Exploiting Inattention & Optimism in DAOs

How I stole from a DAO using standard governance tools
(and how to protect yourself)

Isaac Patka (@isaacpatka)
Logos DAO, Metagov
Proof of Inattention

Real Exploits 😞

Protecting you & your DAOs 🦧
Attention is the most scarce resource in DAOs. Design your governance tooling accordingly.

Optimistic consensus relies on people paying attention 😯

⚡ SafeSnap relies on Reality.ETH, a Q&A oracle with bonded answers

👹 Moloch DAOs use lazy consensus and have no minimum quorum for proposals to pass*

*but they do need to be sponsored by a member
Reality.ETH is a Q&A Oracle

Many DAOs use Reality.ETH to make **off-chain** votes executable **on-chain**

Did the DAO vote to pay me $20k DAI?

1. Ask the question

2. Answer the question

3. Execute
Reality.ETH can execute transactions on a Gnosis Safe through a Zodiac Module.

- DAO Treasury
  - Gnosis Safe
  - Multisig Signers
  - Full Control

- Reality Module Zodiac
  - Instructs
  - Reality.ETH Oracle dApp
  - Asks Questions
  - Anyone

Full Control too
Multisig Owners Choose the Configuration

**Timeout** - Duration during which answers can be submitted

**Cooldown** - Optional duration after Oracle finalization, before execution

**Expiration** - Optional duration during which finalized answer can be executed

**Bond** - Minimum bond for answer to be accepted

**Arbitrator** - Optional 3rd party that can settle Oracle disputes

**Question Template** - How should questions look to Reality.ETH dApp users
Scenario 1 - Happy Path

1. DAO Vote on Snapshot
2. Ask Reality.ETH if it passed
3. Answer honestly
4. Execute

- **Timeout** - Duration during which answers can be submitted
- **Cooldown** - Duration after Oracle finalization, before execution
- **Expiration** - Duration during which finalized answer can be executed
Scenario 2: Dishonest Oracle

1. DAO Vote on Snapshot
2. Ask Reality.ETH if non-existent proposal passed
3. Answer dishonestly
4. 🙏 Hope no one is watching
5. Execute

An attacker can pose a non-existent proposal as a question to Reality.ETH, and submit a fraudulent answer by putting down a bond in ETH.
Scenario 2: Dishonest Oracle - Override

1. DAO Vote on Snapshot
2. Ask Reality.ETH if non-existent proposal passed
3. Answer dishonestly
4. Override malicious answer
5. Execute

An honest person can override the malicious answer and claim the bonded ETH
Scenario 2: Dishonest Oracle - Arbitration

1. DAO Vote on Snapshot
2. Ask Reality.ETH if non-existent proposal passed
3. Answer dishonestly
4. Request arbitration
5. Execute

An arbitrator can step in to override the malicious answer (IF one is configured)
Scenario 2: Dishonest Oracle - Veto

1. DAO Vote on Snapshot
2. Ask Reality.ETH if non-existent proposal passed
3. Answer dishonestly
4. Veto proposal
5. Execute

Timeout

Cooldown

Expiration

Multisig owners can **veto** the malicious answer during **cooldown** (if it is configured)
Misconfiguration can make exploits trivial

**Timeout** - Too short of a timeout can make it hard to catch malicious transactions

**Cooldown** - 0 second cooldown removes veto period

**Bond** - Low minimum bond makes it cheap to try and exploit

**Arbitrator** - Absent arbitrator removes final safeguards

**Vetoer** - Absent or negligent multisig signers remove veto safeguard

We will see examples of ALL of these misconfigurations in mainnet exploits (coming up next...)
Section 2

Real Exploits 😞

How I exploited a DAO and how others are attacking them as we speak

$100Ms of DAO treasuries are at risk of inattention attacks
Exploiting the SafeSnap Honeypot

Gnosis set up a bug bounty for the Reality.ETH module in Spring 2021 and it sat dormant for over a year.

https://etherscan.io/address/0x0a147ddf0817ade664eb9cb343d96a21ed857d11
Crafting the Exploit

On the Etherscan page for the module I was able to call 'addProposal' with a transaction I crafted locally

etherscan.io/address/0x1c51...

This transaction would have sent me ~20k DAI from the safe if it was approved

1. addProposal

The nonce used for the question by this function is always 0

Function to add a proposal that should be considered for execution

proposalId (string)

nothingtoseehere

Id that should identify the proposal uniquely

txHashes (bytes32[])

[0x9f033cd0fe1086d9ec3e83149211ca3c2ff69b6b7667b18ba4ae8db5a6ad8be]

EIP-712 hashes of the transactions that should be executed

Write  View your transaction

I answered the question 'YES' with a bond of 0.1 ETH
Defenders take notice

#nothingtoseehere

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>proposalID</td>
<td>string</td>
<td>nothingtoseehere</td>
</tr>
<tr>
<td>1</td>
<td>txHashes</td>
<td>bytes32[]</td>
<td>8x9f933cd0fe1086d9ec3e83149211ca332cf669b6b7667b18ba4ae8db6e5ad8be</td>
</tr>
</tbody>
</table>

13:42 · 5/24/22 · Twitter Web App

Mkoeppelmann 5:36 PM
potentially someone is trying to drain the honey DAO. cc @Auryn

Auryn.eth 🏛️ 🏛️ 🌤️
@auryn_macmillan

 DAO.eth is actually just a mutlisig controlled by @koeppelmann, @rimeissner, and myself. So first step is to invalidate the proposal.

Notice the proposal ID, "nothingtoseehere"

gnosis-safe.io/app/eth:0x0a14...

Interact with:
Honey DAO SafeSnap
Ether: 0xc51d88ba898b4D9cd9113D13B9c360a02Fc4a1

MARK PROPOSAL AS INVALID
proposalID(string):
nothingtoseehere
    [ 0x9f033cd0fe1086d9ec3e831...

13:42 · 5/24/22 · Twitter Web App

3 Likes
“Is anyone available to sign this transaction?”
But it was too late

This exploit was successful because...

**Arbitration** could not be requested

**Cooldown** was a short 24hr and the vetoers were AFK

**No one** else on Reality.ETH was paying attention
Then it started happening for real...

While monitoring their own Gnosis Safe & Reality module, the Opium Network team discovered a series of fraudulent transactions attacking DAO treasuries.
Attack 1 - easy target

The attacker found a DAO with a Reality module configured with just 24hr cooldowns, little activity, and no minimum bond.

They stole 7.5 ETH after putting down a 0.01 ETH bond.
Attacks 2-7+

The attacker used the 7.5 ETH bond to place fraudulent answers in at least 6 other DAOs.

They primarily targeted NFT collections including SZNS.

🎉 The SZNS team had 7 day voting periods & 1 ETH minimum bonds so the attacker was limited in how many they could attempt.

I was able to thwart the attack by overriding their answer, but if the attacker was more highly capitalized it would have been harder to defend.
$100Ms of DAO treasuries are at risk of inattention attacks

🔔 We need more monitoring infrastructure for DAO treasuries & governance tooling

📝 We need configuration audits, just as much as we need smart contract audits

 Attacks like this are only going to start happening more frequently
Protecting you & your DAOs
10 Steps to keep your DAO Safe:

1) Make a resiliency & continuity plan
2) Keep track of who has administrative controls over smart contracts (ideally 0 or limited multisigs)
3) Set up monitoring infrastructure
   a) Etherscan alerts, OpenZeppelin Sentry
4) Leverage automation tools to pause contracts if exploit conditions are detected
   a) OpenZeppelin Defender
5) Use simulation tools to check what proposals are going to change before you execute them
   a) Tenderly
6) Conduct regular configuration audits, especially focusing on new tools that can execute proposals
7) Minimize cross-chain communication
   a) It’s always the bridges that get hacked
8) Implement spending limits & transaction guards on Safe treasuries
9) Use hardware wallets & never back up your seed phrase online (including password managers)
10) Use on-call shifts to track availability of multisig signers

Regularly Audit your DAO’s tooling stack and set up robust monitoring infrastructure

Reach out to LOGOS DAO & isaac@logos.xyz
Composable governance tooling make DAOs powerful but requires careful configuration
Thank you!

Isaac Patka
Co-summoner, Logos DAO
isaac@logos.xyz

@isaacpatka