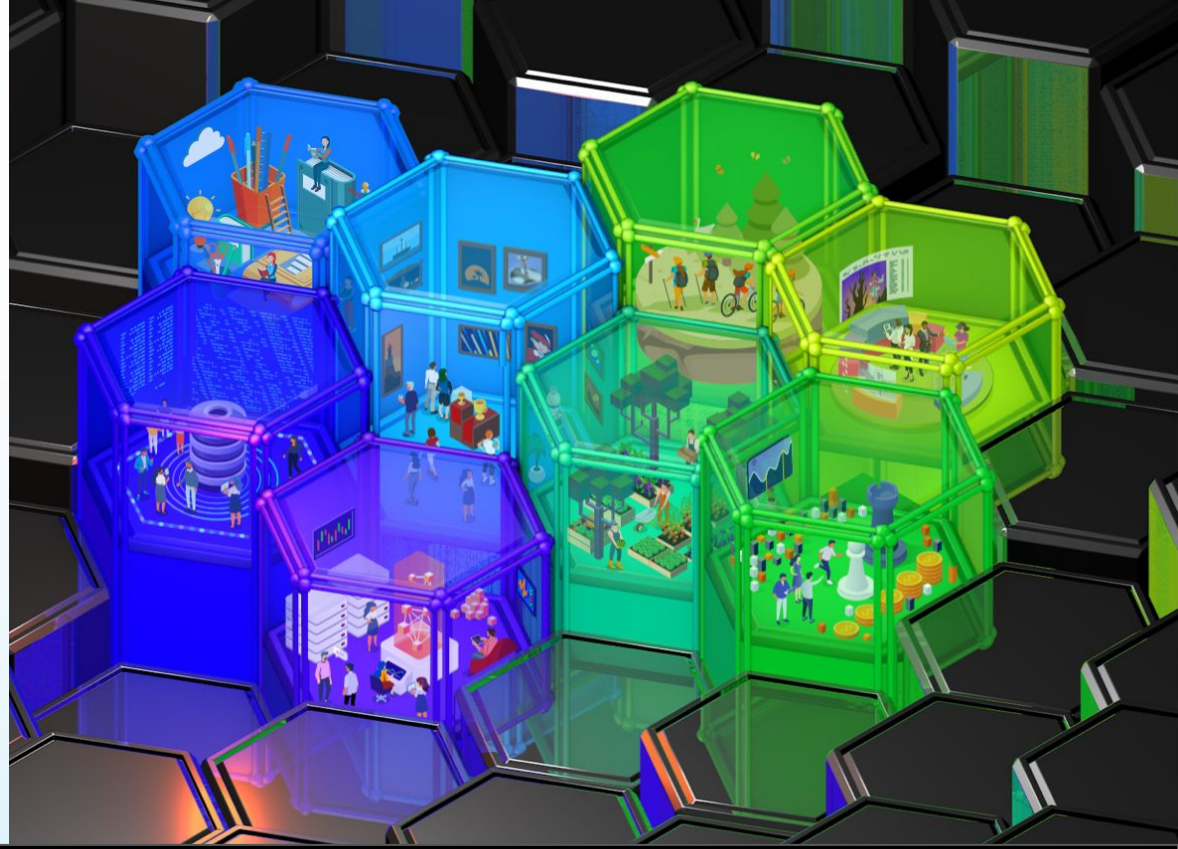
Not Quite Water Under the Bridge: Review of Cross-Chain Bridge Hacks



Martinet Lee
Martin Derka
Sebastian Banescu
Alex Murashkin
Kacper Bak

Why Is This Talk Relevant?

Polygon Bridge: $850M at risk

Ronin Bridge: $600M rekt

Poly Network Bridge: $600M rekt

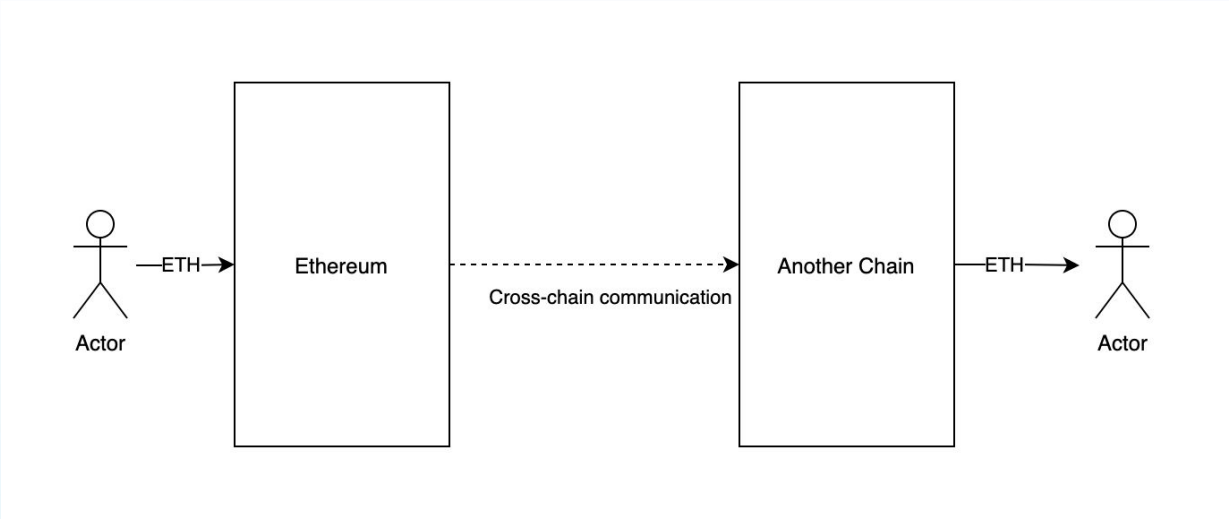
Wormhole Bridge: $325M rekt

Qubit Bridge: $80M rekt

…

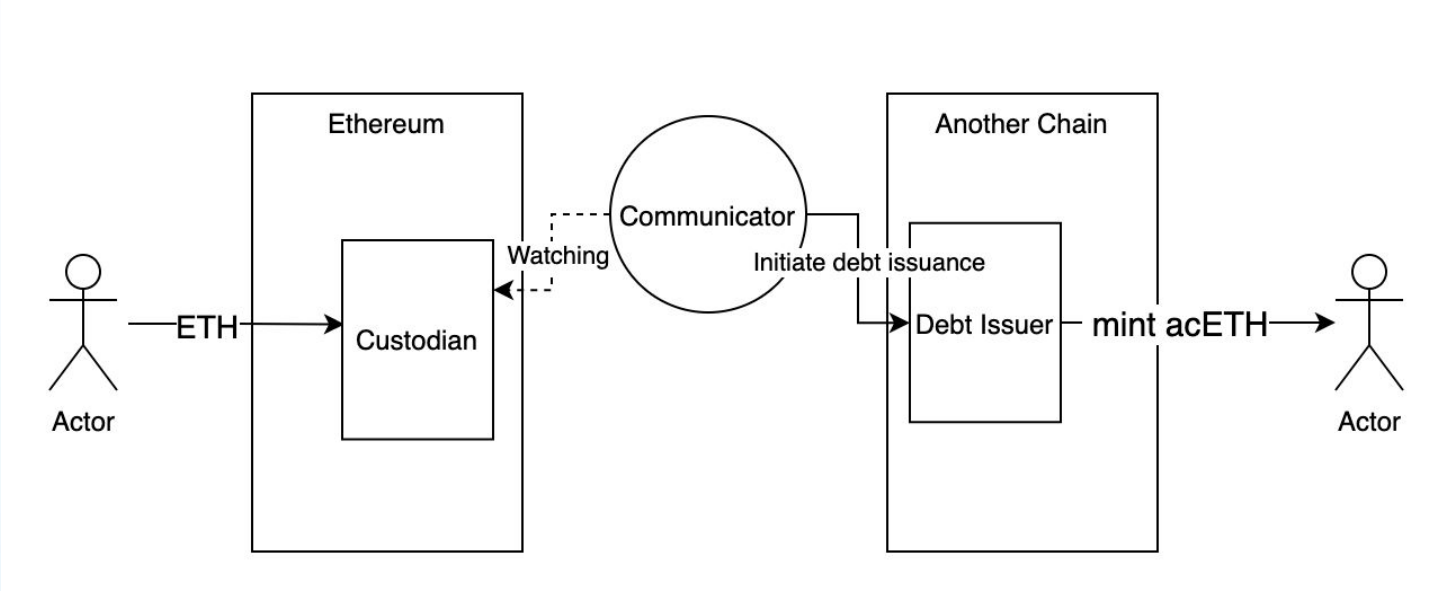
Bridge

A bridge is used to “move” assets from one chain to another



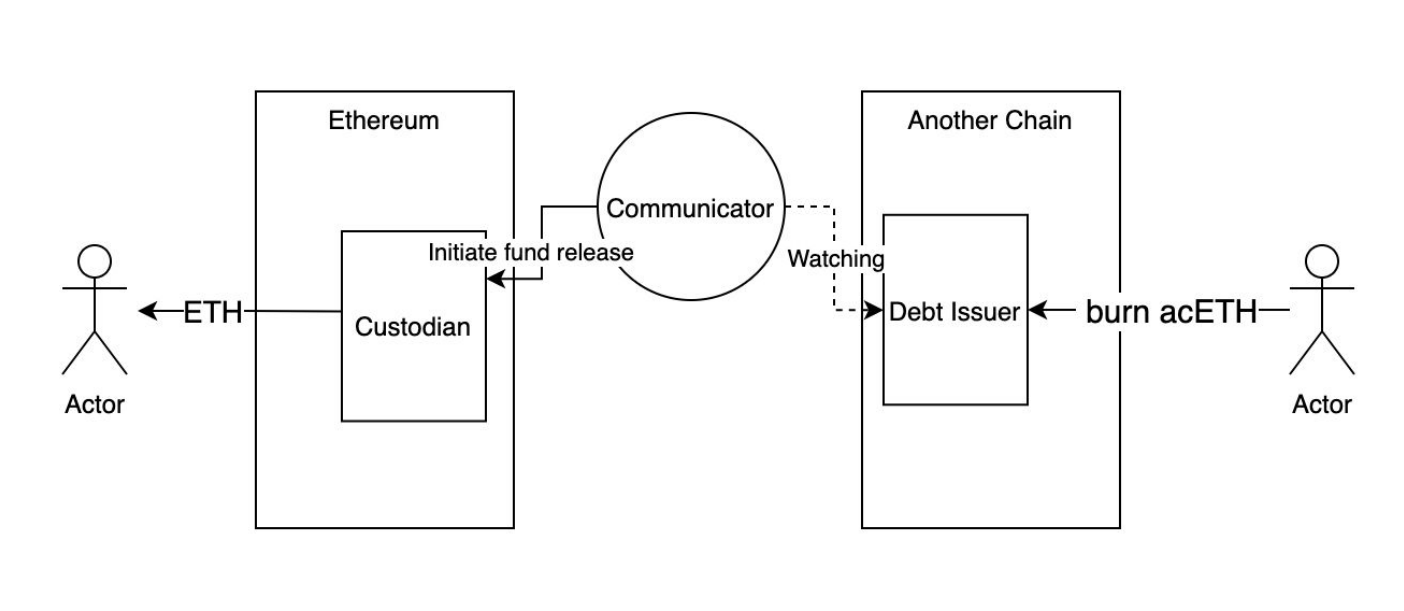
Bridge: Deposit

Assets are locked on one chain, and a debt token is issued on another chain



Bridge: Withdrawal

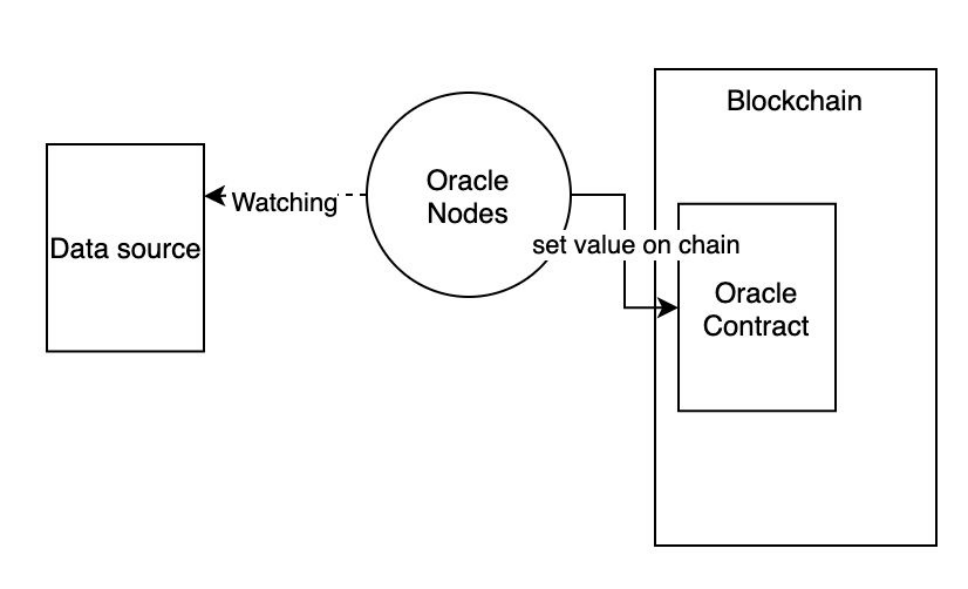
Debt token is burnt on the other chain and
the Communicator informs the custodian to release the deposited assets



Bridge Structure

Contracts
- Asset custodian
- Debt issuer

Communicator
- Oracle



Attack Surfaces

Custodian

Debt Issuer

Communicator

Interfaces

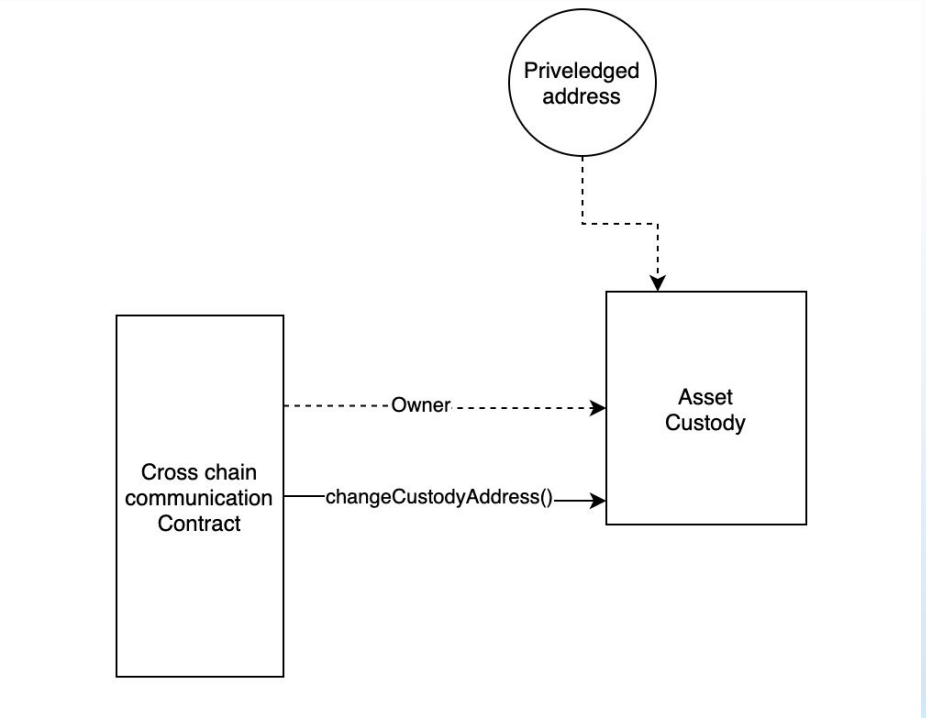
Network

Custodian: Call Relay Attack

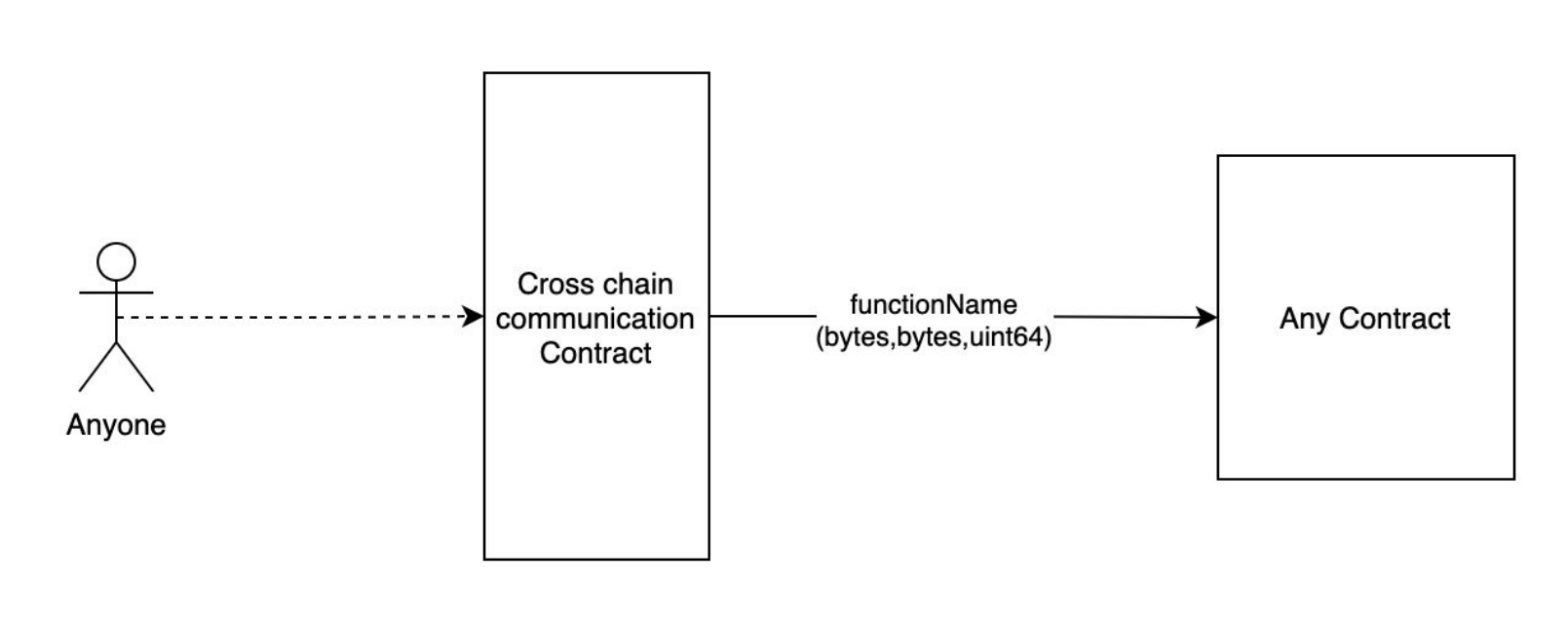
Depending on the custodian, privileged addresses may have access to the assets

Attacker's goal:

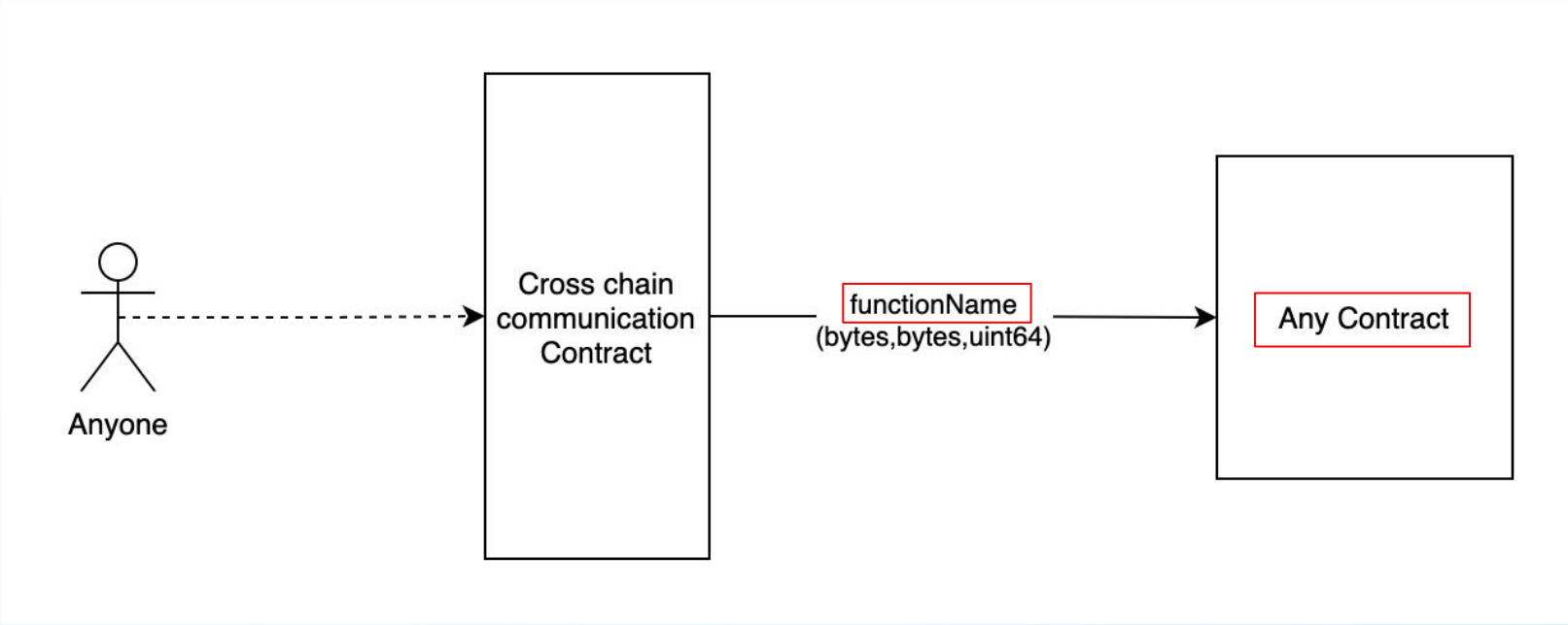
- Take over the privileged addresses
- Change the list of privileged addresses



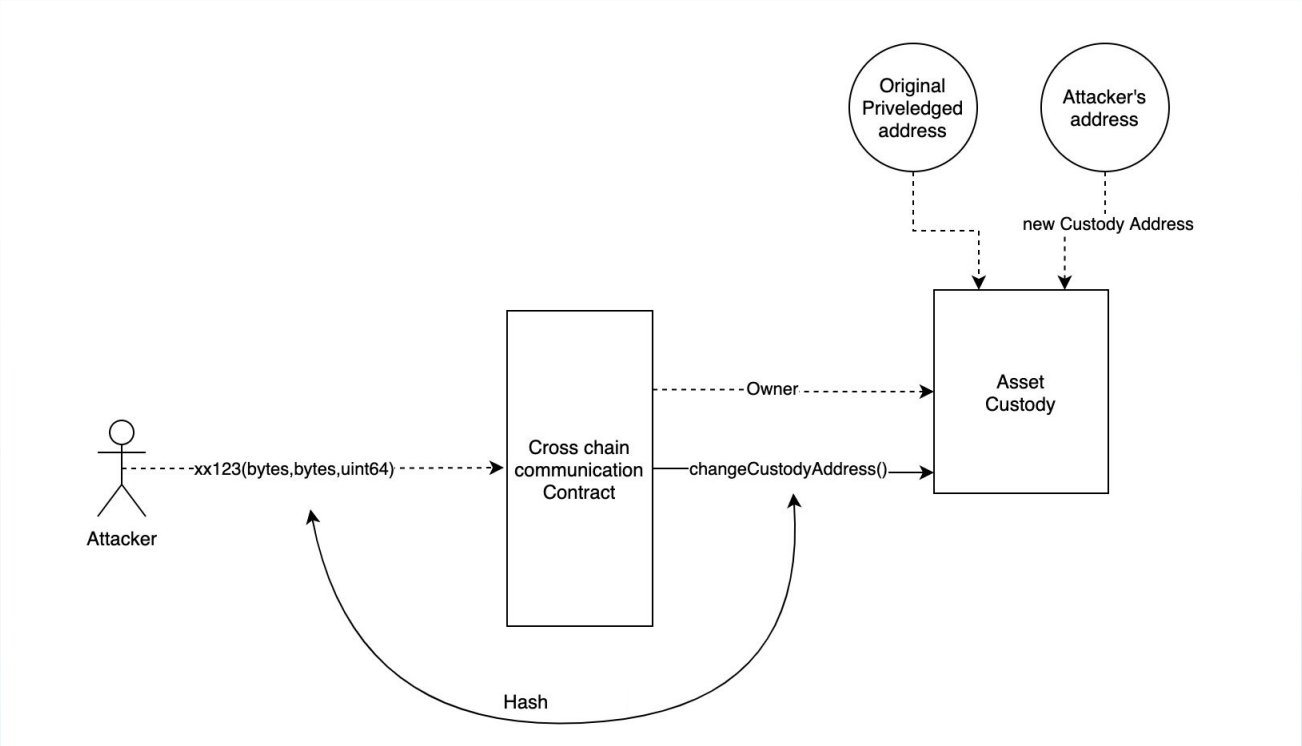
Custodian: Call Relay Attack



Custodian: Call Relay Attack



Custodian: Call Relay Attack

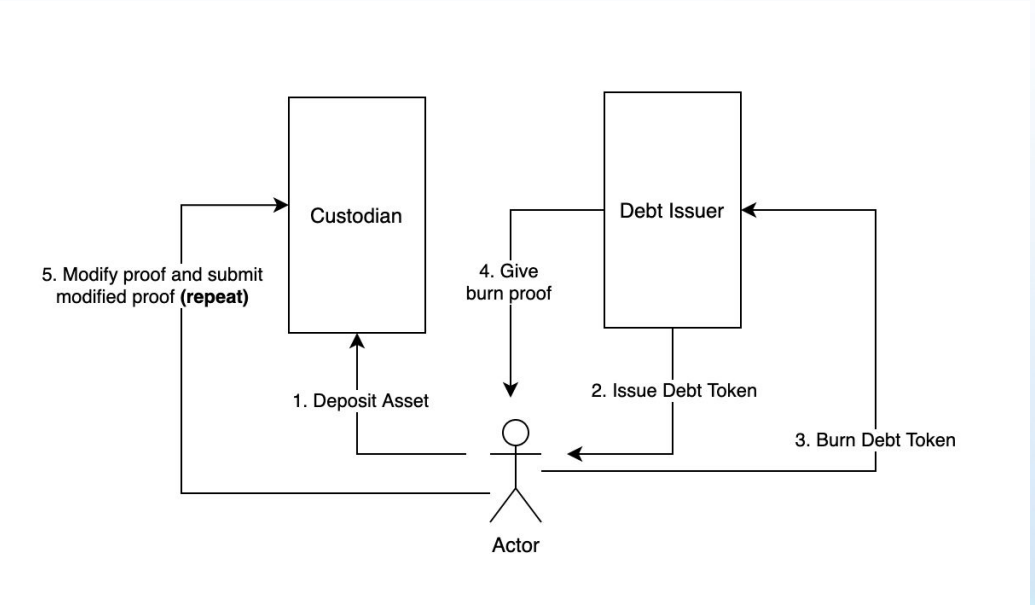


Custodian: Transaction Replay Attack

Depending on the custodian, proofs need be presented to withdraw assets

Attacker's goal:

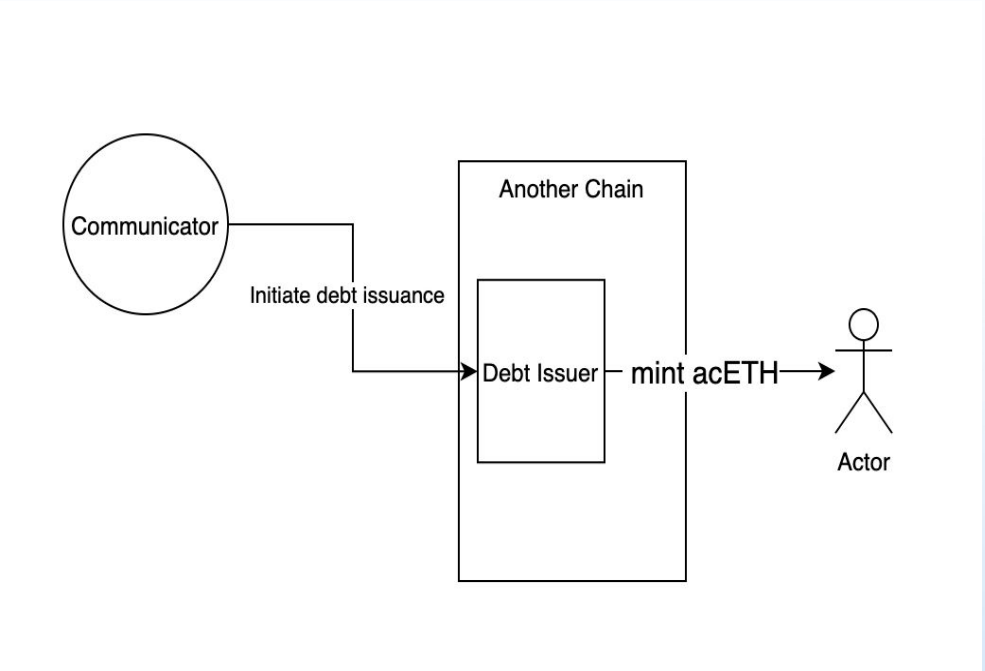
- Receive a valid burn proof
- Exploit the verifier and craft more valid proofs



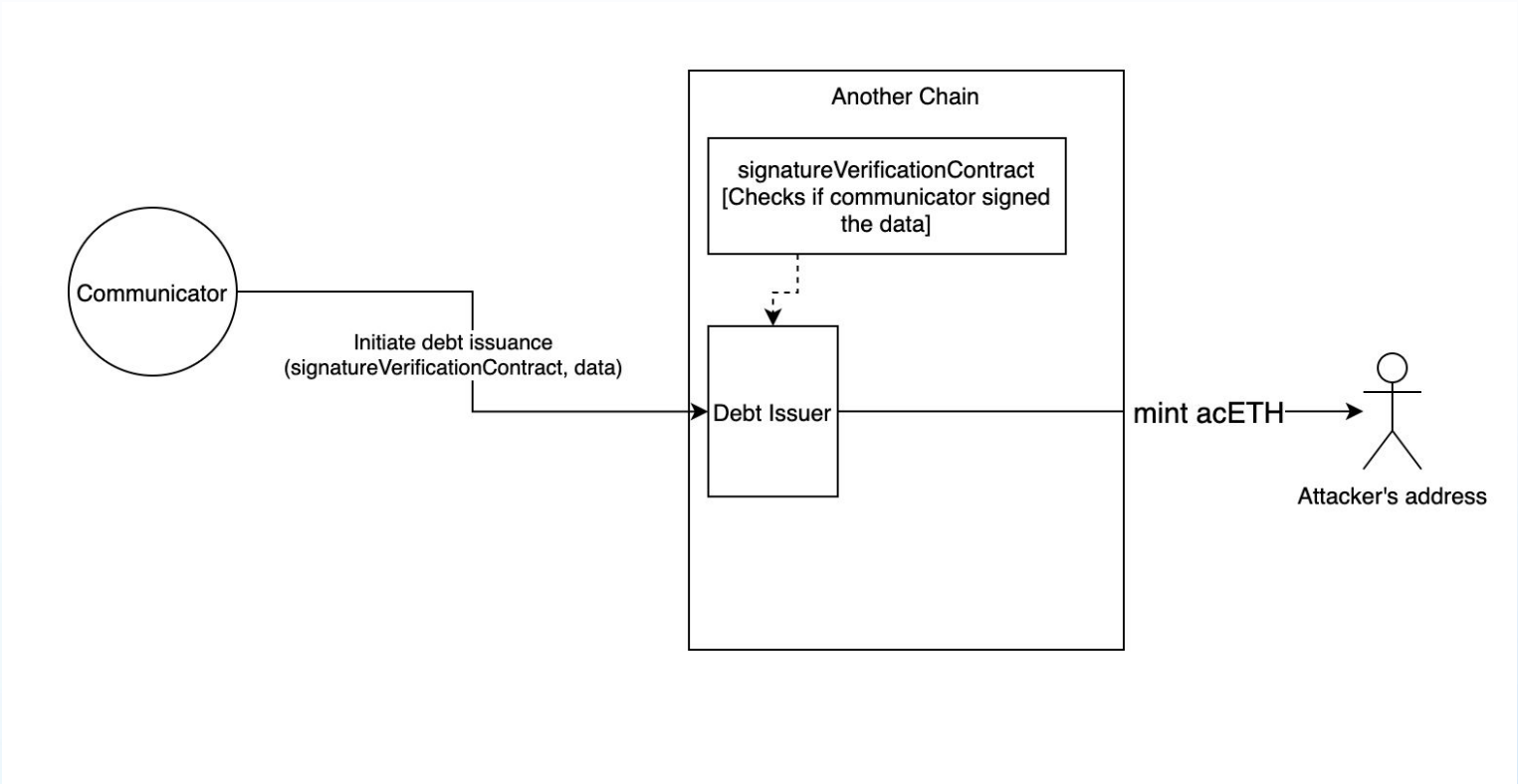
Debt Issuer: Fake Verification Attack

Attacker's goal:

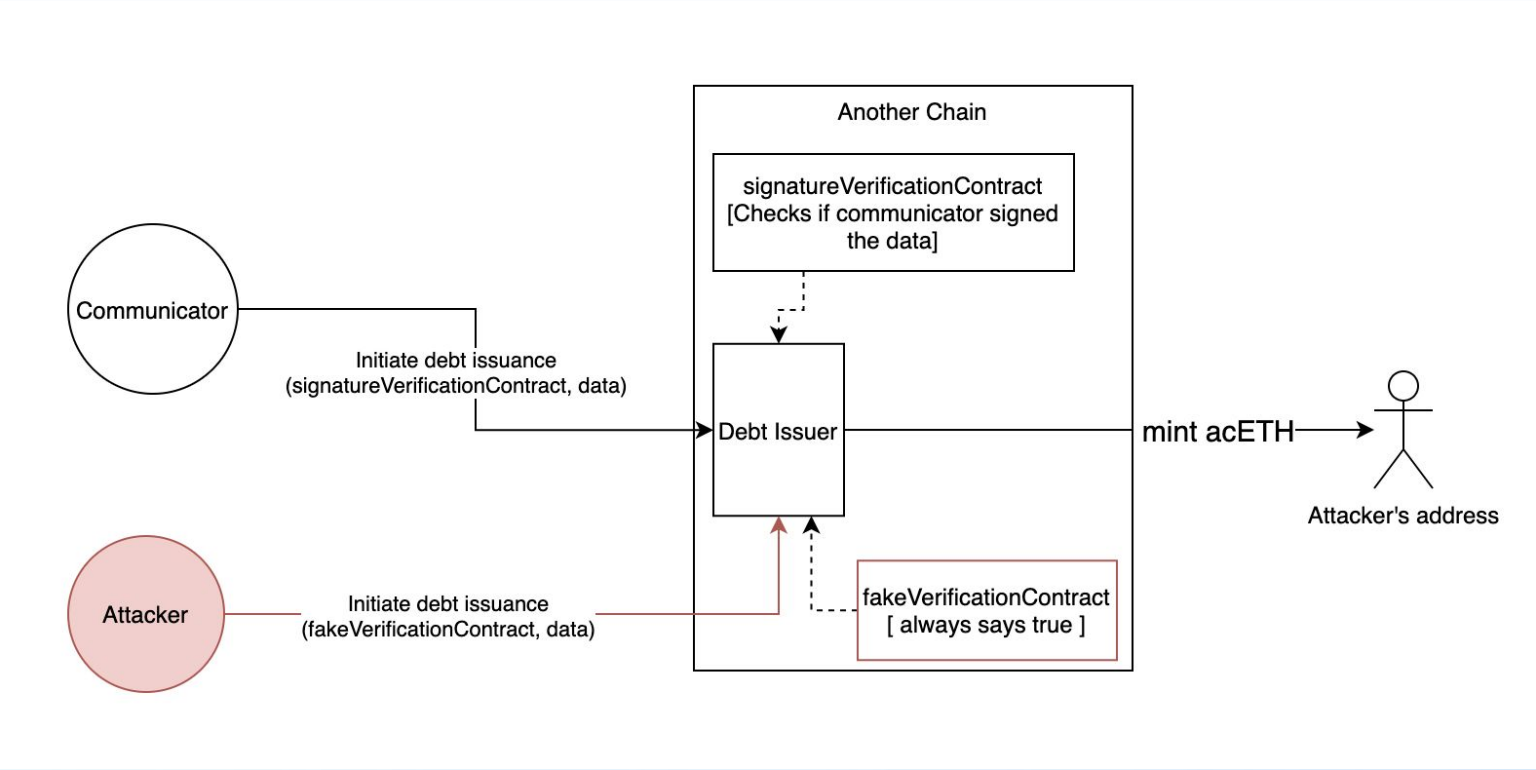
- Bypass signature verification
- Arbitrarily mint debt token



Debt Issuer: Fake Verification Attack



Debt Issuer: Fake Verification Attack



Communicator: Fake Events Attack

Attacker's goal:

- Trick the communicator into forwarding invalid messages
- Mint debt tokens

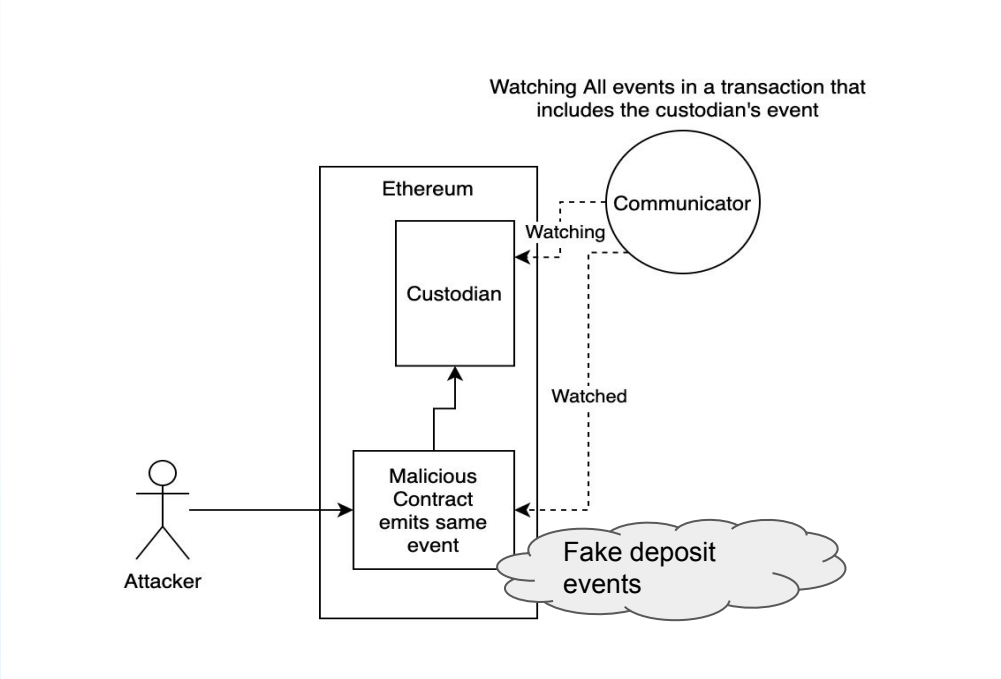
Think: polluting the data source of an oracle

Communicator: Fake Events Attack

Communicator has a bug where it validates only the first contract address of events

Deposit events are faked by crafting a malicious contract that emitted the same deposit event

This creates excessive debt tokens in the bridge

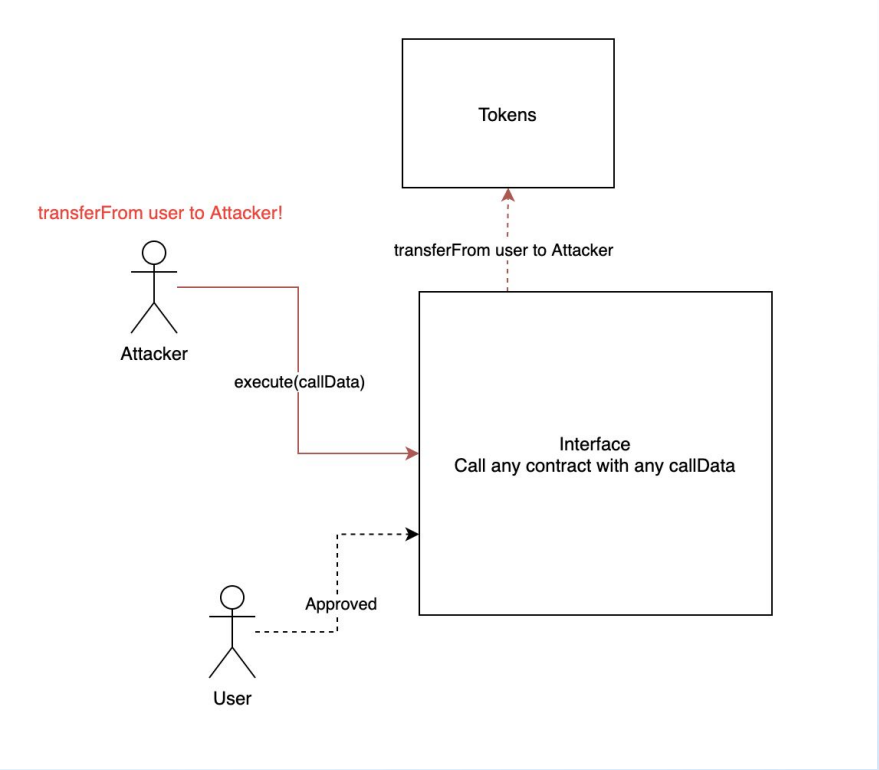


Interface: Infinite Approval Attack (Not Bridge-Specific)

Example:
Interface doesn’t sanitize inputs and could execute function calls to any contract with any data

Attacker crafts transferFrom

"revoke approval"

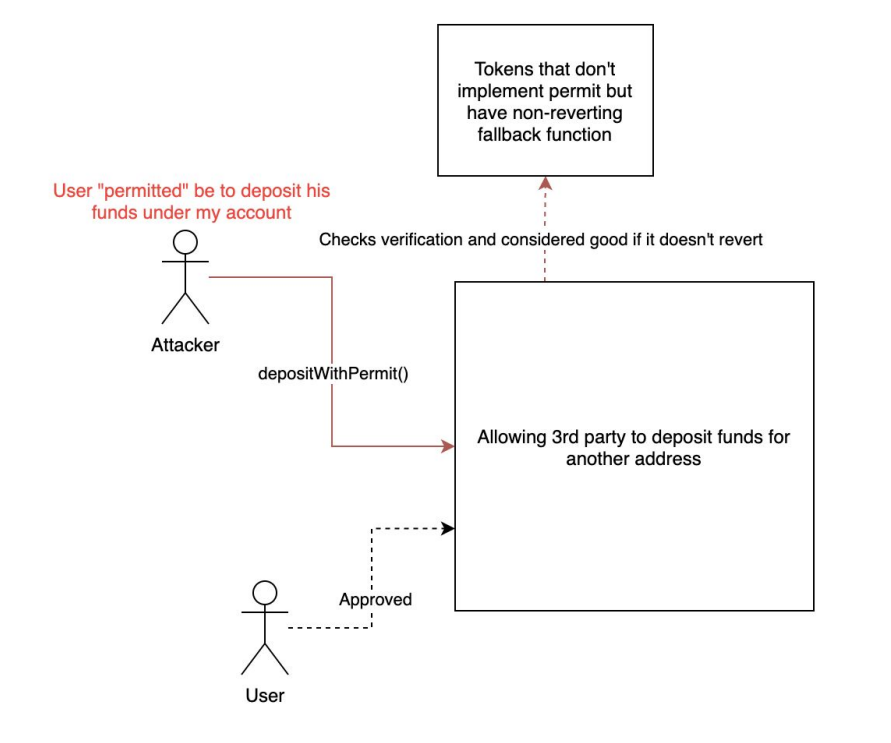


Interface: Permit Attack (Not Bridge-Specific)

Some issues may arise when there's an interface for users to deposit

Example:
Contracts that allow depositing funds from other addresses with permit / allowance

"revoke approval"



Network: 51% Attack

Launch a 51% attack on the L1 chain

Deposit assets to the custodian (receive the debt token)

Withdraw the debt token (receive assets and transfer them out)

Cancel the deposit transaction (the custodian doesn’t have the deposit)

1h of a 51% attack on Ethereum mainnet $1.5-2M

(*source: crypto51.app*)

Summary: a Lot Can Go Wrong With the Bridges

- The Custodian
  - Incorrect asset amount released with respect to the burnt tokens
  - Assets released despite the debt token has not been burnt
  - Asset transaction replay for a single burn transaction

- The Debt Issuer
  - Incorrect amount of debt issued with respect to the deposited assets
  - Debt token issued although the actual verification did not take place
  - Anybody can issue debt tokens

Summary: a Lot Can Go Wrong With the Bridges

- The Communicator
  - Issues debt tokens although no assets have been deposited
  - Issues no debt tokens although assets have been deposited
  - Accepts fraudulent messages from a fake custodian or a debt issuer
  - Does not relay messages
  - The source contract does not emit events upon deposit/withdrawal

- The Interface (could be fixed with "revoke approval")
  - Deposit from another account
  - Execute any calls from any contract

- The Network
  - 51% attack

Q&A / Thank You!

We are hiring!

Learn more about working at Quantstamp

Securing the Metaverse

info@quantstamp.com