Exploiter (0xd215, 50 ETH)

→ Bridged via THORChain (Ethereum → Bitcoin) ([0xed65](#))

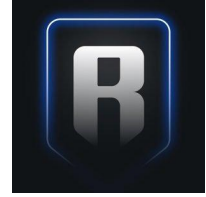
→ Bitcoin ([bc1q5h](#), [bc1qlh](#))

Note

The project seems not operating in its current state, as no updates have been made since Mar 2024.



26. Jeffery Zirlin, Ronin Co-founder (Feb 23)



Underlying Chain: Ronin

Amount: \$9,700,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

On February 23, 2024, @Jihoz_Axie posted that his two personal wallet addresses had been compromised. ([Ref.](#)) He claimed that the leaked private keys are not related to the project.

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#)

Fund Flow

Exploiter ([0x39f8](#))

→ Bridged via Ronin Official Bridge (Ronin → Ethereum)

→ Laundered through Tornado Cash ([0x39f8](#), 946 ETH)



27. Bitforex (Feb 23)



Underlying Chain: Ethereum

Amount: \$56,500,000

Attack Vector: Rugpull

Brief Summary

Bitforex, a Hong Kong-based CEX, shut down its website and social media accounts without any announcement and \$56.5 million was removed from its hot wallets. Since Bitforex held 18% of the \$TRB and 7% of \$OMI supply, the tokens faced immediate price impact due to the exploit.

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#)

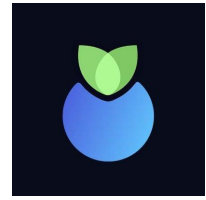
Fund Flow

Exploiter (0x50cb)

→ Deposited to OKX ([0xdcac](#), 350K TRB), Binance ([0xBB21](#), 116K TRB), [0xBB21](#) (9.77M USDT), [0xCcE7](#) (393K USDC), Bitforex Hot Wallets



28. Blueberry Protocol (Feb 23)



Underlying Chain: Ethereum

Amount: \$1,340,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

The newly implemented price oracle of Blueberry protocol had a logical error, as it always returned the prices scaled to 18 decimals. When the borrowed token had less than 18 decimals, they were significantly undervalued, leaving the possibility for the attacker to generate bad debts. Exploiting the vulnerability, the attacker tried to borrow all the liquidity with relatively small collaterals. Fortunately, a whitehat MEV bot c0ffeebabe.eth managed to frontrun the attacker's transaction and returned funds to the project. Blueberry team rewarded c0ffeebabe.eth 10% of returned funds, and promptly released a post-mortem with remediation plans.

Compensation Process

According to the team, 80% of user funds affected were repaid by the DAO multisig, and an additional 20% will be reimbursed gradually.

Pre-Incident Audit History

0x52 (2023.02.05) - [Report](#), Sherlock (2023.02.20) - [Report](#),

Hacken (2023.04.25) - [Report](#), Sherlock (2023.04.30) - [Report](#),

Sherlock (2023.05.11) - [Report](#), Sherlock (2023.08.15) - [Report](#),

Sec3 (2023.12.13) - [Report](#)

Post-Incident Audit History

0x52 (2024.04.19) - [Report](#), cuthalion0x (2024.05.07) - [Report1](#), [Report2](#)



Postmortem

[Link](#) - 2024.02.26

Exploit Transactions - [Tx](#)

Fund Flow

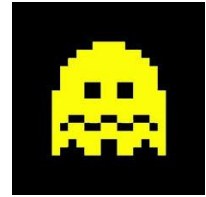
Frontrun by c0ffeebabe.eth → Returned to the Project ([Tx](#))

Note

Blueberry Protocol is no longer operating, but the team has been building a new project, StakeUp Protocol. It seems the project is currently under audits. ([Ref.](#))



29. RiskOnBlast (Feb 25)



Underlying Chain: Ethereum

Amount: \$1,300,000

Attack Vector: Rugpull

Brief Summary

Risk, a gambling protocol on Blast, rugged over 750 wallets after raising 500 ETH during their token presale.

Exploit Transactions - [Tx](#)

Fund Flow

The scammers moved the funds to various CEXs, including ChangeNOW, MEXC, Bybit and Binance.

Exploiter ([0x1EeB](#))

→ Deposited to Bybit ([0x1EeB](#), 98K DAI), MEXC ([0xcF4a](#), 198K DAI), ChangeNOW ([0x84ca](#), 217K DAI), Sideshift / Bridged via THORChain (Ethereum → ATOM) ([cosmos12a](#), 24.8K DAI) / Bridged via RangoBridge (Ethereum → Arbitrum) ([0x09c3](#), 129K USDC) → Deposited to Binance



30. SenecaUSD (Feb 29)



Underlying Chain: Ethereum

Amount: \$6,400,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

An arbitrary call vulnerability existed in SenecaUSD's collateral pool contract named Chamber. Since the Chamber's `performOperations()` function allowed calls to `transferFrom()`, the exploiter could maliciously use Chamber contracts to send tokens to any addresses.

Exploiting the bug, the attacker drained the previously allowed amount of user assets. Even though the Chamber contract had a Pausable modifier, it did not work and caused all the users to revoke their approvals. Fortunately, the attacker accepted a 20% bounty and returned the exploited assets to the project. After the incident, the Seneca team clarified that the exploited function had been audited and it was exactly the same at the time of its deployment.

Compensation Process

On Mar 5 2024, Seneca proceeded on the reimbursement, using the recovered assets (80%) with their assets. ([Ref.](#))

Pre-Incident Audit History

Halborn (2023.12.08) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.03.01

Exploit Transactions - [Tx1](#), [Tx2](#)



Fund Flow

After the bounty negotiation, the exploiter returned 80% of the stolen assets to the project. Rest of the assets are distributed through various routes.

Exploiter ([0x1457](#), [0x9464](#))

→ Returned to the Project ([0xb7af](#), 1537 ETH)

/ Deposited to FixedFloat ([0x78f7](#), 9.7 ETH)

/ Laundered through Railgun ([0x22A8](#), 10 ETH)

/ Bridged via Arbitrum Official Bridge & Laundered through Railgun ([0x128d](#), 20 ETH)

/ Bridged via THORChain (Ethereum → Litecoin) → Litecoin addresses ([ltc1q9h](#), [ltc1q9t](#), [ltc1qzs](#) - 200 ETH)

/ Bridged via Across Protocol (Ethereum → Arbitrum) ([0xa07c](#), 150 ETH) → [0xdbc2](#), [0xa07c](#) → Laundered through Railgun ([0xdbc2](#), [0xd474](#))

Note

Despite the recovery of the exploited assets, the project seems not operating in its current state. Their website is shut down, and no updates have been made since the reimbursements proceeded on Mar 2024.



31. Shido Network (Feb 29)



Underlying Chain: Ethereum

Amount: \$2,400,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

Shido is a layer 1 project that launched its mainnet the day before the exploit. The exploiter upgraded the \$SHIDO token staking contract and drained all the staked tokens. According to the on-chain investigator ZachXBT, the exploiter was funded by lead developer of Injective Labs, implying the exploiter's effort on hiding his identity and social engineering to compromise admin authority of the Shido team. After the incident, the founder promised full compensation of affected assets.

Compensation Process

The token contract was re-deployed, and the token holders were airdropped based on the snapshot. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History

Zokyo (2024.05.28) - [Report](#)

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0x1982](#)) → EOA ([0x4623](#), 956 ETH)



32. WOOFi (Mar 5)



Underlying Chain: Arbitrum

Amount: \$8,500,000

Attack Vector: Contract Vulnerability - Price Dependency

Brief Summary

WOOFi's sPMM algorithm utilized its own oracle based on the token's currently traded price in order to adjust slippage. However, the misimplemented exchange rate calculation mechanism failed to manage the flash loan attack, which allowed the attacker to manipulate the WOO token's price into an extremely low value.

```
function _calcQuoteAmountSellBase(
    address baseToken, uint256 baseAmount, IWoocommerceV2.State memory state) private view returns (uint256 quoteAmount, uint256 newPrice)
{
    require(state.woFeasible, "WooPPV2: !ORACLE_FEASIBLE");

    DecimalInfo memory decs = decimalInfo(baseToken);

    // quoteAmount = baseAmount * oracle.price * (1 - oracle.k * baseAmount * oracle.price - oracle.spread)
    {
        uint256 coef = uint256(1e18) - ((uint256(state.coef) * baseAmount * state.price) / decs.baseDec / decs.priceDec) - state.spread;
        quoteAmount = (((baseAmount * decs.quoteDec * state.price) / decs.priceDec) * coef) / 1e18 / decs.baseDec;
    }

    // newPrice = (1 - 2 * k * oracle.price * baseAmount) * oracle.price
    newPrice = ((uint256(1e18) - (uint256(2) * state.coef * state.price * baseAmount) / decs.priceDec / decs.baseDec) * state.price) / 1e18;
}
```

For additional information, `_calcQuoteAmountSellBase`, the function that calculates the swap rate of BaseToken to QuoteToken, performed multiplication and division without the slippage control. Through frequent token swaps, exploiter could perform the price manipulation. Exploiting the bug, the exploiter drained \$8.7M from WOOFi, leaving bad debts to the protocol.

Compensation Process

So far, there has been no announcement of the compensation for the affected users.

Pre-Incident Audit History

CertiK (2021.10.25, 2021.12.16, 2022.02.07, 2022.07.05, 2022.10.12, 2023.05.08) - [Report](#),

Peckshield (2022.07.05) - [Report](#)



Post-Incident Audit History

Sherlock (2024.03.20) - [Report](#), Zelic (2024.04.15) - [Report](#)

Zelic (2024.08.16) - [Report](#), Sherlock (2024.09.24) - [Report](#)

Postmortem

[Link](#) - 2024.03.06

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#)

Fund Flow

All the exploited assets are laundered through Tornado Cash. Laundry took 66 days.

Exploiter ([0x9961](#))

→ EOAs ([0xe87A](#), [0x0496](#), [0xC12F](#))

→ Bridged via LayerZero Stargate (Arbitrum → Ethereum) ([0x9961](#), [0xC12F](#), [0xe87A](#) - 1610 ETH)

/ Bridged via Synapse (Arbitrum → Ethereum) ([0x0496](#), 510 ETH) → Laundered through Tornado Cash ([0x72Ff](#), [0x3B6C](#) - 2120 ETH)



33. OrdizK (Mar 5)



Underlying Chain: Ethereum

Amount: \$1,000,000

Attack Vector: Rugpull

Brief Summary

The OrdizK team pulled an exit scam by dumping OZK token to the market and withdrawing ETH in its funding contract by invoking an emergency exit function. After the exit scam, the team deleted its website and social media.

Pre-Incident Audit History - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are transferred to 3 EOAs and haven't moved until now. Only 0.5 ETH has been transferred to FixedFloat on Dec 25 2024.

Exploiter ([0xbfdd](#), 252.3 ETH)

→ Multiple EOAs ([0x279b](#), [0x0404](#), [0xAD10](#)), Deposited to FixedFloat ([0x6fB6](#), 0.5 ETH)



34. TGBS Token (Mar 6)

Underlying Chain: BSC

Amount: \$151,000

Attack Vector: Contract Vulnerability - Misimplemented Transfer Function

Brief Summary

The transfer function of the token contract was implemented with a logic that burns certain portions of LP tokens, and this allowed the attacker to manipulate the token swap ratio by just calling the function multiple times.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

All the exploited assets are directly laundered through Tornado Cash. Laundry took 4 days.

Exploiter ([0xFF1d](#))

→ Laundered through Tornado Cash (376.7 BNB)

Note

Since the entire liquidity is drained from the victim, the token price went 0 after the exploit.



35. Humanized AI (Mar 6)

Underlying Chain: Ethereum

Amount: \$665,000

Attack Vector: Rugpull

Brief Summary

The project named Humanized AI performed an exit scam after raising over \$600K through its token presale. The scammer drained the entire balance of the presale contract through `withdrawBalance()`, which could be seen through the verified contract(!).

```
function withdrawBalance(address withdrawer) external onlyOwner(){
    uint256 contractBalance = address(this).balance;
    require(contractBalance > 0, "No ETH balance to withdraw");
    (bool success, ) = payable(withdrawer).call{value: contractBalance}("");
    require(success, "ETH withdrawal failed");
}
```

After the rugpull, the X account was deleted and the assets were transferred to eXch.

Pre-Incident Audit History - N/A

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

Presale contract ([0x98af](#))

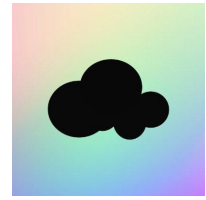
→ Exploiter ([0x24Be](#), 173.8 ETH)

→ [0xef46](#)

→ Distributed to multiple EOAs, 12 ETH each → Deposited to eXch



36. Unizen (Mar 9)



Underlying Chain: Ethereum

Amount: \$2,100,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

Exploiter targeted the arbitrary call vulnerability of Unizen's trade aggregator contract, which was upgraded shortly before the exploit and left unverified.

Due to the exploit, previously approved user assets were drained. Although the previously deployed contract was audited by two firms, we could not identify whether the new contract was audited.

Compensation Process

Unizen founder announced immediate reimbursement for affected users with losses below \$750K, leveraging his personal funds. ([Ref.](#))

Pre-Incident Audit History

CertiK (2021.07.13) - [Report](#), Halborn (2022.09.09) - [Report](#)

Verichains (2022.12.29) - [Report](#)

Post-Incident Audit History

Beosin (2024.04.03) - [Report](#), Verichains (2024.04.11) - [Report](#),

Beosin (2024.04.30) - [Report](#), Beosin (2024.06.21) - [Report](#),

Verichains (2024.06.27) - [Report](#)

Postmortem - N/A

Exploit Transactions - [Tx](#)



Fund Flow

All the exploited assets are laundered through Tornado Cash, 152 days after the exploit.

Exploiter ([0xB660](#)) → [0x8660](#) (1.68M DAI) & Laundered through Tornado Cash (865.4 ETH)



37. Juice Bot (Mar 9)



Underlying Chain: Ethereum

Amount: \$216,000

Attack Vector: Unknown

Brief Summary

Juice, a telegram trading bot application, took an exploit to its staking contract and the staked \$JUICE tokens were drained. The exploiter immediately sold the tokens on Uniswap, making 54 ETH of profit. The details of the exploit were not disclosed.

Compensation Process

The affected \$JUICE holders received token airdrop with 10% additional amount, and the team promised to use 50% of their treasury to buy back tokens from Uniswap and burn them. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

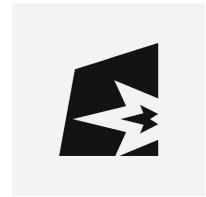
All the exploited assets are deposited to FixedFloat, right after the exploit.

Exploiter ([0x3fA1](#))

→ Distributed to multiple EOAs & Deposited to FixedFloat ([0x90A5](#), [0x23B1](#), [0x6cDD](#), [0x91ed](#), [0x41d4](#) - 46.6 ETH)



38. Blastoff (Mar 11)



Underlying Chain: Blast

Amount: \$601,000

Attack Vector: Unknown

Brief Summary

Blastoff team informed an exploit on the future yield minter vault contract, without the detailed information about the exploit. After the exploit, the team closed the pool contract and promised a thorough investigation. The team offered a 20% bounty, but there has been no response at the time of research.

Compensation Process

Blastoff CEO announced a full reimbursement to the affected users. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History

Zokyo (2024.03.14) - [Report](#), Three Sigma (2024.06.11) - [Report](#)

Three Sigma (2024.08.14) - [Report](#)

Postmortem

[Link](#) - 2024.03.11

Exploit Transactions - Unknown

Fund Flow - Unknown

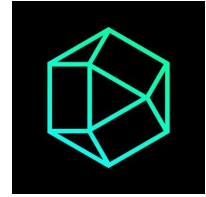


Note

It seems that the project has not sunsetted yet, but the community has left. Its TVL has dropped from \$2.3M to \$90K, and failed to open its first IDO as the number of registrations has not satisfied the minimum number they needed. ([Ref.](#))



39. Polyhedra Network (Mar 13)



Underlying Chain: BSC

Amount: \$700,000

Attack Vector: Control Hijacking - Unknown

Brief Summary

A wallet of Polyhedra Network on BSC was compromised. The exploiter withdrew \$700K of \$THE from the wallet and swapped them into BNB. Right after the attack, the stolen assets were laundered through Tornado.Cash. We could not find any official announcements from the team about the exploit.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#), [Tx5](#), [Tx6](#)

Fund Flow

All the exploited assets are laundered through Tornado Cash, a day after the exploit.

Exploiter (0x36bF) → Laundered through Tornado Cash (1302.6 BNB)



40. Cloud AI (Mar 13)



Underlying Chain: Ethereum

Amount: \$360,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

Cloud AI, a platform for decentralized AI infrastructure, took an exploit on their deployer and treasury account. Drained assets were laundered through Railgun after several transfer transactions.

According to the team, a developer has installed a malicious npm package on a project when invited for a job interview.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.03.13

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#), [Tx5](#), [Tx6](#)

Fund Flow

Exploiter ([0x0996](#)) → ... → Laundered through Railgun ([0x8ab2](#), 91.7 ETH)

Current Status of the Project

The project seems to be inactive, as there has been no update since May 13 2024 from the team's X account.



41. MOBOX (Mar 14)

Underlying Chain: Optimism

Amount: \$750,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The borrow function of the MO token contract had a vulnerability, as the token improperly managed token swap ratio by transferring MO tokens to the burn address when the borrow function was called. By repeated calls of borrow and redeem function, the exploiter manipulated the swap ratio and drained all USDT in the pool with a small amount of MO tokens. After the incident, the X account (@mobox_op) of the project got suspended.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter (0x4e2c)

→ Bridged via Hop Protocol ([0xDd08](#), [0x1009](#) - 140 ETH) / Bridged via Layerzero Stargate ([0x4520](#), 50 ETH)

→ Deposited to FixedFloat ([0xD3d7](#), [0x36c6](#) - 1.8 ETH) / Laundered through Tornado Cash ([0x1009](#), [0x4520](#) - 187 ETH)



42. Mozaic Finance (Mar 14)



Underlying Chain: Arbitrum

Amount: \$2,000,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

Mozaic Finance, a yield farming protocol based on Arbitrum, lost \$2 million from its vaults due to the private key leakage. The team claimed the exploiter to be one of the developers who had illegally obtained the private keys of a security module, by compromising the data of a core team member.

Compensation Process

The compensation has been gradually processed, starting from Apr 3 2024. ([Ref.](#)) In Jul 2024, the team posted a statement that 1.6M USDT were successfully frozen, which is 80% of the exploited assets. ([Ref.](#))

Pre-Incident Audit History

Trust Security (2023.06.05) - [Report](#), Trust Security (2023.08.14) - [Report](#),

Trust Security (2023.09.20) - [Report](#), Trust Security (2024.01.13) - [Report](#)

Trust Security (2024.02.01) - [Report](#), Trust Security (2024.02.12) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.03.16

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#), [Tx5](#)



Fund Flow

Exploiter ([0xD6d4](#))

→ Deposited to MEXC ([0xcC78](#), [0xb86f](#) - 2M USDT), Binance ([0xD6d4](#), 100K USDT)



43. NFP (Mar 15)



Underlying Chain: BSC

Amount: Unknown

Attack Vector: Unknown

Brief Summary

NFP, an AI content creating platform incubated by Binance Labs, experienced a security breach on project wallets including NFP token contract manager. The exploiter gained control of some of the treasury and ecosystem funds. The NFP team conducted an investigation with the FBI and web3 security firms. As team contract addresses were not disclosed, it is hard to figure out the amount and fund flow of the exploit at the time of research.

Update: Investigating the "exploit" transactions, we have suspected that this incident is not just an exploit through the private key leakage. At the day of NFP airdrop, the exploiter-marked address (0x60275d1cc368cf021547a82a51cfb8c055390da3) aggregated the \$NFP token from over 2600 EOAs that were received as airdrops and dumped the tokens to the market. If these are the assets allocated as treasury or ecosystem fund as the team initially announced, it is a critical issue caused by the team as those tokens were distributed as airdrop. If it's not, there is some missing information in the team's announcement, or it is a normal market dump from a whale which is not a hack.

Compensation Process

The token contract is redeployed and the previous tokens are not in use.

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.03.15



Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0x6027](#))

→ [0x0d27](#) (3.46M USDT)

→ [0xBE9b](#)

→ Bridged via Synapse ([0xBE9B](#), 945 ETH)

→ Distributed to multiple EOAs & Deposited to eXch



44. Wilder World (Mar 16)



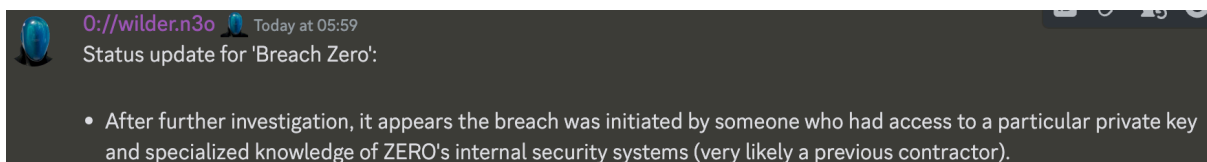
Underlying Chain: Ethereum

Amount: \$3,000,000

Attack Vector: Control Hijacking - Authorization Management Failure

Brief Summary

Wilder World, a blockchain racing game project, suffered a \$1.8 million loss as the exploiter gained access to the project deployer's private key. The exploiter upgraded major contracts of the project and drained \$WILD and \$MEOW tokens. The project suspected one of the devs to be exploiter.



Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0x6584](#))

→ [0x7f07](#) (515.5 ETH)

→ Laundered through Tornado Cash ([0x7f07](#), 515 ETH)



45. Charlotte Fang, Milady Co-founder (Mar 17)



Underlying Chain: Ethereum

Amount: \$3,000,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

Milady cofounder got his password manager compromised, which led to the leakage of seed phrases for project wallets. \$3 million of Milady NFTs and FUMO LP token were dumped to the market, causing irrecoverable damage to the project. The team initially tried to halt the multisig wallet by updating the threshold to 4/4, but it was turned to be ineffective as the exploiter had gained the control for multisig account and added himself as multisig participant and removed the others.

Compensation Process

Since Milady's primary treasury was held off-chain, no user assets are affected by the exploit.

Postmortem

[Link](#) - 2024.05.17

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0xB520](#))

→ Distributed to multiple EOAs

→ [0x8830](#), [0x407b](#) (630 ETH)

→ Laundered through Tornado Cash ([0x407b](#), 42.4 ETH) / Deposited to Kucoin ([0x8830](#) → ~50 EOAs, ~590 ETH)



46. TICKER Token (Mar 18)

Underlying Chain: Base

Amount: \$900,000

Attack Vector: Rugpull

Brief Summary

After the presale of \$TICKER, one of the dev, Jolan Lacroix, stole \$900K worth of TICKER tokens and dumped them directly to the market. After the incident, the scammer posted a tweet saying he is "not sorry" for the exit scam, mocking the presale participants. According to the investigator ZachXBT, the stolen funds were once moved to Solana, bridged back to Ethereum and spent for memecoin tradings and buying Milady NFTs. ([Ref.](#))

Pre-Incident Audit History - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0x88cb](#))

→ Bridged via Orbiter Finance (Base → Ethereum) ([0x88cb](#), 254 ETH)

→ Bridged via MayanSwap (Ethereum → Solana) ([2iLu](#), 254 WETH) → [7DyZ](#)



47. Paraswap (Mar 20)



Underlying Chain: Ethereum, Polygon, Arbitrum, Optimism, Base

Amount: \$4,500,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

Targeted contract, AugustusV6, was the newly implemented contract written in assembly to optimize gas efficiency. However, when the UniswapV3 callback of the contract was called, it failed to verify if the caller is actually the Uniswap V3 pool or not. The vulnerability allowed the exploiter to enter an arbitrary address in the fromAddress to transfer previously approved funds. Fortunately, several transactions of the exploiter were frontrun by MEV bots and the team could retrieve their assets by offering them bounties. As a result, only four addresses with \$24k were affected by the exploit.

However, the follow-up attacks exploiting the same vulnerabilities led to another \$4.5M exploit on Arbitrum, Base, Polygon and Optimism. Through the whitehat operation of SEAL 911, \$3.4M are rescued, including 10 WBTC saved by MEV whitehat 0xc0ffeebabe.eth.

Compensation Process

The team opened a form to let users submit whether they revoked approvals and proceeded for reimbursement. ([Ref.](#)) According to the team's postmortem, the first round of refunds were done on Apr 14, 2024, using the DAO treasury. ([Ref.](#))

Pre-Incident Audit History

Solidified (2021.07.22) - [Report](#), Peckshield (2022.06.10) - [Report](#),

Peckshield (2022.04.22) - [Report](#), Hacken (2024.03.01) - [Report](#),

AstraSec (2024.03.18) - [Report](#)



Post-Incident Audit History

AstraSec (2024.06.08) - [Report](#), Certora (2024.05.22) - [Report1](#), [Report2](#),

Peckshield (2024.06.06) - [Report](#), AstraSec (2024.05.28) - [Report](#),

Hacken (2024.05.13) - [Report](#), Hexens (2024.05.09) - [Report](#),

Peckshield (2024.05.26) - [Report](#)

Postmortem

[Link](#) - 2024.04.22

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#), [Tx5](#), [Tx6](#), [Tx7](#), [Tx8](#)

Fund Flow

According to the team's postmortem, the exploited assets were located in these addresses.

Hacker Address	Approx value stolen (USD)	Networks Exploited
0x000000000d67e5d5f991a40f04ed40fa3b150df	\$565,000	Ethereum, Avalanche, Arbitrum, Binance Smart Chain, Polygon, Optimism
0xe6e83f9dbc454205322c43c861ed5e8745e4e572	\$286,000	Ethereum, Arbitrum, Optimism, Base, Polygon
0x851aa754c39bf23cdaac2025367514dfd7530418	\$122,000	Arbitrum
0xdb125c32be3a183e3e447e3eb15c06103d4d7b3c	\$47,000	Arbitrum
0xcc3a5dc003b3a58621745a39f706ef9646d5c481	\$44,000	Ethereum
0x24bdce7e6e526586047ad557d1146772fa3051e4	\$28,000	Arbitrum
0xfde0d1575ed8e06fbf36256bcdfa1f359281455a	\$26,000	Ethereum
0xb79d9852ec79ee839f717105d0eb73d7cd5b0787	\$1,600	Ethereum
0x0f465dcb5dd4f1ab6c90f21ac9c12ba9627c39b7	\$1,400	Polygon



48. Dolomite Exchange (Mar 21)



Underlying Chain: Ethereum

Amount: \$1,900,000

Attack Vector: Contract Vulnerability - Improper Input Validation / Reentrancy

Brief Summary

The old contracts of Dolomite Exchange were exploited, and previously approved user assets of \$1.9 million were drained. The exploit was due to the faulty function named `callFunction`, which allowed arbitrary calls to the exploiter. Furthermore, the function was guarded with a modifier that prevents reentrancy, but it was not effective for cross-contract reentrancy through low level calls. Claiming the obsolescence of the contract, the project did not make any promise of reimbursements for affected users.

Compensation Process - N

Pre-Incident Audit History

Secbit (2021.08.02) - [Report](#), Guardian Audits (2023.01.11) - [Report](#)

Zokyo (2023.04.19) - [Report](#), Cyfrin (2023.08.23) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.03.21

Exploit Transactions - [Tx](#)



Fund Flow

After negotiation, 90% of the exploited assets are returned to the project. The exploiter laundered the other 10% through Tornado Cash, 13 days after the exploit.

Exploiter ([0x5252](#), 541.4 ETH)

→ Returned to the Project ([0x5225](#), 486.7 ETH) / Laundered through Tornado Cash (57.2 ETH)



49. AirDAO (Mar 22)



Underlying Chain: Ethereum

Amount: \$42,000,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

AirDAO team announced an exploit of 35.2 million AMB tokens and 125.51 ETH from their Uniswap pool, which was \$42 million in total. According to the team's announcement, the exploiter gained access to the liquidity pool using a social engineering scam with a malware-attached email, claiming to be one of the project's partners. The team promised further investigation and recovery of the pool. The exploit had only affected the liquidity pool.

Compensation Process - N, User funds are not affected

Pre-Incident Audit History

Athena Intelligence (2022.12.07) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.03.22

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#)

Fund Flow

Most of the exploited assets headed to ChangeNOW and MEXC, right after the exploit.

Exploiter ([0x35e5](#)) → [0xfd17](#) (126.5 ETH)

→ ... → Deposited to ChangeNOW ([0x24b9](#), [0x8BF9](#) - 30 ETH), MEXC ([0xB582](#), 56.7 ETH) /
EOA ([0x8b1f](#), 20 ETH)



50. Super Sushi Samurai (SSS) (Mar 22)



Underlying Chain: Blast

Amount: \$4,600,000

Attack Vector: Contract Vulnerability - Misimplemented Transfer Function

Brief Summary

A simple bug existed in the SSS token contract, where the exploiter could double its token balance by self-transferring the tokens. Hopefully, the exploiter claimed himself as a whitehat. The SSS team and exploiter proceeded on the reimbursement plan of stolen assets.

Compensation Process

The team redeployed the token contract, and restored the pool liquidity as the same as pre-exploit, based on the snapshot. ([Ref.](#))

Pre-Incident Audit History

Verichains (2024.03.06) - [Report](#), Verichains (2024.03.06) - [Report](#)

Post-Incident Audit History

Cantina (2024.04.18) - [Report](#), Cantina (2024.04.20) - [Report](#)

Postmortem

[Link](#) - 2024.03.22

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#)

Fund Flow

After the negotiation, 95% of the exploited assets are returned to the project. The exploiter has not moved the bounty since then.

Exploiter (0×6a89, 690M SSS)

→ [0xDed8](#) (1310 WETH)

→ EOA ([0×786C](#), 67 WETH) / Returned to the Project ([0×8195](#), 1244 WETH)



51. Ark Token (Mar 24)

Underlying Chain: BSC

Amount: \$200,000

Attack Vector: Contract Vulnerability - Access Control

Brief Summary

As the ARK token contract had set the burn function of the LP token as public, the exploiter could burn the LP tokens and manipulate the swap ratio of the pool. As a result, the contract let its 348 WBNB drained.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)



Fund Flow

The exploiter bridged the exploited assets to Tron and repeatedly deposit & withdraw the assets to CEXs (Binance, OKX, HitBTC). The assets seem to be deposited to other CEXs via Bridgers (CEX bridge) consequently.

Exploiter ([0xDd30](#))

→ [0x78ad](#) (348 BNB)

→ Transferred to SwiftSwap ([0x1ed5](#), 108 BNB) / Deposited to Binance ([0xFd28](#), [0xFd28](#), [0xFd29](#), [0xFd28](#), [0xFd2A](#), [0xFd2A](#), [0xFd27](#), [0xf2b](#) - 240 BNB)

→ Withdrew from Binance ([0x78aD](#), 210 BNB)

→ Bridged via Transit Swap (BSC → Tron) ([TLm8](#), 158K USDT)

→ Deposited to Binance, OKX

→ Repeatedly withdraw & deposit to Binance and HitBTC / Self-transfers

→ Bridged to CEXs through Bridgers



52. Curio Ecosystem (Mar 24)



Underlying Chain: Ethereum

Amount: \$100,000

Attack Vector: Contract Vulnerability - Access Control

Brief Summary

Curio, the RWA infrastructure project, announced a smart contract exploit. Due to the exploit, a MakerDAO-forked smart contract deployed in Ethereum was exploited. The root cause of the exploit was the misimplemented role, as the exploiter could gain access to execute a delegatecall to a malicious contract by holding a small amount of the governance token. Gaining the major role, the exploiter could execute a governance manipulation and mint massive amounts of tokens. After the incident, the team promised to release a new governance token and compensate users.

Compensation Process

Through the exploit recovery plan, the team announced a funds compensation program related to the newly launched governance token. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.03.26

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets haven't moved since the exploit. ([Oxdaaa](#), 26 ETH)



53. ZongZi Token (Mar 25)



Underlying Chain: BSC

Amount: \$223,000

Attack Vector: Contract Vulnerability - Price Dependency

Brief Summary

The swap rate calculation logic of ZongZi was misimplemented, as its `getAmountsOut` was naively calculated by the simple rate of ZongZi and BNB amount. By swapping huge amounts of BNB to ZongZi, the attacker could manipulate the exchange rate and drain the liquidity.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

In order to obfuscate the fund flow, the exploiter distributed the exploited assets to multiple addresses and made a bunch of simple transfers. The assets are consequently deposited to MEXC.

Exploiter ([0x2c42](#)) → [0x20F6](#) (390 BNB) → Distributed to 11 EOAs

→ Multiple (>100) simple transfers for each EOA (e.g. [0x81c6](#) → ... → [0x3e31](#), [0xE347](#) → ... → [0x1720](#))

→ Distributed once again & Deposited to MEXC (e.g. [0x3e31](#), [0x1720](#))



54. Munchables (Mar 27)






Underlying Chain: Blast

Amount: \$62,500,000

Attack Vector: Control Hijacking - Backdoor

Brief Summary

Munchables, a GameFi project on Blast, got a significant amount of its funds drained from the project contract. As the victim contract used an upgradeable proxy and was unverified, people were initially not sure whether it was a rugpull or the contract vulnerability exploit. After the exploit, on-chain investigators had identified that wallets of four project devs were related to the exploit, sharing their GitHub accounts.

 **ZachXBT**  
@zachxbt


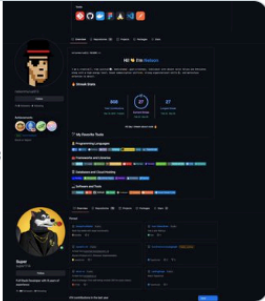
Four different devs hired by the Munchables team and linked to the exploiter are likely all the same person as they:



- >recommended each other for the job
- >regularly transferred payments to the same two exchange deposit addresses
- >funded each others wallets

Github Username
NelsonMurua913
Werewolves0493
BrightDragon0719
Super1114

Payment address
0x4890e32a6A631Ba451b7823dAd39E88614f59C97
0x6BE96b68A46879305c905CcAFF02B2519E78055
0x9976Fe30DAc6063666eEA87133dFad1d5ec27c5E

Exchange deposit address
0x84e86b461a3063ad25575b30756bdc4d051a04b
0xe362130d4718dc9f86c802ca17fe94041f1cfc77

 **Munchables**  @_munchables_ · Mar 27, 2024

Munchables has been compromised. We are tracking movements and attempting to stop the the transactions. We will update as soon as we know more.

1:29 PM · Mar 27, 2024 · 1.4M Views



As the exploiters were suspected to be the North Korean hackers, Blast community was in controversy whether the exploiter should be censored in a chain-level via sequencer or the rollback of the entire chain to consent with the regulation issues. To avoid the extreme case of the chain state rollback, Blast has censored the exploiter's transaction in a sequencer level.

Fortunately, the exploiter came out to return the whole amount of funds stolen by the exploit.



The Munchables developer has agreed to share the keys for the full Munchables funds without any condition.

1:40 PM · Mar 27, 2024 · 626.2K Views

Compensation Process

As the team retrieved their original assets, they announced the reimbursement plan for affected wallets, and finalized verification for user submissions. ([Ref.](#))

Pre-Incident Audit History

EnterSoft (2024.03.20) - [Report](#)

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

All the exploited assets are returned to the project, at the day of the exploit.

Exploiter ([0x6e88](#)) → Returned to the Project ([0x4D2F](#))

Note

On November 26 2024, Munchables announced to sunset the project. ([Ref.](#))



55. LENX Finance (Mar 27)



Underlying Chain: Ethereum

Amount: \$1,000,000

Attack Vector: Rugpull

Brief Summary

An X user @fevrdad accused the LENX Finance co-founder John Kim for cheating the team, allegedly stealing \$10M from the project's treasury. ([Ref.](#)) The stolen funds were directly deposited to Binance. The team claimed that the scammer's Binance account was frozen, but no proof was given.

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

The exploiter deposited 600 ETH to Binance. 250 ETH remains in a Safe Wallet [0x0e57](#).

Project wallet ([0x03f2](#))

→ [0x2eC5](#) (600 ETH), [0x0e57](#) (250 ETH)

→ Deposited to Binance ([0x2eC5](#), 600 ETH)

Note

The project seems not operating in its current state, as there has been no update on the official X account (@LENX_Finance) since the exploit.



56. Prisma Finance (Mar 28)



Underlying Chain: Ethereum

Amount: \$11,600,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

The exploited contract was a contract newly implemented for a special purpose to migrate user positions to another. As the contract lacked input validation on onFlashLoan function, the exploiter could arbitrarily liquidate another user's position and take a portion of the user's collateral.

```
/// @notice Flashloan callback function
function onFlashLoan(
    address,
    address,
    uint256 amount,
    uint256 fee,
    bytes calldata data
) external returns (bytes32) {
    require(msg.sender == address(debtToken), "!DebtToken");
    (
        address account,
        address troveManagerFrom,
        address troveManagerTo,
        uint256 maxFeePercentage,
        uint256 coll,
        address upperHint,
        address lowerHint
    ) = abi.decode(data, (address, address, address, uint256, uint256, address, address));
    uint256 toMint = amount + fee;
    borrowerOps.closeTrove(troveManagerFrom, account);
    borrowerOps.openTrove(troveManagerTo, account, maxFeePercentage, coll, toMint, upperHint, lowerHint);
    return _RETURN_VALUE;
}
```

After the first exploit that stole 3257 ETH, 2 different exploits occurred using the same vulnerability, losing 121 wstETH and 52 wstETH each.

Interestingly, the exploiter blamed the team for inappropriately implementing the contract, and demanded the team to reveal themselves and apologize to the users to get their funds back.

([Ref.](#))

- IDM: Before moving to the next step, I would like to move the funds to a safer place, and please answer my questions. 1, What do you think of the term "Smart Contract"? 2, Have the contract been audited before it was deployed? 3, What are the responsibilities of developers in cases like this? Im not doing this for anything but to raise better awareness on serious contract audits, on developers attitudes towards their work, and on projects responsibility.



On Apr 3, Prisma Finance unpaused the protocol and announced a post-incident audit by MixBytes. ([Ref.](#), [Ref.](#))

On Apr 16 2024, ZachXBT posted a thread about the exploiter, @0x77wn. ([Ref.](#))

Compensation Process

Due to the attack, \$11M was lost with 25 victims. Due to the massive amount of exploited assets, the project decided to proceed on a gradual reimbursement process leveraging the protocol fees. Through the governance, they decided to distribute 30% of the protocol fees to the victims for reimbursement. ([Ref.](#))

Pre-Incident Audit History

Zellic (2023.07.31) - [Report](#), Nomoi (2024.02.14) - [Report](#)

MixBytes (2023.09.01) - [Report](#)

Post-Incident Audit History

MixBytes (2024.04.06) - Report unavailable ([Ref.](#))

Postmortem

[Link](#) - 2024.03.31

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#)

Fund Flow

Most of the exploited assets are laundered through Tornado Cash and Railgun, taking 95 days.

Exploiter1 ([0x7E39](#)) → EOAs ([0x5d00](#), [0x57f7](#), [0x2d41](#) - 3257.7 ETH) → Laundered through Tornado Cash (3250 ETH) / [0x2d41](#) (7.4 ETH)

Exploiter2 ([0x7F38](#)) → Laundered through Tornado Cash (145 ETH) / Railgun (5.9 ETH)

Exploiter3 ([0x7C9F](#)) → Laundered through Tornado Cash (61 ETH) / Deposited to ChangeNOW ([0x077D](#), 0.8 ETH)



57. Lava Lending (Mar 29)



Underlying Chain: Arbitrum

Amount: \$340,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The team announced an exploit in their contract and paused all lending markets. According to the team's postmortem, the root cause of the exploit was a misimplementation in the exchange rate calculation. The donation of assets via trading fees disrupted the exchange rate. The contract had limited trading fees to a maximum APR value, but this safeguard only activated if both token0 and token1 trading fees breached the limit. This mechanism failed when only the token0-generated fees exceeded the limit.

Compensation Process - N

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.04.03

Exploit Transactions - [Tx](#)



Fund Flow

All the exploited assets are laundered through Tornado Cash, 29 days after the exploit.

Exploiter ([0x851a](#))

→ [0x102e](#) (95 ETH)

→ Bridged via deBridge (Arbitrum → Ethereum) / Synapse (Arbitrum → Ethereum) ([0xC782](#), 95 ETH)

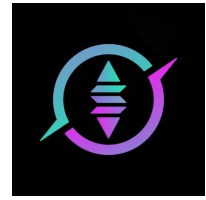
→ Laundered through Tornado Cash

Note

Lava Lending has recently been rebranded as Interflow. ([Ref.](#))



58. Solareum (Mar 30)



Underlying Chain: Solana

Amount: \$208,000

Attack Vector: Rugpull

Brief Summary

Solareum is a telegram trading bot based on Solana. Initially, another Solana trading bot Bonkbot was suspected to be the target of the exploit, but the Bonkbot team promptly denied and claimed that the users who exported the private key and imported into another service got drained. Shortly after, the community had found that the connection point of the exploit was Solareum. Solareum also denied the exploit, but later they shut down their service and websites.



Solareum

@SolareumProject

OP



There were a lot of accusations going on around about [#SOLAREUM](#) exit scam

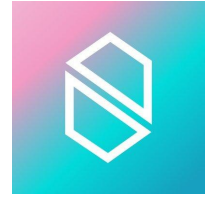
We at [#SOLAREUM](#) team can clarify that we DO NOT steal money. But a lot of [\\$SOLAR](#) users wallet got drained , but this is a part widespread exploit thats happening to other bot projects/Dapp as well.

1:11 AM · Mar 30, 2024 · **20.3K** Views

Pre-Incident Audit History - N/A



59. Nibiru (Mar 31)



Underlying Chain: CEX

Amount: \$58,600,000

Attack Vector: Circulating Supply

Brief Summary

After the CEX listings of \$NIBI, controversies on its circulating supply had emerged as CoinMarketCap showed 187 million of token supply, which was significantly different from the initial supply of 40.5 million announced by the team. Shortly after, the co-founder @ShanghaiYang posted a tweet explaining the situation.



Hi Nibiru community,

Lately, there has been a lot of chatter about @NibiruChain's Circulating Supply showing up as 187M on CoinMarketCap (coinmarketcap.com/currencies/nib...), but our tokenomics (nibiru.fi/docs/learn/tok...) shows that the token supply shouldn't reach this point until about 6 months post-launch. We've also previously claimed that the circulating supply at launch is 40.5M (t.me/nibiruchain/1/...). This has led to rumours that the project team inflated supply and dumped on the community. I'd like to explain where these numbers came from and dispel these rumours.

First, I'll explain the 40.5M number. Originally, the circulating supply at launch time was supposed to be 12M Coinlist (120M * 10%), 11.25M airdrop (50% of 22.5M), and the remainder was CEX inventory + launchpools and estimated Ecosystem Grants for projects that completed their milestones before the mainnet launch.

Based on CMC's definition of Circulating Supply (support.coinmarketcap.com/hc/en-us/artic...), "private sale" tokens are excluded, but the Coinlist sale was considered a "public sale" so we reported those tokens to CMC, even though they are still locked in smart contracts, which increased the Circulating Supply by +108M.



According to his post, the circulating supply was added as CoinMarketCap included private sales of 108 million tokens held in Coinlist, and the tokens given for ecosystem grants and hackathon prizes were included as supply. However, one of the reasons could not persuade users, as he explained that "the team moved NIBI tokens from their strategic treasury to CEX as buy pressure was much higher than expected", so that "the token could be kept liquid."



60. OpenLeverage (Apr 1)



Underlying Chain: BSC

Amount: \$236,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

Misimplemented liquidation logic led to the exploit, resulting in \$236K losses to OpenLeverage. The root cause was the misimplemented `liquidate` function, which calculates if the position is liquidatable. The function should have only used the borrowed amount that was produced by `borrow` function, but it included the assets used for margin trading. Thus, margin traders could abnormally liquidate the entire loan with a small repayment.

```
function liquidate(uint16 marketId, bool collateralIndex, address borrower) external override nonReentrant {
    controller.collLiquidateAllowed(marketId);
    // check collateral
    uint collateral = activeCollaterals[borrower][marketId][collateralIndex];
    checkCollateral(collateral);

    BorrowVars memory borrowVars = toBorrowVars(marketId, collateralIndex);
    LiquidateVars memory liquidateVars;
    liquidateVars.borrowing = OPBorrowingLib.borrowCurrent(borrowVars.borrowPool, borrower);
    liquidateVars.collateralAmount = OPBorrowingLib.shareToAmount(collateral, borrowVars.collateralTotalShare,
    borrowVars.collateralTotalReserve);

    // check liquidable
    require(checkLiquidable(marketId, liquidateVars.collateralAmount, liquidateVars.borrowing, borrowVars
    .collateralToken, borrowVars.borrowToken), "BIH");
    // check msg.sender xOLE
    require(xOLE.balanceOf(msg.sender) >= liquidationConf.liquidatorXOLEHeld, "XNE");
    // compute liquidation collateral
    MarketConf storage marketConf = marketsConf[marketId];
    liquidateVars.liquidationAmount = liquidateVars.collateralAmount;
    liquidateVars.liquidationShare = collateral;
```

Compensation Process - N

The team announced that their insurance funds could cover all the losses. ([Ref.](#)) However, it seems no reimbursement has been done and the project suspended its operation.



Pre-Incident Audit History

CertiK (2021.07.24) - [Report](#), Code4rena (2022.03.09) - [Report](#),

Peckshield (2022.11.06) - [Report](#), Peckshield (2021.12.18) - [Report](#)

Peckshield (2022.11.10) - [Report](#), Peckshield (2022.11.06) - [Report](#)

Peckshield (2023.02.15) - [Report](#), Peckshield (2023.11.02) - [Report](#)

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are directly laundered through Tornado Cash, right after the exploit.

Exploiter ([0x5bB5](#)) → Laundered through Tornado Cash (420.4 BNB)

Note

OpenLeverage has not done the reimbursement they promised. The project seems inactive in its current state, as no update has been made since May 2024. The only post on their X account after May is a retweet of \$OLE listing on Crypto.com. ([Ref.](#))



61. FixedFloat(2) (Apr 1)



Underlying Chain: Ethereum

Amount: \$3,000,000

Attack Vector: Control Hijacking - Third Party Exploit

Brief Summary

FixedFloat, the CEX that experienced a security breach in Feb 2024, again got an exploit of \$3M. FixedFloat team claimed that the exploit was not triggered by the internal system but rooted from the third-party services. The team also claimed that user funds were not affected, as the exploit only drained its liquidity funds.

Compensation Process - N/A, User fund not affected

Postmortem

[Link](#) - 2024.04.02

Exploit Transactions - [Txs](#)

Fund Flow

Exploited assets are deposited to eXch, right after the exploit.

Exploiter ([0xFA02](#)) → [0x9eFB](#) (664 ETH) → Deposited to eXch



62. ATM Token (Apr 2)

Underlying Chain: BSC

Amount: \$182,000

Attack Vector: Contract Vulnerability - Misimplemented Transfer Function

Brief Summary

Misimplemented transfer led to the exploit, resulting in \$182K losses of ATM token. When the transfer function is called, the ATM token is swapped to WBNB, and again into BSC-USD. This logic allowed the exploiter to perform sandwich attacks and manipulate the exchange rate. Interestingly, the exploiter utilized ~100 of previously deployed contracts just to move the assets to ChangeNOW.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are distributed to multiple EOAs with small balances to obfuscate the fund flow. Tracing those EOAs, the exploited assets are consequently deposited to ChangeNOW. (e.g. [0x808a](#))

Exploiter ([0x4990](#)) → Distributed to multiple EOAs → ... → Deposited to ChangeNOW



63. AvoLend Finance (Apr 3)

Underlying Chain: Blast

Amount: \$221,000

Attack Vector: Rugpull

Brief Summary

Avolend Finance, a CompoundV2 fork project on Blast, was rugged by the deployer that listed a fake token and borrowed all funds from the protocol.



CryptoDarkhorse

@Crypt0Darkh0rse OP

...

@Avolend_Finance deployer listed fake token and borrowed all funds from the protocol. Started to bridging funds.

Maybe the private key is compromised...

blastscan.io/address/0x34e4...

26 mins ago	0x34e4bfd076eb8c5284...	OUT	0xee976...
26 mins ago	0x34e4bfd076eb8c5284...	OUT	0x661cf8...
26 mins ago	0x34e4bfd076eb8c5284...	OUT	0x0b5c0...
27 mins ago	0x34e4bfd076eb8c5284...	OUT	0x3daa4...
27 mins ago	0x34e4bfd076eb8c5284...	OUT	0xec9ae...
28 mins ago	0x34e4bfd076eb8c5284...	OUT	0x76dcd...
28 mins ago	0x34e4bfd076eb8c5284...	OUT	0xe20ce...
29 mins ago	0x34e4bfd076eb8c5284...	OUT	0xe1659...
29 mins ago	0x34e4bfd076eb8c5284...	OUT	0x1f958...
29 mins ago	0x34e4bfd076eb8c5284...	OUT	0x31f7f...
29 mins ago	0x34e4bfd076eb8c5284...	OUT	0xe20ce...
29 mins ago	0x34e4bfd076eb8c5284...	OUT	0xe20ce...

11:19 PM · Apr 2, 2024 · **26K** Views

A month before the rugpull, the project was reported by the audit firm BlockApex to claim a fake audit announcement with them.



Pre-Incident Audit History - N/A

Exploit Transactions - [Txs](#)

Fund Flow

Exploited assets are bridged from Blast to Arbitrum, then to BSC. The assets are laundered at BSC Tornado Cash, 3 days after the exploit.

Exploiter ([0x34e4](#))

→ Bridged via Orbiter Finance (Blast → Arbitrum) ([0x34e4](#), 35.7 ETH)

→ Bridged via Axelar Squid (Arbitrum → BSC) ([0xEa74](#), 35 ETH)

→ Laundered through Tornado Cash ([0x34e4](#), [0x6299](#) - 105.5 BNB)



64. Condom Token (Apr 4)

Underlying Chain: Solana

Amount: \$922,000

Attack Vector: Rugpull

Brief Summary

Condom token rugged 4965 SOL in total after its presale on Solana, with their websites and social media deleted.

Pre-Incident Audit History - N/A

Fund Flow

Exploited assets stayed for 4 months in the scammer's wallet. Then, they are distributed to multiple accounts and headed to various destinations, including Stake.com, Binance, ChangeNOW. Rest of the assets are used for memecoin trading.

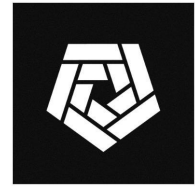
Exploiter ([4gHb](#))

→ [HTKU](#) (1700 SOL), [Cen1](#) (200 SOL), [9zfw](#) (200 SOL), [CZhJ](#) (100 SOL), [6U6E](#) (200 SOL), [ER12](#) (200 SOL), [GUso](#) (200 SOL), [7KVL](#) (1000 SOL), [9pjb](#) (900 SOL), [SGzP](#) (1743 SOL)

→ Stake.com ([SGzP](#), [9pjb](#), [7KVL](#), [DB9K](#) - 5610 SOL) / Binance ([32Lv](#), [9w3Y](#) - 42.5 SOL) / ChangeNOW ([56Lz](#), 80 SOL) / Memecoin Trading ([CZhJ](#))



65. Arkham (Apr 5)



Underlying Chain: Ethereum

Amount: \$56,000,000

Attack Vector: Circulating Supply

Brief Summary

In April 5, 2024, Nansen reported that over 25.2 million \$ARKM was moved from the Arkham's ecosystem fund to Binance, which was not planned in its token distribution plans. ([Ref.](#)) After the report, Arkham posted an announcement that the token transfer was fair, as it was decided by Arkham governance. However, only 14 addresses voted for the token transfer proposal, and an address with 96 million ARKM voted for yes. Since it is over half of ARKM's circulating supply, the proposal was entirely decided by an EOA. ([Ref.](#))



x1 ...

gm

It looks like [@ArkhamIntel](#) has moved over 25.2m \$ARKM (>\$56m) over the past 2 days















Most of these funds have gone to wallets unlabeled on their platform, where a significant portion has moved to Binance 🤔

Let's take a look at what's going on because they won't show you...

TRANSACTIONS	SWAPS	INFLUX	OUTFLOW
18 hours ago	Arkham: Airdrop (0x08c)	OpenSea User (0x1a9)	197,429 ARKM \$424.47
2 days ago	Arkham: Ecosystem Fund (0x08a)	0x0f88...df1c	2,300 ARKM \$501.90K
2 days ago	Arkham: Ecosystem Fund (0x08a)	0x0f88...df1c	1 ARKM \$2.53
2 days ago	Arkham: Ecosystem Fund (0x08a)	0x0f88...df1c	2,300 ARKM \$6.03M
2 days ago	Arkham: Ecosystem Fund (0x08a)	0x0f88...df1c	1 ARKM \$2.54
2 days ago	Arkham: Ecosystem Fund (0x08a)	0x0f88...df1c	2,300 ARKM \$6.04M
2 days ago	Arkham: Ecosystem Fund (0x08a)	0x0f88...df1c	1 ARKM \$2.54
2 days ago	Arkham: Treasury (0x094)	Gnosis Safe Proxy (0x141)	20M ARKM \$52.08M
2 days ago	Arkham: Airdrop (0x08c)	OpenSea User (0x1f4)	197,429 ARKM \$501.47
2 days ago	Arkham: Auction V2 (0x39f)	akusaraya.eth (0x7c4)	53.075 ARKM \$135.77
2 days ago	Arkham: Ecosystem Fund (0x08a)	0x0f88...df1c	19,919 ARKM \$552.08K
2 days ago	Arkham: Airdrop (0x08c)	OpenSea User (0x78a)	197,429 ARKM \$501.47
4 days ago	Arkham: Airdrop (0x08c)	amantaxaglobal.eth (0x096)	197,429 ARKM \$505.42
6 days ago	Arkham: Intel Admin (0x300)	0xA054F4488B101a42963FFFA10F624CC...	500 ARKM \$1.32K
6 days ago	Arkham: Bounty V3 (0x982)	akusaraya.eth (0x7c4)	10 ARKM \$26.30
6 days ago	Arkham: Bounty V3 (0x982)	0xA054F4488B101a42963FFFA10F624CC...	10 ARKM \$26.30

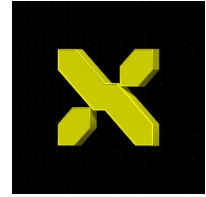
1:17 AM · Apr 5, 2024 · 478K Views



OVERVIEW	VOTES 14	DISCUSSION ↗	
Voter	Choice	Date	Voting po... ↓
<div></div> <div>0x0f88...df1c 0x0f88...df1c</div>	<div></div> For	9m ago Mar 22, 2024	96.1m ARKM 99.989%
<div></div> <div>0xbDd0...Df77 0xbDd0...Df77</div>	<div></div> For	10m ago Mar 17, 2024	3.8k ARKM 0.004%
<div></div> <div>0x83Ec...4e1d 0x83Ec...4e1d</div>	<div></div> For	10m ago Mar 17, 2024	3.6k ARKM 0.004%
<div></div> <div>0x6c04...39B9 0x6c04...39B9</div>	<div></div> For	9m ago Mar 23, 2024	1.6k ARKM 0.002%
<div></div> <div>cryptobrelgin... 0xA1b0...8D7C</div>	<div></div> For I think it's will be cool	9m ago Mar 20, 2024	513.179 ARKM 0.001%
<div></div> <div>0xF20d...5970 0xF20d...5970</div>	<div></div> For	10m ago Mar 19, 2024	300.288 ARKM 0%
<div></div> <div>thegreatlavaro... 0xfbc7c...349d</div>	<div></div> For	10m ago Mar 17, 2024	197.429 ARKM 0%



66. xBlast (Apr 9)





Underlying Chain: Blast



Amount: \$89,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

xBlast, a Telegram project with its contract deployed on Blast, got its wallet drained due to the private key leakage. The team claimed that the developers unintentionally committed the private key to their public GitHub repository. The team promised compensation for the affected users and deployment of a new token pair. ([Ref.](#))

**xBlast | Mine XBL**
@xblast_app 

[Subscribe](#)  

Announcement to the xBlast App community,

After investigating and fixing the issue, we have identified the cause of XBL being hacked and dumped in price.


Cause: During the process of publishing the smart contract source code on GitHub, our developers inadvertently attached a file containing the private key of the deployed wallet in public mode. As a result, attackers exploited the access rights and dumped XBL.

Resolution: We will use a multi-sig wallet for the entire project's wallet to optimize security. Apart from the private key leak, our entire system is secure.

We will compensate the community for damages by adding liquidity back to the new XBL pair using a new contract within the next 12 hours.

All damages will be refunded by us.

We appreciate your understanding and supports. Stay with xBlast!

**xBlast | Mine XBL** @xblast_app · Apr 9, 2024
We got hacked.

This is the wallet through which the attacker compromised the project:
arbiscan.io/address/0xae35...
...
[Show more](#)

2:22 PM · Apr 9, 2024 · **121.9K** Views



Compensation Process

3 days after the exploit, the team initiated the reimbursement program. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.04.09

Exploit Transactions - [TxS](#)

Fund Flow

Exploited assets headed to Rhino.fi Native Deposit, a day after the exploit. ([Ref.](#))

Exploiter ([0xae35](#), 22 ETH) → Bridged via Rhino.fi Native Deposit → ?

Note

xBlast launched its V2 version on Ton.



67. Pac Finance (Apr 11)



Underlying Chain: Blast

Amount: \$26,800,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

On Apr 11, 2024, \$26.8 million of ezETH were liquidated as one of the Pac Finance dev allegedly changed its Loan-To-Value (LTV). The team claimed that they are in contact with the impacted users and developing a compensation plan, mentioning that the change was unexpected without prior notification to the other team members.

Compensation Process

Due to the exploit, \$24M worth of ezETH were liquidated, resulting in a loss of \$800K of its 13 users. On Apr 13, 2024, the team announced the compensation plan.

However, we could not find the reference if the actual compensation was done or not.

Pre-Incident Audit History

Salus Security (2024.04.01) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.04.13

Exploit Transactions - [Tx](#)



Fund Flow

Abnormally liquidated assets are bridged to Ethereum, and merged to 0xF9B5. Some of them are again distributed to Ethereum L2s, including Blast, Linea, Scroll. 100 ETH are laundered through Tornado Cash, 27 days after the exploit.

Abnormal liquidation by [0xae04](#)

→ Bridged via Synapse (Blast → Ethereum) ([0xae04](#), 107 ETH), Blast Official Bridge, Orbiter Finance (Blast → Ethereum) ([0x4883](#), 158 ETH)

→ [0xF9B5](#) (221 ETH, 131K USDC, 86K USDT, 21 BETH, 39K DAI)

/ Bridged to [Blast](#), Linea ([0x00C3](#), 4 ETH), Scroll ([0x00C3](#), 6 ETH)

/ Laundered through Tornado Cash ([0xF9B5](#), 100 ETH)



68. Zest Protocol (Apr 12)



Underlying Chain: Stacks

Amount: \$964,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

Zest Protocol is a lending protocol built on Stacks, the L2(or sidechain) built on Bitcoin. Stacks uses a separate language Clarity to write smart contracts on it. According to the postmortem published by CoinFabrik, who audited Zest Protocol, `borrow-helper` contract which calls `pool-borrow` contract was inappropriately implemented. The contract has an assertion that should revert when the collateral is not enough to establish a borrow position:

Unset

```
(asserts! (<= (get collateral-needed-in-USD amount-collateral-needed) (get total-collateral-balanceUSD user-global-data)) ERR_NOT_ENOUGH_COLLATERAL)
```

Here, the calculation of total collateral had a flaw. For the total collateral, the contract calculates `user-global-data`, which uses a list of user input `assets`. As can be seen in the below code, the contract declares an array of length 100, but executes `check-assets` for only {length of `assets`} times, which can be controlled by the user input. This means the exploiter could use the same position multiple times to inflate the total collateral value.

Unset

```
(define-read-only (validate-assets-order  
  (assets-to-calculate (list 100 { asset: <ft>, lp-token: <ft>, oracle:  
<oracle-trait> })))  
  (let ((assets-used (get-assets)))
```



```

    (fold check-assets assets-used (ok { idx: u0, assets: assets-to-calculate
    }))
  )
)

(define-read-only (check-assets
  (asset-to-validate principal)
  (ret (response { idx: uint, assets: (list 100 { asset: <ft>, lp-token: <ft>,
  oracle: <oracle-trait> })} uint)))
  (let (
    (agg (try! ret))
    (asset-principal (get asset (unwrap! (element-at? (get assets agg) (get idx
agg)) ERR_NON_CORRESPONDING_ASSETS))))
    (asserts! (is-eq asset-to-validate (contract-of asset-principal))
ERR_NON_CORRESPONDING_ASSETS)

    (ok { idx: (+ u1 (get idx agg)), assets: (get assets agg) })
  )
)

```

As a result, the exploiter could borrow much more than the allowed amount for the collateral, leaving the project with a substantial amount of bad debts.

Compensation Process

According to the Zest Protocol's postmortem, the removed 322K STX from the protocol is reimbursed from the project treasury, so user funds are not affected.

Pre-Incident Audit History

CoinFabrik (2022.08.08) - [Report](#), Least Authority (2023.03.16) - [Report](#),

CoinFabrik (2024.02.19) - [Report](#)

Post-Incident Audit History

Thesis Defense (2024.05.09) - [Report](#)

Postmortem

[Link](#) - 2024.04.16 (By Zest Protocol), [Link](#) - 2024.04.12 (By ChainFabrik)



Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#), [Tx5](#)

Fund Flow

Most of the exploited STX remains unmoved in Stacks accounts.

Exploiter ([SP2H](#)) → [SP2M](#) (79K STX)

Exploiter ([SP22](#)) → [SPTC](#) (75K STX)

Exploiter ([SP9D](#)) → [SP2E](#), [SP5S](#) → [SPNM](#), [SPYX](#), [SP3P](#) (116K STX)



69. Sumer.Money (Apr 12)



Underlying Chain: Base

Amount: \$350,000

Attack Vector: Contract Vulnerability - Reentrancy

Brief Summary

Due to the reentrancy in its repay function, Sumer.Money's USDC deposit contract allowed the exploiter to manipulate the swap rate while executing the function. As can be seen in the code below, `repayBorrowBehalf` function allows the caller to repay a loan on behalf of the borrower. The low level call is used to return the residue when the received ETH is more than the debt. Here, the exploiter borrowed a massive amount of tokens via flashloaned collateral, which depleted the liquidity & crashed the swap rate, repaid the entire debt with a very small amount of assets through reentrancy, and redeemed all the collateral used for the exploit.

```
/**
 * @notice Sender repays a borrow belonging to borrower
 * @dev Reverts upon any failure
 * @param borrower the account with the debt being paid off
 */
function repayBorrowBehalf(address borrower) external payable {
    uint256 received = msg.value;
    uint256 borrows = CEther(payable(this)).borrowBalanceCurrent(borrower);
    if (received > borrows) {
        // payable(msg.sender).transfer(received - borrows);
        (bool success, ) = msg.sender.call{value: received - borrows}("");
        require(success, 'Address: unable to send value, recipient may have reverted');
    }
    (uint256 err, ) = repayBorrowBehalfInternal(borrower, borrows);
    requireNoError(err, 'repayBorrowBehalf failed');
}
```

Compensation Process - N

There has been no update nor announcement from the team regarding the exploit.

Pre-Incident Audit History

CertiK (2022.11.20) - [Report](#), CD Security (2023.09.23) - [Report](#)

Post-Incident Audit History - N/A



Postmortem - N

Exploit Transactions - [Tx](#)

Fund Flow

Most of the exploited assets are laundered through Railgun, 63 days after the exploit.

Exploiter ([0xbB34](#))

→ Bridged via Orbiter Finance (Base → Ethereum) ([0xbB34](#), 34.5 ETH), Rhino.fi (Base → Ethereum) ([0xbB34](#), 69 ETH)

→ Laundered through Railgun ([0x6603](#), 101 ETH)

/ Deposited to Binance ([0x4790](#), 0.2 ETH)

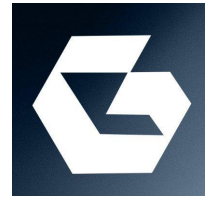
/ [0x6101](#) (2.1 ETH)

Note

Sumer.Money plans to launch on Berachain soon.



70. Grand Base (Apr 15)



Underlying Chain: Base

Amount: \$2,000,000

Attack Vector: Control Hijacking

Brief Summary

Grand Base, a RWA project on Base, experienced a security breach and lost about \$2 million. A massive amount of \$GB tokens were minted by the exploiter and dumped to the market, which caused a 99.9% drop of the token price. The team claimed the incident was caused by the compromised PC of a dev, which was accessible to the wallet with the control over the token contract & LP. The team announced relaunch of the token with post-audits by PaladinSec, and the partial recovery of the project through their \$489K worth treasury.

Compensation Process

According to the CTO, GrandBase used their treasury and team's own funds to repay the bad debt. For the affected users, the team announced to airdrop re-launched tokens based on the snapshot. ([Ref.](#)) We could not identify whether the compensation was done properly since the official X account is deleted, but it doesn't seem to have been smooth as expected. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.04.16

Exploit Transactions - [Tx1](#), [Tx2](#)



Fund Flow

The exploited assets are bridged to Ethereum and merged in an EOA, 0x3Ebc, 11 days after the exploit. The CTO of Grand Base tried to contact the address via on-chain message, but there has been no response till now. ([Ref.](#))

Exploiter ([0xa288](#)) → [0xcFe5](#)

→ Bridged via Across Protocol (Base → Ethereum) ([0xb124](#), [0x8dc2](#) - 91 ETH)

→ [0x6cc1](#)

→ [0x3EbC](#)

Note

The official X account (@Grandbase_Fi) is deleted, and the team has rebranded to Reflect (@RFLonBase).



71. Mars Token (Apr 16)

Underlying Chain: BSC

Amount: \$110,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

Mars, a DeFi project based on BSC, experienced an exploit at the day of its token launch. As MARS token in the liquidity pool was burnt when traded, the exploiter could manipulate the price of the token and drain the pool. The team paused the token contract, and deployed a new token contract with the snapshot airdrop as a compensation.

Compensation Process

The token contract was relaunched and deployed with a new liquidity pool of 100 BNB.

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

Exploited assets are deposited to ChangeNOW, 9 days after the exploit.

Exploiter ([0x810b](#)) → ... → Deposited to ChangeNOW ([0x810b](#), [0xFB38](#), [0x975d](#) - 230 BNB)

Note

The project seems inactive in its current state, since the official X account (@Mars_Defi412) is deleted.



72. Hedgey Finance (Apr 19)



Underlying Chain: Ethereum, Arbitrum

Amount: \$44,700,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

Hedgey Finance, a yield farming project based on Arbitrum and Ethereum, experienced a security breach due to the vulnerability in its token lockup contract. The root cause was in the `cancelCampaign` function, which did not revoke the approval after users withdrew their locked tokens after the campaign. Exploiting the remaining approvals, the exploiter drained approved assets on token lockup contract by transferFrom.

```
function cancelCampaign(bytes16 campaignId) external nonReentrant {
    Campaign memory campaign = campaigns[campaignId];
    require(campaign.manager == msg.sender, '!manager');
    delete campaigns[campaignId];
    delete claimLockups[campaignId];
    TransferHelper.withdrawTokens(campaign.token, msg.sender, campaign.amount);
    emit CampaignCancelled(campaignId);
}
```

Due to the exploit, massive amounts of \$NOBL tokens of Nobleblocks, \$MASA, \$USDC, \$BONUS were drained, which were worth \$42 million at the time of the exploit. Fortunately, MEV whitehat 0xc0ffeebabe successfully frontran several exploit transactions.

Regarding the incident, Nobleblock posted a detailed security analysis and future recovery plans. ([Ref.](#))

Compensation Process

On Apr 30 2024, Hedgey Finance posted a tweet about the progress of the exploit investigation. Here, they promised that they will share new updates with law enforcement and continue to do so until affected users are identified and funds are returned. However, we could not find whether the actual reimbursement process happened, as there was no update on it. ([Ref.](#))



Pre-Incident Audit History

Hacken (2022.03.31) - [Report](#), Hacken (2022.08.09) - [Report](#),

Hacken (2022.08.24) - [Report](#), Hacken (2022.09.23) - [Report](#),

Hacken (2023.01.31) - [Report](#), Hacken (2023.04.06) - [Report](#),

Consensys Diligence (2023.07.21) - [Report](#)

Post-Incident Audit History

Consensys Diligence (2024.04.22) - [Report1](#), [Report2](#),

AuditOne (2024.05.06) - [Report1](#), [Report2](#), [Report3](#), Salus Security (2024.05.17) - [Report](#),

Resonance (2024.06.20) - [Report](#), Resonance (2024.07.03) - [Report](#),

Resonance (2024.11.09) - [Report](#)

Postmortem

[Link](#) - 2024.04.30

Exploit Transactions - [Txs](#) (Provided by audit firm CUBE3.AI)

Fund Flow

Exploiter ([0xDed2](#)) → [0xd84f](#) (2M DAI, 1.7 ETH)

→ ... → Laundered through Tornado Cash ([0xf0F1](#), 664 ETH) / Leftovers on [0xC724](#) (0.87 ETH)

Exploiter ([0xC724](#))

→ Deposited to Bybit (200K BONUS) / Remains in [0xC724](#), [0x6ED7](#), [0xaD96](#) (76M BONUS)



73. ZKasino (Apr 20)



Underlying Chain: Ethereum

Amount: \$33,000,000

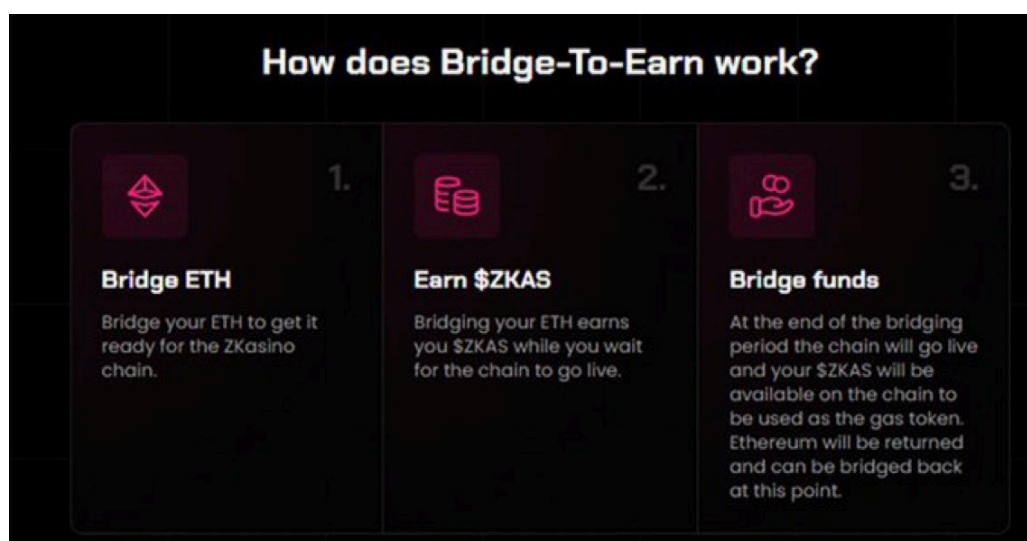
Attack Vector: Rugpull

Brief Summary

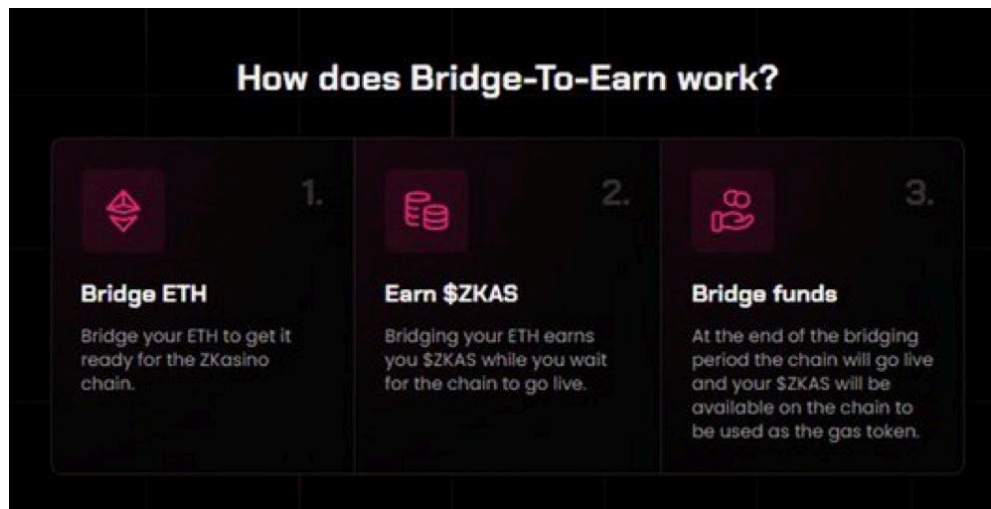
ZKasino is a gambling protocol, which had earned notable hype as it planned to launch its own chain, used ZK as a magic word and announced an integration with EigenDA. The project claimed to have raised \$26 million in its series A, backed by prominent investors such as BigBrain Holdings, MEXC and Trading Axe. ([Ref.](#)) However, suspicions arose as the founder Derivatives Ape had been involved in several failed projects including ZigZag Exchange and Syncus. As rekt.news also mentions, on-chain investigator ZachXBT had once pointed out the maliciousness of the team, such as not repaying borrowed money, not announcing winner for their giveaway, using a gore video to market their casino, etc.

After all user concerns, ZKasino founder promised the users to enable withdrawal of the deposited ETH at the time of token generation. However, the platform converted 10500 ETH into its mainnet token \$ZKAS without any user consent, which is definitely a rugpull action.

(Prev.)



(After)



The project denied the rugpull and claimed the legit launch of their mainnet based on Arbitrum Nitro, but that kinds of chains nowadays could be deployed within minutes through any RaaS. Regarding the incident, the community had identified the real identification of the team and are currently tracing them. ([Ref.](#))

Pre-Incident Audit History

ChatGPT (2022.12.24) - [Report](#), CertiK (2023.01.21) - [Report](#)

Exploit Transactions - [Tx](#)



Fund Flow

The related addresses have been very active since the exploit, as the latest transactions observed happened around Dec 24 2024.

ZKasino ([0x9144](#)) → Exploiter's Safe Wallet ([0x7911](#), 10515 ETH)

→ [0x42dC](#), [0x0Ab4](#) → [0x678f](#), [0x3722](#), [0xAb91](#), [0x678f](#), [0x5763](#), [0x808b](#), [0x6488](#) (wstETH)

→ EOAs / Deposited wstETH to Aave V3 & Borrowed DAI, Bought ETH ([0x678f](#)) / Bridged via zkSync Official Bridge (Ethereum → zkSync Lite → Ethereum) ([0xc420](#), 1427.8 ETH)

→ [0x3722](#) (2315 wstETH), [0x9B1d](#) (27 ETH), [0xAb91](#) (2.9M DAI), [0xB5Df](#) (8.9 ETH) / Distributed to multiple addresses ([0xCCBB](#), 18 ETH), Deposit to Ether.fi ([0xc420](#) - 1899.57 weETH, 2.04M DAI, 1.01M USDC, 57K USDT)

Note

Regarding the incident, Big Brain Holdings posted a tweet that they did not directly invest in ZKasino, but the ZigZagExchange (the previous project of the founder) in 2022 that has sunsetted. ([Ref.](#))

On May 3 2024, Dutch authorities arrested a 26-year-old ZKasino founder (suspected to be @Derivatives_Ape), and \$12.2M of assets were frozen. ([Ref.](#)) On May 9 2024, Binance announced that they will lead the recovery process of the rugpull. ([Ref.](#)) On May 28 2024, @Derivatives_Ape announced the bridge back process to receive deposited ETH at 1:1 ratio. ([Ref.](#)) However, only 72 hours were given to sign up for the process. ([Ref.](#)) According to Cointelegraph's news on Aug 14 2024, users haven't got their ETH back while two months have passed from the claim process (and even now). On Nov 15 2024, @Derivatives_Ape came back to X with a gm post. On Nov 23 2024, @EmberCN posted a tweet that ZKasino has generated \$3.2M of unrealized profit by leverage-trading user assets on Aave. ([Ref.](#))



74. Z123 Token (Apr 22)

Underlying Chain: BSC

Amount: \$136,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

Misimplemented logic in Z213's swap function led to the price manipulation exploit. When the `update` function is called, Z123 tokens in QiaoLP are burned. The exploiter swapped USDT to Z123 and back several times and manipulated the pool's swap ratio.

```
function update(address pair,uint256 amount) public onlyMinter {
    super._transfer(pair, 0x00000000000000000000000000000000dEaD, amount);
}
```

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

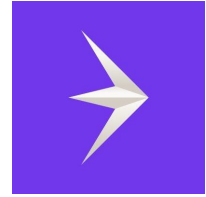
Postmortem - N/A

Exploit Transactions - Tx

Fund Flow

Exploiter ([0x3026](#)) → Bridged via LayerZero Stargate (BSC → Kava) ([0x3026](#))

75. Magpie Protocol (Apr 23)



Underlying Chain: Ethereum, Arbitrum, Avalanche, Optimism, Polygon

Amount: \$129,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

Magpie Protocol is a multichain DEX based on Ethereum. Due to the misimplemented validity check on selector in the router contract, the exploiter could lure the router contract by creating an address which starts with one of the approved selector. Utilizing the exploited router, the exploiter executed transferFrom functions of the tokens and \$129k of previously approved amount of user assets were drained.

Compensation Process

With the publication of the postmortem, the team sent the reimbursements to all the affected wallets. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History

QuillAudits (2024.11.20) - [Report](#)

Postmortem

[Link](#) - 2024.04.27

Exploit Transactions - [0xe8ed](#) on Multiple chains



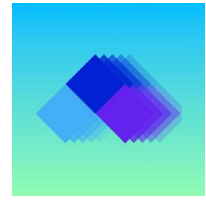
Fund Flow

Exploited assets from various chains were merged to Ethereum and laundered through Tornado Cash, 13 days after the exploit.

Exploiter ([0xe8ed](#)) on multiple chains → Bridged via Jumper Exchange & Merged to [0x3721](#) → Laundered through Tornado Cash (24.3 ETH)



76. YIEDL (Apr 24)



Underlying Chain: BSC

Amount: \$300,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

YIEDL, an AI-powered DeFi protocol based on Optimism and BSC, took an exploit that drained the assets of its spot vault contract on BSC. The root cause of the exploit was lack of input verification in the redeem function. Due to the vulnerability, the exploiter could redeem tokens without any share, making profits of \$300K.

Compensation Process

On Apr 29 2024, the team clarified that the affected assets belong to the team, not the users.. ([Ref.](#))

Pre-Incident Audit History

ImmuneBytes (Unknown) - Report unavailable ([Ref.](#))

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.04.25

Exploit Transactions - [Tx](#)

Fund Flow

All the exploited assets are laundered through Tornado Cash, right after the exploit.

Exploiter ([0x3226](#)) → [0xcAc8](#) → Laundered through Tornado Cash (262.5 BNB)



77. SaitaChain xBridge (Apr 24)



Underlying Chain: Ethereum, BSC

Amount: \$1,600,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

SaitaChain's cross-chain platform xBridge was exploited due to the vulnerability in the bridge contract. As the contract failed to verify whether the msg.sender is the token owner, it allowed the exploiter to drain \$1.6 million worth of \$STC token. After the exploit, the team announced the migration of the project from Ethereum to BSC.

Compensation Process - N/A

We could not find any reference that reimbursement process has been done after the exploit.

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.04.24

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

Exploited assets are laundered through Tornado Cash, right after the exploit.

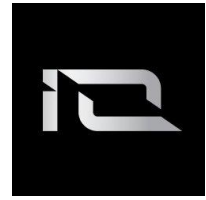
Exploiter ([0x0cFc](#)) → Laundered through Tornado Cash (231 BNB, 20 ETH) / [0cFc](#) (802M STC)

Note

On May 25 2024, SaitaChain announced the sunset of the STC token on Ethereum. ([Ref.](#)) On Dec 25, xBridge was unpaused. ([Ref.](#))



78. IO.NET (Apr 25)



Underlying Chain: Solana

Amount: Unknown

Attack Vector: Web2 Vulnerability

Brief Summary

IO.NET is a decentralized infrastructure project based on Solana, which has been running its incentivized farming program starting from March 2024. While the users should register and run the program via their GPUs, the vulnerability in the node runner API led to the massive amount of fake GPU registration.

The exploiters targeted several APIs of the project to register fake GPUs:

- 1) The API meant to display content within the project explorer unintendedly exposed user IDs when searched by device IDs, and exploiter collected the data to identify who had the most GPUs for the airdrop.
- 2) The API used within the worker routine crons could update any device metadata if you pass the owner's user ID in the header without any user-level authentication.

After 26 hours from the incident, the project ran the proof of work system and blocked fake GPUs from the network. However, as a revenge of being banned, the exploiters manipulated the devices of the other users using the second vulnerable API. After resolving the issue, the IO.NET team implemented an additional user-specific authentication solution using Auth0 with OKTA.

Postmortem

[Link](#) - 2024.04.27



79. FENGSHOU Token (Apr 26)

Underlying Chain: BSC

Amount: \$190,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

A token named FENGSHOU(NGFS) took an exploit shortly after its deployment, due to the vulnerability in its contract. As the token contract deployer did not initialize the contract after deployment, the `delegateCallReserves` function which could be called only once was triggered by the exploiter. ([Ref.](#))

The exploiter called the function, set a malicious address to UniswapV2 proxy, and arbitrarily increased his token balance using `reserveMultiSync` function. The tokens were dumped, and the exploiter transferred the stolen assets through Tornado Cash.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

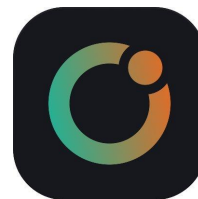
Fund Flow

All the exploited assets are laundered through Tornado Cash, right after the exploit.

Exploiter (0xD03D) → Laundered through Tornado Cash ([0xD03D](#), [0x78e5](#) - 327.2 BNB)



80. Pike Finance (Apr 26, Apr 30)



Underlying Chain: Ethereum, Arbitrum, Optimism

Amount: \$1,980,000

Attack Vector: Contract Vulnerability - Improper Input Validation / Storage Collision

Brief Summary

The first exploit was due to the misimplemented logic of the contract implemented by the team to integrate CCTP led to the exploit. As the function designed for burning USDC on a source chain and minting on a target chain had a critical flaw, the exploiters could manipulate receiver's address and amounts. The stolen assets were laundered through Tornado.Cash.

Four days after its first exploit on its CCTP integration contract, Pike Finance faced its second exploit with over five times the amount. The vulnerability was related to the patch of the first exploit, which added `pause()` and `unpause()` in the contract. Due to the added dependency in the code, the patch modified the storage layout and moved initialized variables. Exploiting this, the attackers initialized the upgraded implementation contract, took the owner authority, and drained the assets in the contract. The stolen assets were laundered through Railgun at this time. After the second exploit, the project announced a detailed reimbursement plan with the user asset snapshot.

Compensation Process

On May 13 2024, the team posted their restitution analysis([Ref.](#)), and the affected users were reimbursed based on their analysis.

Pre-Incident Audit History

OtterSec (N/A) - Report unavailable ([Ref.](#))

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.04.28, [Link](#) - 2024.05.09



Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#) / [Tx4](#), [Tx5](#), [Tx6](#)

Fund Flow

Exploited assets are bridged to Ethereum and laundered through Railgun, right after the exploit.

Exploiter ([0x1906](#))

→ Bridged via LayerZero Stargate (Arbitrum → Ethereum / Optimism → Ethereum) ([0x1906](#))

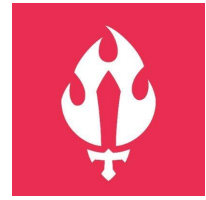
→ Laundered through Railgun (562 ETH)

Note

Pike Finance relaunched the project in Oct 2024. ([Ref.](#))



81. Ember Sword NFT (Apr 28)



Underlying Chain: Polygon

Amount: \$240,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

Ember Sword NFT is a gameFi project based on Polygon. As the contract was left uninitialized after its deployment, the exploiter could claim the owner role by initializing the contract and draining previously approved WETH from 159 victims. However, the team has not been making any response or reimbursement plan regarding the incident.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged via XY Finance and laundered through Tornado Cash, 16 days after the exploit.

Exploiter ([0x268a](#))

→ Bridged via XY Finance (Polygon → BSC) ([0xf077](#))

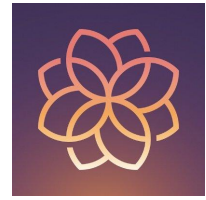
→ Laundered through Tornado Cash ([0xf077](#), 323.4 BNB)

Note

No official announcement has been made from the team regarding the exploit.



82. Rain Exchange (Apr 29)



Underlying Chain: Ethereum

Amount: \$14,100,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

Rain Exchange is a Bahrain-based CEX that runs services in Middle East Asia and Turkey. On May 14, after two weeks from the incident, the team admitted the occurrence of the security breach and announced the resolution of the incident.

According to the seizure warrant filed by the Justice Department, it was Lazarus (North Korean hacker group) that exploited Rain. They contacted an employee on LinkedIn with a job offer, sent a TraderTraitor malware disguised as a coding challenge. ([Ref.](#))

Compensation Process

The team claimed that all the loss resulting from the incident was fully covered by their own assets. ([Ref.](#))

Postmortem

[Link](#) - 2024.05.14

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

Exploited assets on Ethereum are laundered through Tornado Cash, 102 days after the exploit.

Exploiter ([0x197b](#)) → Laundered through Tornado Cash ([0x6b04](#), [0x3751](#) - 1880 ETH)

Exploiter ([bc1q53](#)) → Distributed to various addresses



83. Perpy Finance (May 6)



Underlying Chain: Arbitrum, BSC, Base

Amount: \$132,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

Perpy Finance is a socialFi project based on Arbitrum, Base, and BSC. According to the team's postmortem, uninitialized proxy contract for the staking liquid module led to the exploit. Regarding the exploit, the team bought back \$PRY dumped by the hackers and completed redistribution to the users. Notably, Perpy Finance exploiter is suspected to be the same entity on the Tsuru Exploit, which happened a week after.

Compensation Process - N/A

Pre-Incident Audit History

Peckshield (2023.08.03) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.05.10

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#)



Fund Flow

Exploited assets are bridged to Ethereum and laundered through Tornado Cash and Railgun, 46 days after the exploit.

Exploiter ([0xcbB4](#))

→ Bridged via LayerZero Stargate (Arbitrum → Ethereum) ([0x21bE](#), 41.9 ETH)

→ Laundered through Tornado Cash, Railgun

Note

Perpy Finance has rebranded to Social Trade. ([Ref.](#)) Most of the posts about Perpy Finance posted in 2024 are now deleted.



84. OSN Token (May 6)

Underlying Chain: BSC

Amount: \$110,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

Misimplemented reward distribution logic led to the exploit. As users add liquidity, the `OSNLPRDividendTracker` contract sells its own tokens as a reward. Additionally, the transfer function of the OSN token contract sells its own functions and sends BUSD to the reward tracker. Therefore, as the exploiter repeatedly called those functions to buy and sell OSN tokens, the reward distribution contract accumulated large amounts of rewards. As the accumulated rewards were distributed to the liquidity provider, the exploiter added a relatively small amount of liquidity and drained the rewards in the contract.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [TxS](#)

Fund Flow

Exploited assets are laundered through Tornado Cash, 2 days after the exploit.

Exploiter ([0x835b](#)) → Laundered through Tornado Cash (25 BNB)



85. GNUS (May 6)



Underlying Chain: Fantom

Amount: \$1,280,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

The InterchainProxy contract, one of the core contracts for supporting Axelar bridge, was allegedly initiated by the exploiter and a new \$GNUS token manager contract was deployed on the Fantom chain. With the new token manager contract, the exploiter minted 100M fake GNUS token on Fantom network.

@SuperGeniusEth, the founder of GNUS, stated that a dev's wallet was compromised when their discord was hacked and private messages were exposed. ([Ref.](#))

Compensation Process

The team announced buyback of the exploited \$GNUS based on a snapshot. ([Ref.](#), [Ref.](#), [Ref.](#))

Pre-Incident Audit History

SolidProof (2024.02.28) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.05.06

Exploit Transactions - [Tx](#)



Fund Flow

Exploited assets are bridged to Solana, and deposited to LBank as SOL 5 days after the exploit.

Exploiter ([0x548c](#))

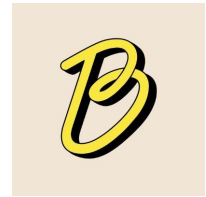
→ Bridged via Axelar (Fantom → Ethereum) ([0x548c](#))

→ Bridge via Allbridge (Ethereum → Solana) ([BnGo](#), [9bDb](#), [5dzS](#), [4KF9](#))

→ Deposited to LBank ([H3Lz](#), [4UiA](#) - 173K USDT & 165K USDC & 455 SOL)



86. Bloom (May 9)



Underlying Chain: Blast

Amount: \$540,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

Misimplemented logic of handling Blast-native yields when updating cash for a position led to the exploit. Since the logic that provided extra collateral through accruing native yields over time for positions in losses was buggy, the exploiter could continuously withdraw USDB repeatedly without limit when a position had a negative PnL.

Compensation Process

As 90% of the exploited assets were recovered, the team released a plan to redistribute LPs.

Pre-Incident Audit History

Zellic (2024.02.23) - [Report](#)

Postmortem

[Link](#) - 2024.05.14

Exploit Transactions - [Tx](#)

Fund Flow

80% of the exploited assets are returned to the project, at the day of the exploit.

Exploiter ([0x60E3](#))

→ Bridged via Synapse (Blase → Ethereum) ([0xd696](#), 200.8 ETH)

→ Returned to the Project ([0x1e40](#), 160 ETH) / Deposited to Binance (40.8 ETH)



Note

Bloom announced the sunset of the project on Jun 25 2024. ([Ref.](#))



87. Galaxy Fox Token (May 10)



Underlying Chain: Ethereum

Amount: \$330,000

Attack Vector: Contract Vulnerability - Access Control

Brief Summary

Due to the access control vulnerability, Galaxy Fox Token had lost over \$300K due to the exploit. As the function that calls setMerkleRoot was open to the public, the exploiter could call the function before the claim function and drain the victim contract.

Compensation Process - N/A

Pre-Incident Audit History

Cyberscope (2024.01.27) - [Report](#)

Post-Incident Audit History

Cyberscope (2024.05.14) - [Report](#), Cyberscope (2024.05.17) - [Report1](#), [Report2](#), [Report3](#)

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets were laundered through Tornado Cash / deposited to Binance, 216 days after the exploit.

Exploiter ([0xfce1](#))

→ Laundered through Tornado Cash (8.1 ETH) / Bridged via Celer Network (Ethereum → BSC) ([0x2529](#), 33.4 ETH), XY Finance ([0x4C6a](#), 179.2 BNB)

→ Laundered through Tornado Cash ([0x4C6a](#), 399.2 BNB)



88. Tsuru (May 10)



Underlying Chain: Base

Amount: \$410,000

Attack Vector: Contract Vulnerability - Access Control

Brief Summary

As the validation lacked for onERC1155Received function, anyone could mint the TSURU token and swap it on Uniswap.

```
function onERC1155Received(
    address,
    address from,
    uint256 id,
    uint256 amount,
    bytes calldata
) external override nonReentrant returns (bytes4) {
    require(id == tokenId, "Token ID does not match");

    if (msg.sender == address(erc1155Contract)) {
        return this.onERC1155Received.selector;
    }

    _safeMint(from, amount * ERC1155_RATIO); // Adjust minting based on the ERC1155_RATIO
    return this.onERC1155Received.selector;
}
```

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.05.11 (Archived, Currently unavailable)

Exploit Transactions - [Tx](#)



Fund Flow

Notably, the exploited assets are merged with the assets stolen from Perpy Finance and laundered through Tornado Cash and Railgun. Laundry was done 42 days after the exploit.

Exploiter ([0x7A5E](#))

→ Bridged via Across Protocol (Base → Ethereum) ([0x5E20](#), 138.7 ETH)

→ Laundered through Tornado Cash, Railgun

Note

The project seems to have sunsetted in its current state.



89. Predy Finance (May 14)



Underlying Chain: Arbitrum

Amount: \$464,000

Attack Vector: Contract Vulnerability - Access Control

Brief Summary

The root cause of the exploit was the misimplemented Locker role of the lockup contract. The exploited contract has a take function that transfers the token, which has an onlyByLocker modifier.

```
/**
 * @notice Transfers tokens. It can only be called from within the `predySettlementCallback` and
 * `predyTradeAfterCallback` of the contract that invoked the trade function.
 * @dev Only the current locker can call this function
 * @param isQuoteAsset Whether the token is quote or base
 * @param to The address to transfer the tokens to
 * @param amount The amount of tokens to transfer
 */
function take(bool isQuoteAsset, address to, uint256 amount) external onlyByLocker {
    globalData.take(isQuoteAsset, to, amount);
}
```

Meanwhile, the locker value can be set at trade function, which included a callback logic that lacked access control.

```
/**
 * @notice Trades perps and squarts. If vaultId is 0, it creates a new vault.
 * @param tradeParams trade details
 * @param settlementData byte data for settlement contract.
 * @return tradeResult The result of the trade.
 */
function trade(TradeParams memory tradeParams, bytes memory settlementData)
    external
    nonReentrant
    returns (TradeResult memory tradeResult)
{
    globalData.validate(tradeParams.pairId);

    if (globalData.pairs[tradeParams.pairId].allowListEnabled && !allowedTraders[msg.sender][tradeParams.pairId]) {
        revert TraderNotAllowed();
    }

    tradeParams.vaultId = globalData.createOrGetVault(tradeParams.vaultId, tradeParams.pairId);

    return TradeLogic.trade(globalData, tradeParams, settlementData);
}
```



Through the callback, anyone could add and register a trading pair via Uniswap and set the locker value. After the locker value is set, the exploiter could gain the locker role and withdraw all the tokens in the contract.

Compensation Process

The project posted a tweet that most of the affected assets are the project funds. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History

Code4rena (2024.07.09) - [Report](#)

Postmortem

[Link](#) - 2024.05.16

Exploit Transactions - [Tx](#)

Fund Flow

All of the exploited assets are bridged to Ethereum and laundered through Tornado Cash, 109 days after the exploit.

Exploiter ([0x76b0](#), [0xe178](#))

→ Bridged via Across Protocol (Arbitrum → Ethereum) ([0xeEe4](#), 101 ETH)

→ Laundered through Tornado Cash ([0xeEe4](#), [0x8D63](#))



90. ALEX & XLink Bridge (May 15)



Underlying Chain: Stacks

Amount: \$31,700,000 (\$4,300,000 & \$27,400,000)

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

Alex is the biggest DeFi protocol in Stacks, the Bitcoin L2/sidechain network. With the private key obtained from a phishing attack, various protocols that are related to Alex took exploits.

On May 15, XLink, the Bitcoin bridge operated by Alex, had its bridge contracts on Ethereum and BSC drained. The stolen assets were about \$4.3 million, but the team announced that the assets were fully recovered with the help of a whitehat.

On May 16, after XLink announced the full recovery of its assets, Alex team informed that one of its vaults associated with the liquidity pool was drained due to the private key exploit. The stolen assets included \$27.4 million of STX while \$6 million of them were sent to various CEXs. The team announced that the other various assets except STX were fully recovered.

Regarding the second exploit, there has been no update on asset recoverage, but the team has been continuously communicating with the community and preparing several treasury grant programs and reopening plan to compensate its users.

Compensation Process

On May 22 2024, AlexLab posted a tweet about asset recovery plan and treasury grant proposal. On Jun 2 2024, the direction of the treasury grant program was decided by the governance. Among the proposals, the plan to fairly distribute the remaining STX in the project vault to cover LP loss and user compensation was selected with 19% of voting power. ([Ref.](#)) Through another proposal on Jun 2 2024, the remaining STX was decided to be directly transferred to affected LP holders. ([Ref.](#)) On Jun 6 2024, affected STX-paired pools were recovered. ([Ref.](#)) Regarding the user compensation, the proposal of utilizing 100% of the protocol revenue for recovery was passed on July 9 2024. ([Ref.](#)) The revenue distribution was done on Sep 29 2024 and Dec 18 2024. ([Ref.](#))



Pre-Incident Audit History

CoinFabrik (2021.11) - [Report](#), CoinFabrik (2022.01.13) - [Report](#),

Least Authority (2022.02.25) - [Report](#), CoinFabrik (2023.10) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.05.16

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

1) XLink Exploit:

Exploited assets from XLink were returned to the project. ([Tx](#))

Exploiter ([0x2705](#)) → Returned to the project ([0x321f](#) - 14.4 BTC, 2.96M USDT, 2.75M SKO)

2) ALEX Exploit:

Exploited assets are laundered through Tornado Cash. Laundry was done 148 days after the exploit.

Exploiter ([0x2705](#))

→ Bridged via XY Finance & Laundered through Tornado Cash ([0xbD86](#), 117.8 ETH) /

Laundered through Tornado Cash ([0x73b7](#), [0x2705](#) - 159.6 BNB)



91. BlockTower Capital (May 15)



Underlying Chain: Web2

Amount: Unknown

Attack Vector: Unknown

Brief Summary

BlockTower Capital, a crypto investment firm with \$1.7B of assets managing, was reported to have its main hedge fund partially drained due to the security breach. ([Ref.](#)) The details of the exploit and the amount of exploited assets were not disclosed.



92. Sonne Finance (May 15)



Underlying Chain: Optimism

Amount: \$20,000,000

Attack Vector: Contract Vulnerability - Precision Loss (Compound V2 Fork)

Brief Summary

Sonne Finance is a DeFi protocol based on Optimism and Base, which is basically a CompoundV2 fork. Compound V2-forked contracts often have a price manipulation vulnerability due to the precision loss when the pool is deployed without initialization & lack of liquidity. To handle this, Compound V2 implemented a timelock contract to deploy and add liquidity to the new pool.

However, in the case of Sonne Finance, the access to execute the transactions after the timelock was open to the public. Therefore, the exploiter could execute the pool deployment transaction before the team and manipulate the swap ratio just as the previous CompoundV2 fork price manipulation exploits.

At the day of the exploit, FuzzLand detected the exploit transaction and soVELO pool could be the next target of the exploit. By adding \$100 of soVELO pool, they saved \$6.5M of assets that might have been affected. ([Ref.](#))

Compensation Process - N/A (@SonneFinance is suspended, so we could not find reference)

Pre-Incident Audit History

yAudit (2023.05.21) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.05.15

Exploit Transactions - [Tx](#)



Fund Flow

Exploited assets were bridged from Optimism to Ethereum and laundered through Tornado Cash, 10 days after the exploit.

Exploiter ([0x5D0D](#))

→ Bridged via Across Protocol ([0x6277](#), [0x87fE](#), [0x0Dea](#), [0xc443](#), [0xd9Cf](#), [0x44Bc](#), [0xd0b7](#) - 41 WBTC, 1862 ETH, 310K DAI), LayerZero Stargate ([0x5D0D](#), [0x44Bc](#), [0xd0b7](#) - 2336.5 ETH)

→ [0xB238](#) (1001 ETH) / Laundered through Tornado Cash ([0x606c](#), [0xfA62](#), [0xdBBC](#), [0xd0b7](#), [0x524f](#))

Note

It seems that Sonne Finance is not operating in its current state. All the liquidity in the pools are shown to be depleted. ([Ref.](#))



93. Pump.fun (May 17)



Underlying Chain: Solana

Amount: \$2,000,000

Attack Vector: Control Hijacking - Authorization Management Failure

Brief Summary

Pump.fun is a memecoin trading platform based on Solana. As the ex-employee @STACCoverflow allegedly established privileged positions, the project experienced a critical security breach. The exploiter flashloaned SOL from marginfi, bought massive amounts of memecoin so that the bonding curve goes 100% and gains access to the liquidity pools. Profiting by draining the liquidity, the exploiter gained profits and repaid the flashloan.

Compensation Process

Regarding the exploit, the project announced the buyback of affected liquidity pools. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History- N/A

Postmortem

[Link](#) - 2024.05.17

Note

The exploiter, @STACCoverflow (Jarett Dunn) was arrested in London. ([Ref.](#)) In Aug 2024, Jarett Dunn pleaded guilty in London Court. ([Ref.](#)) In Oct 2024, right before his sentencing, he withdrew his guilty plea, and his legal team has quit the case with this move. ([Ref.](#)) If the court sentences him guilty for his two charges, Dunn will face a minimum of seven years in prison.



94. TCH Token (May 17)

Underlying Chain: BSC

Amount: \$19,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

Due to the vulnerability in its ECDSA-based signature verification, TCH token contract allowed the exploiter to call burnToken function which was set to be called only by the authorized signer. As the token contract stored used signatures in a mapping to prevent signature replay, the exploiter utilized it for the attack by modifying v of the ECDSA signature, leveraging malleability of ECDSA signatures. As a result, the exploiter burnt massive amounts of TCH token, manipulated the token price and took profit.

Compensation Process- N/A

Pre-Incident Audit History- N/A

Post-Incident Audit History- N/A

Postmortem- N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are directly laundered through Tornado Cash, right after the exploit.

Exploiter ([0xB959](#)) → Laundered through Tornado Cash (32 BNB)



95. Gala (May 20)



Underlying Chain: Ethereum

Amount: \$21,800,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

Due to the private key leakage, \$GALA token contract allowed the exploiter to mint 5 billion additional \$GALA tokens and dump 10% of them to Uniswap. After 45 minutes from the exploit, the team managed to lock 90% of allegedly minted tokens. Fortunately, the exploiter returned his entire profits, which were used for \$GALA buyback. As post-exploit actions, the team upgraded the token contract and burned almost 1 billion of \$GALA token.

Compensation Process

Exploiter returned the entire exploited assets to the team. Using this ETH, the team purchased \$GALA tokens and burned them. ([Ref.](#))

Pre-Incident Audit History

Anchain.ai (2023.02.17) - [Report](#), CertiK (2023.04.20) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.05.27

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0xEC78](#)) → Returned to the Project ([0xfdF0](#), 5914.9 ETH)



96. YESorNO Token (May 22)

Underlying Chain: BSC

Amount: \$118,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

Poorly implemented transferFrom function in the unverified \$YON token contract led to the exploit. Leveraging the vulnerability, the exploiter directly transferred \$YON in the LP contract to the exploit contract, manipulating the exchange rate.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

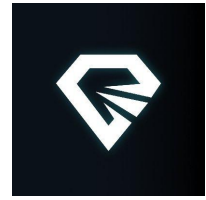
Fund Flow

Exploiter staked the exploited assets at ListaDAO, 83 days after the exploit.

Exploiter ([0xabc92](#)) → Staked at ListaDAO ([0x02ac](#), 88.1 BNB)



97. TonUP (May 23)



Underlying Chain: Ton

Amount: \$107,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

TonUP, a token launchpad platform based on Ton, experienced a security breach as the misconfigured parameters in the contract deployment script allowed users to allegedly claim staked \$UP.

Compensation Process

The team utilized project funds, \$107K for token buyback procedure and 307K \$UP to be returned to the staking contract. User assets are not affected.

Pre-Incident Audit History - N/A

Post-Incident Audit History

TonBit (2024.5.24) - [Report](#)

Postmortem

[Link](#) - 2024.05.24



98. Normie Token (May 26)



Underlying Chain: Base

Amount: \$882,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

Normie, a memecoin project based on Base, was exploited due to the vulnerability in its token contract. They implemented a premarket user list that makes the balance of token contract increases when the premarket user calls transfer function. However, it was poorly verified as it only checked if the transfer amount is the same as the balance of the team wallet.

```
function _get_premarket_user(address _address, uint256 amount) internal {
    premarket_user[_address] = !premarket_user[_address]
        ? (amount == balanceOf(teamWalletAddress))
        : premarket_user[_address];
}
```

Furthermore, premarket users could bypass the checks in swapAndLiquify function, which is authorized to mint additional tokens. To satisfy the condition, the exploiter flashloaned the tokens, manipulated the token supply and dumped 4.65 million allegedly minted NORMIE tokens.

Right after the exploit, @sizeab1e came out to be the exploiter ([Ref.](#)).

Compensation Process

The team relaunched the token contract utilizing the 500+ ETH in the deployer wallet and refunded the users via new token on Jun 7 2024. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A



Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0xf7f3](#))

→ [0xbDfC](#) (200 ETH)

→ Returned to the Project ([0x7E20](#), 192 WETH) / [0xF183](#) (8 ETH)



99. Orion Protocol (May 28)



Underlying Chain: Ethereum

Amount: \$616,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

Orion Protocol, which previously experienced a security breach in Feb 2023, took its second exploit due to the mishandling of the staked token. The liability of the token, which should have been set only once, could be set again through requestReleaseStake function. This allowed the exploiters to withdraw their staked tokens multiple times.

Compensation Process - N

The team has not announced anything regarding the exploit.

Pre-Incident Audit History

Certik (2021.07.22) - [Report](#)

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged to Ethereum and deposited to ChangeNOW, 7 days after the exploit.

Exploiter ([0x5117](#))

→ Bridged via Connex (BSC → Ethereum) ([0xcFC9](#), 25 ETH)

→ Deposited to ChangeNOW ([0x077D](#), 25 ETH)



100. MetaDragon (May 29)



Underlying Chain: BSC

Amount: \$180,000

Attack Vector: Contract Vulnerability - Access Control

Brief Summary

MetaDragon is a ERC-404 based project, which allows conversion of an ERC-721 token with a certain amount of ERC-20 tokens. In the `_erc721ToErc20` function of the MetaDragon token contract, the require statement that should have checked whether the caller is token owner was simply commented out. Due to this simple vulnerability, the exploiter could allegedly mint additional amounts of token and dump them to the market.

Compensation Process - N/A, the X account of the project has been deleted.

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are directly laundered through Tornado Cash, right after the exploit.

Exploiter ([0xc468](#)) → Laundered through Tornado Cash (321.5 BNB)

Note

The project seems to be inactive in its current state, since the project's X account has been deleted.



101. DMM Bitcoin (May 31)



Underlying Chain: Bitcoin

Amount: \$305,000,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

DMM Bitcoin, a Japanese CEX, was exploited and lost 4502 BTC, which is worth more than \$304 million, from its hot wallet.

On Dec 23 2024, FBI's publication revealed that the exploiters of DMM Bitcoin were Lazarus. Similar to the Rain Exchange's case, the TraderTraitor malware that was disguised as a legit file compromised the server of multiple employees. According to the publication, in Mar 2024, Lazarus disguised as a recruiter on LinkedIn and compromised the system of Ginco, the wallet management service provider of DMM. Leveraging the compromised authorization, the exploiters injected a link with malicious Python script on a GitHub page, under disguise as a pre-employment test. In May 2024, exploiters hacked session cookie info to impersonate Ginco's employee and gained access to the communications system. Then, they contacted a DMM employee with a legit transaction request and manipulated it, which drained DMM Bitcoin's assets. ([Ref.](#))

On Dec 2 2024, DMM Bitcoin announced its liquidation and ceased the operation. ([Ref.](#))

Compensation Process - N

Postmortem

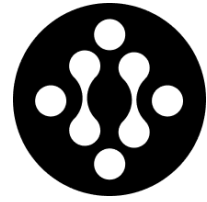
[Link](#) - 2024.05.31

Fund Flow

According to on-chain investigator @zachxbt, the exploited assets were laundered to the online marketplace Huione Guarantee. ([Ref.](#))



102. Velocore (Jun 2)



Underlying Chain: Linea, zkSync

Amount: \$6,850,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The root cause of the exploit was in feeMultiplier, which was implemented to increase fees for withdrawals in the same block. Since no upper bound was set for feeMultiplier, it led to the underflow of LP withdrawal amount. For the exploit, the exploiter manipulated feeMultiplier to be over 100%, flashloaned LP tokens, and withdrew tokens in the pool to make the exploit feasible. Triggering the underflow, the exploiter gained a massive amount of LP tokens as a result. While taking actions, Linea chain paused its sequencer to prevent the exploiter from withdrawing assets to Ethereum. ([Ref.](#))

**Linea**   @LineaBuild · Jun 3, 2024 OP ...

The sequencer was paused from block 5081800 and 5081801.

During this pause, we gave the [@Velocorexyz](#) time team to support their efforts of triaging the vulnerability.

We also censored the hacker's addresses. This significantly reduced the ecosystem impact on Linea users.

 2  20  57  29K  

**Linea**   @LineaBuild · Jun 3, 2024 OP ...

Linea's team made a decision to halt block production by pausing the sequencer and censor attacker addresses to protect the users and builders in our ecosystem. Like other L2s, we are still in the "training wheels" phase of existence, giving us safeguards to use.

 45  140  157  612K  

**Linea**   @LineaBuild · Jun 3, 2024 OP ...

One of the key drivers in our decision to pause the sequencer was that the hacker had acquired and was beginning to sell a large sum of tokens into ETH. This would have created other issues in the ecosystem for users beyond the liquidity pool draining exploit.

 3  4  46  29K  



Despite Linea's efforts, the exploiter managed to bypass their restrictions using a third-party bridge. Regarding the sequencer-level censorship of Linea, lots of researchers arose controversy over the decentralization of L2 chains.

Compensation Process

The team decided to liquidate their treasury and distribute the compensation based on a snapshot. ([Ref.](#))

Pre-Incident Audit History

Zokyo (2023.08.14) - [Report](#), Scalebit (2023.08.16) - [Report](#)

Hacken (2023.10.25) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.06.02

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#)

Fund Flow

Exploited assets are bridged to Ethereum and laundered through Tornado Cash, right after the exploit.

Exploiter ([0x8CDc](#))

→ Bridged via Across Protocol (Linea, zkSync → Ethereum) ([0x8CDc](#), 1206 ETH)

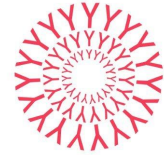
→ Laundered through Tornado Cash (1205.1 ETH)

Note

With the decision to liquidate the team treasury for reimbursement, Velocore decided to cease its operation rather than relaunching the new token, due to the competitive DEX market situation. ([Ref.](#))



103. Lykke (Jun 4)



Underlying Chain: Bitcoin, Ethereum, Tron, Litecoin, Bitcoin Cash

Amount: \$19,500,000

Attack Vector: Control Hijacking - Web2 Exploit

Brief Summary

Swiss-based CEX Lykke suffered an exploit on their infrastructure, losing 158 BTC and 2161 ETH which are more than \$22 million in total. Exploited funds were distributed to various Bitcoin addresses and then to Ethereum addresses via crypto mixer.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.06.11

Fund Flow

@tayvano_ actively traced the destination of the fund and released it in public. ([Ref.](#))

Exploited assets in Bitcoin are bridged to Ethereum via THORChain, 7 days after the exploit.

([Ref.](#)) We found that the assets on Ethereum are deposited to eXch on Dec 15 & 29 2024.

Note

On Jul 9 2024, Lykke announced a temporary shutdown of the exchange. ([Ref.](#)) However, there has been no update until now.



104. Gemholic (Jun 8)



Underlying Chain: zkSync

Amount: \$3,400,000

Attack Vector: Rugpull


Brief Summary

The developers of Gemholic, a DEX based on zkSync, performed \$3.4 million rugpull by allegedly withdrawing the funds in their contract and deleting all of their social channels. Gemholic once got its 921 ETH stuck in a contract due to the usage of the `.transfer()` function, which was not supported in the previous versions of zkSync.

ZKsync (α, Δ) @zksync · Apr 7, 2023 **OP**

TLDR:

- 1) Funds are safe. We found an elegant solution how to unfreeze the contract.
- 2) Era is not EVM equivalent. This is a deliberate design choice with specific upsides and trade-offs.
- 3) Following best practices would've prevented the issue:



From diligence.consensys.io

28 233 492 135K

ZKsync (α, Δ) @zksync · Apr 7, 2023 **OP**

@GemholicECO is a multi-product platform that recently deployed their token contract and performed a pre-sale launch on zkSync Era. After deployment they encountered an issue with their contract due to the usage of the `.transfer()` function.

4 73 233 100K

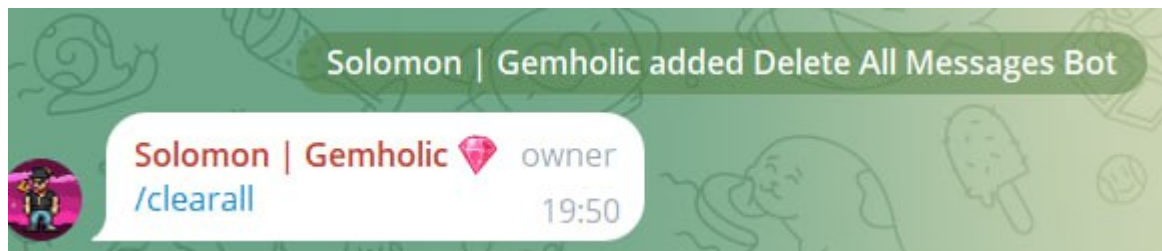
ZKsync (α, Δ) @zksync · Apr 7, 2023 **OP**

Unfortunately, the contracts were deployed on Era mainnet without having been tried on the testnet or local node, which would have immediately caught the problem.

1 24 139 31K



On the day before the exploit, the newest upgrade of zkSync resolved this issue, but the team decided to rug and withdraw rather than compensating their early supporters.



As the audit firm SolidProof previously announced the KYC completion of the Gemholic team through their service, SolidProof promised future actions with Vietnam authorities where the team members are based. ([Ref.](#))

Pre-Incident Audit History

SolidProof(2023.03.13) - Unavailable ([Ref.](#))

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged to Ethereum and laundered through Tornado Cash, right after the exploit.

Project Fund ([0xf1aB](#))

→ [0x3c67](#) (921 ETH)

→ Distributed to multiple EOAs ([Tx](#))

→ Bridged via Across Protocol (zkSync → Ethereum)

→ Merged at [0x7660](#) & Laundered through Tornado Cash (443.6 ETH)



105. YYS Token (Jun 9)

Underlying Chain: BSC

Amount: \$29,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The root cause of the exploit was in liquidity calculation logic in the token contract. The exploiter swapped a massive amount of USDT to YYS, to manipulate the pool balance, and called the sell function to profit. The exploited funds were moved to Railgun.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged to Arbitrum and laundered through Railgun, right after the exploit.

Exploiter ([0x1017](#))

→ Bridged via LayerZero Stargate (BSC → Arbitrum) ([0x72c6](#), 29K USDT)

→ Laundered through Railgun



106. Loopring (Jun 9)



Underlying Chain: Ethereum

Amount: \$5,000,000

Attack Vector: Control Hijacking - Web2 Exploit

Brief Summary

The exploiter compromised Loopring's 2FA service and impersonated the wallet owner to get approval for the recovery functionality from one of the official guardians of Loopring. Exploiting the guardian, the exploiter could initiate the recovery process that resets ownership of the wallet and withdraw the assets.

Compensation Process

Loopring asked affected users to contact the team for further recovery. ([Ref.](#)) However, we could not identify whether the actual reimbursement was done or not.

Postmortem

[Link](#) - 2024.06.09

Exploit Transactions - [Txs](#)

Fund Flow

The exploited assets, 1373.4 ETH, haven't moved from the exploiter [0x44F8](#) since the exploit.



107. UwU Lend (Jun 10, Jun 13)



Underlying Chain: Ethereum

Amount: \$22,720,000

Attack Vector: Contract Vulnerability - Price Dependency

Brief Summary

The root cause of the exploit was the misimplemented price oracle of \$sUSDe. UwU Lend utilized fallback oracles for some pools, which calculated token prices from multiple Curve pools. The exploiter broke the pool balance through the flashloan and manipulated the token price, and borrowed \$sUSDe at price 0.99 while its liquidation price was about 1.03. 3 days after the first exploit, UwU Lend experienced the second exploit due to the same vulnerability(!), as the exploiter borrowed all the assets with \$uSUSDE which was stolen from the first exploit.

Compensation Process

From Jul 3 to Aug 10 2024, the team had announced the repayment of the bad debts generated by the exploit. ([Ref.](#), [Ref.](#)) They have repaid a total of ~\$20.6M since the exploit.

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.06.11

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#)

Fund Flow

Exploited assets are laundered through Tornado Cash, 121 days after the exploit.

Exploiter ([0x841d](#))

→ ... → Laundered through Tornado Cash ([0x77fB](#), [0x81c9](#), [0xB4D0](#) - 6357 ETH)



108. JokInTheBox (Jun 11)



Underlying Chain: Ethereum

Amount: \$34,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The root cause of the exploit was misimplemented `unstake` function of the staking contract. Since the `unstake` function does not check the state of the variable "unstake", the exploiter could unstake multiple times and drain the assets.

Compensation Process

The team announced the airdrop of the exact amount of tokens for affected users, and burn 15% of token supply. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0xFcd4](#))

→ Laundered through Tornado Cash (10 ETH)



109. Bazaar (Jun 11)



Underlying Chain: Blast

Amount: \$1,500,000

Attack Vector: Contract Vulnerability - Access Control

Brief Summary

The exploited contract was the Bazaar's LBP contract for YOLO Games. The root cause was the lack of a permission check in the exitPool function, which allowed anyone to impersonate liquidity providers and drain the pool.

```
function exitPool(bytes32 poolId, address sender, address payable recipient, ExitPoolRequest memory request)
    external
    override
    nonReentrant
    registeredPool(poolId)
{
    (address[] memory tokens, uint256[] memory balances,) = getPoolTokens(poolId);
    require(tokens.length == TOKENS_LENGTH && tokens.length == request.tokens.length, "mismatch tokens length");

    IBazaarLBP lbp = IBazaarLBP(_getPoolAddress(poolId));
    (uint256[] memory amountsOut,) = lbp.onExitPool(poolId, sender, recipient, balances, 0, 0, request.userData);

    for (uint256 i = 0; i < tokens.length; i++) {
        uint256 amountOut = amountsOut[i];
        require(amountOut >= request.minAmountsOut[i], "limit not reached");
        require(tokens[i] == _translateToErc20(request.tokens[i]), "token mismatch");

        address token = tokens[i];
        poolBalances[poolId][token] = poolBalances[poolId][token].sub(amountOut);

        // use original address such that ETH is supported
        _sendFunds(request.tokens[i], recipient, amountOut);
    }
}
```

Leveraging the vulnerability, the exploiter drained 392 ETH and 880M rYOLO by calling BazaarVaultBlast.exitPool(). Right after the exploit, the exploiter claimed himself as a whitehat and reached out to the team. Accepting the Bazaar team's bounty offer, the exploiter returned 90% of stolen assets.

Compensation Process - N/A

Pre-Incident Audit History

OtterSec (2024.03.09) - [Report](#), OtterSec (2024.06.07) - [Report](#)

Post-Incident Audit History - N/A



Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Accepting the team's negotiation offer, the exploiter returned 90% of the exploited assets to the team. The exploiter laundered his bounty, 11 days after the exploit.

Exploiter ([0x3cf5](#))

→ Returned to the Project ([0xDE4d](#), 353 ETH)

/ Bridged via Blast Official Bridge (Blast → Ethereum) ([0x7813](#), 41.2 ETH) → Laundered through Tornado Cash

Note

The project seems inactive in its current state. The website doesn't work, and no update has been made in their X account since its launch.



110. NFTPerp (Jun 14)



Underlying Chain: Blast

Amount: \$772,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The clearingHouse contract which handles trader margins and open positions was exploited due to the bug in safeTransferFrom function. According to one of the devs, the bug allowed users to place limit orders and immediately cancel to receive collateral without any deposit. With the help of on-chain investigator ZachXBT, the exploiter was identified and got 90% of the stolen assets (200 WETH) returned.

Compensation Process

On Jun 15 2024, the team announced that the lost funds were fully recovered. ([Ref.](#)) On Jun 18 2024, the team announced the airdrop of tokens utilizing the deposits in their vault. ([Ref.](#))

Pre-Incident Audit History

Peckshield (2022.08.28) - [Report](#) (Unavailable), Code4rena (2024.01.26) - [Report](#) (Unavailable),

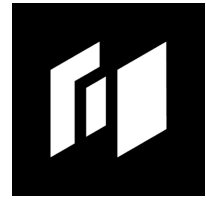
Three Sigma (2024.02.02) - [Report](#)

Post-Incident Audit History - N/A

Postmortem - N/A



111. Holograph (Jun 14)



Underlying Chain: Ethereum, Mantle

Amount: \$14,400,000

Attack Vector: Control Hijacking - Authorization Management Failure

Brief Summary

A former developer of Holograph deployed an unverified contract on Mantle to mint additional \$HLG token. With an allegedly gained authority, the exploiter bypassed the verification process and minted 1 billion additional \$HLG, which was worth \$14 million. Minted \$HLG were bridged to Ethereum and sent to various CEX and DEX, including Bybit, Gate.io, Kucoin, Bltget and Backpack. Due to the exploit, \$HLG's FDV plummeted by 79%.

On Aug 15 2024, the team announced that multiple arrests tied to the exploit have been made by French OFAC and European investigation agencies. ([Ref.](#))

Compensation Process

To resolve the over-supplied HLG due to the exploit, Holograph announced a plan to burn 1B HLG through open market purchases and buybacks through the team treasury. ([Ref.](#)) In Jul 17 2024, the team announced that the circulating supply returned to its original due to the team buyback and burning. ([Ref.](#))

Pre-Incident Audit History

Code4rena (2022.12.15) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.07.02

Exploit Transactions - [Tx](#)



Fund Flow

Exploiter ([0xa198](#)) → [0xFF8c](#) (1B HLG)

→ Deposited to Bitget ([0x126e](#), 500M HLG), Gate.io ([0x0465](#), 100M HLG), Kucoin ([0xF009](#), 100M HLG), Bybit ([0xcd2B](#), 100M HLG), [0x2228](#) (100M HLG), ChangeNOW ([0xC1F3](#), 142 ETH)

/ Laundered through Tornado Cash ([0x12d8](#), 1 ETH), Railgun ([0xC1F3](#), 25 ETH)

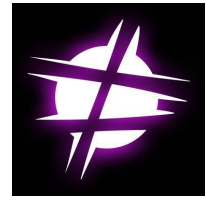
/ Bridged via Optimism Official Bridge & LayerZero Stargate (Ethereum → Optimism → Ethereum) ([0x2E7F](#), [0x09Ca](#), [0x6a22](#), [0xdEe3](#), [0xB8aB](#), [0x01dE](#), [0xA513](#), [0x1609](#), [0xd6C5](#), [0x81E5](#) - 49.9 ETH) → ... → Laundered through Railgun ([0x9f35](#), [0x1c83](#), [0x6813](#), [0x077D](#) - 49.9 ETH)

/ Bridged via THORChain (Ethereum → Bitcoin) ([0xC1F3](#), 5 ETH) → Bitcoin address ([bc1q3e](#))

/ [0xc527](#) (100 ETH), [0x077D](#) (1 ETH), [0x87D6](#) (50 ETH)



112. Dyson (Jun 17)



Underlying Chain: BSC

Amount: \$31,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

One of the deprecated vaults of Dyson was compromised, as the vault allowed the exploiter to withdraw a large amount of unharvested rewards with a small position. The exploiter could drain all the rewards of the contract by depositing a small amount of tokens in the pool and calling harvest function.

Compensation Process - N/A, User funds not affected

Pre-Incident Audit History

Peckshield (2023.02.24) - [Report](#), Prisma Shield (2023.03.03) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.06.19

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

Exploited assets were distributed to multiple EOAs, and then moved to ChangeNOW right after the exploit.

Exploiter ([0x4CeD](#)) → [0x1A4a](#) (59 BNB) → Distributed to EOAs → Deposited to ChangeNOW



113. Farcana (Jun 19)



Underlying Chain: Polygon

Amount: \$440,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

One of the market maker's wallets of Farcana, which is an EOA, was drained due to the private key leakage.

Regarding the incident, Farcana uploaded a post that one of the team's wallets has been compromised, but deleted after 8 hours. ([Ref.](#)) Later, they uploaded the revised version of the post that claims none of the team's wallet or contract had been compromised and the victim was one of the market makers. ([Ref.](#))

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)



Fund Flow

Exploited assets were distributed in various addresses, partially laundered through Railgun. Related wallets were active till 152 days after the exploit.

Exploiter ([0x9681](#))

→ [0xB6A2](#) (8.6M FAR), [0x6454](#) (161K USDT), [0x29eD](#) (5M FAR)

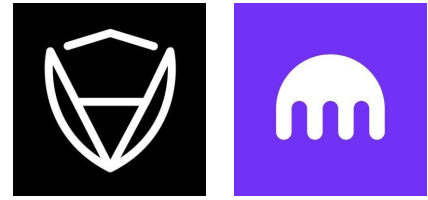
→ Laundered through Railgun ([0x935e](#), 92K USDC)

/ Bridge via Across Protocol (Polygon → Ethereum) ([0xc801](#), 4.2 ETH)

→ Deposited to zkBOB ([0xAC9C](#), 26K USDC), Binance ([0x18C4](#), 3280 POL), ChangeNOW ([0x6D48](#), [0x546E](#) - 2330 POL)



114. CertiK <> Kraken (Jun 20)



Underlying Chain: Ethereum

Amount: \$3,000,000

Attack Vector: Misc.

Brief Summary

On June 19 2024, Kraken CSO @c7five tweeted that Kraken received a report claiming a critical vulnerability in the platform. [\(Ref.\)](#) However, the reporter had not been disclosed due to disagreements in the mediation process. Kraken's deposit transaction handling logic had a critical flaw, which accepted deposit transactions that were reverted in internal transactions thus no actual transfers of the assets were made.

Shortly after, CertiK posted a tweet that they had identified the vulnerability in Kraken with the details, disclosing all the test transactions. [\(Ref.\)](#) CertiK's test transactions on Kraken's system lasted about 5 days, which resulted in the drainage of \$3 million worth of assets from Kraken.

Regarding the incident, several points of controversy had aroused among the community.

First, CertiK withdrew ~\$3M leveraging the vulnerability, which is a huge amount for just the proof of concept. Kraken's bug bounty rule does not allow exploiting more than what is needed to prove the vulnerability. [\(Ref.\)](#) @c7five claimed that a KYC-ed individual also discovered and reported the same vulnerability with \$4 in crypto, which was sufficient to prove the flaw. [\(Ref.\)](#)

Second, CertiK sent the exploited MATIC and USDT to various CEXs including ChangeNOW and swapped them into ETH and XMR. Some of the assets had even been transferred through Tornado Cash, which is an OFAC-sanctioned crypto mixer. [\(Ref.\)](#) This is totally unnecessary behavior for proving a vulnerability, and the exploited assets during PoC should not have been transferred or swapped if they were planned to be returned.

Third, similar activities were identified on Base, BSC, Optimism, Arbitrum, Avalanche, Linea a month before the incident, as @Meir_Dv, founder of Cyvers.AI reported. [\(Ref.\)](#)



Fourth, CertiK claimed that "fabricated tokens" were "minted out of air" by the incident ([Ref.](#)), but the exploited assets were real assets of Kraken, not the fake tokens that had not existed.

More details of the incident are described in Rekt's article ([Link](#)).

Compensation Process

On June 20 2024, @c7five reported that the assets were returned to Kraken. ([Ref.](#))



115. BtcTurk (Jun 22)



Underlying Chain: Bitcoin

Amount: \$90,000,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

BtcTurk, one of the largest Turkish CEX, got its 10 hot wallets exploited, losing assets worth \$90 million. In response, the exchange has halted deposits and withdrawals and is working with law enforcement. The hacker sold significant amounts of several cryptocurrencies, leading to major price fluctuations in those tokens.



Wu Blockchain 
@WuBlockchain **OP**



BtcTurk, one of Turkey's largest cryptocurrency exchanges, said it was attacked on June 22, and cryptocurrency deposits and withdrawals have been stopped. Officials said that some balances in the hot wallets of 10 cryptocurrencies were affected, and the cold wallets that hold most of the assets are safe. status.btcturk.com

9:12 PM · Jun 22, 2024 · **460.8K** Views



27



119



300



23



Post your reply

Reply



Wu Blockchain 
@WuBlockchain · Jun 22, 2024 **OP**



Today, 54 million USD of AVAX was sold. The AVAX at 0x32...4f30 originated from the transfer of X-avax1qsnptt6yxnmh8up7xn5u3sajz37lmyv8u3dmu on the Avalanche-X chain. @VeriDelisi believes that this address is the address of a Turkish exchange. veridelisi.medium.com/avalanche-x-ch...



11



18



69



45K



@CryptoEvgen, @somaxbt, @tayvano_ had actively analyzed the incident. ([Ref.](#)) According to Tay, the BtcTurk incident was not done by Lazarus, but involves the victim of the LastPass exploit.

Compensation Process - N/A

Regarding the incident, no details on the exploit or reimbursement have been announced. Since it is normally operating in its current state, they may have compensated the affected assets through their own assets.

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transaction - [Tx](#) ([Ref.](#))

Fund Flow

According to the on-chain investigator @CryptoEvgen, the exploited assets are transferred to THORChain, bridged to Ethereum and Arbitrum.

According to the Binance CEO Richard Teng, Binance froze \$5.3M related to the exploit. ([Ref.](#))



116. CoinStats (Jun 22)



Underlying Chain: Ethereum

Amount: \$2,000,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

CoinStats, a cryptocurrency portfolio management service, experienced a security breach which involved its 1,590 user wallets. CoinStats stated that only 1.3% of all wallets were affected. The root cause was the compromised AWS infrastructure, as one of the employees was socially engineered into downloading malicious software onto his work computer.



narek ✓

@narek_gevorgyan OP

xl ...

What a week it's been.

I've been working diligently on CoinStats for the last 6 years. We've experienced many highs and lows, but I believe we've created the best portfolio tracker on the market.

Our AWS infrastructure was hacked, with strong evidence suggesting it was done through one of our employees who was socially engineered into downloading malicious software onto his work computer.

Seeing all this happen to something you've worked hard on for 6 years is tough, especially since it occurred because of a secondary feature. The CoinStats Wallet, used by no more than 1% of all our users, was certainly not the reason people loved our product.

I empathize with those who lost money; I'm sure their situation is just as difficult. CoinStats will definitely support the victims of the hack, and we've been discussing options internally.

We're waiting for a few details from law enforcement to be finalized before we can share a more detailed post-mortem of the hack.

5:01 PM · Jun 26, 2024 · **16.5K** Views



Compensation Process - N/A

Postmortem

[Link](#) - 2024.07.12

Fund Flow

After being distributed into various Ethereum EOAs ([Ref.](#)), most of the exploited assets are merged at [0xe099](#). They are laundered through Tornado Cash, 16 days after the exploit. Investigators traced the assets after the laundry. ([0x99A8](#)) They are bridged to Arbitrum ([0xEC80](#)), distributed to 3 EOAs again. They are deposited to CEXs at different times - HTX ([0x3d2b](#), [0x0C51](#) - 267.2 ETH), 45 days after the exploit / Gate.io ([0x731D](#), 122 ETH), 152 days after the exploit.



117. Sportsbet (Jun 23)



Underlying Chain: Tron

Amount: \$3,500,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

SportsBet, a cryptocurrency sports gambling project, was exploited by the same exploiter of the BtcTurk incident, reported by @zachxbt.

Investigations by ZachXBT

Investigations by ZachXBT
I conducted a timing analysis and found highly probably B...

TIME	FROM	TO	VALUE	TOKEN	USD
2024-06-22 01:44:09	Sportsbet.io: Hot Walle...	TDgZKxhyFQWCsNK1p7d1tVife...	410K	USDT	\$410.00K
2024-06-22 01:41:12	Sportsbet.io: Hot Walle...	TJZ8NNxJETGDzGaWwSHwjGrzz...	1.3M	TRX	\$153.56K
2024-06-22 01:28:09	Sportsbet.io: Hot Walle...	TQWSmSqns2BLczLEmpy96tNq3...	1.1M	USDT	\$1.10M
2024-06-21 23:46:39	Sportsbet.io: Hot Walle...	TJZ8NNxJETGDzGaWwSHwjGrzz...	4.6M	TRX	\$544.24K
2024-06-21 23:46:06	Sportsbet.io: Hot Walle...	TJZ8NNxJETGDzGaWwSHwjGrzz...	680K	USDT	\$680.00K
2024-06-21 20:38:00	Sportsbet.io: Hot Walle...	TMANGxX6DK81Jd5r8uYiPmRzQ...	654.431K	USDT	\$654.43K

Online casino Sportsbet was likely hacked for \$2.9M+ by the same threat actor as BTCTurk two hours before as funds from the thefts comingled.

Theft address
TDgZKxhyFQWCsNK1p7d1tVifeuW2DJTUEo
TQWSmSqns2BLczLEmpy96tNq3MagM66H4b
TJZ8NNxJETGDzGaWwSHwjGrzzz2Zhvexo2

61 24 11 7 7 7 2 2 2

100 1

3964 11:27 AM

Exploited assets were transferred to Tron wallets. Regarding the incident, the team had not made any official announcements.



Compensation Process - N/A

Postmortem - N/A

Fund Flow

Exploited assets are deposited to CEXs including FixedFloat, ChangeNOW, and MEXC at different times.

Exploiter

→ [TGiT](#), [TQWS](#), [TKZ8](#) -

> FixedFloat ([TGiT](#), [TGtk](#), [TC8y](#), [TX2U](#), [TDGe](#), [TGCU](#), [TJZ8](#) - 1.507M USDT, 1.03M TRX), 48 days after the exploit

/ ChangeNOW ([TY47](#), [TLrj](#), [TJSA](#), [TJZ8](#) - 1.25M USDT, 5.21M TRX), right after the exploit / MEXC ([TBZy](#) - 204K TRX, 10K USDT), 94 days after the exploit



118. Shentu OpenBounty (Jun 26)



Underlying Chain: -

Amount: Unknown

Attack Vector: Misc.

Brief Summary

On June 26 2024, Security researcher @h0wlu pointed out that OpenBounty, a bug bounty platform operated by Shentu Chain, seems to frontrun the user reports. ([Ref.](#)) Shentu Chain has been known to have a close relationship to CertiK. ([Ref.](#), [Ref.](#))

```
29 // PrefixToShentu convert certik prefix address to shentu prefix address
30 func PrefixToShentu(address string) (string, error) {
31     hrp, data, err := bech32.DecodeAndConvert(address)
32     if err != nil {
33         return "", err
34     }
35
36     if strings.HasPrefix(hrp, Bech32MainPrefix) {
37         return address, nil
38     }
39     if !strings.HasPrefix(hrp, "certik") {
40         return "", fmt.Errorf("invalid address:%s", address)
41     }
42     newhrp := strings.Replace(hrp, "certik", "shentu", 1)
43     shentuAddr, err := bech32.ConvertAndEncode(newhrp, data)
44     if err != nil {
45         return "", err
46     }
47     return shentuAddr, nil
48 }
```

This brought a controversy among security researchers, as frontrunning bounties can lead to serious consequences since the bug bounty platform could acknowledge critical bugs before and exploit the project. ([Ref.](#)) After the incident, CertiK deleted the blog posts about OpenBounty and changed the API domain to non-CertiK domain without any official announcement regarding the incident. ([Ref.](#)) More details can be found in Rekt's article. ([Ref.](#))



119. MintRisesPrices Token (Jul 2)



Underlying Chain: BSC

Amount: \$11,000

Attack Vector: Contract Vulnerability - Reentrancy

Brief Summary

In the token contract, a simple reentrancy vulnerability existed as the exploiter could call the receive fallback from the `_transfer` function, and there was no reentrancy guard.

```
receive() external payable {
    if (tradingStartTime >= block.timestamp || getAddLiquidityTrigger(_msgSender())) {
        _addLiquidity();
    } else {
        _buy();
    }
    _offAddLiquidityTrigger(_msgSender());
}
```

Leveraging vulnerability, the exploiter drained all BNB in the contract.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

The exploited assets haven't moved elsewhere since the exploit.

Exploiter ([0x132D](#)) → [0x36FE](#) (23.7 BNB)



120. Bittensor (Jul 2)



Underlying Chain: Bittensor

Amount: \$8,000,000

Attack Vector: Control Hijacking - Supply Chain Attack

Brief Summary

On Jun 2 2024, Bittensor notified its users that the project had experienced a security breach. The root cause was the upload of a malicious PyPI package for Bittensor v6.12.2, indicating that the exploiter gained access to the Bittensor PyPI account. When users downloaded the specific version of the package, the compromised code stole unencrypted user private keys and sent them to a remote server controlled by the exploiter. To handle the situation, Opentensor Chain Validators decided the entrance of all the subnets of Bittensor into safe mode, halting all the user transactions for a while.



Opentensor Foundation 
@opentensor 

x1 ...

Bittensor Community Update

Yesterday at 7:41 PM UTC, we took the decision to place the Opentensor Chain Validators behind a firewall and entered safe-mode on Subtensor due to an attack that affected multiple participants in the Bittensor community.

We have put together a comprehensive post-mortem including remediation and next steps on Medium: blog.bittensor.com/bittensor-community-update

Last edited 4:40 AM · Jul 4, 2024 · **109K** Views

Regarding the team's response, the community had raised concerns on the centralization of the Bittensor ecosystem.



Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.07.04



121. DefiPlaza (Jul 5)



Underlying Chain: Ethereum

Amount: \$2,000,000

Attack Vector: Contract Vulnerability - Precision Loss

Brief Summary

The root cause was the rounding but that existed in removeLiquidity function in their contract. When a certain token's liquidity gets depleted in the pool, it leads to the rounding down of LPamount and allows the price manipulation attack.

```
302  ✓ function removeLiquidity(uint256 LPamount, address outputToken, uint256 minOutputAmount)
303      external
304      onlyListedToken(outputToken)
305      override
306      returns (uint256 actualOutput)
307  ✓ {
308      // Checks the initial balance of the token desired as output token
309      uint256 initialBalance;
310  ✓  if (outputToken == address(0)) {
311      |   initialBalance = address(this).balance;
312  ✓  } else {
313      |   initialBalance = IERC20(outputToken).balanceOf(address(this));
314      }
315
316      // Calculates intermediate variable F = (1-R)^16 and then the resulting output amount.
317      uint256 F_;
318      F_ = (1 << 64) - (LPamount << 64) / totalSupply(); // (1-R)      (0.64 bits)
319      F_ = F_ * F_; // (1-R)^2    (0.128 bits)
320      F_ = F_ * F_ >> 192; // (1-R)^4  (0.64 bits)
321      F_ = F_ * F_; // (1-R)^8    (0.128 bits)
322      F_ = F_ * F_ >> 192; // (1-R)^16 (0.64 bits)
323      actualOutput = initialBalance * ((1 << 64) - F_) >> 64;
324      require(actualOutput > minOutputAmount, "DFP: No deal");
325
326      // Burns the LP tokens and sends the output tokens
327      _burn(msg.sender, LPamount);
328  ✓  if (outputToken == address(0)) {
329      |   address payable sender = payable(msg.sender);
330      |   sender.transfer(actualOutput);
331  ✓  } else {
332      |   IERC20(outputToken).safeTransfer(msg.sender, actualOutput);
333      }
334
335      // Emitting liquidity removal event to enable better governance decisions
336      emit LiquidityRemoved(msg.sender, outputToken, actualOutput, LPamount);
337  }
```



With flashloaned assets, the exploiter depleted the ETH pool and manipulated the ETH-related pool price. The team uploaded the postmortem 49 days after the exploit.

Compensation Process

According to the postmortem, DefiPlaza plans to “retrieve any stolen funds distributed through the Lido validator by proposing a DAO governance vote to recover the funds.”, but we could not identify if the actual compensation was done to the affected users.

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.08.23

Exploit Transactions - [Tx](#)

Fund Flow

An MEV bot ([0xFDe0](#)) had frontrun the exploit transaction. We could not identify if the refund was made from the address.



122. LW Token (Jul 8)

Underlying Chain: BSC

Amount: \$80,000

Attack Vector: Contract Vulnerability - Misimplemented Transfer Function

Brief Summary

A misimplemented transferFrom function in the LW token contract allowed an underflow in its logic, despite being compiled in Solidity v0.7.6.

```
// 转账
function transferFrom(
    address sender,
    address recipient,
    uint256 amount
) public override returns (bool) {
    _transfer(sender, recipient, amount);
    if (_allowances[sender][msg.sender] != MAX) {
        _allowances[sender][msg.sender] =
            _allowances[sender][msg.sender] -
            amount;
    }
    return true;
}
```

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

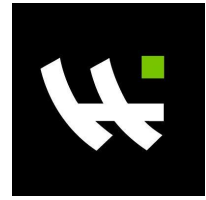
Fund Flow

Exploited assets are directly laundered through Tornado Cash, 7 days after the exploit.

Exploiter ([0x56b2](#)) → Laundered through Tornado Cash (13.9 BNB)



123. Wasabi Wallet (Jul 10)



Underlying Chain: -

Amount: Unknown

Attack Vector: Control Hijacking - Supply Chain Attack / DoS

Brief Summary

According to Wasabi Wallet, they suffered three types of attacks, including DoS on free coordinators, supply chain compromise and malicious coordinator attacking its liquidity.

1. DoS Attack: Several free coordinators with most liquidity was taken down by DDoS attack.
2. Supply Chain Attack: Wasabi Wallet's Windows installer has been replaced with malware.
3. Malicious Coordinators: A vulnerability in the project allowed a malicious coordinator to charge a higher fee than user specification. Exploiting the vulnerability, one of the coordinators (wasabicoordinator.io) was gradually exploiting user assets. The total amount of exploited assets is not publicly announced.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.07.10



124. Dough Finance (Jul 12)



Underlying Chain: Ethereum

Amount: \$1,810,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

The ConnectorDeleverageParaswap contract lacked verification on calldata and msg.sender for the callback on Paraswap. The exploiter flashloaned 930K USDC from Aave, called flashloanReq function of the vulnerable contract with maliciously constructed calldata. The dedicatedly constructed calldata included settlement of all USDC debt and withdrawal of all WETH collateral in the contract, and transfer of WETH to the exploiter.

Compensation Process

On Aug 8 2024, the team opened a governance proposal for fund recovery, but it seems that no actual reimbursement has been made. ([Ref.](#)) The project seems to be inactive since the vote.

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.07.12

Exploit Transactions - [Tx](#)

Fund Flow

The exploited assets are laundered through Tornado Cash, 6 days after the exploit.

Exploiter ([0x6998](#), [0x2913](#)) → [0x3461](#) (608 ETH)

→ Laundered through Tornado Cash ([0x3461](#), [0x932e](#), [0xba25](#))



125. Minterest (Jul 15)



Underlying Chain: Mantle

Amount: \$1,400,000

Attack Vector: Contract Vulnerability - Reentrancy

Brief Summary

Minterest, a cross-chain lending protocol, was exploited due to the reentrancy in its flashloan function. Exploiting the vulnerability, the attacker flashloaned a massive amount of \$RUSDY to crash the exchange rate through the lendRUSDY function, and called the function again to lend tokens in the manipulated exchange rate. The team has offered 10% as a bounty, but the exploiter bridged all stolen funds from Mantle to Ethereum and laundered them through Tornado Cash.

Compensation Process

On Jul 31 2024, Minterest posted the remediation plan on its blog. The detailed remediation plan is as follows: 15% decrease was applied for all users WETH & mETH, affected users are compensated with \$MINTY equal to their share with a 6-month vesting, 40% boost was applied for WETH & mETH suppliers. ([Ref.](#))

Pre-Incident Audit History

Trail of Bits (2022.03.29) - [Report](#), Hacken (2022.04.13) - [Report](#),

Peckshield (2022.06.28) - [Report](#), Zokyo (2022.11.30) - [Report](#),

Zokyo (2023.02.24) - [Report](#), Zokyo (2023.03.14) - [Report](#),

Zokyo (2023.10.18) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.07.16



Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged to Ethereum and laundered through Tornado Cash, at the day of the exploit.

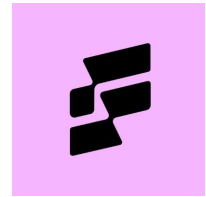
Exploiter ([0x618F](#))

→ Bridged via LayerZero Stargate (Mantle → Ethereum) ([0x618F](#), 223.5 ETH)

→ Laundered through Tornado Cash



126. LI.FI (Jul 16)



Underlying Chain: Ethereum

Amount: \$10,000,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

LI.FI, a cross-chain bridge aggregation protocol, was exploited as the swap function in one of its newly deployed contracts did not verify call target and calldata.

```
function depositToGasZipERC20(
    LibSwap.SwapData calldata _swapData,
    uint256 _destinationChains,
    address _recipient
) public {
    // get the current native balance
    uint256 currentNativeBalance = address(this).balance;

    // execute the swapData that swaps the ERC20 token into native
    LibSwap.swap(0, _swapData);

    // calculate the swap output amount using the initial native balance
    uint256 swapOutputAmount = address(this).balance -
        currentNativeBalance;

    // call the gas zip router and deposit tokens
    gasZipRouter.deposit{ value: swapOutputAmount }(
        _destinationChains,
        _recipient
    );
}

// solhint-disable-next-line avoid-low-level-calls
(bool success, bytes memory res) = _swap.callTo.call{
    value: nativeValue
}(_swap.callData);
if (!success) {
    LibUtil.revertWith(res);
}
```

As a result, users who had set approvals on the contract had their assets drained, which was over \$10 million in total.



Compensation Process

On Jul 18 2024, the team opened a voluntary compensation scheme. ([Ref.](#))

Pre-Incident Audit History

Code4rena (2022.03.30) - [Report](#), Quantstamp (2022.05.06) - [Report](#),

Spearbit (2022.10.18) - [Report](#), Spearbit (2023.04.11) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.07.18

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are distributed multiple times and laundered through Tornado Cash. The whole laundry process took 69 days.

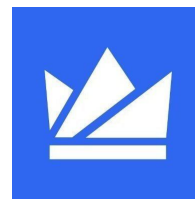
Exploiter ([0x8B3C](#), 1910.84 ETH, 956K USDC, 169K DAI)

→ Distributed to multiple EOAs

→ Laundered through Tornado Cash ([0xa926](#), [0x9593](#), [0x3462](#), [0xeb36](#), [0x6780](#), [0x08Cf](#), [0xe051](#), [0x12b6](#), [0x71a8](#), [0x4ca4](#), [0x1e17](#), [0xCc54](#), [0x8769](#), [0x8eB3](#), [0x91F6](#), [0x596F](#), [0x183b](#), [0xE9f7](#), [0x5fD0](#), [0xbc71](#), [0xE6b2](#), [0x5bfA](#), [0x2dbD](#), [0x0cEf](#), [0x0fF2](#), [0x485E](#), [0x574a](#), [0xfB6a](#), [0xb02f](#), [0x1293](#), [0x727f](#), [0xeA73](#), [0x96c3](#), [0x500b](#), [0x02ea](#), [0xa502](#), [0xf9b5](#), [0x8a4b](#), [0x8004](#), [0xDc43](#), [0x32b9](#), [0x937B](#), [0x21f5](#), [0x07f6](#), [0x24Fe](#), [0x879E](#), [0x625c](#), [0xF373](#), [0x2885](#)) / Leftovers deposited to eXch ([0xbc9d](#), [0x4864](#), [0xbC8a](#)) / [0xc43b](#) (0.1 ETH)



127. WazirX (Jul 18)



Underlying Chain: CEX

Amount: \$230,000,000

Attack Vector: Control Hijacking

Brief Summary

WazirX, one of the largest CEX based in India, suffered a security breach and lost the control of one of its multisig wallets which owned over \$230 million of assets. According to the team, the multisig wallet had 6 signatories including 5 Wazir team and 1 Liminal team.

WazirX and Liminal (The wallet management service that WazirX used) had a controversy around the responsibility of the exploit.

WazirX states: ([Ref.](#))

- The exploit stemmed from the discrepancy between the actual transaction & Liminal's interface
- The replacement of payload is suspected

Liminal States: ([Ref.](#))

- Liminal's interface was not breached, including the WazirX's other Safe Wallets.
- The contract was set as 4-out-of-6 multisig, while WazirX controls 5 of them. The only key that Liminal controlled was an HSM key.
- From Liminal's detailed investigation, the exploit stems from WazirX's 3 compromised devices.
- Once the contract was deployed, all transactions for withdrawal were made outside Liminal's infrastructure.

On Jul 18 2024, WazirX announced the recovery of account balances and the rollback of the trades made after the exploit. ([Ref.](#))

On Jul 19 2024, WazirX reported the incident to FIU and investigation agencies. ([Ref.](#))



On Nov 15 2024, an Indian suspect of the WazirX exploit, Masud Alam, was arrested by Indian police. ([Ref.](#)) The suspect allegedly set up a fake account on WazirX under the name of "Souvik Mondal", and sold the account to the main exploiter M Hasan. Delhi police concluded that no criminal case could be substantiated for the case.

On Dec 18 2024, Delhi High Court rejected the findings and required an updated state report by Feb 2025. ([Ref.](#))

On Dec 18 2024, Binance delisted WazirX's native token \$WRX.

On Dec 23 2024, Times of India reported that M Hasan fled to Bangladesh in Oct 2024. ([Ref.](#))

Compensation Process

WazirX announced the recovery of account balances and the rollback of the trades made after the exploit.

Postmortem

[Link](#) - 2024.07.25

Exploit Transactions - [Tx](#)

Fund Flow

Most of the assets were laundered through Tornado Cash, 56 days after the exploit.

WazirX ([0x27fD](#))

→ Exploiter ([0x04b2](#))

→ Remains in [0x04b2](#) (\$5.3M), [0x35fe](#) (\$2.48M)

/ Laundered through Tornado Cash ([0x5FE2](#), [0xa4d1](#), [0xd70D](#), [0x5990](#), [0x46b9](#), [0xFb4f](#), [0xdBdE](#), [0x0641](#), [0xa6e8](#), [0x6683](#) - 46392.4 ETH)



128. RHO Markets (Jul 19)



Underlying Chain: Scroll

Amount: \$7,600,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

RHO Markets, a lending protocol based on Scroll, experienced a security breach due to the price oracle misconfiguration vulnerability that existed in a newly deployed market contract. The vulnerability was simple: the contract allowed ETH suppliers to mint rETH at BTC oracle price(!), pumping the collateral 20x more than the original.

Fortunately, the exploiter, operating an MEV bot, offered to return the funds through an on-chain message, with conditions on the team to publicly admit their mistakes and provide future actions. ([Ref.](#))

Txn Type: 2 (EIP-1559)

Nonce: 221

Position In Block: 1

Hello RHO team, our MEV bot have profited from your price oracle misconfiguration. We understand that the funds belong to users and are willing to fully return. But first we would like you to admit that it was not an exploit or a hack, but a misconfiguration on your end. Also, please provide what are you going to do to prevent it from happening again.

While handling the incident, Scroll had artificially delayed the finalization of the chain, raising concerns on L2 centralization issues.



Scroll

@Scroll_ZKP

OP

x1 ...

Scroll was notified of a potential exploit within our ecosystem.

After verification with the RhoMarket team, we initiated a coordinated response. To thoroughly assess the situation, Scroll decided to temporarily delay the finalization of the chain.

We have confirmed that the exploit was application-specific.

Currently, RhoMarket is leading the response efforts, and we can confirm that finalization is no longer delayed.

9:26 PM · Jul 19, 2024 · 456.7K Views



Compensation Process

Shortly after the exploit, the team announced the full recovery of stolen funds and repayment on the exploited pools. ([Ref.](#))

Pre-Incident Audit History

Dedaub (2024.05.17) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.07.24

Exploit Transactions -

Fund Flow

MEV bot ([0xe000](#)) → Returned to the Project ([0x2a9c](#), 2202.8 ETH)



129. DeltaPrime (Jul 23)



Underlying Chain: Arbitrum

Amount: \$1,000,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

The root cause was a storage collision vulnerability in DeltaPrime's PrimeAccount contract due to the design mistake. DeltaPrime implemented a diamond beacon proxy, dividing contracts into storage(PrimeAccounts) and implementation contracts. PrimeAccounts are designed to call methods in implementation contract via delegatecalls. In order to mitigate the possibility of storage collision, DeltaPrime utilized custom storage slots, which is commonly used in proxy patterns. In order to initialize the custom slots, DiamondBeacon contract's init() function and PrimeAccount's initialize() function are implemented. These functions should serve only at the deployment of the contract and should be removed after. However, since PrimeAccount does not own the implementation of the method(as it only serves as the state storage), it called DiamondBeacon for the logic. Unfortunately, a bug existed in the logic that checks if the initialize function was executed, as it checked the initialize variable's storage slot for DiamondBeacon, not the PrimeAccount. Exploiting the vulnerability, the exploiter could re-initialize, hijack the PrimeAccounts and execute PrimeAccount-specific methods. Due to the safeguards, the exploiter had to unwind positions and repay the borrowed amounts to the pools before draining the collaterals. Regarding the incident, DeltaPrime deployed a transaction that removes the init method from the DiamondBeacon contract. Fortunately, the exploiter accepted \$100K as a bug bounty and returned exploited assets to the project.

Compensation Process

On Jul 25 2024, all the affected accounts were reimbursed. ([Ref.](#))



Pre-Incident Audit History

Piotr Szlachciak (2021.12.04) - [Report](#), Peckshield (2022.02.12) - [Report](#),

Chainsulting (2022.04.11) - [Report](#), Peckshield (2022.12.01) - [Report](#),

Peckshield (2023.05.27) - [Report](#), Peckshield (2023.09.08) - [Report](#),

AstraSec (2024.06.19) - [Report](#), ABDK (2024.06.24) - [Report](#)

Post-Incident Audit History

AstraSec (2024.08.07) - [Report](#), BlockSec (2024.10.01) - [Report](#)

Postmortem

[Link](#) - 2024.07.27

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

Exploiter ([0x3c61](#), [0x4Fb2](#))

→ ... → Merged at [0xc95b](#)

→ Bridged via Orbiter Finance (Arbitrum → Ethereum)

→ Deposited to CoinsPaid, Revolut



130. Spectra (Jul 23)



Underlying Chain: Ethereum

Amount: \$550,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

The root cause was a vulnerability in its contract due to lack of verification on calldata. Due to the exploit, previously approved user assets totaling about \$550K were drained. Exploited assets were sent to Tornado Cash, while the project offered 10% as a bug bounty.

Compensation Process - N/A

Pre-Incident Audit History

Pashov Audit Group (2024.03.02) - [Report](#), Code4rena (2024.04.05) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.07.24

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are directly laundered through Tornado Cash, right after the exploit.

Exploiter ([0x5363](#)) → Laundered through Tornado Cash (168.5 ETH)



131. MonoSwap (Jul 25)



Underlying Chain: Blast

Amount: \$1,300,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

Monoswap, a DeFi protocol on Blast, was exploited due to a social engineering attack that targeted one of the devs. The victim has installed malware to join a call with exploiters who pretended to be a VC.

Compensation Process - N

MonoSwap has not made any updates after the last announcement on Aug 6 2024. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.07.25

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged to Ethereum and laundered through Tornado Cash, right after the exploit.

Exploiter ([0x895A](#))

→ Bridged via Synapse (Blast → Ethereum) ([0xd30e](#), 333 ETH & 100K USDT)

→ Laundered through Tornado Cash ([0xd30e](#), [0x90D2](#) - 359.6 ETH)



Note

Monoswap seems to be inactive in its current state.



132. Casper Network (Jul 28)



Underlying Chain: Casper Network

Amount: \$6,700,000

Attack Vector: Control Hijacking - Web2 Exploit

Brief Summary

On July 26, 2024, Casper Network experienced a security breach, leading to a temporary halt of its network. According to the team's report, 13 wallets were affected, with illicit transactions totaling approximately \$6.7 million. The breach occurred when malicious actors exploited a vulnerability that allowed a contract installer to bypass access rights checks on urefs. This enabled them to grant unauthorized access to uref-based resources, including the ability to transfer tokens.

Compensation Process - N/A

In the team's postmortem, the team has promised a further reimbursement procedure for affected wallets, but we could not find out whether the actual recovery was done or not.

Pre-Incident Audit History

QuantStamp (2021.05.07) - [Report](#), QuantStamp (2022.07.15) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.07.31



133. Compound Governance (Jul 28)



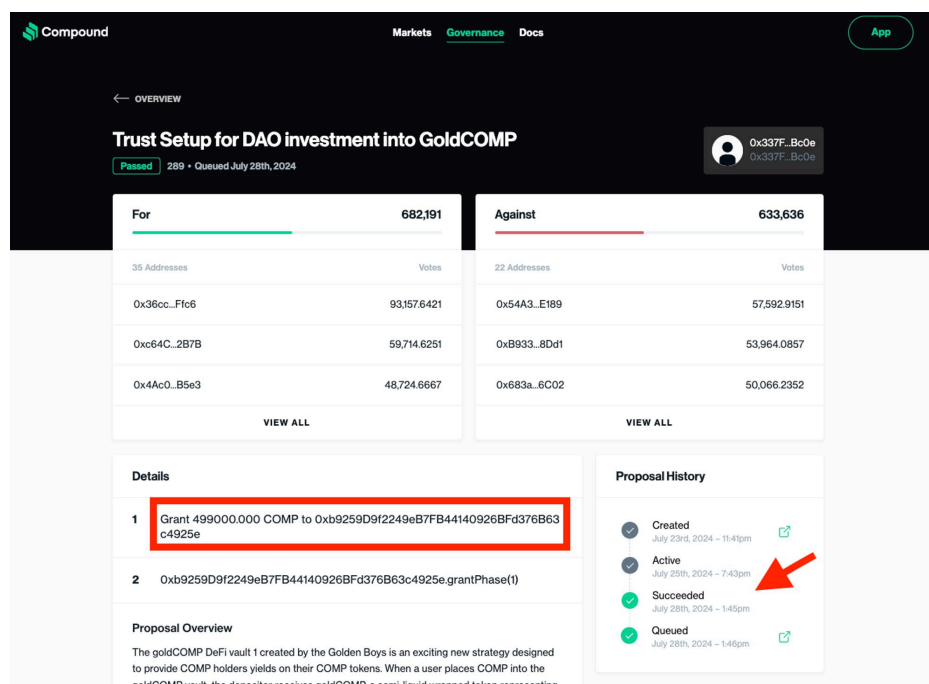
Underlying Chain: Ethereum

Amount: \$25,000,000

Attack Vector: Malicious Governance Proposal

Brief Summary

On July 28, 2024, Compound Governance passed a proposal to grant \$25 million worth of \$COMP to a "goldenboys" vault, which should have been rejected on common sense grounds. On May 6, Goldenboys proposed investing 5% of Compound Finance's treasury into a goldCOMP vault, but the community rejected it due to insufficient security measures and lack of assurance from Goldenboys. Humpy, the leader of Goldenboys, resubmitted a proposal with a "Trust Setup" to address these concerns. As a crypto whale, Humpy controlled over 10% of \$COMP's total supply, enough to push his proposal through. ([Ref.](#))

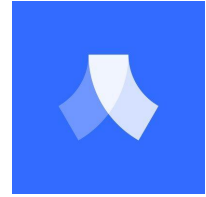


Compensation Process

Compound Finance offered to send \$3.5 million worth of reserves to the Goldenboys vault. Humpy accepted this offer and withdrew his previous proposal.



134. Anzen Finance (Jul 30)



Underlying Chain: Base

Amount: \$500,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

RWA protocol Anzen Finance announced that one of its Blast vault contracts was exploited and lost 500,000 \$USDz. The details of the exploit are yet disclosed, but it seems to be related to the decimal configuration of the stablecoin. Fortunately, the exploiter accepted the bug bounty of 10% and returned 450,000 \$USDz.

Compensation Process

On Jul 31 2024, Anzen Finance announced that the exploited funds have been returned. ([Ref.](#)) Also, as the team purchased 500K USDz immediately to minimize the sell pressure, the price impact of the exploit was negligible.

Pre-Incident Audit History

Peckshield (2024.04.28) - [Report](#), Halborn (2024.05.07) - [Report](#),

Zellic (2024.05.21) - [Report](#)

Post-Incident Audit History - N/A

Postmortem - N/A

Fund Flow

The exploiter returned 90% of the exploited assets to the project.

Exploiter ([0xDC79](#)) → Returned to the Project ([0x18B7](#), 450K USDC) / [0xB47f](#) (50K USDC)



135. Terra (Jul 31)



Underlying Chain: Terra

Amount: \$6,700,000

Attack Vector: Contract Vulnerability - Reentrancy

Brief Summary

The root cause was a known reentrancy vulnerability in the timeout callback on IBC hook, which was reported by Asymmetric Research in April 2024. While Cosmos promptly patched this bug, Terra did not include the patch in its upgrade in June 2024.

ASA-2024-007: Potential Reentrancy using Timeout Callbacks in ibc-hooks

Critical crodriguezvega published GHSA-j496-crgh-34mx on Apr 5, 2024

Package	Affected versions	Patched versions
github.com/cosmos/ibc-go (Go)	< 4.6.0	4.6.0
	< 5.4.0	5.4.0
	< 6.3.0	6.3.0
	< 7.4.0	7.4.0
	< 8.2.0	8.2.0

Severity

Critical

CVE ID

No known CVE

Weaknesses

CWE-696

Credits

mdulin2 Finder

Description

Name: ASA-2024-007: Potential Reentrancy using Timeout Callbacks in ibc-hooks
Component: ibc-go
Criticality: Critical (ACMV1: I:Critical; L:AlmostCertain)
Affected versions: < v4.6.0, < v5.4.0, < v6.3.0, < v7.4.0, < v8.2.0
Affected users: Chain Builders + Maintainers

Summary

Through the deployment and subsequent use of a malicious CosmWasm contract via IBC interactions, an attacker could potentially execute the same `MsgTimeout` inside the IBC hook for the `OnTimeout` callback before the packet commitment is deleted. On chains where `ibc-hooks` wraps ICS-20, this vulnerability may allow for the logic of the `OnTimeout` callback of the transfer application to be recursively executed, leading to a condition that may present the opportunity for the loss of funds from the escrow account or unexpected minting of tokens.

Exploiting the vulnerability, the exploiter stole 3.5 million \$axlUSDC, 500k \$USDT, 2.7 \$BTC and 60M \$ASTRO from Astroport, the major DEX protocol on Terra. Due to the exploit, \$ASTRO price plummeted over 60%.



Compensation Process - N

Pre-Incident Audit History

SCV Security - Unavailable ([Ref.](#))

Post-Incident Audit History - N

Postmortem - N

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are initially bridged to Neutron and Ethereum. Terra blacklisted the Neutron addresses, and froze the stolen 20M ASTRO. Ethereum assets are bridged to Litecoin through THORChain, 108 days after the exploit.

Exploiter ([1wrv](#))

→ Bridged to Neutron ([16wy](#), 20M ASTRO) via IBC (Frozen)

/ Ethereum ([0xBDe1](#)) via Axelar Squid

→ Deposited to [eXch](#) ([0xCE66](#), [0x7eB8](#), [0xc94b](#), [0x4c63](#), [0x3818](#) - 54 ETH)

/ Bridged via THORChain (Ethereum → Litecoin) ([0x81BE](#), [0x5939](#), [0xe357](#) - 273.5 ETH) →
Litecoin accounts [1qtt](#), [1qqf](#), [1qln](#)

Note

Terraform Labs suspended its operation due to the settlement with SEC and its bankruptcy. Therefore, Terra blockchain is now driven by the community. ([Ref.](#)) On Sep 25, they announced the shutdown of several ecosystem services, including Enterprise Protocol, Warp Protocol, Station Wallet, Alliance Hub, Finder Block Explorer, Foundation, IBC Relayers and several API services. ([Ref.](#))



136. Convergence Finance (Aug 1)



Underlying Chain: Ethereum

Amount: \$210,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

The root cause was lack of input validation in the function that mints \$CVG for eligible stakers, which allowed a malicious contract as `claimContracts`. As an arbitrary value as `cvgClaimable` could be passed to the function, the exploiter could claim rewards for all the staked tokens in the contract.

```
function claimMultipleStaking(
    ICvxStakingPositionService[] calldata claimContracts,
    address _account,
    uint256 _minCvgCvxAmountOut,
    bool _isConvert,
    uint256 cvxRewardCount
) external {
    require(claimContracts.length != 0, "NO_STAKING_SELECTED");

    /// @dev To prevent an other user than position owner claims through swapping and grief the user rewards in cvgCVX
    if (_isConvert) {
        require(msg.sender == _account, "CANT_CONVERT_CVG_FOR_OTHER_USER");
    }
    /// @dev Accumulates amounts of CVG coming from different contracts.
    uint256 _totalCvgClaimable;

    /// @dev Array merging & accumulating rewards coming from different claims.
    ICommonStruct.TokenAmount[] memory _totalCvxClaimable = new ICommonStruct.TokenAmount[] (cvxRewardCount);

    /// @dev Iterate over all staking service
    for (uint256 stakingIndex; stakingIndex < claimContracts.length; ) {
        ICvxStakingPositionService cvxStaking = claimContracts[stakingIndex];

        /** @dev Claims Cvg & Cvx
         * Returns the amount of CVG claimed on the position.
         * Returns the array of all CVX rewards claimed on the position.
         */
        (uint256 cvgClaimable, ICommonStruct.TokenAmount[] memory _cvxRewards) = cvxStaking.claimCvgCvxMultiple(
            _account
        );
        /// @dev increments the amount to mint at the end of function
        _totalCvgClaimable += cvgClaimable;
    }
}
```

Convergence stated that the vulnerability was made after the code audits for gas optimization, which removed the line of code that was validating the given input(!). ([Ref.](#))

Compensation Process - N/A



Pre-Incident Audit History

Halborn (2023.04.26) - [Report](#), Hats Finance (2023.09.19) - [Report](#),

Sherlock (2023.11.30) - [Report](#), Hats Finance (2024.05.16) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.08.02

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are laundered through Tornado Cash, right after the exploit.

Exploiter ([0x0356](#)) → Laundered through Tornado Cash (65.8 ETH)



137. Satoshi Token (Aug 3)

Underlying Chain: Ethereum

Amount: \$70

Attack Vector: Contract Vulnerability - Misimplemented Transfer Function

Brief Summary

The root cause was a misimplemented transfer function that automatically decreased the balance if the recipient was a Uniswap pair.

```
function _transferFrom(address sender, address recipient, uint256 amount) internal {
    if(limitsInEffect){
        if (sender == pair || recipient == pair){
            require(amount <= maxTxnAmount, "Max Transaction Amount Exceeded");
        }
        if (recipient != pair){
            require(balanceOf(recipient) + amount <= maxWallet, "Max Wallet Amount Exceeded");
        }
    }

    if(recipient == pair){
        if(autoBlockReward && shouldReward()){
            blockReward();
        }
        if (shouldReward()) {
            autoSatoshi();
        }
    }
}
```

```
function autoSatoshi() internal {
    nextSatoshi = block.timestamp + satoshiFrequency;

    uint256 liquidityPairBalance = balanceOf(pair);

    uint256 amountToContribute = liquidityPairBalance * percentForSatoshi / 10000;

    if (amountToContribute > 0) {
        uint256 unitAmountToContribute = amountToContribute * _unitsPerCoin;
        _unitBalances[pair] -= unitAmountToContribute;
        _unitBalances[address(0xdead)] += unitAmountToContribute;
        emit Transfer(pair, address(0xdead), amountToContribute);
    }

    InterfaceLP _pair = InterfaceLP(pair);
    _pair.sync();
    emit SatoshiContribution(amountToContribute);
}
```

Exploiting this vulnerability, the attacker swapped some WETH for Satoshi tokens and repeatedly called the transfer function. Subsequently, they exchanged some Satoshi tokens back to WETH.



Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0xc9a5](#))



138. OCF Token (Aug 5)

Underlying Chain: BSC

Amount: \$578,000

Attack Vector: Contract Vulnerability - Price Dependency

Brief Summary

The root cause was the use of the `getReserves` function in the OCF staking contract, which calculates the token price. In the `stake` and `withdraw` functions, the OCF token price is determined using the `getReserves` function from the Uniswap pair. The exploiter manipulated this by swapping large amounts of tokens, allowing them to withdraw significantly more tokens than they staked.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Most of the exploited assets are laundered through Tornado Cash, 7 days after the exploit.

Exploiter ([0xb96e](#))

→ Laundered through Tornado Cash ([0xDDa7](#), 83.3 BNB), Leftover deposited do Kucoin ([0xeF77](#))



139. Ronin Bridge (Aug 6)



Underlying Chain: Ethereum

Amount: \$11,800,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

The root cause is the failure to properly initialize the operator weight configuration needed for cross-chain transaction confirmation during the Ronin Bridge's contract upgrade. The `minimumVoteWeight` parameter is set to zero, allowing any signature to pass cross-chain verification. The actors withdrew approximately 4,000 ETH and 2 million USDC, valued at around \$12 million. This is the maximum amount ETH and USDC allowed for a single transaction withdrawal from the bridge, as the bridge limit was set as a safeguard.

Compensation Process

User funds are not affected by the exploit, since the exploited assets were fully returned.

Pre-Incident Audit History

Verichains (2022.06.28) - [Report](#), Beosin (2023.09.08) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.08.06

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#)

Fund Flow

Fortunately, the exploit transaction was frontrun by an MEV bot and the bot returned the entire amount of exploited assets to the project with \$500K (~4.2%) as a bug bounty. ([Ref.](#))



140. OMPX Token (Aug 7)

Underlying Chain: Ethereum

Amount: \$10,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The root cause was a faulty logic in the token sale contract that discounted the token price when the purchase amount exceeded a specific threshold.

```
function getPurchasePrice(uint256 purchaseValue, uint256 amount) public view returns(uint256 price_) {
    require(purchaseValue >= 0);
    require(amount >= 0);
    uint256 buyerContributionCoefficient = getDiscountByAmount(amount);
    uint256 price = getBuyBackPrice(purchaseValue).mul(buyerContributionCoefficient).div(descPrecision);
    if (price <= 0) {price = 1e11;}
    return price;
}
```

An exploiter took advantage of this by buying large quantities of tokens, selling them, and repeating the process multiple times. Ultimately, he gained approximately 4.3 ETH.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter laundered the assets through Railgun and ChainFlip, and deposited half to Binance, 2 days after the exploit.

Exploiter (0x40D1) → Laundered through Railgun (4.46 ETH) / Sent to [0xd1ca](#), distributed to various EOAs & Deposited to Binance (4.44 ETH) / Laundered via ChainFlip ([0x5E8A](#), 0.79ETH)



141. Nexera (Aug 7)



Underlying Chain: Ethereum

Amount: \$1,830,000

Attack Vector: Control Hijacking

Brief Summary

An external attacker could access the credentials for managing Nexera Funder platform's smart contracts. With these credentials, the attacker transferred NXRA tokens from Fundrs' staking contracts on Ethereum. Of the 47.24 million NXRA tokens stolen, the attacker sold only 14.75 million tokens.

Compensation Process

After the exploit, Nexera promptly retrieved the remaining 32.5 million NXRA from the attacker's wallet and prevented further loss.

Pre-Incident Audit History

Omniscia (2023.11.17) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.08.09

Exploit Transactions - [Tx](#)



Fund Flow

Right after the exploit, the team promptly retrieved 32.5M NXRA and burned them. Others were swapped to USDT, bridged multiple times and moved to Tron. The assets are consequently moved to Binance, OKX and other CEX, 108 days after the exploit.

Exploiter ([0xe697](#))

→ Bridge via LayerZero Stargate (Ethereum → BSC) ([0xE697](#))

→ [0xd138](#), [0xaDB7](#), [0x1E6c](#), [0xf8d0](#)

> Bridged via Allbridge (BSC → Tron) ([TCVg](#), 203K USDT) / Bridged via LayerZero Stargate, Allbridge (BSC → Polygon → Tron) ([TS33](#), 550K USDT)

→ Deposited to Blance ([TBJm](#)) / OKX ([TMoy](#)) / CEX-suspected address ([TYQL](#))

Exploiter ([0xe697](#)) → Burned (32.5M NXRA)



142. iVest DAO (Aug 12)

Underlying Chain: BSC

Amount: \$172,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The root cause was the misimplemented donation function, which burned more tokens of the source address than intended, crashing the exchange rate of the token.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are directly laundered through Tornado Cash, 71 days after the exploit.

Exploiter ([0x4645](#)) → Laundered through Tornado Cash (103.3 BNB)

Note

The project seems inactive in its current state, since its official X account has been deleted.



143. Vow (Aug 14)



Underlying Chain: Ethereum

Amount: \$1,200,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

The exchange rate of \$VOW to \$vUSD was changed from 5 to 100, enabling a bot to buy \$VOW from Uniswap and gain 2 billion \$vUSD. The team claimed it was not a rugpull trial, but an error during testing of their new lending contract. It is quite questionable that the team tested such an important factor directly to a live contract on mainnet, not on the testnet contract.



Subject: Update on Recent Market Incident and Ongoing Efforts

Dear Community,

We want to clarify the recent incident that has occurred while our team was testing the USD rate setter function of the v\$ contract in order to mint v\$ for the new lending pool and oracle functions.

The USD rate was amended and 1m VOW was sent to the contract to test that everything was working as expected. This created, as expected v\$100m, which is subsequently to be burned.

This time gap of making this change, testing, and reverting was around 15-30 seconds.

During this process, an unexpected event occurred where a bot acquired around 20 million VOW tokens from Uniswap and sent them to the contract resulting in nearly v\$2 billion being created. The bot subsequently then sold the \$2 billion v\$ into the Uniswap pool.

Compensation Process

To minimize the impact of oversupplied \$vUSD, the team swapped 20M VOW into vUSD and burned 700M vUSD. vUSD's burn rate is also increased to 50% to bring the supply back. ([Ref.](#))



Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.08.14

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are laundered through Tornado Cash, 6 days after the exploit.

Exploiter ([0x48de](#)) → Laundered through Tornado Cash ([0x48de](#), [0xa153](#) - 488.8 ETH)



144. HFLH Token (Aug 23)



Underlying Chain: BSC

Amount: \$5,300

Attack Vector: Contract Vulnerability - Price Dependency

Brief Summary

The price calculation function of the token contract was misimplemented, as it was naively implemented to divide one token's balance into another. Through donations, the exploiter could crash the exchange rate of the token and drain the pool.

```
function getPrice() public view returns (uint256) {
    uint256 usdtBalance = IERC20(usdtAddress).balanceOf(lpAddress);
    uint256 HFLHBalance = IERC20(HFLHAddress).balanceOf(lpAddress);

    require(usdtBalance > 0, "USDT balance is zero");

    uint256 price = 1e18*usdtBalance/ HFLHBalance;
    return price;
}
```

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are moved to Binance, right after the exploit.

Exploiter ([0x9F3e](#)) → Deposited to Binance ([0xfE64](#))



145. Aave Peripheral Contract (Aug 28)



Underlying Chain: Ethereum

Amount: \$56,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

Aave's peripheral contract, which interacted with the Paraswap Contract, lacked proper calldata validation, leading to an arbitrary call vulnerability. If a swap failed or was only partially executed, the contract left a high token allowance unadjusted. This oversight allowed attackers to withdraw unauthorized tokens. The vulnerability stemmed from the function's failure to properly validate or sanitize `paraswapData` and verify the swap outcome. The attacker exploited this by crafting malicious `paraswapData`, manipulating the swap process, or bypassing it entirely. By taking advantage of the unchecked token allowance, the attacker circumvented the intended swap logic, enabling unauthorized fund transfers from the contract.

Compensation Process

The victims were Aave peripheral contracts, not the contracts that Aave operates. The smart contracts affected interact exclusively with the user using it, and with the contract itself.

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are deposited to Defiway, a centralized crosschain bridge, 26 days after the exploit.

Exploiter ([0x6ea8](#)) → Deposited to Defiway ([0x2e13](#), 15 ETH)



146. Penpie (Sep 3)



Underlying Chain: Ethereum

Amount: \$27,000,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

Penpie, a project for yield farming on Pendle, suffered a \$27M loss due to vulnerabilities in its reward contract. The issue stemmed from the ability for anyone to register arbitrary pools in the PendleMarketRegisterHelper and PendleMarketFactoryV3 contracts. Pool verification merely checked for the pool's existence in the allMarkets data structure of the PendleMarketFactory contract. However, the createNewMarket function, which adds new pools to allMarkets, was also permissionless, essentially rendering the verification process ineffective. The reward claim function in PendleStaking calculated rewards based on balance differences before and after calling the redeemRewards function. Exploiting the re-entrancy vulnerability in PendleStaking's depositMarket, the attacker registered a malicious pool to drain rewards.

```
poolInfo.lastHarvestTime = block.timestamp;
address[] memory bonusTokens = IPendleMarket(_markets[i]).getRewardTokens();
uint256[] memory amountsBefore = new uint256[](bonusTokens.length);
for (uint256 j; j < bonusTokens.length; j++) {
    if (bonusTokens[j] == NATIVE) bonusTokens[j] = address(WETH);
    amountsBefore[j] = IERC20(bonusTokens[j]).balanceOf(address(this));
}
IPendleMarket(_markets[i]).redeemRewards(address(this));
for (uint256 j; j < bonusTokens.length; j++) {
    uint256 amountAfter = IERC20(bonusTokens[j]).balanceOf(address(this));
    uint256 originalBonusBalance = amountAfter - amountsBefore[j];
    uint256 leftBonusBalance = originalBonusBalance;
    uint256 currentMarketHarvestPendleReward;
    if (originalBonusBalance == 0) continue;
    if (bonusTokens[j] == PENDLE) {
        currentMarketHarvestPendleReward =
            (originalBonusBalance * harvestCallerPendleFee) /
```

Malicious Pendle Market returns legit Pendle Market as reward tokens

Reentrancy from malicious SY contract to deposit legit Pendle Market

Reward amount is manipulated by reentrancy attack

Compensation Process

Through a governance proposal, Penpie announced its recovery plan. They shared their team allocation and treasury assets to the users and decided to use 20% of their future revenues for the recovery. ([Ref.](#), [Ref.](#))



Pre-Incident Audit History

WatchPug (2023.06.05) - [Report](#), WatchPug (2023.06.28) - [Report](#),

Zokyo (2023.06.22) - [Report](#), Zokyo (2023.07.13) - [Report](#),

AstraSec (2024.07.24) - [Report](#)

Post-Incident Audit History - N.A

Postmortem

[Link](#) - 2024.09.05

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#)

Fund Flow

Exploited assets are laundered through Tornado Cash, 4 days after the exploit.

Exploiter ([0x7A2f](#), [0xc0eb](#))

→ [0x28e3](#) (11113.6 ETH)

→ Laundered through Tornado Cash ([0xD440](#), [0x415a](#))



147. Pythia (Sep 3)

Underlying Chain: Ethereum

Amount: \$53,000

Attack Vector: Contract Vulnerability - Reentrancy

Brief Summary

The algorithmic stablecoin Pythia suffered a loss of \$53,000 due to a vulnerability in its staking contract. The issue was caused by a reentrancy vulnerability in the reward claim function.

```
function claimRewards() external {
    distributeRewards();
    uint256 rewardAmount = setupClaim(_msgSender());
    uint256 escrowedRewardAmount = rewardAmount * escrowPortion / 1e18;
    uint256 nonEscrowedRewardAmount = rewardAmount - escrowedRewardAmount;

    if(escrowedRewardAmount != 0 && address(escrowPool) != address(0)) {
        escrowPool.vestingLock(_msgSender(), escrowedRewardAmount);
    }

    // ignore dust
    if(nonEscrowedRewardAmount > 1) {
        rewardToken.safeTransfer(_msgSender(), nonEscrowedRewardAmount);
    }

    emit RewardsClaimed(_msgSender(), escrowedRewardAmount, nonEscrowedRewardAmount);
}
```

Since the claimRewards() is reentrant & distributeRewards function is called before the setupClaim, the exploiter could retrieve rewards multiple times via claimRewards().

Compensation Process - N

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [TxS](#)



Fund Flow

Exploited assets are laundered through Tornado Cash, right after the exploit.

Exploiter ([0xd861](#)) → Laundered through Tornado Cash (21 ETH)

Note

Following the incident, the project's Twitter account has been deleted, suggesting the possibility of rugpull.



148. DAI (Sep 4)



Underlying Chain: Polygon, Base

Amount: Unknown

Attack Vector: Control Hijacking - Third Party Exploit

Brief Summary

On Sep 4 2024, @godsflaw posted a tweet that the authority over L2 DAI deployment has been compromised, affecting all Layer 2 networks except Optimism and Arbitrum. The issue arose because DAI addresses were generated using a vanity address tool called Profanity, which was later found to have a vulnerability. Exploiting this flaw, an attacker has been able to deploy unauthorized DAI honeypots on other chains where DAI had not been officially deployed, potentially leading to user confusion and financial risks. ([Ref.](#))

**Christopher Mooney**  @godsflaw · Sep 4, 2024 

 DAI Security PSA for wallet providers, block explorers, and users. The L2 DAI deployer (0x075da589886BA445d7c7e81c472059dE7AE65250) for the DAI vanity address (0xDA10009cBd5D07dd0CeCc66161FC93D7c9000da1) has been compromised. (1/7)

 7  111  268  80K  

[Show more replies](#)

**Christopher Mooney**  @godsflaw · Sep 4, 2024 

The attacker can now deploy malicious contracts on other chains with the same address as DAI. They've already deployed a honeypot on Base: [basescan.org/address/Oxda10...](https://basescan.org/address/Oxda10009cbd5d07dd0cecc66161fc93d7c9000da1) (6/7)



basescan.org

Fake_Phishing38349 | Address Oxda10009cbd5d07dd0cecc66161fc93d7c9000da1

The Contract Address

Oxda10009cbd5d07dd0cecc66161fc93d7c9000da1

 2  5  23  10K  



149. CUT Token (Sep 10)

Underlying Chain: BSC

Amount: \$1,400,000

Attack Vector: Contract Vulnerability - Price Manipulation

Brief Summary

The root cause of the exploit was misimplemented price dependency in the token contract. When removing liquidity from the contract, the (leftAmount - amount) is added to the recipient's balance. As leftAmount is calculated through the \$CUT token amount of the pancakeswap pair, the exploiter could manipulate the value through the flash loan attack.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are transferred multiple times, and currently staying at 0x8383. Last transaction was sent 103 days after the exploit.

Exploiter ([0x560a](#))

→ [0x5766](#)

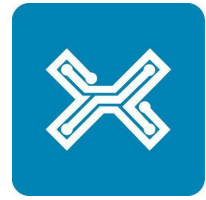
→ Distributed to multiple EOAs

→ Merged to [0xF008](#) (644K USDT)

→ [0x8383](#)



150. Indodax (Sep 11)



Underlying Chain: Bitcoin, Ethereum, Polygon, Optimism, BSC, Tron

Amount: \$22,000,000

Attack Vector: Control Hijacking - Web2 exploit

Brief Summary

Indonesian CEX Indodax suffered a hot wallet drainage and lost \$22M worth of its cryptocurrency. Indodax stated that it was due to the systematic vulnerabilities of its platform.

Compensation Process

Indodax claimed that user assets are not affected and disclosed their proof of reserves. ([Ref.](#)) Plus, the Indodax cofounder @WilliamSutant0 stated that all the loss from the exploit will be covered. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.09.11

Exploit Transactions - [Ref.](#)

Fund Flow

On Sep 11 2024, @tayvano_ provided the addresses where the exploited assets headed to.

Exploited assets headed to multiple addresses, and the laundry seems to be still ongoing.

[bc1q5u](#) (25 BTC) → Deposited to Bybit ([bc1qsr](#))

[TBoo](#)(16.7M TRX) → Deposited to FixedFloat, OKX, Binance, Gate.io



[0x90ff](#) (6.9M POL) → Deposited to Bitget Wallet ([0x7da7](#), 144K POL)

/ Bridged via Across Protocol (Polygon → Arbitrum → Ethereum) ([0xae39](#)) → ... → [0x249B](#) (847K USDT)

/ Bridged via Across Protocol, OKX Web3 Proxy (Polygon → Ethereum) ([0xDC18](#), [0x0653](#), [0xC683](#), [0x2B92](#), [0xD2bc](#), [0xDF1B](#), [0x79C6](#), [0x65da](#)) → Deposited to eXch (397 ETH, 320K DAI)

[0x5910](#), [0xb0a2](#) (5204 ETH & \$1.2M of ERC 20 tokens)

→ Deposited to Bitget Wallet ([0x7668](#), [0xed58](#) 721K USDT), [0xCb6b](#) (52.6 ETH), [0x367B](#) (60 ETH), Binance ([0xF97f](#), 30 ETH)

/ Bridged via THORChain (Ethereum → Bitcoin) ([bc1qhu](#), [bc1q4a](#), [bc1qef](#), [bc1qy0](#), [bc1qhw](#), [bc1qc4](#), [bc1qtu](#), [bc1qw5](#), [bc1pag](#), [bc1p60](#) - 2510.27 ETH)

/ Laundered through Chainflip ([0x9Fd1](#), [0x9035](#), [0xd004](#), [0x0614](#), [0x5744](#), [0x6162](#) - 247 ETH)

/ Bridged via OKX Web3 Proxy (Ethereum → Solana) ([Gg9g](#), 142K USDC)

/ Bridged via THORChain (Ethereum → Litecoin) ([ltc1qk2](#), 93 ETH)

/ Bridged via Across Protocol (Ethereum → Arbitrum) ([0x618F](#), 20 ETH) & Deposited to Bitget Wallet ([0x618F](#), 66K USDT)

[0x3b8f](#) (380 ETH)

→ Bridged via Across Protocol (Optimism → Ethereum) ([0xF5f5](#), [0x8ddF](#), [0xef06](#), [0x3b19](#) - 261.62 ETH), LayerZero Stargate ([0x11E1](#), 79.7 ETH)

/ Bridged via Across Protocol (Optimism → Arbitrum) ([0x8058](#), 38.75 ETH)

→ Deposited to eXch ([0xF5f5](#), [0x11E1](#), [0x8ddF](#), [0xef06](#), [0x3b19](#) - 341.31 ETH), Bitget Wallet ([0x8058](#), 38.75 ETH)



151. BaseBros Finance (Sep 13)

Underlying Chain: Base

Amount: \$130,000

Attack Vector: Rugpull

Brief Summary

BaseBros Finance, a yield farming project on Base, rugged through the backdoor of its unverified vault contract. After the exit scam, the audit firm ChainAudits published a post mortem that the contract was out of the audit scope. ([Ref.](#))



Incident Report

Yesterday on 13.09.2024, @BaseBrosFi, a DeFi project on [@base](#), executed a rug pull by gaining control of and draining ecosystem funds via an unaudited and unverified Vault contract.

The BaseBrosFi team exploited the unverified Vault Contract by overriding critical functions in the Strategy Contract, allowing them to withdraw funds and “retire” all Strategy Contracts. This attack led to the draining of multiple pools associated with BaseBrosFi, while the Seamless protocol was mistakenly believed to be affected due to the contract titles.

ChainAudits had no involvement with the unverified contracts used in the exploit, and the contracts deployed after the audit were not provided to ChainAudits, therefore not audited and not included in our audit report.

For full clarification including how the team was able to gain access to ecosystem funds, please view the full report on our GitHub repository:

github.com/ChainAudits/Pr...

Since ChainAudits made all their reports and post-mortem unavailable, they are not available in their current state.



Pre-Incident Audit History

ChainAudits (2024.08.20) - [Report](#) (Unavailable)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.09.14 (Unavailable)

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged to Ethereum and laundered through Tornado Cash, right after the exploit.

Exploiter → Laundered through Tornado Cash ([0x022E](#), 54.3 ETH)



152. OTSea (Sep 13)



Underlying Chain: Ethereum

Amount: \$26,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

OTSea's staking contract had a simple but critical logic bug that did not reflect the withdrawal from the user deposit amount, therefore allowed multiple withdrawals for a single deposit.

```
function _withdrawMultiple(
    uint256[] calldata _indexes
) private returns (uint88 totalAmount, uint256 totalRewards) {
    uint256 length = _indexes.length;
    _validateListLength(length);
    uint256 total = getTotalDeposits(_msgSender());
    uint32 currentEpoch = _currentEpoch;
    for (uint256 i; i < length; ) {
        if (total <= _indexes[i]) revert DepositNotFound(_indexes[i]);
        totalRewards += _calculateRewards(_msgSender(), _indexes[i]);
        Deposit memory deposit = _deposits[_msgSender()][_indexes[i]];
        if (deposit.rewardReferenceEpoch == 0) revert OTSeaErrors.NotAvailable();
        _deposits[_msgSender()][_indexes[i]].rewardReferenceEpoch = 0;
        /**
         * @dev if the rewardReferenceEpoch is in the future, it means that the user deposited in the current
         * epoch (currentEpoch). Therefore next epoch's total stake needs to be reduced by the user's deposit.
         *
         * If the rewardReferenceEpoch is less than or equal to the currentEpoch it means that the user
         * either deposited or claimed rewards in a past epoch. Either way it means that the user's
         * deposit cannot possibly be in the future therefore the current epoch's total stake needs to be reduced
         */
        _epochs[
            currentEpoch < deposit.rewardReferenceEpoch
                ? deposit.rewardReferenceEpoch
                : currentEpoch
        ].totalStake -= deposit.amount;
        totalAmount += deposit.amount;
        unchecked {
            i++;
        }
    }
    return (totalAmount, totalRewards);
}
```

Regarding the exploit, there are no official announcements from the team.

Compensation Process - N/A

Pre-Incident Audit History

Dedaub (2023.12.18) - [Report](#), Peckshield (2024.01.30) - [Report](#)

Post-Incident Audit History - N/A



Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

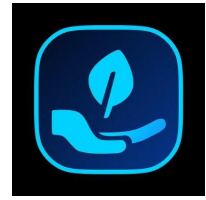
Exploiter deposited the exploited assets to various destinations, 56 days after the exploit.

Exploiter ([0x0000](#))

→ Deposited to HTX ([0x6aaF](#), [0xF555](#), [0xd661](#) - 5 ETH), SideShift ([0xedBb](#), 0.07 ETH), Tap ([0xbf68](#), 0.01 ETH)



153. MintStakeShare (Sep 15)



Underlying Chain: BSC

Amount: \$50,000

Attack Vector: Contract Vulnerability - Price Dependency

Brief Summary

The \$MSS token contract included a function called 'buyWithMint', which simultaneously received \$BNB, minted \$MSS, and provided liquidity to the Pancakeswap pool. However, the function failed to consider the pool's state when calculating the \$MSS mint amount, allowing an attacker to use a \$BNB flash loan to mint \$MSS, swap it all, and drain \$180K from the LP pool. Interestingly, \$130K was transferred to the token deployer during the attack, though it's unclear if this was intentional. Notably, an identical attack occurred on August 19th, with a post-mortem published, raising questions about the effectiveness of their patch. The project is currently undergoing an audit by PeckShield to address the security concerns.

Compensation Process - N/A

Pre-Incident Audit History

G0 Group (2021.10) - [Report](#), SolidProof (2024.07.26) - [Report](#)

Post-Incident Audit History

Peckshield (2024.10.25) - N/A ([Ref.](#))

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0x6F9E](#)) → [0xA889](#) (7.27 WETH)



154. WXETA Token (Sep 16)

Underlying Chain: Ethereum

Amount: \$65,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

As the token contract did not set `initialized` to true after executing the initialize function, it allowed anyone to initialize the token contract.

```
function initialize(uint256 max) public {
    WXETASTORAGE storage s = getWXETASTorage();
    require(!s.initialized, "WXETA: already initialized");
    s._maxSupply = max;
    s.owner = msg.sender;
    s.authorized[msg.sender] = true;
    s.name = "Wrapped Xeta";
    s.symbol = "WXETA";
    s.decimals = 18;
}
```

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter bridged the exploited assets to Bitcoin, 30 days after the exploit.

Exploiter ([0x6d30](#)) → Bridged via THORChain (Ethereum → Bitcoin) ([bc1q7e](#), 27.6 ETH)



155. DeltaPrime(2) (Sep 16)



Underlying Chain: Arbitrum

Amount: \$6,000,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

Due to the private key leakage, the exploiter could set himself as the admin account, upgrade the contract through proxy and withdraw the entire liquidity from the exploited contract. While DeltaPrime is being operated also on Avalanche, the Avalanche contract was not affected by the exploit as it is operated by multisigs and cold wallets. Regarding the exploit, on-chain investigator ZachXBT raised a potential linkage with Lazarus. ([Ref.](#))

Compensation Process

On Oct 9 2024, DeltaPrime published its reimbursement plan with postmortem. It is as follows:

1. Utilization of \$1.33M available assets in the stability pool to restore the liquidity
2. Allocating 33% of the future revenues for reimbursement
3. Additional 40% compensation for affected users

Pre-Incident Audit History

Piotr Szlachciak (2021.12.04) - [Report](#), Peckshield (2022.02.12) - [Report](#),

Chainsulting (2022.04.11) - [Report](#), Peckshield (2022.12.01) - [Report](#),

Peckshield (2023.05.27) - [Report](#), Peckshield (2023.09.08) - [Report](#),

AstraSec (2024.06.19) - [Report](#), ABDK (2024.06.24) - [Report](#),

AstraSec (2024.08.07) - [Report](#)



Post-Incident Audit History

BlockSec (2024.10.01) - [Report](#)

Postmortem

[Link](#) - 2024.10.09

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged to Ethereum and laundered through Tornado Cash, right after the exploit.

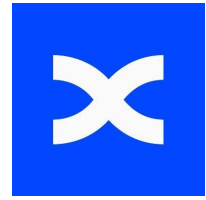
Exploiter ([0xd550](#))

→ Bridged via LayerZero Stargate (Arbitrum → Ethereum)

→ Laundered through Tornado Cash ([0x0BCf](#), 1337.5 ETH)



156. BingX (Sep 20)



Underlying Chain: Ethereum

Amount: \$44,000,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

Singaporean CEX BingX suffered hot wallet drainage as the system allowed abnormal access to the exploiter.

Compensation Process

BingX promised full compensation of user assets via their treasury, while claiming the total loss minimal and manageable. ([Ref.](#))

On the day of the exploit, BingX CEO Vivien Lin held an AMA as a response to the exploit. The details of the exploit are not disclosed.

Postmortem

[Link](#) - 2024.09.20

Exploit Transactions - [Txs](#)

Fund Flow

Exploited assets had moved with a very similar pattern to the Indodax incident. The assets were swapped into ETH, bridged to Bitcoin / Litecoin via THORChain, and deposited to OKX. ([Ref.](#))



157. BananaGun (Sep 20)



Underlying Chain: Ethereum

Amount: \$1,900,000

Attack Vector: Misc.

Brief Summary

BananaGun, a token swap sniping tool, allowed unauthorized token transfers from several user wallets. Promptly after the exploit, the backend system was turned off and limited the affected user number lower than 10. The team claims that the exploit stems from a frontend vulnerability.



Banana Gun 🍌🔫🏆
@BananaGunBot OP



UPDATE ON BOT SITUATION

Today, some users of Banana Gun experienced unauthorized transfers from their wallets. Promptly after the first incident, we immediately switched off the bot and began diligently checking our back-end.

We have confirmed that our back-end is not compromised. Both the router and database have been thoroughly inspected, and only a very small number of users (fewer than 10) were affected. Additionally, the transfers appear to have been executed manually.

This leads us to believe the issue may stem from a front-end vulnerability.

As we prioritize security, we will keep our bot offline while we investigate the root cause. The amount of support we've received, particularly from our partners, has been truly heartwarming. If you have any insights that may help us, feel free to send us a direct message here on Twitter.

12:27 AM · Sep 20, 2024 · 89.1K Views



💬 96

↻ 99

❤️ 452

🔖 21



158. Shezmu (Sep 20)



Underlying Chain: Ethereum

Amount: \$638,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

Shezmu, a platform for earning interest on cryptocurrency, experienced a \$4.9 million hit in their \$ShezUSD when an intruder identified a loophole, enabling them to create unlimited collateral to borrow ShezUSD. While these tokens weren't very liquid, meaning the hacker probably couldn't cash out the full amount, the incident was still significant. In an effort to reclaim the lost funds, Shezmu initially proposed a 10% reward for their return. However, the hacker indicated they'd consider no less than a 20% cut. Eventually, Shezmu accepted the renegotiated bounty and later informed their community about recuperating the funds with assistance from the "whitehat" hacker.

Compensation Process

Since most of the exploited assets were returned by the hacker, the team fully recovered their assets thus user assets were merely affected. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History

Hashlock (24.09.25) - [Ref.](#)

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

The exploited assets were initially laundered through Railgun. ([0x412C](#), 261 ETH) 80% of the exploited assets were returned, a day after the exploit. ([Tx](#))



159. Bankroll Network (Sep 22)



Underlying Chain: BSC

Amount: \$230,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The victim contract took charge of the token swap as buy and sell functions are implemented in it. Users buying tokens send WBNB to the pool, where a portion is deducted as a fee, rewarding Liquidity Providers. As a result, "profitPerShare_" grows with each token purchase.

```
function allocateFees(uint fee) private {  
  
    // 1/5 paid out instantly  
    uint256 instant = fee.div(5);  
  
    if (tokenSupply_ > 0) {  
        // Apply instant divs  
        profitPerShare_ = SafeMath.add(profitPerShare_, (instant * magnitude) / tokenSupply_);  
    }  
  
    // Add 4/5 to dividend drip pools  
    dividendBalance_ += fee.safeSub(instant);  
}
```

The flaw that allowed hacking lies within the "buyFor" function, which permits token purchases on another user's behalf. If the "_customerAddress" matches the victim contract's address, it allows anyone to inflate "profitPerShare_" without needing actual WBNB tokens.

```
/// @dev Converts all incoming eth to tokens for the caller, and passes down the referral addy (if any)  
function buyFor(address _customerAddress, uint buy_amount) public returns (uint256) {  
    require(token.transferFrom(_customerAddress, address(this), buy_amount));  
    totalDeposits += buy_amount;  
    uint amount = purchaseTokens(_customerAddress, buy_amount);  
  
    emit onLeaderBoard(_customerAddress,  
        stats[_customerAddress].invested,  
        tokenBalanceLedger[_customerAddress],  
        stats[_customerAddress].withdrawn,  
        now  
    );  
  
    //distribute  
    distribute();  
  
    return amount;  
}
```



The attacker pre-acquired tokens before the exploit and repeatedly triggered the "buyFor" function, significantly inflating "profitPerShare_". With this manipulation, they executed the "withdraw" function to extract a substantial amount of WBNB tokens, ultimately reaping \$230k.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets were laundered through Tornado Cash, 29 days after the exploit.

Exploiter ([0x4645](#)) → Laundered through Tornado Cash (103.3 BNB)

Note

The X account of the project has been inactive since Feb 2022, so it seems the project has been deprecated for a long time.



160. Inferno (Sep 24)



Underlying Chain: Ethereum

Amount: \$440,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The victim's contract includes a "swapTitanXForInfernoAndBurn" feature, which allows for the conversion of "Titan" tokens to "Inferno" tokens, followed by burning the "Inferno" tokens. This function is publicly accessible, making it vulnerable to a sandwich attack.

```
/**
 * @notice Swaps TitanX for Inferno and burns the Inferno tokens
 * @param _amountBlazeMin Minimum amount of Blaze tokens expected
 * @param _deadline The deadline for which the passes should pass
 */
function swapTitanXForInfernoAndBurn(uint256 _amountBlazeMin, uint32 _deadline)
    external
    nonReentrant
    intervalUpdate
{
    if (!liquidityAdded) revert NotStartedYet();
    Interval storage currInterval = intervals[lastIntervalNumber];
    if (currInterval.amountBurned != 0) revert IntervalAlreadyBurned();

    currInterval.amountBurned = currInterval.amountAllocated;

    uint256 incentive = (currInterval.amountAllocated * INCENTIVE_FEE) / BPS_DENOM;

    uint256 titanXToSwapAndBurn = currInterval.amountAllocated - incentive;

    uint256 blazeAmount = _swapTitanForBlaze(titanXToSwapAndBurn, _amountBlazeMin, _deadline);
    uint256 infernoAmount = _swapBlazeForInferno(blazeAmount, _deadline);

    burnInferno();

    TransferHelper.safeTransfer(address(titanX), msg.sender, incentive);

    emit BuyAndBurn(titanXToSwapAndBurn, infernoAmount, msg.sender);
}
```

Malicious actors managed to swap a significant amount of "Titan" for "Blaze," then triggered the "swapTitanXForInfernoAndBurn" operation. Subsequently, they converted "Blaze" back into "Titan" and then into "Eth." To safeguard against unauthorized usage, implementing a restriction such as "onlyOwner" as a modifier on the "swapTitanXForInfernoAndBurn" feature would be prudent.



Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

The exploit involved hundreds of transactions, and the exploiter's profit had transferred multiple times. The exploited assets are periodically deposited to Binance, and the latest transaction was done 74 days after the exploit.

Exploiter ([0xb84a](#)) → Deposited to Binance ([0x08E5](#), 40.5 ETH)



161. ReHold (Sep 26)



Underlying Chain: BSC

Amount: \$700,000

Attack Vector: Rugpull

Brief Summary

ReHold's CTO Michael Semin revealed on the X platform that on July 15, 2024, Renat Gafarov, the CEO and co-founder of ReHold, removed in excess of \$700,000 from the company's smart contract without Semin's consent. Following this incident, ReHold transitioned to a different domain. Meanwhile, ReHold website has announced "private keys have been compromised by Michael Semin, CTO of ReHold" and "Michael is attempting a hostile takeover of the company by threatening our CEO, Renat Gafarov, and his family, while extorting a large sum of money.", confusing the community.



Important Security Alert!

We regret to inform you that the private keys have been compromised by Michael Semin, CTO of ReHold. He currently has control over users' funds and the company's servers.

Michael is attempting a hostile takeover of the company by threatening our CEO, Renat Gafarov, and his family, while extorting a large sum of money. A criminal case has been filed, and the police are actively searching for his whereabouts.

If you have any information regarding Michael Semin's location, please contact us immediately at support@rehold.io.

Until the situation is officially resolved, we urge all users not to follow any links or conduct any transactions. Your security is our top priority, and we will keep you updated as events unfold.

ReHold Team.

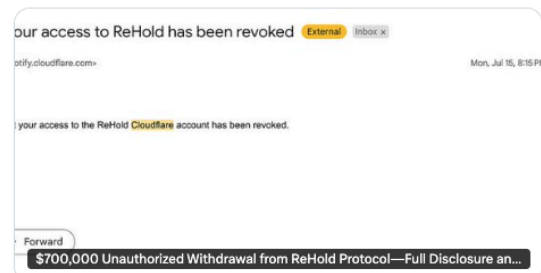
7:32 AM · Sep 26, 2024 · 55.3K Views

x1 ...



Michael Semin | ReHold @michael_rehold · Sep 26, 2024
Important Update from ReHold

On July 15, 2024, @renat_rehold, our co-founder, withdrew over \$700,000 from the company's smart contracts without my approval. We've moved to a new domain: app.reholdio.com because Renat revoked the team access to the original domain.



From medium.com

14

18

29

4.7K

1

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A



Compensation Process

On Nov 26, ReHold announced a 50% refund until the end of 2024. The team stated that the remaining 50% will be refunded through ReHold's future revenue. ([Ref.](#)) On Jan 2 2025, ReHold announced that all the refund requests have been resolved. ([Ref.](#))

Postmortem

[Link](#) - 2024.09.26

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#)

Fund Flow

Exploited assets were bridged to Ethereum, and laundered through Tornado Cash, 31 days after the exploit.

Exploiter ([0x2587](#))

→ Bridged via LayerZero Stargate (BSC → Ethereum) ([0x6930](#))

→ Laundered through Tornado Cash (77.4 ETH)



162. Onyx DAO (Sep 26)



Underlying Chain: Ethereum

Amount: \$4,000,000

Attack Vector: Contract Vulnerability - Precision Loss (Compound V2 Fork)

Brief Summary

Following two prior exploits, Onyx DAO, a project where the community took control of Onyx Protocol, encountered a significant security incident, losing more than \$3.8 million. The root cause was the known precision loss issue of CompoundV2 forks, combined with a flaw in the NFTLiquidation contract that did not accurately verify untrusted user inputs. This enabled the attacker to artificially raise the self-liquidation reward, compounding the overall financial damage.

Compensation Process

On Sep 27, OnyxDAO announced that all the affected users will be reimbursed at a 1:1 payment of the supplied assets. ([Ref.](#))

Pre-Incident Audit History

CertiK (2021.11.08, 2021.11.15, 2022.03.07, 2022.03.28, 2022.10.07, 2023.02.28) - [Reports](#)

Post-Incident Audit History

Peckshield (24.10.24) - [Report](#)

Postmortem

[Link](#) - 2024.09.27

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#), [Tx4](#), [Tx5](#)



Fund Flow

Exploiter ([0x6723](#), [0x0000](#), [0xfD47](#), [0x6809](#))

→ [0xb6d](#) (266 ETH) → Multiple transfers via Maya Protocol (still ongoing)

/ [0x0D03](#) (8.1 ETH, 9.7K DAI)

/ Bridged via Across Protocol (Ethereum → Arbitrum → Ethereum) ([0x38b1](#)) → Laundered through Railgun ([0x34Ed](#), 323 ETH), Chainflip ([0x34Ed](#), 50 ETH), Transferred multiple times via Maya Protocol & Laundered through Railgun ([0x34Ed](#), 266.78 ETH)

/ Bridged via Blast Official Bridge (Ethereum → Blast → Ethereum) ([0x0000](#)) → Bridged via Arbitrum Official Bridge (Ethereum → Arbitrum) ([0x0000](#)) → Laundered via Chainflip ([0xF15e](#), [0x8b37](#) - 746 ETH)



163. Truflation (Sep 26)



Underlying Chain: Ethereum

Amount: \$5,000,000

Attack Vector: Control Hijacking - Web2 Exploit

Brief Summary

On-chain investigator ZachXBT reported that the Truflation project suffered a hack exceeding \$5 million, breaching the treasury multisig and individual wallets. The team attributed this breach to malware used by the perpetrators, assuring that user assets remained untouched.

Compensation Process

On Oct 24 2024, Truflation stated that no user funds were affected due to the exploit. ([Ref.](#))

Pre-Incident Audit History

Sherlock (2024.01.26) - [Report](#), Stronghold Security (24.05.16) - [Report](#)

Post-Incident Audit History

Decurity (2024.10.18) - [Report](#)

Postmortem

[Link](#) - 2024.09.29

Exploit Transactions - [Tx](#)

Fund Flow

Over 500 ETH are deposited to eXch, while the rest are sent to Ethereum and Bitcoin EOAs. The assets were sent to eXch 25 days after the exploit.

Exploiter ([0x2867](#)) → ... → Bridged via THORChain (Ethereum → Litecoin) ([ltc1q96](#), 158.4 ETH) / [0x7A28](#), [0x3FEf](#) (4.28 BTC, 30.17 ETH, 21K USDT) / Deposited to eXch ([0xCb45](#), 500.33 ETH)



164. Bedrock (Sep 26)



Underlying Chain: Ethereum

Amount: \$1,700,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

Bedrock, a protocol focused on liquidity across different blockchains, revealed on social media that its team has identified a security flaw tied to uniBTC. This breach has resulted in estimated damages totaling nearly \$2 million. While uniBTC of Bedrock is designed to be converted at 1:1 ratio with BTC, the contract allowed other assets to be converted with uniBTC at the same ratio(!).

```
163      /**
164      * @dev mint uniBTC with native BTC tokens
165      */
166      function _mint(address _sender, uint256 _amount) internal {
167          (, uint256 uniBTCAmount) = _amounts(_amount);
168          require(uniBTCAmount > 0, "USR010");
169
170          require(address(this).balance <= caps[NATIVE_BTC], "USR003");
171
172          IMintableContract(uniBTC).mint(_sender, uniBTCAmount);
173
174          emit Minted(NATIVE_BTC, _amount);
175      }
176
177      /**
178      * @dev mint uniBTC with wrapped BTC tokens
179      */
180      function _mint(address _sender, address _token, uint256 _amount) internal {
181          (, uint256 uniBTCAmount) = _amounts(_token, _amount);
182          require(uniBTCAmount > 0, "USR010");
183
184          require(IERC20(_token).balanceOf(address(this)) + _amount <= caps[_token], "USR003");
185
186          IERC20(_token).safeTransferFrom(_sender, address(this), _amount);
187          IMintableContract(uniBTC).mint(_sender, uniBTCAmount);
188
189          emit Minted(_token, _amount);
190      }
```

The bug was due to the misimplementation of the mint function that handled the native tokens. While Bedrock was a Bitcoin L2 project, they utilized the same contract that uses ETH as native token. Therefore, the exploiter could mint uniBTC with ETH, which led to the 30x inflation.



Compensation Process

In the postmortem they published, they promised to guarantee uniBTC to be 1:1 redeemed to native BTC. (Ref.) They also proceeded on airdrop of their tokens based on a snapshot. ([Ref.](#))

Pre-Incident Audit History

Blocksec (2024.06.12) - [Report](#), Peckshield (2024.02.15) - [Report](#)

Post-Incident Audit History

Peckshield (2024.10.01) - [Report](#)

Postmortem

[Link](#) - 2024.09.28

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are laundered through Tornado Cash, 5 days after the exploit.

Exploiter ([0x1e1d](#)) → Laundered through Tornado Cash ([0xEE80](#), 650.1 ETH)



165. FIRE Token (Oct 1)

Underlying Chain: Ethereum

Amount: \$24,000

Attack Vector: Contract Vulnerability - Misimplemented Transfer Function

Brief Summary

Just moments after its debut on Ethereum, the Fire (\$FIRE) token fell victim to an exploit, with the attacker making off with 9 ETH, valued at around \$24,000. The breach arose from a flaw in the token's burn feature embedded within the transfer() function. Specifically, when Fire tokens were sent to a Uniswap pair, the pair's balance and reserves were automatically reduced as "sync" was triggered.

```
// Deduct tokens from the liquidity pair and transfer to the dead address
uint256 sellAmount = amount.sub(taxAmount);
if (sellAmount > 0) {
    uint256 liquidityPairBalance = balanceOf(uniswapV2Pair);
    if (liquidityPairBalance >= sellAmount) {
        _balances[uniswapV2Pair] = _balances[uniswapV2Pair].sub(sellAmount);
        _balances[DEAD_ADDRESS] = _balances[DEAD_ADDRESS].add(sellAmount);
        emit Transfer(uniswapV2Pair, DEAD_ADDRESS, sellAmount);

        // Call sync to update the pair
        IUniswapV2Pair(uniswapV2Pair).sync();
    }
}
```

By sending a significant amount of Fire tokens to the pair, the hacker minimized the token's reserves. Initially, the hacker swapped 20 ETH for Fire, leaving roughly 200 tokens in the pair. By transferring this amount back to the Uniswap pair, the reserve value for Fire decreased considerably. Utilizing the "swap" function with the reduced reserves allowed him to acquire a substantially larger amount of tokens. He exploited this loophole repeatedly, ultimately amassing about \$20,000.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A



Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are directly laundered through Tornado Cash, right after the exploit.

Exploiter ([0x81F4](#)) → Tornado Cash (9.1 ETH) / Leftover sent to eXch ([0x12D6](#), 0.08 ETH)



166. EGA Token (Oct 5)

Underlying Chain: BSC

Amount: \$554,000

Attack Vector: Contract Vulnerability - Access Control

Brief Summary

In the unverified victim contract, a function that swaps the contract's WBNB into EGA was implemented, which was mistakenly set as a public function. Anyone could trigger the function with just 1 wei, thus the exploiter could proceed on a sandwich attack by manipulating the balance of the contract. The exploiter converted a significant amount of WBNB to EGA and reverted EGA back to WBNB, and profited 506 WBNB effortlessly.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Right after the exploit, the exploited assets were bridged to Ethereum and Polygon, then laundered through Railgun.

Exploiter ([0xe4Fb](#)) → Bridged via LayerZero Stargate (BSC → Ethereum) ([0xe4Fb](#), 435.6 BNB), (BSC → Polygon) ([0xe4Fb](#), 45 BNB) → Laundered through Railgun



167. P719 (Oct 11)

Underlying Chain: BSC

Amount: \$315,000

Attack Vector: Contract Vulnerability - Misimplemented Transfer Function

Brief Summary

The main issue stems from the `transfer()` function within the P719's unverified contract. When tokens are sent to P719, the transfer is treated as a sale, employing a Uniswap-like swap process to determine the BNB amount for exchange. Yet, after this swap occurs, P719 destroys most of the tokens sold and sends fee tokens from its store, boosting the token's value. By continuously executing buying and selling transactions, the attacker exploited the resulting price difference to gain profit.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0xFeb1](#)) → Exploit transactions ([Ref.](#)) → Deposit to FixedFloat (475.4 BNB)



168. Radiant Capital(2) (Oct 16)



Underlying Chain: Arbitrum, BSC

Amount: \$58,000,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

Radiant faced a \$50 million setback on Arbitrum and BSC after a hacker altered the owner of the poolConfigurator. They upgraded the lending pool's contract setup, enabling them to empty all funds by using the "transferUnderlyingTo" function. For the ownership change, approval from three owners of the multisig was required.

Compensation Process

On Dec 30 2024, the team updated the remediation plan stating a long-term recovery.

Pre-Incident Audit History

Pre-Incident Audit History

Zokyo (2022.05.06) - [Report](#), SourceHat (2022.05.12) - [Report](#),

Peckshield (2022.07.28) - [Report](#), BlockSec (2023.03.22) - [Report](#),

Peckshield (2023.03.05) - [Report](#), OpenZeppelin (2023.10.18) - [Report](#),

BlockSec (2024.06.28) - [Report](#) , Pashov (2024.07.26) - [Report](#),

OpenZeppelin (2024.07.23) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.10.18

Exploit Transactions - [Tx1](#), [Tx2](#)



Fund Flow

Exploited assets are bridged to Ethereum through various bridging protocols, 7 days after the exploit. The bridged assets are now located at some EOAs, not moving for over 70 days.

Exploiter ([0x0629](#))

→ [0xb777](#) (12239 ETH)

→ Bridged via LayerZero Stargate, Hop Protocol, Synapse (Arbitrum → Ethereum)

→ [0x491C](#) (2722.35 ETH), [0x8035](#) (1829 ETH), [0xdEcF](#) (2496 ETH), [0x5B9a](#) (3091 ETH),
[0x4AFb](#) (2213 ETH)

Exploiter ([0x0629](#))

→ [0x62dC](#) (32779 BNB)

→ Bridged via LayerZero Stargate, DODO, ...

→ [0x759F](#) (3635.8 ETH)

Note

On Dec 30 2024, the team updated the remediation plan stating a long-term recovery. However, the project is not currently operating.



169. IBX (Oct 18)



Underlying Chain: Solana

Amount: \$24,000,000

Attack Vector: Rugpull

Brief Summary

Crypto analyst Anon Vee shared on X that numerous users have flagged the Orderly Network's IBX project for potentially executing a rugpull. The situation began when IBX launched a \$ARTIC pre-sale three days back, aiming to gather 2,000 SOL (approximately \$3.2 million), promising to refund those not selected. Instead, the pre-sale amassed over 160,000 SOL (around \$24 million), with participants believing they'd receive \$21.8 million back. Contrary to expectations, rather than issuing refunds, the IBXtrade team conducted a poll on a site they controlled, proposing an increase in the pre-sale cap. This poll passed, and IBX stated they had refunded 65,000 SOL (\$9.7 million) to participants. Following the chaos, \$ARTIC distanced itself from any links with IBX.

Pre-Incident Audit History

Halborn (2024.09.02) - [Report](#)

Exploit Transactions - [TxS](#)

Fund Flow

From the presale account([Ctf2L](#)), the exploited assets were sent to various Solana accounts. Some of them are bridged through Wormhole and deBridge, and deposited into CEXs such as Kucoin and Gate.io. A substantial amount of the assets are deposited to TradeOgre or bridged to CEXs through Bridgers. Here are the destination addresses I identified at the day of the exploit.

Solana: [Dest1](#), [Dest2](#), [Dest3](#), [Dest4](#), [Dest5](#), Arbitrum: [Dest](#)

Ethereum: [Dest1](#), [Dest2](#), [Dest3](#), [Dest4](#), [Dest5](#), [Dest6](#), [Dest7](#), [Dest8](#), [Dest9](#), [Dest10](#), [Dest11](#), [Dest12](#), [Dest13](#), [Dest14](#), [Dest15](#), [Dest16](#), [Dest17](#), [Dest18](#), [Dest19](#), [Dest20](#), [Dest21](#), [Dest22](#), [Dest23](#)



170. Tapioca DAO (Oct 19)



Underlying Chain: Ethereum

Amount: \$4,400,000

Attack Vector: Control Hijacking - Social Engineering

Brief Summary

Tapioca DAO suffered a significant security compromise due to a social engineering attack, with thieves making off with roughly \$4.4 million in digital assets, which sent the TAP token's value crashing by more than 95%.

According to the team's postmortem, one of the core contributors (a lead smart contract engineer of Pearl Labs) got his private key compromised due to the social engineering of North Korean hackers. The methodology was similar to that of Rain Exchange, as the exploit involved a malware injection through the job interview.

Thanks to the joint efforts of the project team and security firm SEAL911, they managed to secure 1,000 ETH (about \$2.7 million), preventing further depletion of resources. Although the hacker still retains some of the stolen funds, efforts are underway to reclaim the remaining assets.

Compensation Process

On Oct 23 2024, Tapioca Foundation announced the token migration plan. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.10.25

Exploit Transactions - [Txs](#)



Fund Flow

Exploiter ([0x7028](#))

→ Bridged via LayerZero Stargate (Arbitrum → BSC) ([0x69D9](#), [0x3c6c](#))

→ Bridged via LayerZero Stargate (BSC → Arbitrum) ([0x55e5](#)) / (BSC → Polygon) ([0x1a9e](#), [0x4131](#), [0x9ace](#), [0x88fb](#), [0xa029](#), [0x5800](#), [0x4bb5](#), [0x129c](#), [0x6b11](#), [0x7fba](#), [0x9463](#), [0x0082](#))

→ Bridged via deBridge (Arbitrum, Polygon → BSC) ([0x5933](#), [0x1fe0](#), [0x74b5](#), [0xCb74](#), [0xf675](#), [0xA4Ad](#) - 1.65M USDT), Across Protocol & deBridge (Polygon → Optimism → BSC) ([0xcd16](#), [0xc461](#), [0xE254](#), [0x0424](#) - 1.06M USDT), Rhino.fi bridge (Polygon → BSC) ([0xA4Ad](#), [0xB815](#) - 420K USDT), Axelar Squid (Polygon → BSC) ([0xf675](#), 70K USDT), Synapse (Polygon → BSC) ([0x1fe0](#), 350K USDT)

→ Distributed to multiple EOAs & Transfers ongoing ...



171. Cryptobottle (Oct 22)



Underlying Chain: Polygon

Amount: \$527,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

In October 2024, Cryptobottle on Polygon experienced two distinct attacks, resulting in total losses of about \$577K.

The first exploit was performed at Cryptobottle's unverified "Navitagor's Advantage" contract. The contract allowed users to set `fixedPriceEnabled`, to choose the user to mint the token at a fixed price or swap at a pool to acquire the token. However, as balance check in the swap function was disabled after callback, users could buy a massive amount of NAS tokens for arbitrary price. These tokens were then sold, leading to a loss of approximately \$490K for the project.

The second exploit was performed at the NFT sale of Cryptobottle. The root cause of the exploit was misconfigured decimals; They set the NFT price based on the decimal of EUROc, 12, while their sale contract treated the price the same as other ERC20 tokens (18). Therefore, each NFTs were sold at 0.0003 EUROc, which was intended to be sold for 300 EUROc for each. The exploit led to 971 NFTs of their collection bought at near 0 price. Notably, their contract was under audit by Wagmi Studio, but the team confessed that they decided to launch the contract before the audit was done. ([Ref.](#))

Compensation Process - N/A

Pre-Incident Audit History

Wagmi Studio (N/A) - Unavailable

Post-Incident Audit History - N/A



Postmortem

[Link](#) - 2024.10.22, but this postmortem does not mention the first exploit.

Exploit Transactions - [Txs](#)

Fund Flow

The exploited assets were sent to an EOA 0x4d80, and swapped to various ERC20 tokens. Currently the address holds the entire assets as Aave USDT & Uniswap V3 LP.

Exploiter ([0x5Ec5](#)) → [0x4d80](#)



172. Ramses Exchange (Oct 24)



Underlying Chain: Arbitrum

Amount: \$93,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

Ramses Exchange recently experienced an exploit due to a critical flaw in their reward distribution system. The vulnerability stemmed from the contract's failure to properly update its state after distributing rewards. Specifically, the `tokenTotalSupplyByPeriod` variable, essential for reward calculations, wasn't updated after rewards were sent to users. This oversight created a loophole, enabling attackers to repeatedly claim rewards by using different token IDs for the same period. Essentially, this allowed multiple claims for the same reward allocation.

Compensation Process - N/A

Pre-Incident Audit History

yAudit (2023.07.30) - [Report](#)

Post-Incident Audit History

code4rena (2024.10.30) - [Report](#)

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0x1d8b](#))

→ Bridged via LayerZero Stargate (Arbitrum → Ethereum) (37 ETH)

→ Bridged via OKX Web3 Proxy (Ethereum → BSC → Ethereum) ([0x9C9D](#), 20 ETH)

→ Bridged to CEXs via Bridgers



173. U.S. Government (Oct 24)



Underlying Chain: Ethereum

Amount: \$20,000,000

Attack Vector: Control Hijacking

Brief Summary

A significant security breach occurred involving a US government wallet, resulting in the theft of \$20 million in assets. According to SlowMist, nearly \$20 million was transferred to `0x3486ee700ccaf3e2f9c5ec9730a2e916a4740a9f`, including 5.4M USDC, 1.12M USDT, 13.7M aUSDC and 178 ETH.

Compensation Process

Arkham later reported that approximately 88% of the stolen funds were later returned.

Exploit Transactions - [Ref.](#)

Fund Flow

Exploited assets are initially deposited into various exchanges. The other related assets are moved to Ethereum addresses that are labeled as `Bitfinex Hacker Seized Funds` in Arkham.

Exploiter ([0x3486](#), [0xBf6F](#), [0x15D0](#))

→ Deposited to Binance, N.exchange, HitBTC ([0x3486](#), [0x2434](#), [0x15d0](#) - 458.87 ETH)

/ Bridged via Maya Protocol (Ethereum → Solana) ([EVek](#), 62K USDC) → Burned ([Tx](#))

/ [0x4bee](#) (1.8 ETH), [0xc7A2](#) (5939 ETH, 1066 WETH, 900K USDC, 296K USDT), [0x0CaB](#) (2411.75 ETH, 13M aUSDC, 7K USDC)



174. Unverified Contract (Oct 25)

Underlying Chain: Base

Amount: \$1,000,000

Attack Vector: Contract Vulnerability - Precision Loss (Compound V2 Fork)

Brief Summary

An unverified contract on the Base chain was exploited for \$1 million due to a fundamental design flaw in its price mechanism. This project, a fork of the CompoundV2 lending protocol, depended on CLPool's pricing to set token exchange rates. The critical issue arose from the pool's insufficient liquidity, leaving it vulnerable to price manipulation. The attacker took advantage of this weakness by inflating the pool's price and using the artificially increased collateral to create malicious positions, ultimately resulting in the loss of funds.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged to Ethereum via various bridge protocols and laundered through Tornado Cash, right after the exploit.

Exploiter (0x81d5)

→ Bridged via Rhino.fi Bridge, Across Protocol, Reservoir, MaestroBots, Owlto, Orbiter Finance (Base → Ethereum) ([0x81d5](#), [0xD477](#), [0xf70d](#), [0x0166](#) - 380 ETH)

→ Laundered via Tornado Cash



175. Essence Finance (Oct 26)



Underlying Chain: Scroll

Amount: \$20,000,000

Attack Vector: Rugpull

Brief Summary

Essence Finance, a stablecoin project on Scroll, rugpulled and its stablecoin \$CHI has fallen by more than 92%. More than \$20 million of collateral is suspected to have been removed.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

176. M2 (Oct 31)



Underlying Chain: Bitcoin, Ethereum, Solana

Amount: \$13,700,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

According to on-chain detective ZachXBT on his personal channel, cryptocurrency exchange M2 was hacked, resulting in the theft of approximately \$13 million from several on-chain hot wallets.

Compensation Process

On Nov 1 2024, M2 announced that the exploit had been fully resolved. ([Ref.](#))

Postmortem - N/A

Fund Flow

Exploited assets are mostly laundered through Tornado Cash, 11 days after the exploit.

Ethereum exploiter: [0x968b](#), Bitcoin exploiter: [bc1qu4](#), Solana Exploiter: [EKko](#)

→ Deposited to Binance (17 ETH) / Laundered through Tornado Cash ([0x6066](#), 4010 ETH) / Leftovers deposited to eXch ([0x826F](#))



177. Sunray Finance (Oct 31)



Underlying Chain: Arbitrum

Amount: \$2,850,000

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

Private key leakage allowed the exploiter to upgrade the token management contract and mint a massive amount of \$SUN and \$ARC. The team tweeted that future plans would be announced soon, but no updates have been made so far. ([Ref.](#))

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx1](#), [Tx2](#)

Fund Flow

Exploited assets are bridged to BSC and moved to an EOA, 2 days after the exploit.

Exploiter ([0xb1BB](#)) → Bridged via LayerZero Stargate (Arbitrum → BSC) ([0xc405](#)) → [0xd7b0](#)
(2.85M USDT)

Note

The last tweet of the team was made on Nov 5 2024, stating that they have contacted the Binance security team and are waiting for the result. ([Ref.](#))



178. Metawin (Nov 4)



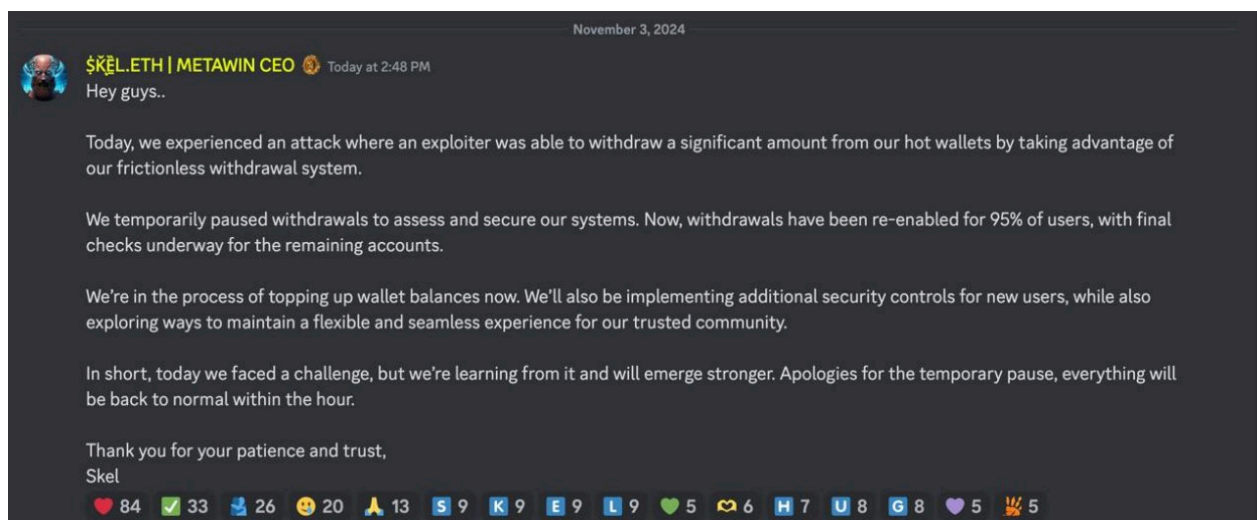
Underlying Chain: Ethereum

Amount: \$4,000,000

Attack Vector: Control Hijacking - Web2 Exploit

Brief Summary

According to on-chain detective ZachXBT, the cryptocurrency gambling platform Metawin was reportedly attacked, resulting in the theft of over \$4 million on the Ethereum and Solana blockchains. The exploiter targeted the platform's instant withdrawal mechanism that had vulnerable security checks that could be bypassed. This allowed the exploiter to access Ethereum and Solana hot wallets.



Compensation Process

Shortly after the exploit, Metawin announced that withdrawals are restored for 95% of users.

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A



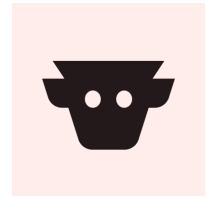
Exploit Transactions - [Ref.](#) (Reported by ZachXBT)

Fund Flow

Exploited assets are known to be deposited to Kucoin and HitBTC.



179. CowSwap (Nov 7)



Underlying Chain: Ethereum

Amount: \$59,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

GPv2Settlement contract got its approved assets drained, as its UniswapV3 callback function allowed the exploiter to bypass the verification of msg.sender. Several MEV bots took profits exploiting this vulnerability. (Ref: [TenArmor](#))

```
function uniswapV3SwapCallback(int256 amount0Delta, int256 amount1Delta, bytes data) public nonPayable {
    require(msg.data.length - 4 >= 96);
    require(data <= uint64.max);
    require(4 + data + 31 < msg.data.length);
    require(msg.data[4 + data] <= uint64.max);
    require(v0.data <= msg.data.length);
    require(4 + data + 32 + msg.data[4 + data] - (4 + data + 32) >= 128);
    require(data.word2 == address(data.word2));
    require(data.word3 == address(data.word3));
    require(data.word4 == address(data.word4));
    require(msg.sender == address(data.word4), Error('Invalid pool'));
    v1 = v2 = 0;
    if (amount0Delta <= v2) {
        if (amount1Delta > 0) {
            0x4c3(amount1Delta, msg.sender, data.word2, data.word3);
        }
    } else {
        0x4c3(amount0Delta, msg.sender, data.word2, data.word3);
    }
    require(v1 >= data.word1, Error('Slippage tolerance exceeded'));
}
```

Pre-Incident Audit History

G0 Group (2021.05.21) - [Out of Scope](#), Hacken (2021.12.10) - [Report](#)

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)



180. CoinPoker (Nov 8)



Underlying Chain: Ethereum, BSC, Polygon

Amount: \$2,000,000

Attack Vector: Control Hijacking

Brief Summary

On Nov 8, \$2M worth of assets were allegedly transferred from CoinPoker's hot wallets. Regarding the exploit, there has been no announcement from the team. However, on Nov 16 2024, the team contacted the exploiter through on-chain message, for negotiation. ([Ref.](#))

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

The exploited assets are laundered through Tornado Cash, 9 days after the exploit.

Hot wallet ([0x3c17](#)) → [0x4C1e](#)

→ Bridged via LayerZero Stargate (BSC, Polygon → Ethereum) ([0x4C1e](#))

→ Laundered through Tornado Cash (437.3 ETH)

On Dec 17 2024, exploited 3.7M CoinPoker Chips tokens were returned to the project. ([Tx](#))



181. BGM Token (Nov 10)

Underlying Chain: BSC

Amount: \$458,000

Attack Vector: Contract Vulnerability - Price Dependency

Brief Summary

When executing the LP withdrawal function, rewards are calculated based on the liquidity provision period and sent to users. However, the reward calculation relied solely on one Pancakeswap pool's exchange rate. The attacker manipulated the pool's exchange rate through a flash loan and immediately called withdraw, claiming a massive amount of reward tokens.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are distributed into multiple addresses and deposited to FixedFloat, and the rest of the assets remain in an EOA. The last activity was done 49 days after the exploit.

Exploiter ([0x7824](#))

→ ... → Deposited to FixedFloat ([0xb0A7](#), [0x9525](#) - 139 BNB) / [0x7663](#) (375K USDT)



181. DeltaPrime(3) (Nov 11)



Underlying Chain: Arbitrum

Amount: \$4,750,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

This was the third exploit for DeltaPrime in 2024.

The vulnerability stemmed from inadequate input validation in the LP reward claiming system. The contract failed to properly verify the legitimacy of trading pairs used for reward calculations. In detail, their contract set `getHooks()` function in public, which accepts a pair address as an input and returns details about hook and rewarders. Since there were no checks whether the pair was created by `TraderJoe`, it accepted an arbitrary pair to mimic a legit pair and rewarder.

```
function getHooks(address pair) public view returns (Hooks.Parameters memory) {
    if (msg.sender == address(this)) {
        bytes32 hooksParameters = ILBPair(pair).getLBHooksParameters();

        return Hooks.decode(hooksParameters);
    } else {
        try this.getHooks(pair) returns (Hooks.Parameters memory hooksParameters) {
            return hooksParameters;
        } catch {
            return
                Hooks.Parameters(address(0), false, false, false, false, false, false, false, false, false, false);
        }
    }
}
```

The attacker exploited this by executing a sophisticated series of transactions: they first took out an ETH flash loan to use as collateral, then borrowed WBTC. With these funds, they created an artificial trading pair between WBTC and their own token. This malicious pair was then used to claim rewards from the system. After claiming the rewards, they withdrew the liquidity and repaid the flash loan, effectively stealing \$4.5M in rewards without having any legitimate stake in the protocol.



Compensation Process

At the time of the exploit, DeltaPrime was already under the compensation process. The team decided to drop more reimbursement tokens and proceed on the planned compensation (33% of future revenues goes to reimbursement token holders).

Pre-Incident Audit History

Piotr Szlachciak (2021.12.04) - [Report](#), Peckshield (2022.02.12) - [Report](#),

Chainsulting (2022.04.11) - [Report](#), Peckshield (2022.12.01) - [Report](#),

Peckshield (2023.05.27) - [Report](#), Peckshield (2023.09.08) - [Report](#),

AstraSec (2024.06.19) - [Report](#), ABDK (2024.06.24) - [Report](#),

AstraSec (2024.08.07) - [Report](#), BlockSec (2024.10.01) - [Report](#)

Post-Incident Audit History

DeltaPrime got its every method re-audited by BlockSec. (2024.11.25) - [Ref.](#)

Postmortem

[Link](#) - 2024.12.08

Exploit Transactions - [Tx](#)

Fund Flow

Notably, the exploiter directly moved his asset to an EOA and provided liquidity to LFJ. ([Ref.](#))

Exploiter ([0xDA81](#))

→ [0x2D81](#)

→ [0xd3d5](#) (69401 AVAX)

/ Deposited to LFJ ([0xd3d5](#) - 3220 WAVAX, 73.1 WETH, 623K USDC, 423K USDT)

/ Stargate Staking ([0xd3d5](#) - 600K USDC)



182. vETH Token (Nov 14)

Underlying Chain: Ethereum

Amount: \$450,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The vulnerability centered around the `takeLoan` function in the token contract. While the `takeLoan` function has a modifier that restricts the caller to a valid factory (`onlyValidFactory`), the factory contract had a flawed function that adds liquidity to the Uniswap vETH-BIF pair using `takeLoan` function and user's BIF tokens. Thus, the exploiter could flashloan to purchase BIF tokens, borrow vETH tokens through the `takeLoan` function, then manipulate the vETH-BIF price on Uniswap. This manipulation allowed them to profit from vETH arbitrage using the increased k of the pool.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter laundered the assets through Tornado Cash, 2 days after the exploit.

Exploiter ([0x713d](#)) → Laundered through Tornado Cash (150 ETH)



183. Thala (Nov 15)



Underlying Chain: Aptos

Amount: \$25,500,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The vulnerability emerged in an upgraded contract. The critical flaw was in the unstaking mechanism - the contract failed to verify whether the withdrawal amount was less than the user's currently staked tokens. This basic validation oversight allowed attackers to withdraw more tokens than they had actually staked. ([Ref.](#))

Compensation Process

On Nov 15, Thala team stated that user assets are not affected by the exploit.

Pre-Incident Audit History

Zellic (2023.02.28) - [Report](#), OtterSec (2023.03.14) - [Report](#),

OtterSec (2023.05.31) - [Report](#), [Report](#), [Report](#)

Post-Incident Audit History

OtterSec (2024.11.20) - [Report](#)

Postmortem

[Link](#) - 2024.11.16

Exploit Transactions - [TxS](#)



Fund Flow

Shortly after the exploit, MOD and THL tokens were frozen by the team. Substantial amounts of USDC are bridged to Ethereum or sent to the other Aptos accounts, but it seems that most of them are returned to the team, at the day of the exploit.

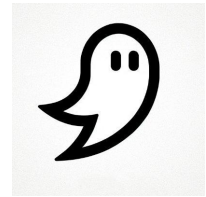
Exploiter ([0xf723](#))

→ Bridged via LayerZero Aptos Bridge (Aptos → Ethereum) ([0x5606](#), 8.08M USDC)

→ Safe Wallet (Returned?) ([0x039c](#), 3143.2 ETH), Returned ([Ref.](#), 8M USDC), Frozen ([Tx1](#), [Tx2](#), \$9m MOD & \$2.5m THL)



184. Polter Finance (Nov 18)



Underlying Chain: Fantom

Amount: \$7,000,000

Attack Vector: Contract Vulnerability - Price Dependency

Brief Summary

The exploit involved a straightforward price manipulation vulnerability. The contract's critical flaw was its dependence on BOO token pricing from the Spooky LP contract. The attacker executed their exploit by taking out a flash loan to borrow a large amount of BOO tokens, depleting the LP's BOO reserves. They then deposited 1 BOO and immediately removed liquidity, effectively draining the liquidity pool's assets.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged to Ethereum and laundered through Tornado Cash, right after the exploit.

Exploiter ([0x511f](#))

→ ... → Bridged via LayerZero Stargate, Axelar Squid (Fantom → Ethereum) ([0xce16](#), [0x8928](#), [0x959c](#), [0x3856](#), [0x47b9](#), [0xB5Cd](#), [0xC805](#), [0x51c2](#), [0x7BDe](#), [0x988D](#), [0xE12C](#), [0x9a7B](#), [0x2F44](#), [0xb17E](#), [0x141C](#), [0x05C8](#), [0xFDD3](#), [0x4E04](#), [0x8f8f](#), [0xbd53](#))

→ Laundered through Tornado Cash ([0x9c20](#), [0xb827](#), [0xDA3D](#), [0x302a](#), [0x2F44](#), [0xb17E](#), [0x141C](#), [0x05C8](#), [0xFDD3](#), [0x4E04](#), [0x8f8f](#), [0xbd53](#) - 2305.7 ETH)



Note

On Nov 29, a proposal to decide the percentage of the bounty that the team would offer to the exploiter passed, and as a result, the team offered a 10% bug bounty to the exploiter. However, it seems that the exploiter has not taken any action. The last tweet of the team was posted on Dec 7 2024, for a bounty submission form of the exploit. Polter Finance has been indefinitely suspended since the exploit.



185. BTB Token (Nov 18)

Underlying Chain: BSC

Amount: \$5,000

Attack Vector: Contract Vulnerability - Price Dependency

Brief Summary

Since the pool contract's exchange rate of BTB and USDT was dependent on a single Pancakeswap pair, it was exposed to a classic price manipulation exploit.

```
function getPrice() public view returns (uint256) {
    IUniswapV2Pair pair = IUniswapV2Pair(btbPackAddress);
    (uint112 reserve0, uint112 reserve1, ) = pair.getReserves();

    require(reserve1 > 0, "Reserve1 must be greater than zero");

    return getAmountOut(1 ether, reserve0, reserve1); // 每1 BTB 兑换的 USDT 价格
}
```

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are directly laundered through Tornado Cash, right after the exploit.

Exploiter ([0x8464](#)) → Laundered through Tornado Cash (11 BNB)



186. DCF Token (Nov 25)

Underlying Chain: BSC

Amount: \$442,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

DCF token had two vulnerabilities in its contract. First, the transfer function allows anyone to burn tokens when the tokens are transferred to its LP pool, as the 5% of the amount is automatically swapped as USDT and added to the DCF-USDT pool. Second, as the addLiquidity process in the transfer function did not have a slippage protection. Exploiting the two vulnerabilities, the attacker could perform a classic price manipulation attack through flashloan.

```
// buying restriction
if (from == pairAddress) {
    require(false, "buy error");
}

// swap token for usdt
if (to == pairAddress && !swapping) {
    swapping = true;
    uint256 fee = (amount * 5) / 100; // 5%
    uint256 deadAmount = (amount - fee) / deadCfg;
    amount -= fee;
    super._transfer(from, address(this), fee);

    uint256 initialUsdtBalance = IERC20(USDT).balanceOf(helperAddress);
    swapTokensForUSDT(fee, helperAddress);
    uint256 newUsdtBalance = IERC20(USDT).balanceOf(helperAddress) -
        initialUsdtBalance;
    liquidityHelper.addLiquidity(newUsdtBalance);
    swapping = false;
    if (balanceOf(pairAddress) > deadAmount) {
        burnPair(deadAmount);
    }
}

// proceed transfer
super._transfer(from, to, amount);
}

function swapTokensForUSDT(uint256 _tokenAmount, address _to) private {
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = USDT;
    _approve(address(this), router, _tokenAmount);
    uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(
        _tokenAmount,
        0,
        path,
        _to,
        block.timestamp
    );
}
```



Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are merged to the GAGAW exploiter-labeled address, and then deposited to FixedFloat 2 days after the exploit.

Exploiter ([0x00c5](#))

→ GAGAW exploiter ([0x86c5](#))

→ [0x9bCC](#) (252 BNB)

→ Distributed to multiple EOAs & Deposited to FixedFloat



187. Pump.Science (Nov 26)



Underlying Chain: Tron




Amount: 0

Attack Vector: Control Hijacking - Private Key Leakage

Brief Summary

On Nov 26 2024, DeSci project Pump Science announced that their wallet (T5j2) was exploited as their private key was included in their source code & uploaded in GitHub. Despite being the deprecated address, it was the first wallet that purchased \$RIF and \$URO, so it might have caused severe confusion to the community.



Pump Science  
@pumpdotscience 



The T5j2UBTvLYPCwDP5MVkSALN7fwuLFDL9jUXJNjib8sc wallet was exploited today from an oversight in our github repository. The exploiter was able to find the keypair in the source code of our website

This keypair has been in our github for testing purposes from the start and was considered as non-important by our developer team, [@builderzdotdev](#)

There was an oversight realizing this wallet address was marked on [pump.fun](#) as the off-chain token-creator of URO and RIF

This oversight was made because the on-chain history shows that BLDRZQiqT4ESPz12L9mt4XTBjeEfjoBopGPDMA36KtuZ was the first wallet to buy both URO and RIF and is therefore considered the token-deployer on all other trading apps

1:10 AM · Nov 26, 2024 · 102.1K Views



188. XT Exchange (Nov 28)



Underlying Chain: Ethereum

Amount: \$1,700,000

Attack Vector: Control Hijacking

Brief Summary

XT Exchange, a Dubai-based CEX, suspended withdrawals after the hot wallet exploit. XT Exchange stated that user assets are safe as they constantly maintain 1.5x greater reserves than user assets. ([Ref.](#)) On the day of the exploit, XT Exchange stated that the issue was identified and fixed, and its withdrawal service would resume the next day.

Fund Flow

Exploited assets are bridged to Arbitrum and Litecoin. The last transaction happened 28 days after the exploit.

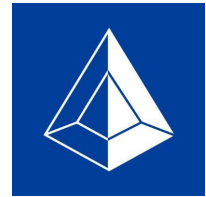
[0x4CB6](#) → [0xB43f](#)

→ Bridged via Across Protocol (Ethereum → Arbitrum) ([0x6eCE](#), 56.7 ETH) → Returned? ([0xA8C1](#), 37 WETH), Deposited to Bitget Wallet ([0x1123](#), 180K USDT)

→ Bridged via THORChain (Ethereum → Litecoin) ([ltc1qm8](#), [ltc1qyp](#), [ltc1qx3](#), [ltc1qqf](#), [ltc1qfu](#), [ltc1qn3](#), [ltc1qph](#), [ltc1q95](#), [ltc1qlq](#), [ltc1q0c](#), [ltc1qs5](#), [ltc1q8q](#))(521.57 ETH)



189. Spectral (Dec 1)



Underlying Chain: Base

Amount: \$250,000

Attack Vector: Contract Vulnerability - Misconfiguration

Brief Summary

On Dec 1 2024, Spectral's Syntax V2 bonding contract got an exploit, due to the redundant approval settings of AgentToken contract for AgentBalances contract. When a user swaps SPEC token for AgentToken, AgentBalances contract charges the user a transfer fee. When the fee calculation and charge happens, AgentToken was implemented to set the max approval.

```
super.transferFrom(sender, address(this), taxAmount/2);  
// Send half the tax to the agent  
approve(address(agentBalances), type(uint256).max);  
agentBalances.deposit(address(this), address(this), address(this), taxAmount/2);
```

Exploiting the max approval, the exploiter called deposit function to transfer all approved AgentToken balances of AutonomousAgentDeployer to AgentBalances, manipulating the value of AgentToken. Since AgentToken's price was inflated, the exploiter could drain \$250K worth of SPEC tokens from the bonding curve.

```
function deposit(address from, address token, address agent, uint256 amount)  
public {  
    if(agentWallets[agent] != address(0)){  
        //Pipe it straight through if the agent's wallet is set  
        IERC20Upgradeable(token).safeTransferFrom(from, agentWallets[agent],  
            amount);  
    }  
    else  
    {  
        //Otherwise store it here for later  
        IERC20Upgradeable(token).safeTransferFrom(from, address(this), amount);  
        agentBalance[agent][token] += amount;  
    }  
  
    emit Deposit(agent, token, amount);  
}
```



Compensation Process

Spectral announced the replenishment of SPEC tokens to restore the drained tokens in the bonding curve. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.12.03

Exploit Transactions - [Tx](#)

Fund Flow

The exploited assets were bridged to Ethereum and laundered through Railgun, a day after the exploit.

Exploiter ([0x0000](#))

→ Bridged via Orbiter Finance (Base → Ethereum)

→ Laundered through Railgun ([0x3332](#), 41.74 ETH)



190. Clipper DEX (Dec 1)



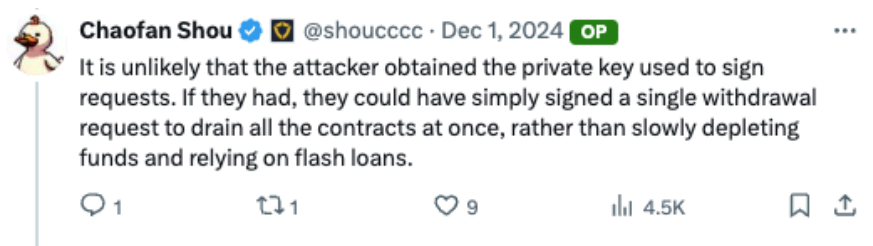
Underlying Chain: Optimism, Base

Amount: \$457,000

Attack Vector: Control Hijacking - Web2 Exploit

Brief Summary

After losing 6% of its TVL on Optimism and Base, Clipper paused its operation on all chains. While the team claims no private key leakages, @shouccccc suggests an API vulnerability due to the victim contract's reliance on an API. ([Ref.](#))



On Dec 5 2024, Clipper published a postmortem with details of the exploit, and here they admitted that the exploit was due to the attack on their public API. To summarize, the exploiters performed a price manipulation attack via their public API.

First, the exploiters requested a signature for single asset deposits through the API. Then they manipulated the pool value through swaps, as the pool value could easily be inflated due to the low liquidity of the Optimism and Base pools, and sent extra ETH. Then they requested a signature for single asset withdrawals with the inflated value. As the previously requested deposit signature was still valid, the withdrawal amount was much greater than the assets actually sent.

Compensation Process

Clipper is still working to recover. ([Ref.](#))



Pre-Incident Audit History

Quantstamp (2021.07.08) - [Report](#), Solidified (2021.07.10) - [Report](#)

Solidified (2021.11.18) - [Report](#), Solidified (2023.08.10) - [Report](#)

Post-Incident Audit History - N/A

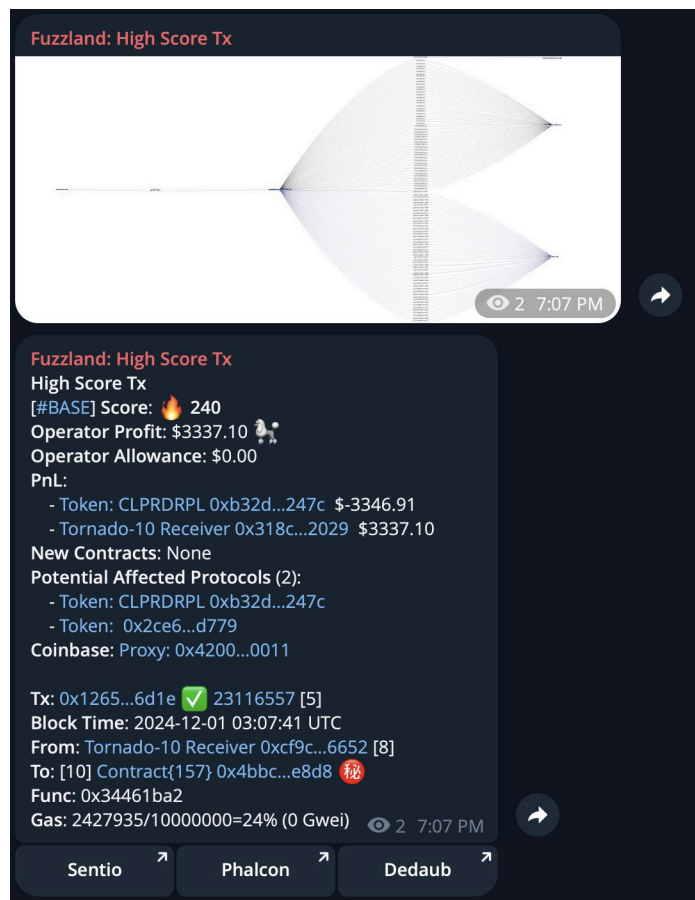
Postmortem

[Link](#) - 2024.12.05

Exploit Transactions - [Ref.](#)

Fund Flow

According to the information provided by Fuzzland, the exploited assets are laundered through Tornado Cash, 16 days after the exploit. ([Ref.](#)) ([Exploiter](#) → Tornado Cash ([0x1eb8](#), 114.39 ETH)



191. GAGAW Token (Dec 2)

Underlying Chain: BSC

Amount: \$70,000

Attack Vector: Contract Vulnerability - Misimplemented Transfer Function

Brief Summary

The token contract had a very basic vulnerability. When the token was transferred, the recipient's balance was updated to that of the dead address. Therefore, anyone who burns the token could acquire more GAGAW tokens.

```
    private {  
        _balances[sender] = _balances[sender].sub(tAmount);  
        _balances[recipient] = _balances[_destroyAddress].add(tAmount);  
        _balances[recipient] = _balances[recipient].add(0);  
        emit Transfer(sender, _destroyAddress, tAmount);  
    }
```

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are merged with the DCF exploiter, and then deposited to FixedFloat 2 days after the exploit.

Exploiter ([0x00c5](#))

→ GAGAW exploiter ([0x86c5](#)) → [0x9bCC](#) (252 BNB)

→ Distributed to multiple EOAs & Deposited to FixedFloat



192. Vestra DAO (Dec 4)



Underlying Chain: Ethereum

Amount: \$500,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

Misimplemented unstaking logic of Vestra DAO's staking contract led to the exploit. It was a simple vulnerability, as the unStake function did not update the user's staking information after unstake. Thus, anyone who staked could stake multiple times. The exploiter stole 73M VSTR from the exploit, swapping the assets to \$500K ETH from Uniswap.

Detecting the exploit, the team blacklisted the locked staking contract, removing 755M VSTR from the circulating supply. ([Ref.](#))

Compensation Process

On Dec 10 2024, Vestra DAO announced the full compensation to the affected users. ([Ref.](#))

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.12.05

Exploit Transactions - [Tx](#)

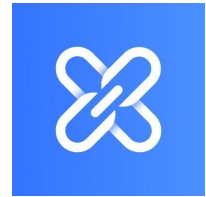
Fund Flow

Exploited assets are directly laundered through Tornado Cash, right after the exploit.

Exploiter ([0x9543](#)) → Laundered through Tornado Cash (128 ETH)



193. Clober (Dec 10)



Underlying Chain: Base

Amount: \$501,000

Attack Vector: Contract Vulnerability - Reentrancy

Brief Summary

Clober, a DEX on Base, suffered an exploit due to the reentrancy of Rebalancer contract's burn function. The exploiter created a pool of its own token and manipulated the token price by repeatedly burning the token.

```
function _burn(bytes32 key, address user, uint256 burnAmount)
    public
    selfOnly
    returns (uint256 withdrawalA, uint256 withdrawalB)
{
    Pool storage pool = _pools[key];
    uint256 supply = totalSupply(uint256(key));

    (uint256 canceledAmountA, uint256 canceledAmountB, uint256 claimedAmountA, uint256 claimedAmountB) =
        _clearPool(key, pool, burnAmount, supply);

    uint256 reserveA = pool.reserveA;
    uint256 reserveB = pool.reserveB;

    withdrawalA = (reserveA + claimedAmountA) * burnAmount / supply + canceledAmountA;
    withdrawalB = (reserveB + claimedAmountB) * burnAmount / supply + canceledAmountB;

    _burn(user, uint256(key), burnAmount);
    pool.strategy.burnHook(msg.sender, key, burnAmount, supply);
    emit Burn(user, key, withdrawalA, withdrawalB, burnAmount);

    IBookManager.BookKey memory bookKeyA = bookManager.getBookKey(pool.bookIdA);

    pool.reserveA = _settleCurrency(bookKeyA.quote, reserveA) - withdrawalA;
    pool.reserveB = _settleCurrency(bookKeyA.base, reserveB) - withdrawalB;

    if (withdrawalA > 0) {
        bookKeyA.quote.transfer(user, withdrawalA);
    }
    if (withdrawalB > 0) {
        bookKeyA.base.transfer(user, withdrawalB);
    }
}
```

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A



Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged to Ethereum and laundered through Tornado Cash, 19 days after the exploit.

Exploiter ([0x012F](#))

→ Bridged via Across Protocol (Base → Ethereum) ([0x2d27](#), [0x711C](#), [0x7760](#))

→ Laundered through Tornado Cash ([0x2d27](#), [0x7760](#) - 154 ETH)



194. Haven Protocol (Dec 11)



Underlying Chain: Monero

Amount: Unknown

Attack Vector: Misc.

Brief Summary

Haven Protocol is a privacy-focused stablecoin project built on Monero. On Dec 11 2024, Haven Protocol stated that they were hit by an exploit, and this led to the excessive supply of 500M XHV. According to the team, they had a vulnerability in the “range proof validation”, which was introduced after their update, and the vulnerability allowed the exploiter to mint XHV allegedly without getting detected.

Compensation Process - N

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.12.11

Note

On Dec 13 2024, Haven Protocol announced the project closure. ([Ref.](#))



195. DCF Token (Dec 15)

Underlying Chain: BSC

Amount: \$9,000

Attack Vector: Contract Vulnerability - Price Dependency

Brief Summary

The root cause was simple, as the exchange rate of the token pool solely relied on a pancakeswap v2 pair. The exploiter could profit by performing huge swaps on pancakeswap, manipulating the pool state. Notably, the exploiter is the same entity as the vETH token exploiter.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are directly laundered through Tornado Cash, a day after the exploit.

Exploiter (0x7a5b) → Laundered through Tornado Cash ([0x988c](#), 20.6 BNB)



196. bnbs Token (Dec 15)

Underlying Chain: BSC

Amount: \$20,000

Attack Vector: Contract Vulnerability - Reentrancy

Brief Summary

The root cause was simple, as the token had a basic reentrancy vulnerability in its `removeLiquidity` function.

```
function removeLiquidity(
    uint liquidity,
    uint amountNativeMin,
    uint amountTokenMin,
    address to,
    uint deadline
)
{
    public
    payable
    ensure(deadline)
    returns (uint amountNative, uint amountToken)
{
    require(
        LPBalanceOf(msg.sender) >= liquidity,
        "ERC20S: INSUFFICIENT_LIQUIDITY"
    );

    (uint112 _reserveNative, uint112 _reserveToken, ) = getReserves();
    uint balanceNative = address(this).balance;
    uint balanceToken = balanceOf(address(this));

    _mintFee(_reserveNative, _reserveToken);
    uint lpSupply = LPTotalSupply();

    amountNative = (liquidity * balanceNative) / lpSupply; // using balances ensures pro-rata distribution
    amountToken = (liquidity * balanceToken) / lpSupply; // using balances ensures pro-rata distribution
    require(
        amountNative > 0 && amountToken > 0,
        "ERC20S: INSUFFICIENT_LIQUIDITY_BURNED"
    );
    require(amountNative >= amountNativeMin);
    require(amountToken >= amountTokenMin);
    _LPBurn(msg.sender, liquidity);

    to.safeTransferETH(amountNative);
    _update(address(this), to, amountToken);

    balanceNative = address(this).balance;
    balanceToken = balanceOf(address(this));

    kLast = uint256(reserveNative) * uint256(reserveToken);
    _syncReserve(balanceNative, balanceToken);
    emit Burn(amountNative, amountToken, to, _msgSender(), block.timestamp);
}
```

Compensation Process - N/A

Pre-Incident Audit History - N/A



Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets haven't moved since the exploit.

Exploiter ([20Fa](#))



197. LABUBU Token (Dec 17)

Underlying Chain: BSC

Amount: \$120,000

Attack Vector: Contract Vulnerability - Misimplemented Transfer Function

Brief Summary

The root cause was simple, as anyone could double their balance when the tokens were self-transferred due to the misimplemented transfer function.

```
function _transfer(address sender, address recipient, uint256 amount) internal {
    require(sender != address(0), "Xfer from zero addr");
    require(recipient != address(0), "Xfer to zero addr");

    uint256 senderBalance = _balances[sender];
    uint256 recipientBalance = _balances[recipient];

    uint256 newSenderBalance = SafeMath.sub(senderBalance, amount);
    if (newSenderBalance != senderBalance) {
        _balances[sender] = newSenderBalance;
    }

    uint256 newRecipientBalance = recipientBalance.add(amount);
    if (newRecipientBalance != recipientBalance) {
        _balances[recipient] = newRecipientBalance;
    }

    if (_balances[sender] == 0) {
        _balances[sender] = 16;
    }

    emit Transfer(sender, recipient, amount);
}
```

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are directly laundered through Tornado Cash, a day after the exploit.

Exploiter ([0x2744](#)) → Laundered through Tornado Cash (17.4 BNB)



198. GemPad (Dec 17)



Underlying Chain: BSC

Amount: \$1,900,000

Attack Vector: Contract Vulnerability - Reentrancy

Brief Summary

The root cause was a reentrancy bug in collectFee function of the project's lock contract. The exploiter minted a fake token and locked its LP in the lock contract. By re-entering collectFee via callback, the project's vault could be drained.

```
function collectFees(
    uint256 lockId
) external isLockOwner(lockId) validLockLPv3(lockId) returns (uint256 amount0, uint256 amount1) {
    Lock storage userLock = _locks[lockId];
    // set amount0Max and amount1Max to uint256.max to collect all fees
    // alternatively can set recipient to msg.sender and avoid another transaction in `sendToOwner`
    INonfungiblePositionManager.CollectParams
    memory params = INonfungiblePositionManager.CollectParams({
        tokenId: userLock.nftId,
        recipient: address(this),
        amount0Max: type(uint128).max,
        amount1Max: type(uint128).max
    });
    // send collected feed back to owner
    (
        ,
        address token0,
        address token1,
        ,
        ,
        ,
        ,
        ,
        ,
    ) = INonfungiblePositionManager(userLock.nftManager).positions(
        userLock.nftId
    );
    uint256 originalAmount0 = IERC20(token0).balanceOf(address(this));
    uint256 originalAmount1 = IERC20(token1).balanceOf(address(this));
    INonfungiblePositionManager(userLock.nftManager).collect(params);
    amount0 = IERC20(token0).balanceOf(address(this)) - originalAmount0;
    amount1 = IERC20(token1).balanceOf(address(this)) - originalAmount1;
    IERC20(token0).safeTransfer(userLock.owner, amount0);
    IERC20(token1).safeTransfer(userLock.owner, amount1);
}
```

Compensation Process - N/A



Pre-Incident Audit History

ContractWolf (2023.05.31) - [Report](#), SolidProof (2024.01.30) - [Report](#)

Cyberscope (2024.03.15) - [Report](#)

Post-Incident Audit History - N/A

Postmortem

[Link](#) - 2024.12.21

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged to Ethereum and laundered through Tornado Cash, 2 days after the exploit.

Exploiter (Various addresses)

→ Bridged via LayerZero Stargate (BSC → Ethereum)

→ Laundered through Tornado Cash ([0xFDd9](#), 410.2 ETH)



199. Moonwell Third-Party Vault (Dec 23)



Underlying Chain: Optimism

Amount: \$320,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

Moonhacker, a third-party vault on Moonwell, was exploited due to the improper input validation. The executeOperation function, which helps the contract supply and borrow, failed to check the input for mToken address and allowed exploiter's own mToken contract to be used.

```
function executeOperation(
    address token,
    uint256 amountBorrowed,
    uint256 premium,
    address initiator,
    bytes calldata params
) external returns (bool) {
    (SmartOperation operation, address mToken, uint256 amountToSupplyOrReedem) = abi.decode(params, (SmartOperation, address, uint256));
    uint256 totalAmountToRepay = amountBorrowed + premium;

    if (operation == SmartOperation.SUPPLY) {
        //get amount to supply from user
        //IERC20(token).transferFrom(owner, address(this), amountToSupplyOrReedem); ==> removed, we do transfer instead of approve from outside

        //approve total amount to supply
        uint256 totalSupplyAmount = amountBorrowed + amountToSupplyOrReedem;
        IERC20(token).approve(mToken, totalSupplyAmount);

        //supply total amount
        require(IMToken(mToken).mint(totalSupplyAmount) == 0, "mint failed");

        //borrow amount borrowed from aave plus aave fee
        require(IMToken(mToken).borrow(totalAmountToRepay) == 0, "borrow failed");

        //pay back to aave
        IERC20(token).approve(address(P00L), totalAmountToRepay);
    } else if (operation == SmartOperation.REDEEM) {
```

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets haven't been moved since the exploit. ([0x3649](#), 319K DAI)



200. Nani Finance (Dec 23)

Underlying Chain: Ethereum

Amount: \$2,000

Attack Vector: Contract Vulnerability - Logic Bug

Brief Summary

The NLP contract which manages the liquidity of the UniswapV3 pool had a vulnerability. The `contribute` function in the contract was implemented to supply liquidity when users send ETH to the contract, but the contract supported its own \$NANI token in addition to the ETH that user sent. Since this means the user earns \$NANI for free when ETH is supplied by `contribute`, the exploiter supplied a large amount of ETH through `contribute` and removed liquidity, draining all the \$NANI in the pool.

```
function contribute(address to, uint256 minOut) public payable {
    unchecked {
        assembly ("memory-safe") {
            pop(call(gas(), WETH, callvalue(), codesize(), 0x00, codesize(), 0x00))
        }

        (uint160 sqrtPriceX96, int24 currentTick,,,,) = IUniswapV3Pool(LP).slot0();
        uint256 random = _randomish(sqrtPriceX96, currentTick);

        if (random % 2 == 0) {
            uint256 liquidityPortion = (msg.value * 4) / 5;

            // Calculate discounted NANI amount for LP position:
            uint256 naniForLP = (liquidityPortion * TWO_192)
                / (uint256(sqrtPriceX96) * uint256(sqrtPriceX96)) * 100
                / (MIN_DISCOUNT + (random % (MAX_DISCOUNT - MIN_DISCOUNT + 1)));

            INonfungiblePositionManager.MintParams memory params = INonfungiblePositionManager
                .MintParams({
                    token0: NANI,
                    token1: WETH,
                    fee: 3000,
                    tickLower: (currentTick - 600) / 60 * 60,
                    tickUpper: (currentTick + 600) / 60 * 60,
                    amount0Desired: naniForLP,
                    amount1Desired: liquidityPortion,
                    amount0Min: 0,
                    amount1Min: 0,
                    recipient: to,
                    deadline: block.timestamp
                });

            INonfungiblePositionManager(POS_MNGR).mint(params);

            (int256 swapNANI,) = IUniswapV3Pool(LP).swap(
                to, false, int256(msg.value - liquidityPortion), MAX_SQRT_RATIO_MINUS_ONE, ""
            );
            if (naniForLP + uint256(-(swapNANI)) < minOut) revert InsufficientOutput();
        } else {
            (int256 amount0,) = IUniswapV3Pool(LP).swap(
                to, false, int256(msg.value), MAX_SQRT_RATIO_MINUS_ONE, ""
            );
            if (uint256(-(amount0)) < minOut) revert InsufficientOutput();
        }
    }
}
```



Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploiter ([0xB1E2](#)) hasn't moved the assets since the exploit.



201. BIZNESS Token (Dec 28)

Underlying Chain: Base

Amount: \$16,000

Attack Vector: Contract Vulnerability - Reentrancy

Brief Summary

In the `splitLock` function, which splits the user's lock position, `_feeHandler()` function which is used to send fees to treasury & send remains to the user was executed. Since it had no reentrancy guard and did not follow the CEI pattern, the exploiter could call `withdrawLock` multiple times through reentrancy attack.

Compensation Process - N/A

Pre-Incident Audit History - N/A

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx](#)

Fund Flow

Exploited assets are bridged to BSC and laundered through Tornado Cash, 15 days after the exploit..

Exploiter ([0x3Cc1](#))

→ Bridged via Symbiosis (Base → BSC)

→ [0x3Cc1](#) (26.3 BNB)

→ ... → Laundered through Tornado Cash ([0x67A5](#), 124 BNB)



202. FEG (Dec 29)



Underlying Chain: Ethereum, BSC, Base

Amount: \$1,300,000

Attack Vector: Contract Vulnerability - Improper Input Validation

Brief Summary

The root cause was the lack of validation on Wormhole bridge messages. According to the analysis of BlockSec and TenArmor, when receiving a bridge message, the relayer did not appropriately check if the source address is allowed to trigger the withdrawal registration. There was a check implemented to validate if the message is valid or not, but the main problem was that the contract registered an address to its allowlist when a legitimate message was sent once. Therefore, the exploiter could bypass the check by sending one legitimate message and a fake withdrawal message after that. On Jan 14 2025, FEG announced the migration to BSC.

Compensation Process

FEG team announced a 48 hour buyback program of FEG with an additional 25%.

Pre-Incident Audit History

Peckshield (2024.05.14) - [Report](#)

Post-Incident Audit History - N/A

Postmortem - N/A

Exploit Transactions - [Tx1](#), [Tx2](#), [Tx3](#)

Fund Flow

Exploited assets are bridged and laundered through Tornado Cash, right after the exploit.

Exploiter ([0xCB96](#)) → Bridged via SmartBridge (Base → Ethereum) → Laundered through Tornado Cash (ETH [0xCB96](#), [0xe971](#), BSC [0xCB96](#) - 169.3 ETH, 712 BNB)

End of the Document

