From Exploit to Recovery: Unraveling DeFi Incidents

@spreekaway

Outline

- Platypus Finance Case Study
- Best practices for auditors/advisors/teams to react to and/or recover from an exploit or attack
- Q&A

About Me

Twitter: **@spreekaway**
Telegram: **@Spreek**

- Academic background in math and statistics
- In defi since 2019, primarily doing **risk/governance** from 2019-2021.
- Since late 2021, **independent researcher** doing risk/governance/threat detection/on chain sleuthing

Platypus Case Study (Background/Timeline)

Dec 2021: Platypus Finance, a stableswap (e.g. Curve) fork on Avalanche launches

Feb 6, 2023: Platypus launches USP, a new stablecoin

Feb 15, 2023: USP total supply reaches $3.25m

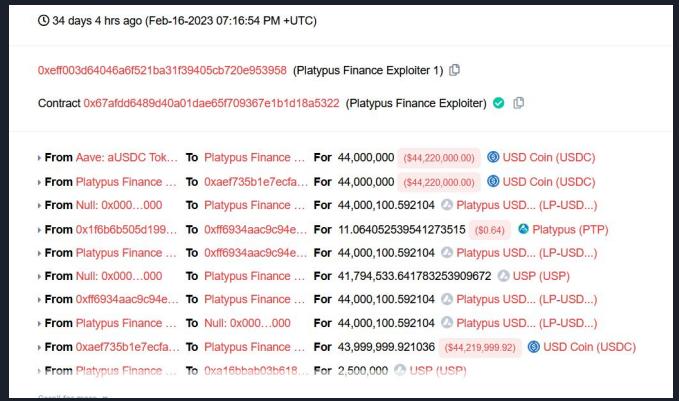
Attack #1 Timeline (Feb 16)

18:51 UTC: Attacker funds wallet using Fixed Float

Transfer	26342857	34 days 5 hrs ago	FixedFloat	IN	Platypus Finance Exploit…	5.864973 AVAX

19:16 UTC: Attacker deploys Attack Contract #1, and calls Hello(), executing the first attack 3 blocks later.

Due to error in attack contract, the ~$8m in stolen stables cannot be withdrawn

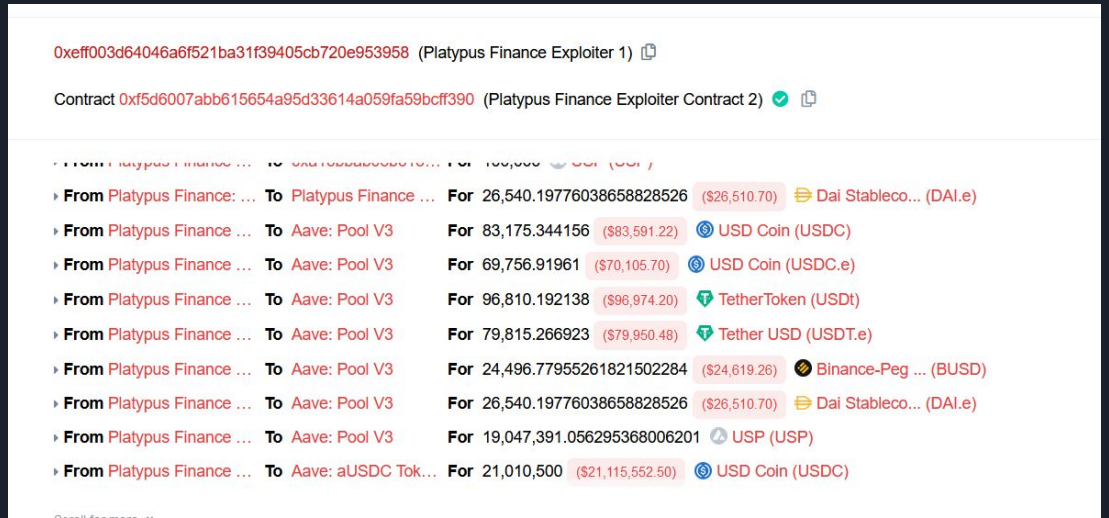


Attack #2 Timeline

19:32 UTC: Attacker deploys Attack Contract #2

19:38 UTC: Attacker calls Hello(), executing attack #2.

This time, due to a different error in the attack contract, the proceeds are sent to Aave reserves rather than to the attacker



Attack #3 Timeline

19:51 UTC: Attacker deploys Attack Contract #3 and calls Hello() executing the third attack

This time, there are no errors and the attacker manages to escape with the money.

He spends the next hour or so converting the proceeds into 14,314 AVAX or approx $225k

20:32 UTC First mention of attack in public channels

20:36 UTC: I tweet about it

Attack Aftermath

21:38 UTC: USDT in Attack Contract #1 frozen by Tether

Feb 17th, 04:12 UTC:
Attacker publicly identified by @ZachXBT

Feb 17th, 16:20 UTC:
Platypus team upgrades their proxy w/ help of Blocksec

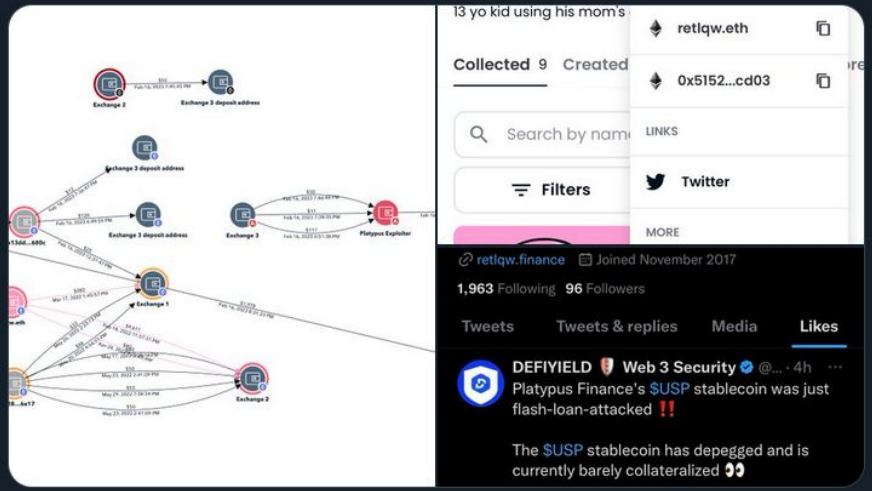
Feb 17th, 16:21 UTC:
Using upgraded proxy, they counterexploit attacker, recovering 2.4m USDC

Feb 25th: Attacker arrested in France

ZachXBT
@zachxbt

I've reviewed your transaction history across multiple chains which lead me to your ENS address retlqw.eth

Your OpenSea account links directly to your Twitter and you liked a Tweet about the Platypus exploit.



10:12 PM · Feb 16, 2023 · **89K** Views

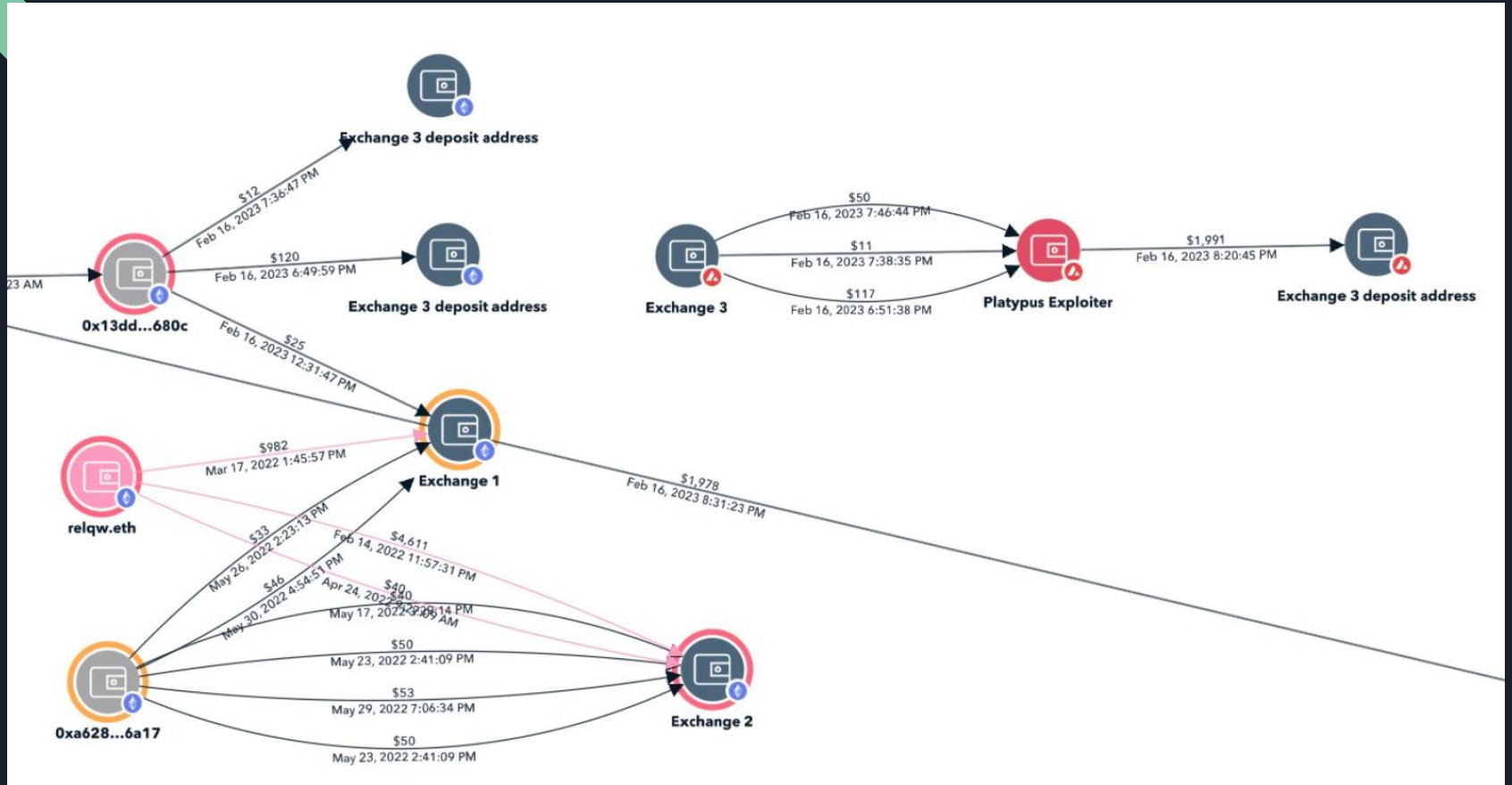
26 Retweets **14** Quotes **655** Likes **9** Bookmarks

Root cause of exploit? HT @danielvf

```
/// @notice Withdraw without caring about rewards. EMERGENCY ONLY.
/// @param _pid the pool id
function emergencyWithdraw(uint256 _pid) public nonReentrant {
    PoolInfo storage pool = poolInfo[_pid];
    UserInfo storage user = userInfo[_pid][msg.sender];

    if (address(platypusTreasure) != address(0x00)) {
        (bool isSolvent, ) = platypusTreasure.isSolvent(msg.sender, address(poolInfo[_pid].lpToken), true);
        require(isSolvent, 'remaining amount exceeds collateral factor');
    }

    // ...

    // SafeERC20 is not needed as Asset will revert if transfer fails
    pool.lpToken.transfer(address(msg.sender), user.amount);
    // update non-dialuting factor
    pool.sumOfFactors -= user.factor;
    user.amount = 0;
    user.factor = 0;
    user.rewardDebt = 0;
    emit EmergencyWithdraw(msg.sender, _pid, user.amount);
}
```

No check or adjustment for borrowed funds

How was attacker identified? @ZachXBT



How were funds recovered? HT @danielvf

Decompile of the upgraded proxy

ByteCode Decompilation Result:

```
17    return 0
18
19  def unknown09a5fca3(uint256 _param1, uint256 _param2) payable:
20    require calldata.size - 4 >=ΓÇ▒ 160
21    require _param1 == addr(_param1)
22    require _param2 == addr(_param2)
23    require ext_code.size(0xb97ef9ef8734c71904d8002f8b6bc66dd9c48a6e)
24    static call 0xb97ef9ef8734c71904d8002f8b6bc66dd9c48a6e.allowance(address tokenOwner, address spender) with:
25           gas gas_remaining wei
26          args caller, this.address
27    if not ext_call.success:
28        revert with ext_call.return_data[0 len return_data.size]
29    require return_data.size >=ΓÇ▒ 32
30    require ext_code.size(0xb97ef9ef8734c71904d8002f8b6bc66dd9c48a6e)
31    call 0xb97ef9ef8734c71904d8002f8b6bc66dd9c48a6e.transferFrom(address from, address to, uint256 tokens) with:
32         gas gas_remaining wei
33        args caller, 0x416a7989a964c9ed60257b064efc3a30fe6bf2ee, ext_call.return_data[0]
34    if not ext_call.success:
35        revert with ext_call.return_data[0 len return_data.size]
36    require return_data.size >=ΓÇ▒ 32
37    require ext_call.return_data == bool(ext_call.return_data[0])
38    return 0
39
40
41
```

USDC

Takeaways

- Be especially careful with functions like emergencyWithdraw() that are not intended to be called often
- Don’t hesitate to reach out for help from public and security community after experiencing an exploit. ZachXBT, Blocksec, French law enforcement, and many others were instrumental in achieving as good an outcome as happened here.
- Hope whoever attacks you makes as many mistakes as this guy did

Part 2: Best Practices

Before an attack occurs...

- Have robust internal notifications system for potentially material events related to your project
  - E.g. large TVL changes, multisig transactions, new governance votes, etc.

Before an attack occurs...

- Verify all your contracts on Etherscan. Security by obscurity will not help you vs attackers, but will frustrate whitehats trying to help you

- PLEASE include your project name and good contact info (even if it is just the most active chat channel for your project). Seconds and minutes count!

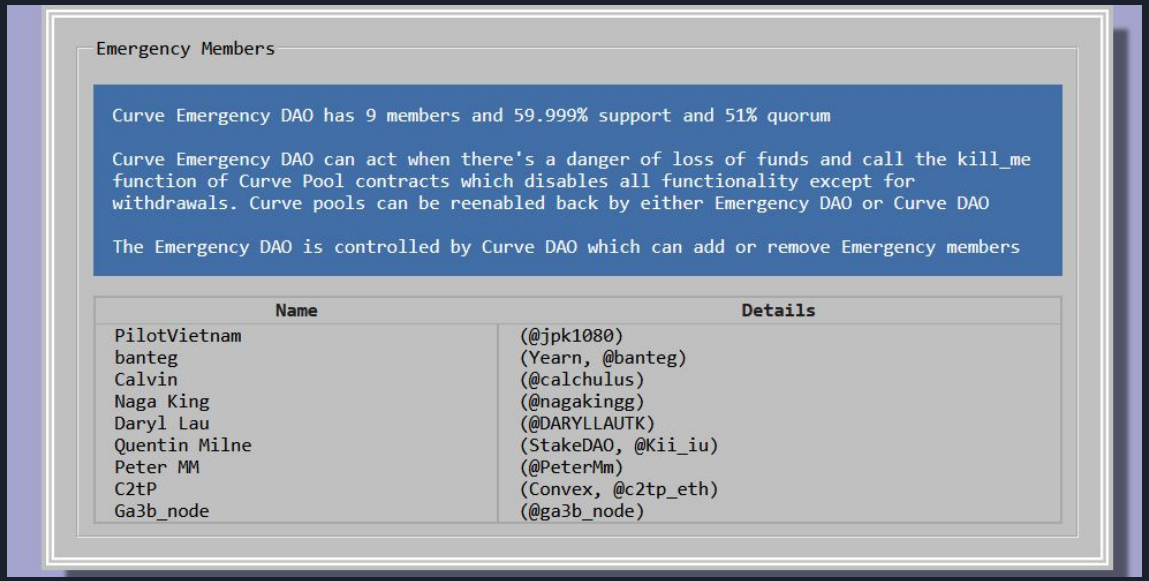
Before an attack occurs...

- Make sure you can respond to an attack regardless of what time zone it occurs in. “Devs are sleepy” is not a good excuse!

- If you have heavy concentration in one time zone, consider rotating someone to be on call, and/or enlisting community mods that can wake up someone if necessary

Before an attack occurs:

- Consider designing in circuit breakers, emergency DAOs that have strictly risk reducing governance powers, or similar to allow for rapid derisking of the system on short notice



Emergency Members

Curve Emergency DAO has 9 members and 59.999% support and 51% quorum

Curve Emergency DAO can act when there's a danger of loss of funds and call the kill_me function of Curve Pool contracts which disables all functionality except for withdrawals. Curve pools can be reenabled back by either Emergency DAO or Curve DAO

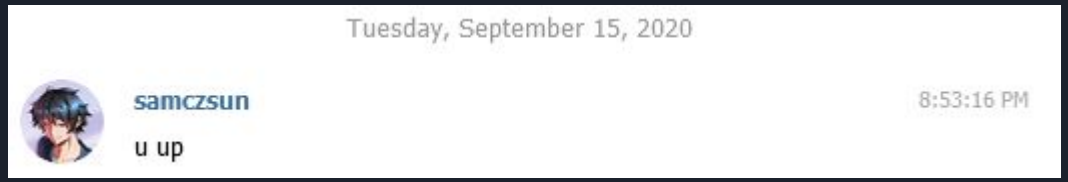
The Emergency DAO is controlled by Curve DAO which can add or remove Emergency members

Name	Details
PilotVietnam	(@jpk1080)
banteg	(Yearn, @banteg)
Calvin	(@calchulus)
Naga King	(@nagakingg)
Daryl Lau	(@DARYLLAUTK)
Quentin Milne	(StakeDAO, @Kii_iu)
Peter MM	(@PeterMm)
C2tP	(Convex, @c2tp_eth)
Ga3b_node	(@ga3b_node)

While an attack is underway:

- Make sure community mods and frontline people in your telegram or discord are taught what kinds of things are concerning and encouraged to escalate suspicious activity ASAP

- Most of the time, your first notice that something is happening will be from one of your users (or an outside whitehat group) posting in one of your chat channels or DMing one of you.

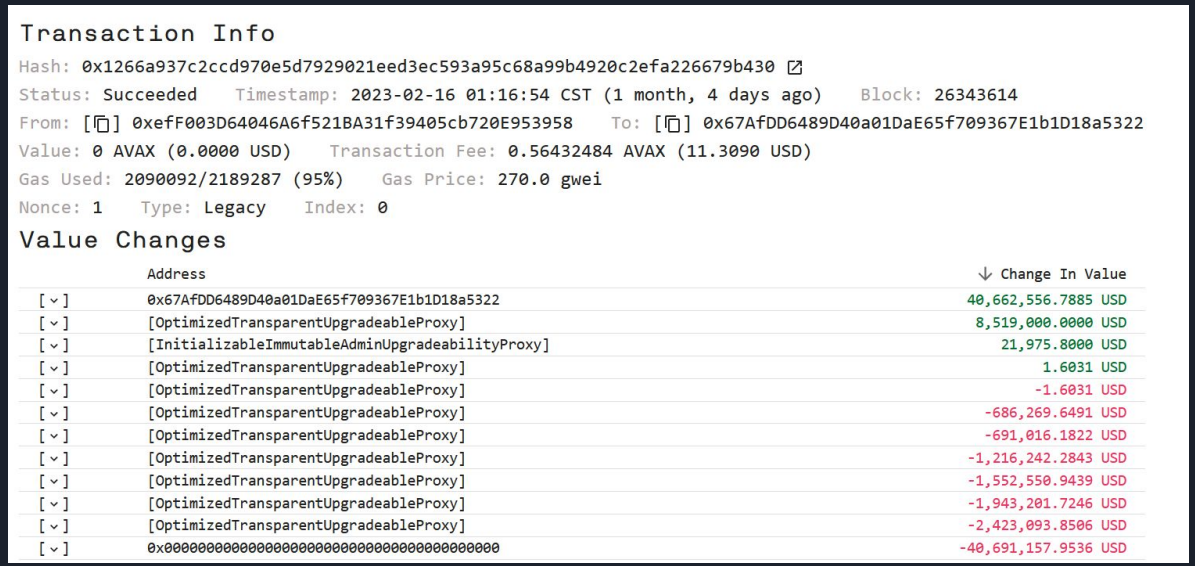


While an attack is underway:

- What to look for when sent a suspicious user/contract/tx
  - User's funding source and history. New accounts funded by tornado/changenow/fixedfloat/etc much more likely to be malicious than old accounts with long history of CEX funding
  - If they have deployed a contract:
    - Which (if any) of your contracts does it reference in its bytecode/calldata?
    - Does it seem to be using flashloans? Flashloans more likely malicious
    - How complex is the contract? More complex → more likely to be malicious
    - Does it reference block.coinbase? Is the owner EOA prefaced with tons of 0s? If yes, more likely to be benign MEV related

While an attack is underway:

- If one or more suspicious transactions have gone through:
  - Do the transactions show a large profit for the user at the expense of your contracts? https://openchain.xyz **very useful tool for this**



Transaction Info

Hash: 0x1266a937c2ccd970e5d7929021eed3ec593a95c68a99b4920c2efa226679b430

Status: Succeeded Timestamp: 2023-02-16 01:16:54 CST (1 month, 4 days ago) Block: 26343614

From: [] 0xefF003D64046A6f521BA31f39405cb720E953958 To: [] 0x67AfDD6489D40a01DaE65f709367E1b1D18a5322

Value: 0 AVAX (0.0000 USD) Transaction Fee: 0.56432484 AVAX (11.3090 USD)

Gas Used: 2090092/2189287 (95%) Gas Price: 270.0 gwei

Nonce: 1 Type: Legacy Index: 0

Value Changes

	Address	↓ Change In Value
[⌄]	0x67AfDD6489D40a01DaE65f709367E1b1D18a5322	40,662,556.7885 USD
[⌄]	[OptimizedTransparentUpgradeableProxy]	8,519,000.0000 USD
[⌄]	[InitializableImmutableAdminUpgradeabilityProxy]	21,975.8000 USD
[⌄]	[OptimizedTransparentUpgradeableProxy]	1.6031 USD
[⌄]	[OptimizedTransparentUpgradeableProxy]	-1.6031 USD
[⌄]	[OptimizedTransparentUpgradeableProxy]	-686,269.6491 USD
[⌄]	[OptimizedTransparentUpgradeableProxy]	-691,016.1822 USD
[⌄]	[OptimizedTransparentUpgradeableProxy]	-1,216,242.2843 USD
[⌄]	[OptimizedTransparentUpgradeableProxy]	-1,552,550.9439 USD
[⌄]	[OptimizedTransparentUpgradeableProxy]	-1,943,201.7246 USD
[⌄]	[OptimizedTransparentUpgradeableProxy]	-2,423,093.8506 USD
[⌄]	0x0000000000000000000000000000000000000000	-40,691,157.9536 USD

While an attack is underway:

- Alternatively, do the transactions show a pattern of uneconomic activity (especially routed through user created contracts)? Example: 0.1 USDC deposits through your bridge
- Often a sign of an attack in progress (or at the very least, someone testing a potential exploit “in prod”)

After an attack is confirmed:

- Create war room with devs/auditors/security researchers/etc ASAP
- Often times this might be when you as the auditor for the project are called in to help
- Limit further damage by pausing contracts/stopping new trading/lending etc.
- Make public statement acknowledging the attack, giving any important action items for public ASAP. It is in the public interest and also will often mobilize community to help you

After an attack is confirmed:

- Reach out to attacker offering bounty for return of funds (not often successful, but worth a shot)
- Contact third parties that may be able to thwart efforts to launder money (analytics companies like Chainalysis/TRM, stablecoin issuers like Tether and Circle, centralized exchanges like Binance, third party swaps like Fixedfloat/ChangeNow, bridges like Celer, Wormhole, etc.)

After an attack is confirmed:

- Do root cause analysis on how exploit was done. Are any of your other products/contracts vulnerable to similar exploits? What about forks or competitors?
- If attacker has not fully divested your projects tokens, consider whether a snapshot + reissuance could be appropriate. Do you have governance powers to rug him/her?

Questions?