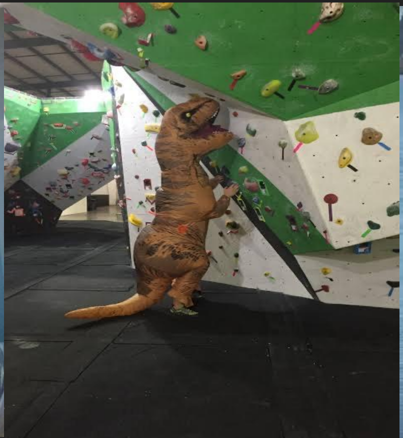
Hack The Bridge

whoami

- Anto Joseph
- Security Engineer @ Coinbase
- Speaker / Trainer @ Blackhat / Defcon / Nullcon / HITB/ HIP/ HackLu / PHdays / c0c0n….

Interested in distributed systems, machine learning , linux, radios and biotechnology



Disclaimer

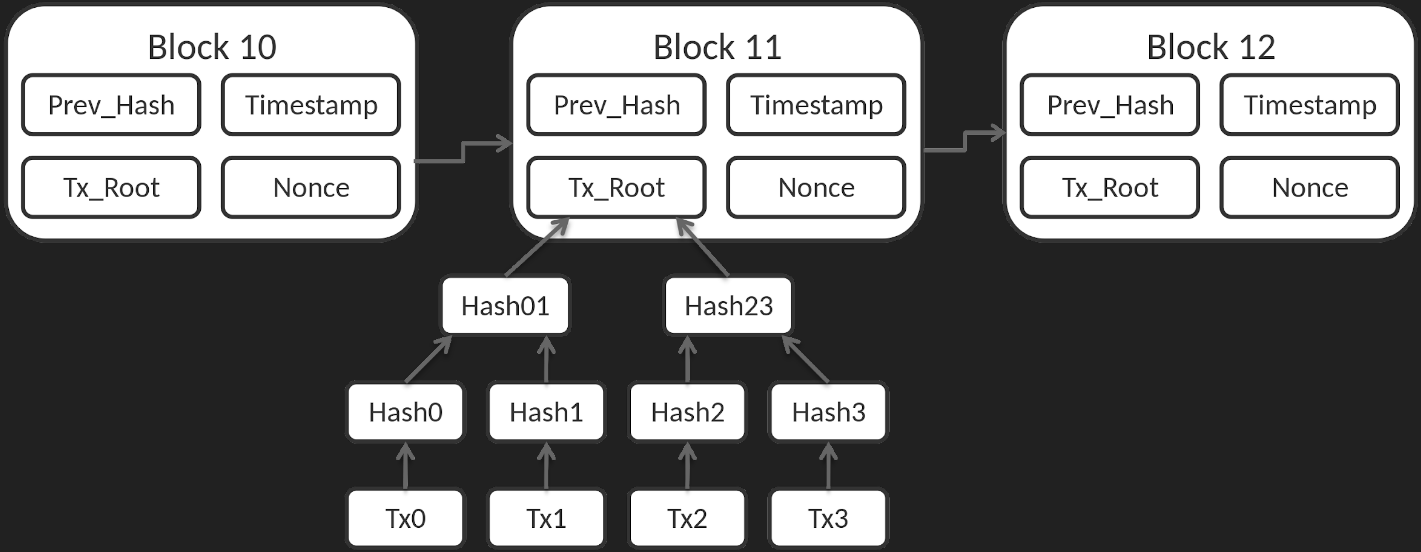
- This is NOT financial advice
- This is NOT legal advice
- These are purely my opinions/ comments and in no way reflect my employers
- This is purely meant for educational purposes!

Blockchains 101

At its most basic, a blockchain is a list of transactions that anyone can view and verify. The **Bitcoin blockchain**, for example, contains a record of every time someone sent or received bitcoin.

The **Ethereum blockchain** is a further evolution of the distributed ledger idea, Think of it as a powerful and highly flexible computing platform that allows coders to easily build all kinds of applications leveraging the blockchain.

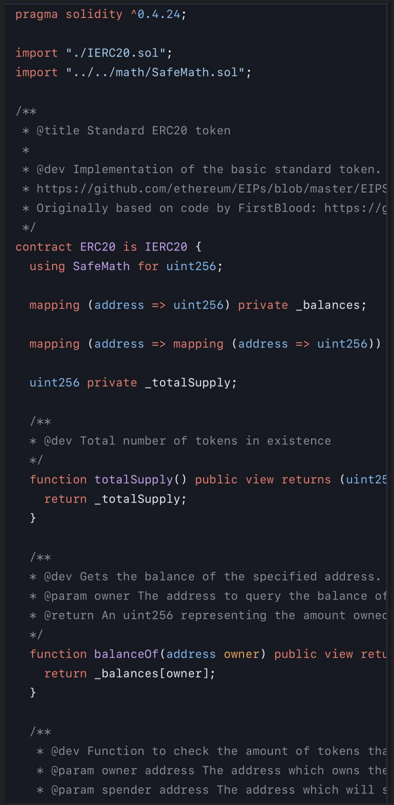
Blockchains 101



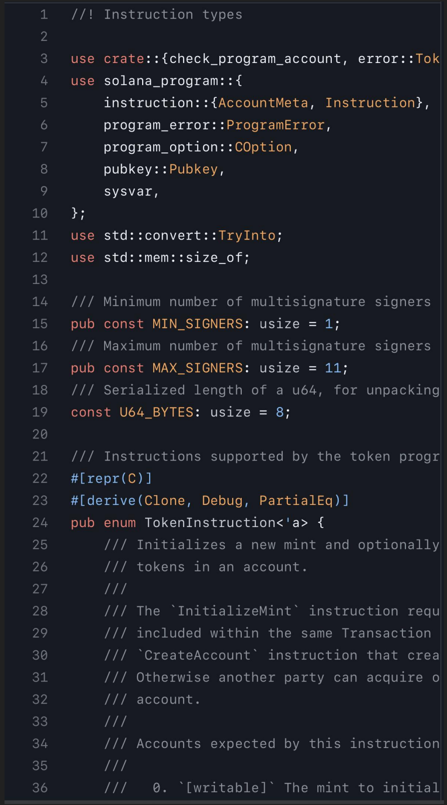
Smart Contracts

A smart contracts are(sometimes immutable) code running on a blockchain like Ethereum, Solana , Cosmos etc. They allow developers to build d(apps) that take advantage of blockchain security, reliability, and accessibility while offering sophisticated peer-to-peer functionality — everything from exchanges, loans and insurance to logistics and gaming.

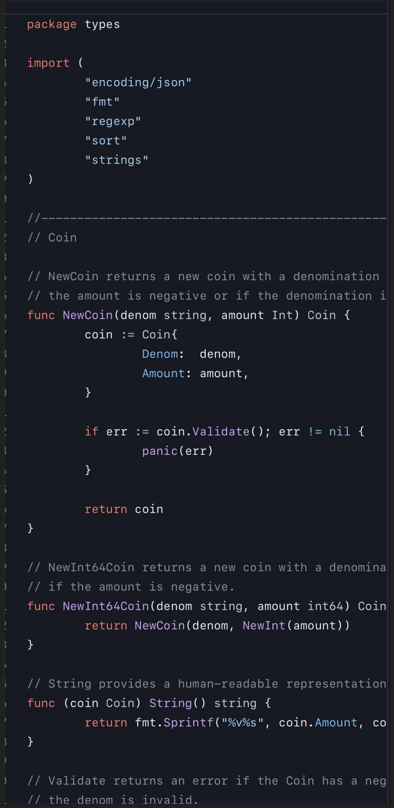
What do they look like?



```solidity
pragma solidity ^0.4.24;

import "./IERC20.sol";
import "../../math/SafeMath.sol";

/**
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * https://github.com/ethereum/EIPs/blob/master/EIPS
 * Originally based on code by FirstBlood: https://g
 */
contract ERC20 is IERC20 {
  using SafeMath for uint256;

  mapping (address => uint256) private _balances;

  mapping (address => mapping (address => uint256))

  uint256 private _totalSupply;

  /**
  * @dev Total number of tokens in existence
  */
  function totalSupply() public view returns (uint25
    return _totalSupply;
  }

  /**
  * @dev Gets the balance of the specified address.
  * @param owner The address to query the balance of
  * @return An uint256 representing the amount ownec
  */
  function balanceOf(address owner) public view retu
    return _balances[owner];
  }

  /**
   * @dev Function to check the amount of tokens tha
   * @param owner address The address which owns the
   * @param spender address The address which will s
```



```rust
//! Instruction types

use crate::{check_program_account, error::Tok
use solana_program::{
    instruction::{AccountMeta, Instruction},
    program_error::ProgramError,
    program_option::COption,
    pubkey::Pubkey,
    sysvar,
};
use std::convert::TryInto;
use std::mem::size_of;

/// Minimum number of multisignature signers
pub const MIN_SIGNERS: usize = 1;
/// Maximum number of multisignature signers
pub const MAX_SIGNERS: usize = 11;
/// Serialized length of a u64, for unpacking
const U64_BYTES: usize = 8;

/// Instructions supported by the token progr
#[repr(C)]
#[derive(Clone, Debug, PartialEq)]
pub enum TokenInstruction<'a> {
    /// Initializes a new mint and optionally
    /// tokens in an account.
    ///
    /// The `InitializeMint` instruction requ
    /// included within the same Transaction
    /// `CreateAccount` instruction that crea
    /// Otherwise another party can acquire o
    /// account.
    ///
    /// Accounts expected by this instruction
    ///
    ///   0. `[writable]` The mint to initial
```

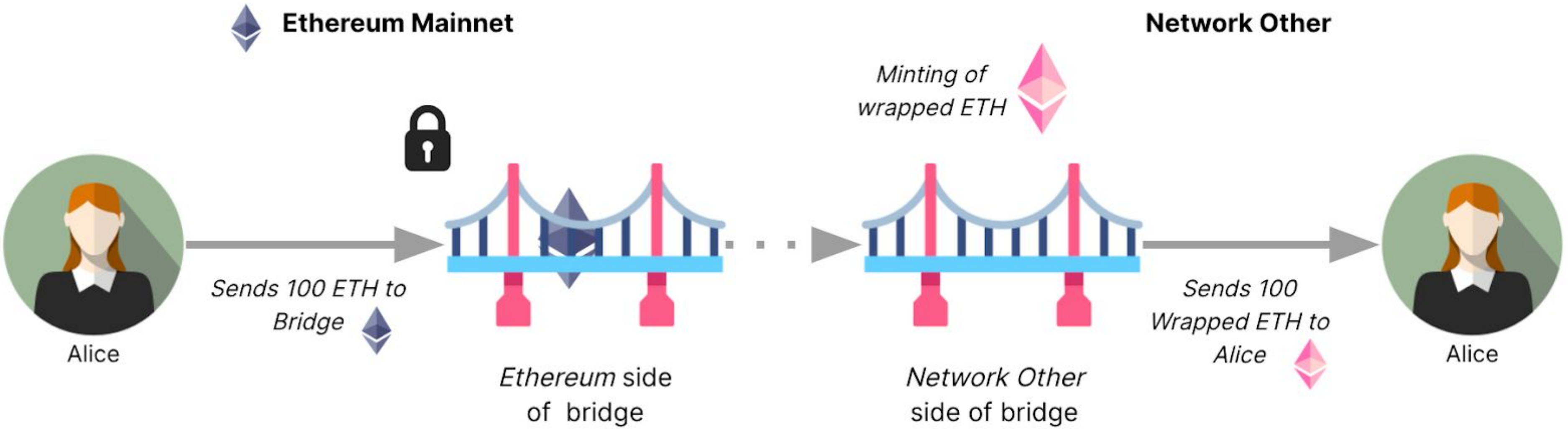


```go
package types

import (
        "encoding/json"
        "fmt"
        "regexp"
        "sort"
        "strings"
)

//-----------------------------------------------
// Coin

// NewCoin returns a new coin with a denomination
// the amount is negative or if the denomination i
func NewCoin(denom string, amount Int) Coin {
        coin := Coin{
                Denom:  denom,
                Amount: amount,
        }

        if err := coin.Validate(); err != nil {
                panic(err)
        }

        return coin
}

// NewInt64Coin returns a new coin with a denomina
// if the amount is negative.
func NewInt64Coin(denom string, amount int64) Coin
        return NewCoin(denom, NewInt(amount))
}

// String provides a human-readable representation
func (coin Coin) String() string {
        return fmt.Sprintf("%v%s", coin.Amount, co
}

// Validate returns an error if the Coin has a neg
// the denom is invalid.
```

Popular smart contract programming languages

- Solidity
- Rust
- Go

Allows users to transfer value from one chain to the other. if you have ether but want to use it on solana, you can do that through a bridge.



Why bridge?

- Reducing transaction fees
- speeding up transactions
- Utilizing dapps on different networks
- Better trade execution with larger liquidity pools
- NFT’s launching on different blockchains
- Better UX (think wallets / rpc nodes / even uptime)

The future of bridges

Cross chain bridges

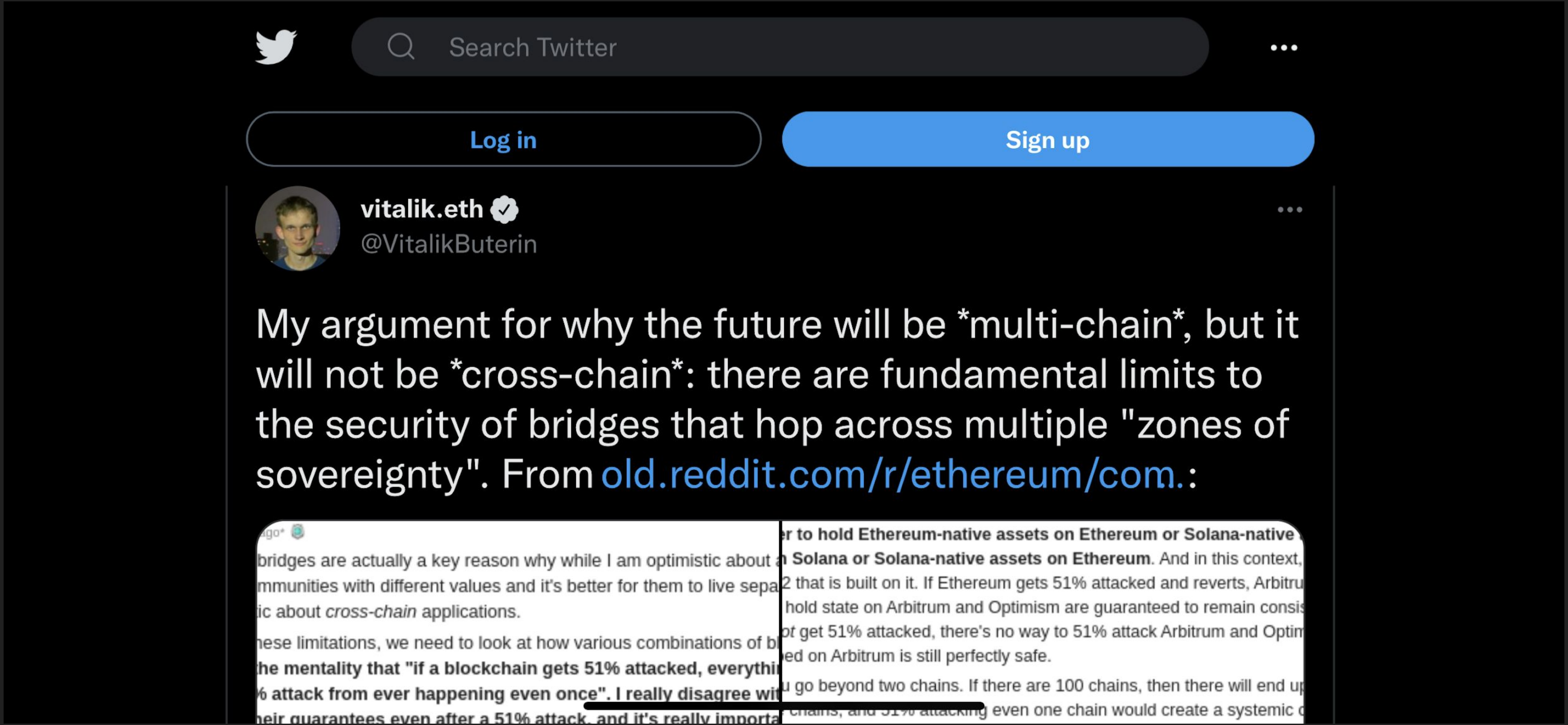
- Bridge across different kind of blockchains like ethereum to solana
  - Wormhole, Nomad

Multi Chain bridges

- Moving assets from l1 to l2 and back
  - (bridging from ethereum to optimism / arbitrum etc)
- Optimism and Arbitrum are layer 2 scaling solutions on ethereum using optimistic rollup technology
- Cosmos IBC
- Polkadot

Cross chain bridges
Multi chain bridges
JAKE-CLARK.TUMBLR
imgflip.com

What’s better? An opinion



vitalik.eth
@VitalikButerin

My argument for why the future will be *multi-chain*, but it will not be *cross-chain*: there are fundamental limits to the security of bridges that hop across multiple "zones of sovereignty". From old.reddit.com/r/ethereum/com.:

go*

bridges are actually a key reason why while I am optimistic about a
mmunities with different values and it's better for them to live sepa
ic about *cross-chain* applications.

nese limitations, we need to look at how various combinations of bl
he mentality that "if a blockchain gets 51% attacked, everythi
% attack from ever happening even once". I really disagree wit
heir guarantees even after a 51% attack, and it's really importa

r to hold Ethereum-native assets on Ethereum or Solana-native
Solana or Solana-native assets on Ethereum. And in this context,
2 that is built on it. If Ethereum gets 51% attacked and reverts, Arbitru
hold state on Arbitrum and Optimism are guaranteed to remain consis
ot get 51% attacked, there's no way to 51% attack Arbitrum and Optim
ed on Arbitrum is still perfectly safe.

u go beyond two chains. If there are 100 chains, then there will end up
chains, and 51% attacking even one chain would create a systemic c

Wait, whaaat?

A 51% attack (or majority attack)

refers to a potential attack on the integrity of a pow blockchain system in which a single entity controls more than half of the total hashing power of the network, potentially causing double spends / censorship etc

A Reorganization attack

refers to nodes receiving blocks from a new chain while the old chain continues to exist. In this case, the chain would be split and create a fork, or a duplicate version of the blockchain

The Longest Chain Rule

This rule kick in when forks appear. Each fork will have its own chain and miners can pick which one to apply their work on. But eventually the longer of the chains will be declared the winner – and all miners will apply their work onto that chain.

Scenario 1

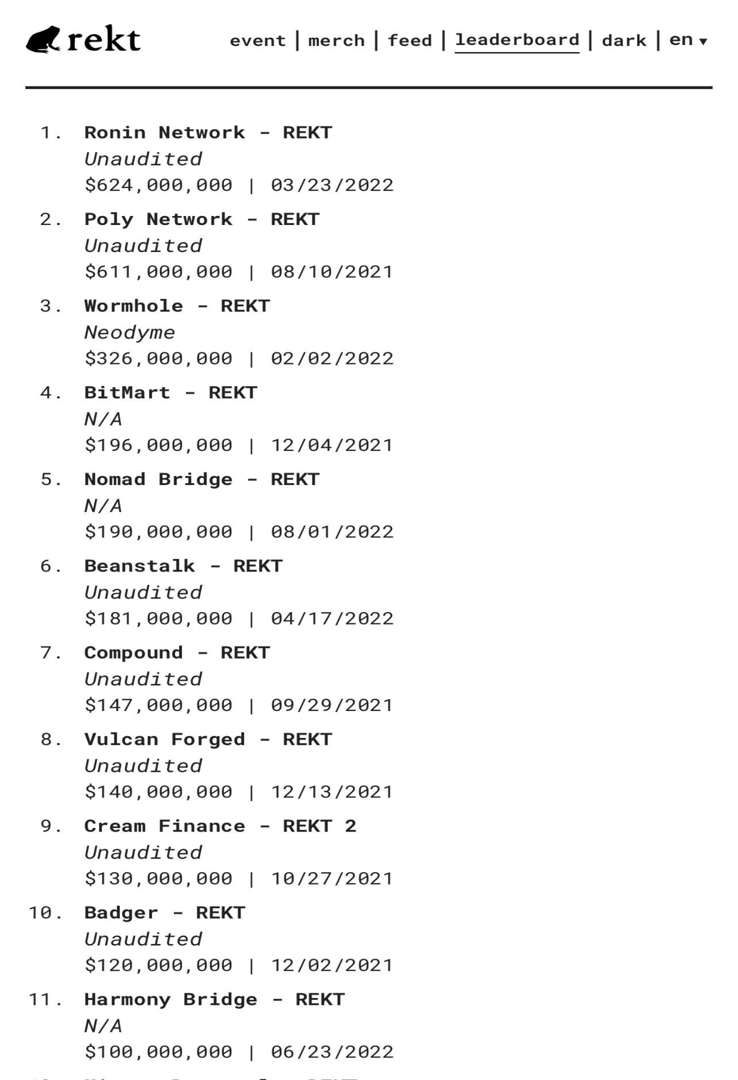
Imagine this

- Bridge 100 ETH from ethereum to solana
- Swap eth on solana , let’s call it sETH to USDC
- Ethereum goes through a reorg and the bridge transaction is no longer part of the canonical chain
- Now you have 100 ETH on ethereum and $150,000 USDC on solana (assuming 1ETH = $1500 USDC)

Cross chain bridges try to mitigate this by waiting for multiple block confirmations before they credit the deposit on the destination chain.

Block confirmations : number of blocks that were build on the block in question , as more blocks are build (more pow accumulated) , it becomes harder to reorg the chain. POW chains have probabilistic finality unlike certain POS chains.

Let’s look into cross chain bridges , they seem to have topped the leaderboard



rekt

event | merch | feed | leaderboard | dark | en ▾

1. **Ronin Network - REKT**
   Unaudited
   $624,000,000 | 03/23/2022
2. **Poly Network - REKT**
   Unaudited
   $611,000,000 | 08/10/2021
3. **Wormhole - REKT**
   Neodyme
   $326,000,000 | 02/02/2022
4. **BitMart - REKT**
   N/A
   $196,000,000 | 12/04/2021
5. **Nomad Bridge - REKT**
   N/A
   $190,000,000 | 08/01/2022
6. **Beanstalk - REKT**
   Unaudited
   $181,000,000 | 04/17/2022
7. **Compound - REKT**
   Unaudited
   $147,000,000 | 09/29/2021
8. **Vulcan Forged - REKT**
   Unaudited
   $140,000,000 | 12/13/2021
9. **Cream Finance - REKT 2**
   Unaudited
   $130,000,000 | 10/27/2021
10. **Badger - REKT**
    Unaudited
    $120,000,000 | 12/02/2021
11. **Harmony Bridge - REKT**
    N/A
    $100,000,000 | 06/23/2022

Rekt.news maintains a leaderboard of protocols including bridges that were **rekt**.

5 cross chain bridges made it to the top 11 category, (there is more in this leaderboard, it’s clipped for readability)

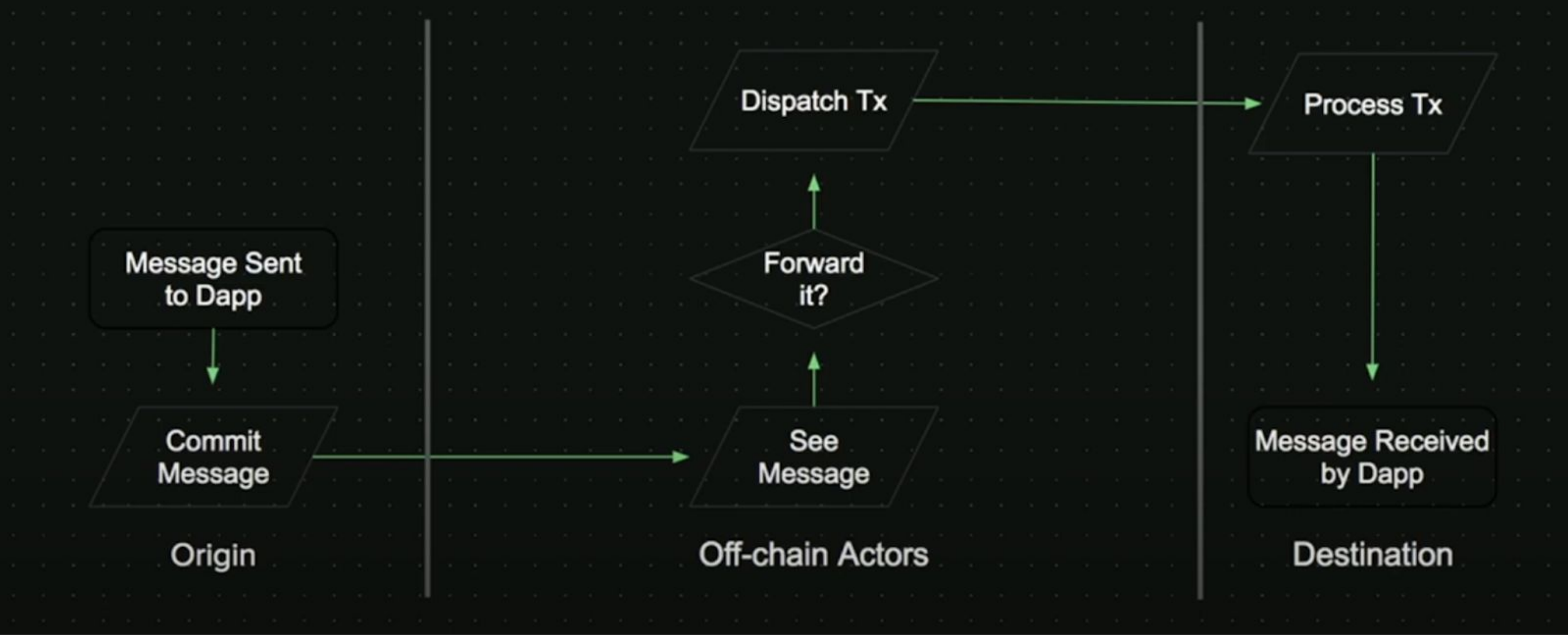
Visit the leaderboard at https://rekt.news

How do bridges work?

Since blockchain assets are often not compatible with one another, bridges create synthetic derivatives that represent an asset from another blockchain.

They have either a trusted or varying degrees of decentralised message passing techniques

Examples of trusted bridges include wbtc (custodied by bitGo) or bridging using crypto exchanges.



Simplified message passing bridge

Where them bugs at?

- Key management & cryptography
  - Issues with custody / implementation / operation of signing tx’s
    - Private key / Multisig key compromise
      - Axie infinity Ronin bridge
      - Harmony bridge
    - MPC keyshares compromise / cryptography bugs
      - Fire blocks MPC bug
    - Upgrade keys for smart contracts
    - Bugs in proof systems
      - Fraud/ fault proofs used by optimistic rollups
      - zkP’s used by zeroKnowledge rollups

Off Chain systems

- The relayer
  - Watches events on source chain and initiates a transaction on destination chain
  - Fake events or the compromise of these systems can lead to a loss
  - For some bridges, this is a group of nodes that validate the tx and reach consensus before relaying the tx to the target chain , often called guardians
- The validator
  - Validates signatures / blocks for cryptographic correctness
  - Merkel trees are commonly used to prove inclusion
  - Signature replay / verification bugs affect these systems
- The watcher
  - They can pause the bridge if they detect fraud in optimistic bridge designs
  - They have Permissioned watchers to prevent griefing attacks
  - Do not confuse optimistic bridges with Optimistic roll ups as the latter allows anyone to post a fraud proof , this is more inclusive than the above approach

Smart Contracts bridge contracts

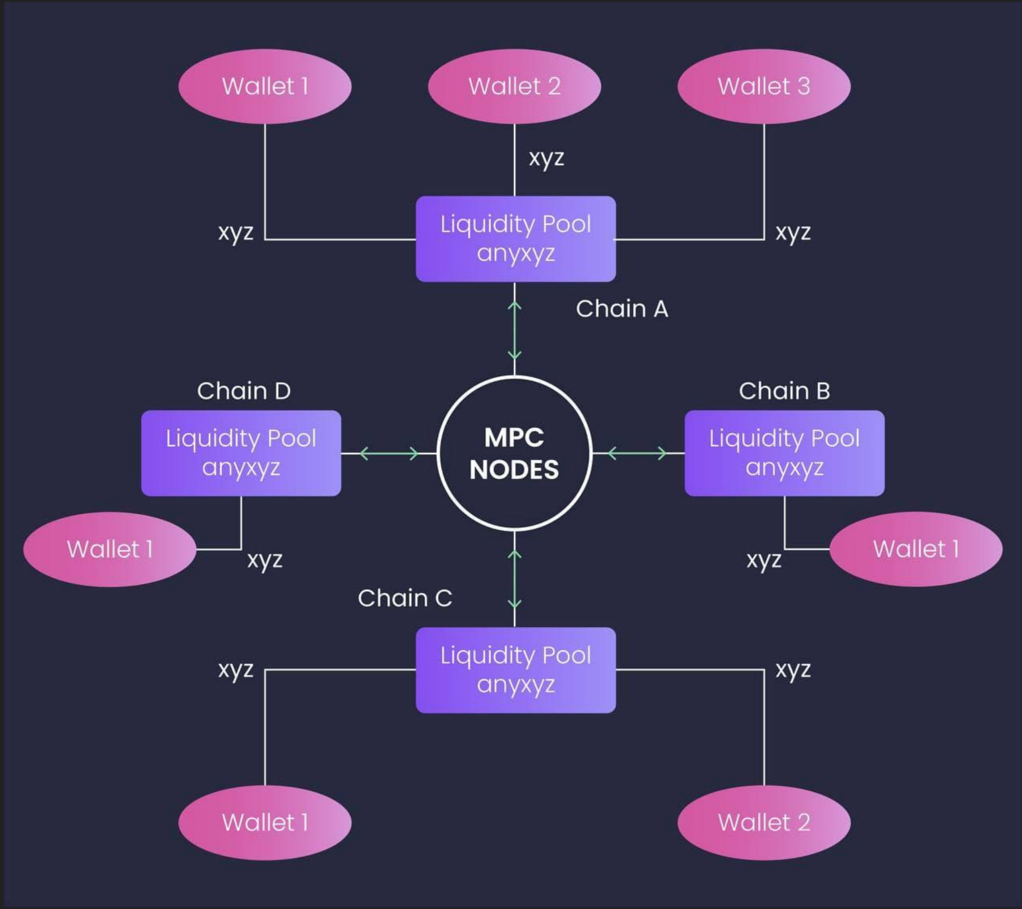
- Operational issues with smart contracts
  - Uninitialized proxy contracts
  - Wormhole bridge exploit
- Mint without deposit
  - tokenAddress.safeTransferFrom() doesn’t revert for EOA’s
  - Qubit finance hack
- Toxic privilege combination
  - Allowing user calls to be relayed via privileged contracts, thereby giving these actions admin privileges
  - Poly chain hack
- Lack of input validation
  - Using address returned by an Attacker supplied input for token swaps
  - Multichain hack
- Logic bugs in smart contract
  - Nomad bridge hack
  - We will explore this one in detail

Case Study

Multichain
(anyswap)
Bridge

MultiChain bridge

Multichain allows users to swap between supported chains. To do so, the router wraps the actual token with its “anyToken”. For example, the DAI token is wrapped as anyDAI. The wrapped token is used for internal accounting and when user “transfers” DAI from Ethereum to BSC, actually anyDAI is added on Multichain’s anyDAI BSC contract and burned on anyDAI Ethereum contract.



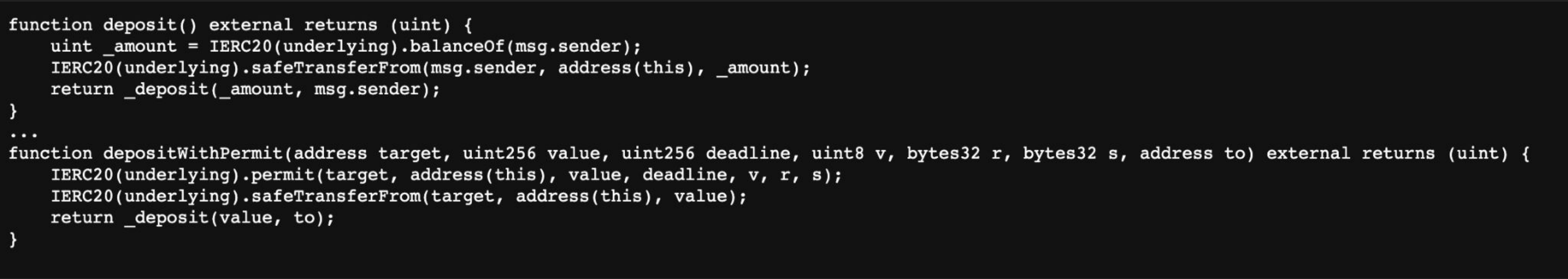
Erc-20 permit

Implementation of the ERC20 Permit extension allowing approvals to be made via signatures, as defined in EIP-2612.

Adds the permit method, which can be used to change an account’s ERC20 allowance by presenting a message signed by the account. By not relying on IERC20.approve, the token holder account doesn’t need to send a transaction, and thus is not required to hold Ether at all.

The Bug

Attacker controls the token parameter which is inturn used by the bridge contract to identify the underlying token. A malicious contract returns `**weth**` which doesn't have a permit function. Solidity calls the fallback function when the function that’s called on the contract can’t be triggered and as such , this successfully returns without errors. The last step of the exploit abuses unlimited token approvals by the dapp to drain funds from victim to attackers contract



```
function deposit() external returns (uint) {
    uint _amount = IERC20(underlying).balanceOf(msg.sender);
    IERC20(underlying).safeTransferFrom(msg.sender, address(this), _amount);
    return _deposit(_amount, msg.sender);
}
...
function depositWithPermit(address target, uint256 value, uint256 deadline, uint8 v, bytes32 r, bytes32 s, address to) external returns (uint) {
    IERC20(underlying).permit(target, address(this), value, deadline, v, r, s);
    IERC20(underlying).safeTransferFrom(target, address(this), value);
    return _deposit(value, to);
}
```

Why did the exploit work?

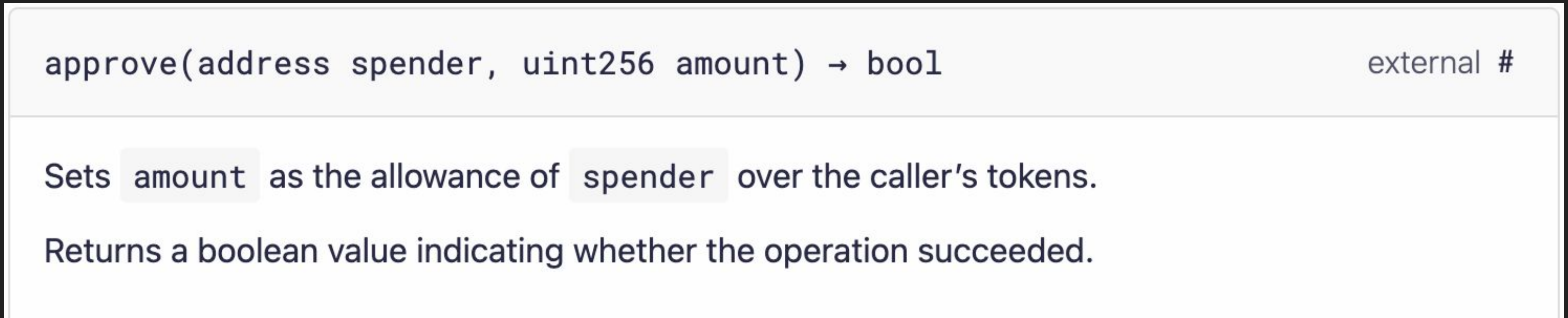
Do not trust user input without validation

Callers should not rely on permit reverting for arbitrary tokens.The call token.permit(...) never reverts for tokens that

- do not implement permit
- have a (non-reverting) fallback function.

Unlimited token approvals

- Smart contracts could get hacked and the approvals for this smart contract can be abused to drain funds from wallets that have approved this contract already



```
approve(address spender, uint256 amount) → bool
```

Sets `amount` as the allowance of `spender` over the caller's tokens.

Returns a boolean value indicating whether the operation succeeded.

Demo time

Case Study

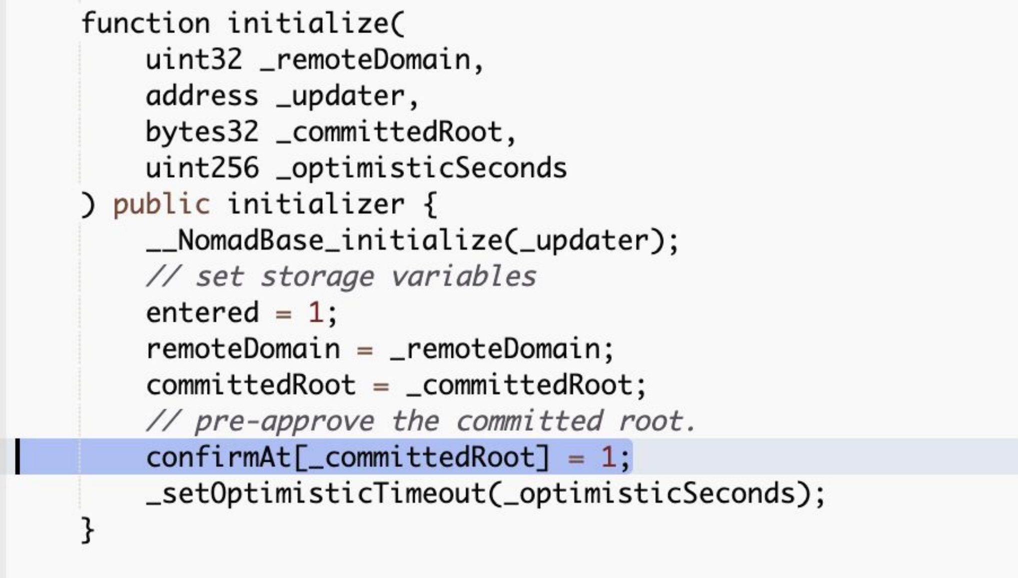
Nomad Bridge

Nomad Bridge Components

- Replica contract
  - Validates and stores messages

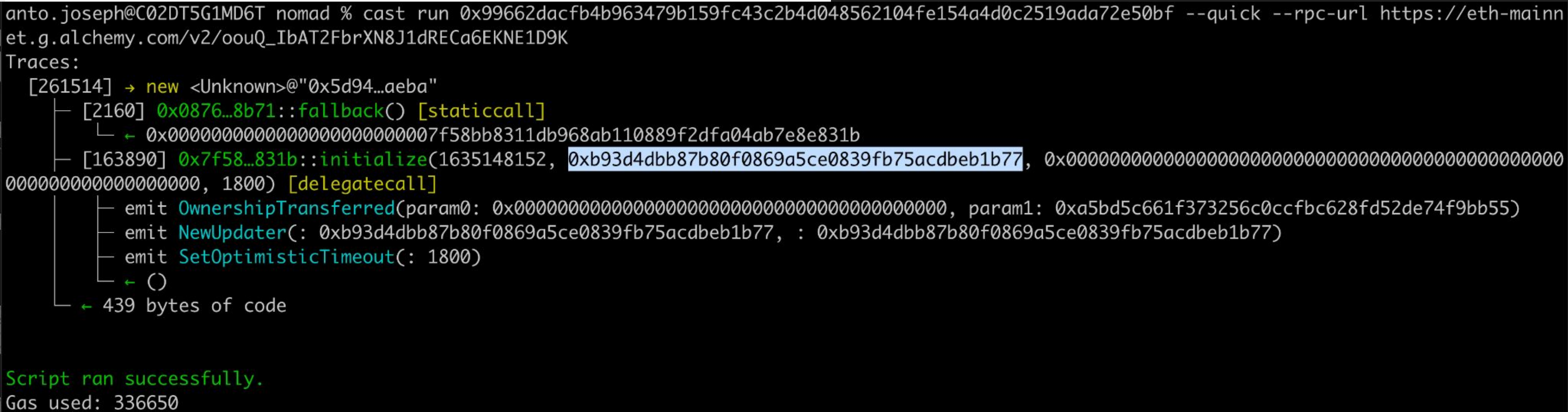
- BridgeRouters
  - Enables users to “send” tokens from Chain A to Chain B via a lock-and-mint mechanism.
    - NomadBridgeRouter Contract
      - Sender Bridge
    - ERC20 Router Contract
      - Receiver Bridge
- Off-Chain systems
  - Used for Message Passing between chains
  - Watcher nodes to report fraud

The setup



```
function initialize(
    uint32 _remoteDomain,
    address _updater,
    bytes32 _committedRoot,
    uint256 _optimisticSeconds
) public initializer {
    __NomadBase_initialize(_updater);
    // set storage variables
    entered = 1;
    remoteDomain = _remoteDomain;
    committedRoot = _committedRoot;
    // pre-approve the committed root.
    confirmAt[_committedRoot] = 1;
    _setOptimisticTimeout(_optimisticSeconds);
}
```

- confirmAt map sets _committedRoot to 1
- _committedRoot is set to 0 during initialization

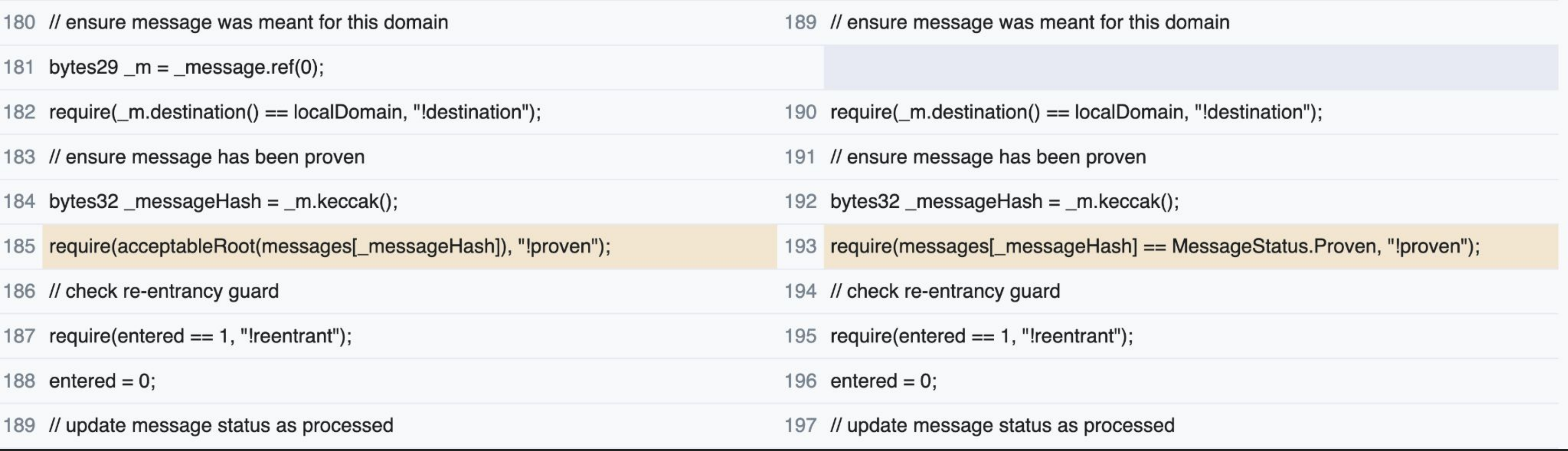


```
anto.joseph@C02DT5G1MD6T nomad % cast run 0x99662dacfb4b963479b159fc43c2b4d048562104fe154a4d0c2519ada72e50bf --quick --rpc-url https://eth-mainn
et.g.alchemy.com/v2/oouQ_IbAT2FbrXN8J1dRECa6EKNE1D9K
Traces:
  [261514] → new <Unknown>@"0x5d94…aeba"
    ├─ [2160] 0x0876…8b71::fallback() [staticcall]
    │   └─ ← 0x0000000000000000000000007f58bb8311db968ab110889f2dfa04ab7e8e831b
    ├─ [163890] 0x7f58…831b::initialize(1635148152, 0xb93d4dbb87b80f0869a5ce0839fb75acdbeb1b77, 0x0000000000000000000000000000000000000000000000
000000000000000000, 1800) [delegatecall]
    │   ├─ emit OwnershipTransferred(param0: 0x0000000000000000000000000000000000000000, param1: 0xa5bd5c661f373256c0ccfbc628fd52de74f9bb55)
    │   ├─ emit NewUpdater(: 0xb93d4dbb87b80f0869a5ce0839fb75acdbeb1b77, : 0xb93d4dbb87b80f0869a5ce0839fb75acdbeb1b77)
    │   ├─ emit SetOptimisticTimeout(: 1800)
    │   └─ ← ()
    └─ ← 439 bytes of code


Script ran successfully.
Gas used: 336650
```

The Bug

Replica contract was upgraded recently

The Diff

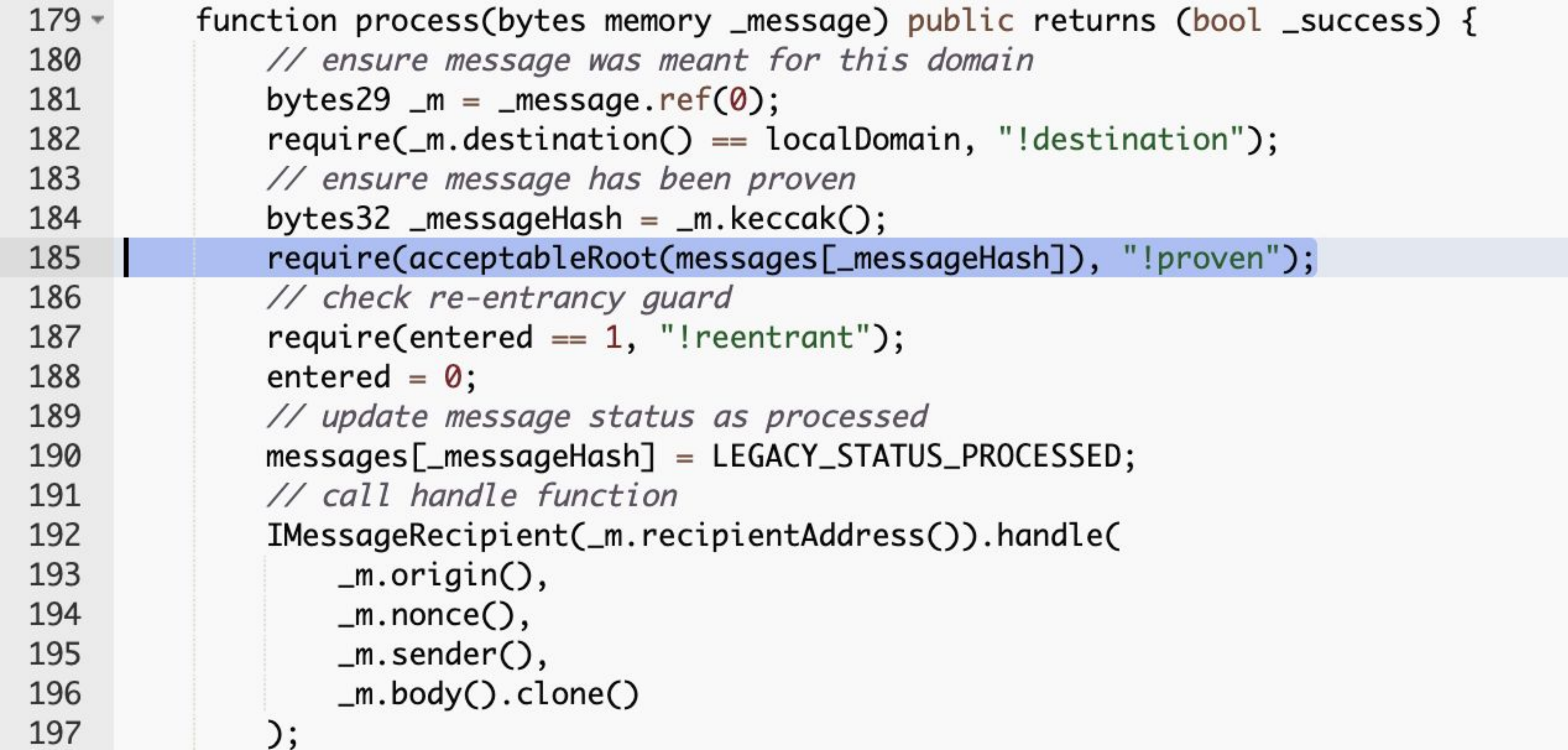


```
// ensure message was meant for this domain
bytes29 _m = _message.ref(0);
require(_m.destination() == localDomain, "!destination");
// ensure message has been proven
bytes32 _messageHash = _m.keccak();
require(acceptableRoot(messages[_messageHash]), "!proven");
// check re-entrancy guard
require(entered == 1, "!reentrant");
entered = 0;
// update message status as processed
```

```
// ensure message was meant for this domain

require(_m.destination() == localDomain, "!destination");
// ensure message has been proven
bytes32 _messageHash = _m.keccak();
require(messages[_messageHash] == MessageStatus.Proven, "!proven");
// check re-entrancy guard
require(entered == 1, "!reentrant");
entered = 0;
// update message status as processed
```

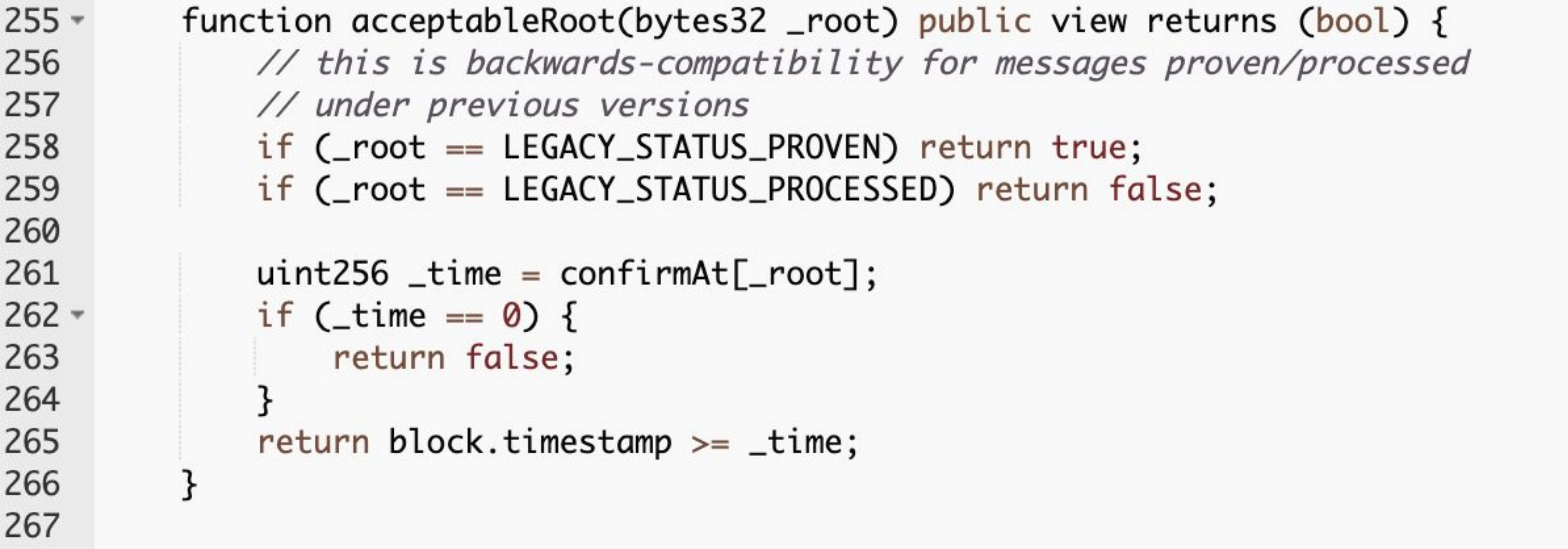
Verified messages can be submitted to the **process()** method.

- **process()** method internally calls **acceptableRoot()**
- “when called with an item that doesn't exist in a map , the map returns 0”



```
    function process(bytes memory _message) public returns (bool _success) {
        // ensure message was meant for this domain
        bytes29 _m = _message.ref(0);
        require(_m.destination() == localDomain, "!destination");
        // ensure message has been proven
        bytes32 _messageHash = _m.keccak();
        require(acceptableRoot(messages[_messageHash]), "!proven");
        // check re-entrancy guard
        require(entered == 1, "!reentrant");
        entered = 0;
        // update message status as processed
        messages[_messageHash] = LEGACY_STATUS_PROCESSED;
        // call handle function
        IMessageRecipient(_m.recipientAddress()).handle(
            _m.origin(),
            _m.nonce(),
            _m.sender(),
            _m.body().clone()
        );
```

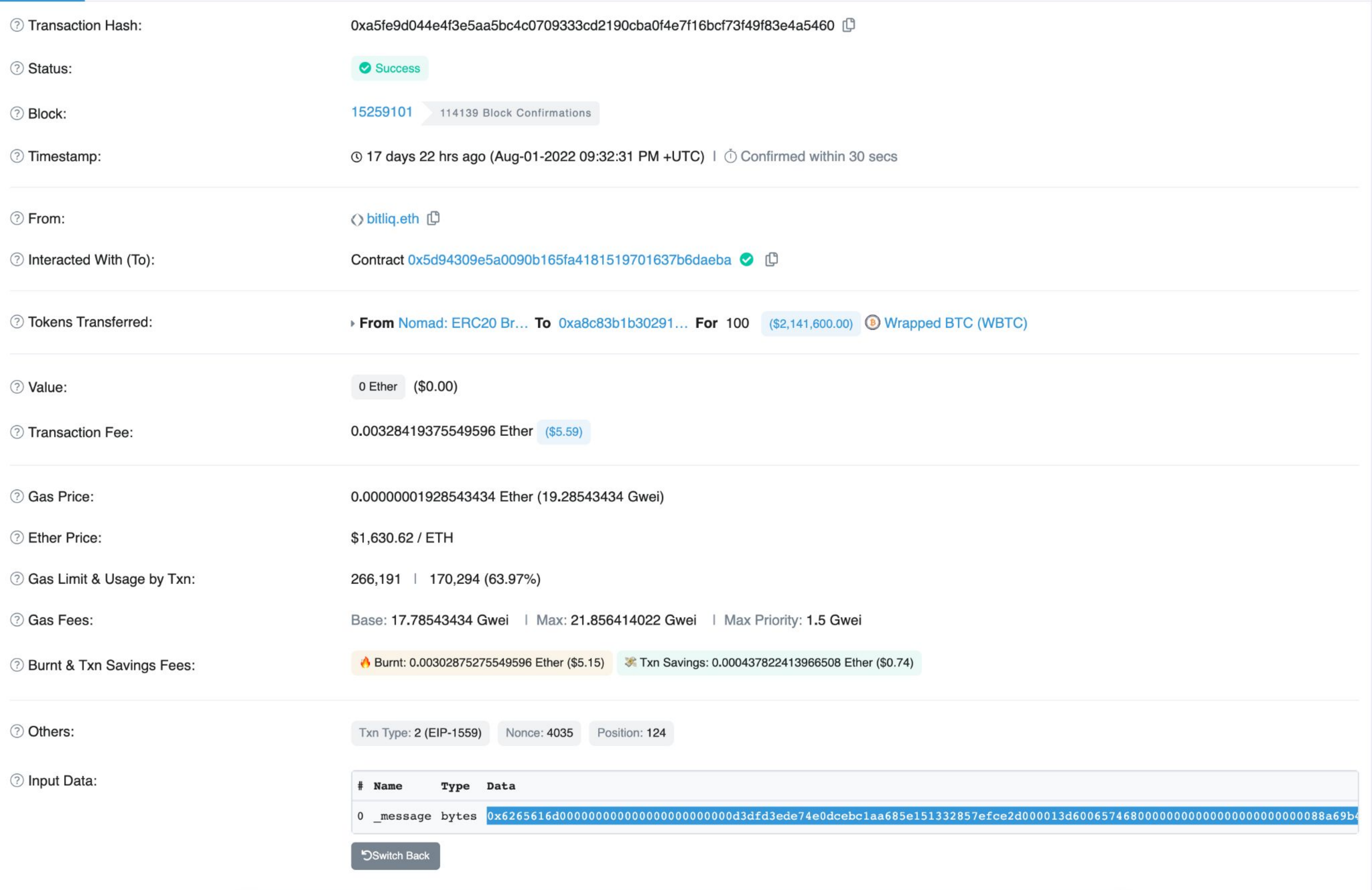
- **acceptableRoot()** references the **confirmAt** map
- require(acceptableRoot(messages[_messageHash]),!proven);
  - => require(acceptableRoot(0),"!proven");
  - => confirmAt[0] = 1



```
    function acceptableRoot(bytes32 _root) public view returns (bool) {
        // this is backwards-compatibility for messages proven/processed
        // under previous versions
        if (_root == LEGACY_STATUS_PROVEN) return true;
        if (_root == LEGACY_STATUS_PROCESSED) return false;

        uint256 _time = confirmAt[_root];
        if (_time == 0) {
            return false;
        }
        return block.timestamp >= _time;
    }
```

The Exploit

- Easy way
  - Copy hack txn , search and replace recipient addr
  - https://etherscan.io/tx/0xa5fe9d044e4f3e5aa5bc4c0709333cd2190cba0f4e7f16bcf73f49f83e4a5460
- Exploitorr way
  - Craft token transfer request struct yourself



Transaction Hash:	0xa5fe9d044e4f3e5aa5bc4c0709333cd2190cba0f4e7f16bcf73f49f83e4a5460
Status:	✓ Success
Block:	15259101 114139 Block Confirmations
Timestamp:	17 days 22 hrs ago (Aug-01-2022 09:32:31 PM +UTC) | Confirmed within 30 secs
From:	bitliq.eth
Interacted With (To):	Contract 0x5d94309e5a0090b165fa4181519701637b6daeba ✓
Tokens Transferred:	From Nomad: ERC20 Br… To 0xa8c83b1b30291… For 100 ($2,141,600.00) Wrapped BTC (WBTC)
Value:	0 Ether ($0.00)
Transaction Fee:	0.00328419375549596 Ether ($5.59)
Gas Price:	0.00000001928543434 Ether (19.28543434 Gwei)
Ether Price:	$1,630.62 / ETH
Gas Limit & Usage by Txn:	266,191 | 170,294 (63.97%)
Gas Fees:	Base: 17.78543434 Gwei | Max: 21.856414022 Gwei | Max Priority: 1.5 Gwei
Burnt & Txn Savings Fees:	🔥 Burnt: 0.00302875275549596 Ether ($5.15) Txn Savings: 0.000437822413966508 Ether ($0.74)
Others:	Txn Type: 2 (EIP-1559) Nonce: 4035 Position: 124
Input Data:	

#	Name	Type	Data
0	_message	bytes	0x6265616d000000000000000000000000d3dfd3ede74e0dcebc1aa685e151332857efce2d000013d60065746800000000000000000000000088a69b4

Demo time

For developers

- Smart Contract Security Verification Standard
- Use safe audited libraries (OpenZepplin)
- Get audits, even better if you have a product security team
- Minor updates to a smart contracts can wreak havoc
- Write tests , invariant testing is especially useful
- Fuzz your contracts (use foundry , echidna)
- Have a meaningful bug bounty program
- Have a monitoring program, they might help
- Test your projects end to end including deployment/ initialisation

For whitehats

- Bridges are an attractive target because they custody lots of assets
- Most protocols including bridges have great bug bounty programs
- They are important in growing the crypto ecosystem, why not hack on systems where you can clearly demonstrate impact and get paid for it (generously , something upto 10% of the value secured) while securing the future of money for the masses ?

- Tools that may help you in the process
  - Foundry
  - Tenderly debugger
  - Echidna/ Certora
  - Learning resources: immunify write ups , BlockThreat Newsletter
  - CTF : capture the ether, crypto zombies , ethernaught , paradigm ctf

Questions?

Tweet @ pwnfooo
Telegram @ blocksek

References

- https://docs.nomad.xyz/the-nomad-protocol/cross-chain-messaging/lifecycle-of-a-message
- https://rekt.news/
- https://blog.coinbase.com/nomad-bridge-incident-analysis-899b425b0f34?gi=3d3d942484d8
- https://blog.coinbase.com/what-are-bridges-bridge-basics-facts-and-stats-8dd9449066a0
- https://medium.com/zengo/without-permit-multichains-exploit-explained-8417e8c1639b
- https://media.dedaub.com/phantom-functions-and-the-billion-dollar-no-op-c56f062ae49
- https://docs.multichain.org/getting-started/how-it-works/cross-chain-router
- https://github.com/SunWeb3Sec/
- https://gist.github.com/yoavw/160d5dadb37fbd0d1ec04e69951edafd
- https://gist.github.com/yajin/0f1a7acfd54adce02422298a1dea8d89
- https://docs.openzeppelin.com/contracts/4.x/api/token/ERC20
- https://blog.trailofbits.com/2022/04/18/the-frozen-heart-vulnerability-in-plonk/
- https://www.fireblocks.com/blog/vulnerabilities-discovered-and-patched-in-legacy-mpc-algorithm-fireblocks-urges-move-to-mpc-cmp/

Thank you Nullcon for organising a fantastic event