Nomad Bridge Hack

Sergios - Anestis Kefalidis
cs32200036

Outline

- Blockchain Bridges
- Blockchain Bridge Hack
- Nomad Bridge
  - Nomad Bridge Optimistic Verification
  - Fraud Detection
  - Lifecycle of a message
- The Nomad Bridge Hack
- Aftermath

Blockchain Bridges

- Each blockchain is an independent, siloed environment.
- Bridges enable connectivity and interoperability between blockchains.
  - Transfer of assets and/or data.
  - Centralized/Trusted - Decentralized/Trustless.
    - speed/cost vs security
  - Wrapped assets - Liquidity pools
- High volume of funds.
  - Lucrative targets for bad-faith actors.
  - Over $2.5 billion have been stolen from cross-chain bridges.

Blockchain Bridges

Blockchain Bridge Hacks

Target	Money stolen	Hackers
Ronin Bridge	$625,000,000	1
PolyBridge	$610,000,000	1
Wormhole	$320,000,000	1
Nomad Bridge	$186,000,000	>300
Horizon Bridge	$100,000,000	1

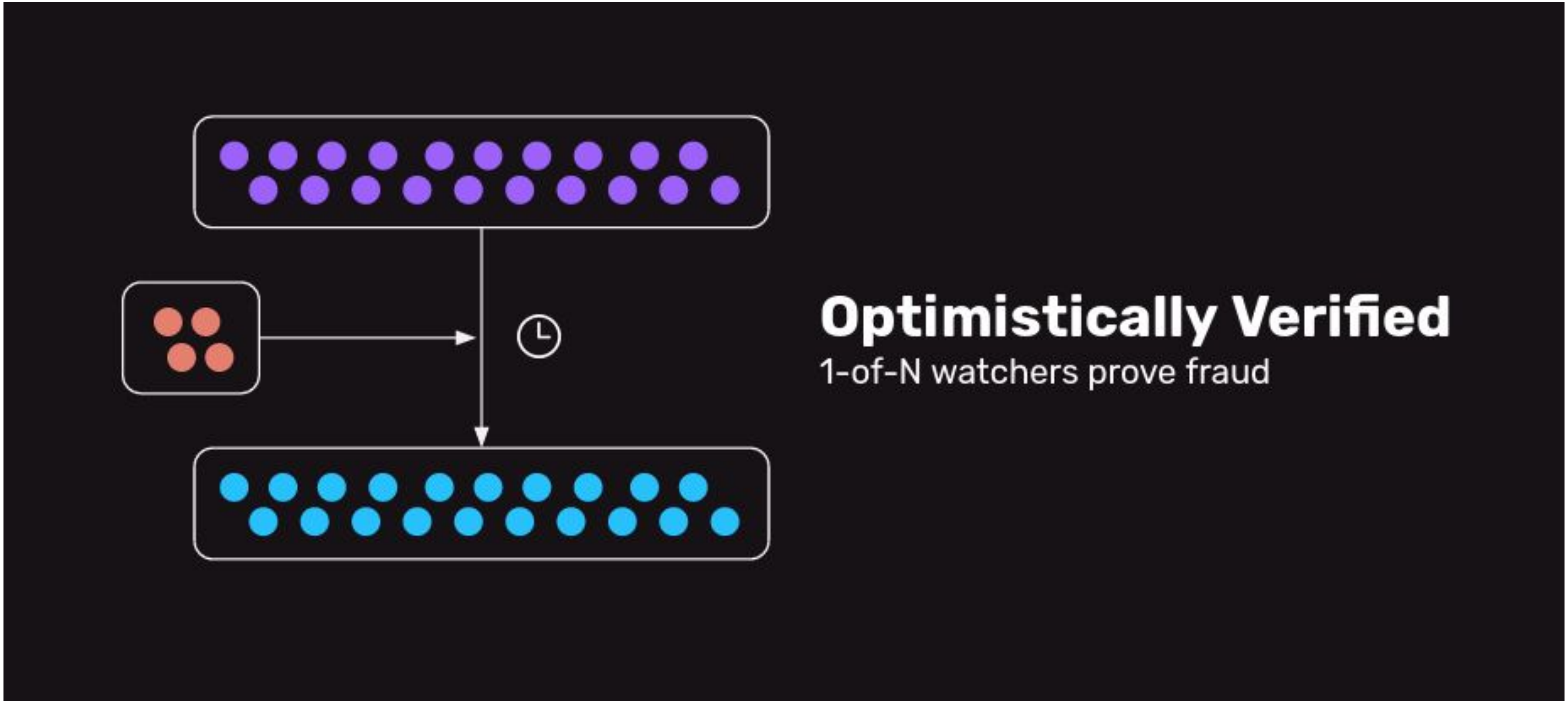
Nomad Bridge

- Cross-chain communication between:
    - **Ethereum** (only one affected by the hack)
    - Avalanche
    - Evmos
    - Milkomeda C1
    - Moonbeam
- Lock & Mint transfer of tokens.
    - Minted tokens are burned to unlock original tokens.

Nomad Bridge

- To push data we must rely on someone to verify and relay the data.
  - Goal: Minimize the trust assumptions in the verification process.
- Nomad uses an optimistic mechanism. It consists of the following actors:
  - **Home** contract.
    - **Send** messages.
    - Data Structures:
      - A **Merkle Tree** that holds all messages. New messages are stored as leafs.
      - A **Queue of Roots** that contains all roots of the Merkle Tree. Used to prove fraud.
    - Updates to the Merkle Tree are signed (**Updater**) and relayed to **Replicas** deployed to destination chains.
  - **Replica** contract.
    - **Receive** messages from a specific **Home** contract.
    - Data Structures:
      - A **Queue of Pending Updates** used to identify fraud.
    - Signed updates are accepted after a timeout (**optimistic dispute window**).
  - **Updater** (off-chain).
    - Signs new roots and publishes them to the home chain.
      - Listens to **Home** *Dispatch* events, baches them and signs updates by calling *Update* on the **Home** contract.
  - **Watcher** (off-chain).
    - Observes **Home** and **Replicas** to detect fraud.

Nomad Bridge: Optimistic Verification



Nomad Bridge: Fraud Detection

- **Updaters** can attempt to commit fraud.
    - When an Updater signs an attestation to a merkle root that did not actually exist on the Home chain. This would mean that malicious messages would be authenticated and executed.
- To detect fraud **Watchers** are used. At least 1 agent is required to act honestly to detect fraud.
    - We only need to check that the state in the **Replicas** is equal to the **Home**.
- **Optimistic Timeout Period**: A time window during which **Watchers** can submit fraud proofs.
    - 30 minutes, it is prohibitively expensive for an attacker to buy the blockspace for 30 minutes
- If a fraud attempt by an **Updater** is detected, the **Updater** is slashed.

Nomad Bridge: Lifecycle of a message

1. User initiates action on chain A.
2. Business logic is executed on chain A.
3. The message is enqueued on the **Home** contract.
4. Nomad's work begins.
    a. New Merkle Tree root on the **Home** contract.
    b. The **Updater** signs the new root.
    c. The update is relayed to the **Replica** on chain B.
    d. The **dispute window** elapses.
    e. The message can now be proven on chain B.
5. Business logic is executed on chain B.

The Nomad Bridge Hack

- Vulnerable **Replica** contract upgrade on June 21st, 2022.
    - An implementation bug caused the **Replica** contract to fail to authenticate messages properly.
    - This issue allowed any message to be forged as long as it had not already been processed.
    - Only Ethereum was affected.
- First malicious transaction: August 1st, 2022, 21:32:31.
- After the initial vulnerability was discovered a lot of people copied it.
    - Decentralized Finance means that anyone can join :-)
- $186M stolen by over 300 hackers.
    - 960 transactions
    - In a few hours only $1,794 dollars were left

The Nomad Bridge Hack

- For a message to be accepted the following conditions must apply:
  - It exists in the **Merkle Tree** (Merkle proof).
  - The **Optimistic Timeout Period** has elapsed.

```
function process(bytes memory _message) public returns (bool _success) {
    // ...
    require(acceptableRoot(messages[_messageHash]), "!proven");
    // ...
}

function acceptableRoot(bytes32 _root) public view returns (bool) {
    // ...
    uint256 _time = confirmAt[_root];
    if (_time == 0) {
        return false;
    }
    return block.timestamp >= _time;
}
```

The Nomad Bridge Hack

- When a **Replica** is deployed after its associated **Home** contract, the **Replica** contract is initialized with a specific state.
    - This way deployments don't have to replay all past updates.
    - The deployer may pass a *committedRoot* at which the message tree's history begins receiving Updates by setting:
        - `confirmAt[_committedRoot] = 1`
- Deploying both **Home** and **Replica** contracts at the same time means that there are no messages.
    - In the Nomad implementation this means a **Merkle Tree** with a root of *bytes32(0)*.
    - `confirmAt[bytes32(0)] = 1`
- If a message hash doesn't exist in the *messages* mapping it will return a value of *bytes32(0).*
    - This will be passed to *acceptableRoot* which will return *true*.
    - Vulnerability!

The Nomad Bridge Hack

- Why did the **Watchers** not take action here?
    - Respond to compromises of the Updater key.
    - Unable to detect suspicious activity arising from smart contract bugs.
    - This exploit didn't require a fraudulent Updater signature.
- Anyone who knew how to encode a message for Nomad could just send it to the **Replica** contract (by calling its vulnerable *process* function) and it would not be checked for its authenticity.
- Attack Example

Aftermath

- Recovered Funds
  - 20% in 6 days
  - 21% as of 6/11/2022
- Radio silence after December 2022.
- The last Nomad Bridge blog post is about ongoing work on relaunching the bridge.