Quantstamp

Chromia - Postchain

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	Bridge
Timeline	2024-08-26 through 2024-09-13
Language	Rell, Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Postchain EIF Docs
Source Code	• https://gitlab.com/chromaway/postchain-eif #f2272ca
Auditors	• Ibrahim Abouzied Auditing Engineer • Roman Rohleder Senior Auditing Engineer • Ed Zulkoski Senior Auditing Engineer

Documentation quality	Medium
Test quality	Low
Total Findings	16 Fixed: 11 Acknowledged: 4 Mitigated: 1
High severity findings	1 Fixed: 1
Medium severity findings	5 Fixed: 4 Acknowledged: 1
Low severity findings	8 Fixed: 5 Acknowledged: 2 Mitigated: 1
Undetermined severity findings	0
Informational findings	2 Fixed: 1 Acknowledged: 1

Summary of Findings

The Chromia Token Bridge allows users to transfer tokens between EVM-compatible chains and Chromia using a Lock & Mint model for L1 to L2 transfers and a Burn & Release model for L2 to L1. Chromia monitors token deposits for L1 to L2 transfers, while L2 to L1 transfers require submitting a Merkle Proof on L1 to confirm tokens were burned on L2. Token withdrawals include a holding period, allowing the contract owner to flag and deny suspicious transactions before execution.

A key feature is the *mass exit*, which the bridge owner can trigger if Chromia Validators are compromised. During a mass exit, normal bridging is halted, and users can only withdraw their token balances based on a Chromia node snapshot. Pending withdrawals are allowed if they predate the *mass exit block*—the last valid block before the compromise, as determined by the owner.

While the core functionality of the EVM bridge contracts was robust, the mass exit feature suffered from many issues. In the ideal scenario, the integrity of the validators is maintained and a mass exit is never triggered. However, should a mass exit occur, all parties have a means of draining the contract. The compromised validators will still be able to forge blocks (CHR-4). Given that the withdrawal snapshots are not validated against the mass exit block, any user can forge their snapshots as well (CHR-1). If the contract is ever paused as a precautionary measure, the token bridge owner can withdraw all tokens for themselves (CHR-2). The ability for the token bridge owner to trigger and revert a mass exit at will allows them to extract the contract's liquidity (CHR-3).

Regarding the Rell files, the current mapping of snapshots may send funds to the wrong chain in a mass exit for contracts with the same address (CHR-6). The signature format for linking EVM accounts to FT4 accounts allows for an attacker to steal a deposit if they get ahold of the signature (CHR-7). The attack surface can also be further reduced by more strictly determining the origins of deposits (CHR-14).

The Chromia team was in communication and helped answer all of our questions throughout the audit. The documentation was helpful, though the codebase could benefit from more in-line comments and NatSpec. The test suite was missing tests on the new validator contracts as well as the snapshot withdrawal logic. We were unable to assess the code coverage of the rell test suite. We strongly recommend that tests be written for all features before deployment.

Fix-Review Update: The Chromia team has addressed or acknowledged all of the listed vulnerabilities.

ID	DESCRIPTION	SEVERITY	STATUS
CHR-1	**Anyone Can Forge a Withdrawal in `withdrawBySnapshot()`**	• High	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
CHR-2	The `TokenBridge` Owner Can Drain All Funds if the Contract Is Paused	● Medium	Fixed
CHR-3	The `TokenBridge` Owner Can Double-Spend Withdrawals	● Medium	Fixed
CHR-4	Malicious Validators Can Forge Old Blocks During a Mass Exit	● Medium	Fixed
CHR-5	Withdrawal by Snapshot Can Be Double-Spent Across Different Bridges	● Medium	Acknowledged
CHR-6	Signature Format for Linking EVM Accounts May Lead to Stolen Deposits	● Medium	Fixed
CHR-7	The `TokenBridge` Owner Can Lock All Funds	● Low	Fixed
CHR-8	`triggerMassExit()` Does Not Scale with `withdrawOffset`	● Low	Acknowledged
CHR-9	`TokenMinter` May Be Unable to Transfer Roles	● Low	Fixed
CHR-10	The `dayLimit` Can Quickly Increase Despite a Recent Update	● Low	Fixed
CHR-11	Withdrawals May Accidentally Resume Despite Suspicious Validators	● Low	Fixed
CHR-12	`setBlockchainRid()` Is Front-Runnable by Anyone	● Low	Fixed
CHR-13	`process_deposit_erc20_event` Incorrectly Assumes Deposit Origin	● Low	Mitigated
CHR-14	An `evm_block` May Be Associated with Multiple Hashes	● Low	Acknowledged
CHR-15	`postponeMassExit()` May Create a Deficit	● Informational	Fixed
CHR-16	Potential Storage Collisions	● Informational	Acknowledged

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Disclaimer
Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
   1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

Files Included

Repo: https://gitlab.com/chromaway/postchain-eif(f2272ca63e4f8ccca7e866f4d209693780bcef0a)
Files:

- `postchain-eif-contracts/*`
- `postchain-eif-rell/rell/src/*`

Files Excluded

Repo: https://gitlab.com/chromaway/postchain-eif(ddae59f1b83a4a4ad3d789582278f1d251a80167)
Files:

- `postchain-eif-contracts/contracts/Data.sol`
- `postchain-eif-contracts/contracts/Validator.sol`
- `postchain-eif-contracts/contracts/utils/cryptography/*`
- `postchain-eif-contracts/contracts/token/AliceToken.sol`
- `postchain-eif-contracts/contracts/token/TestToken.sol`
- `postchain-eif-contracts/contracts/anchoring/Anchoring.sol`
- `postchain-eif-rell/rell/src/transaction_submitter/*`

Key Actors And Their Capabilities

There are a several key actors in the Chromia bridge:

1. **The Chromia Validators**: Validators authenticate the blocks that users submit along with their withdrawals. In order for validators to act maliciously, a supermajority of 2/3rds of the validators would have to be compromised. Documentation further details that "validators are required to go through identity verification, and required to have a stake in CHR tokens, and are economically incentivized to act honestly." If validators are compromised, a mass exit can be triggered, allowing users to reclaim their funds according to the last valid snapshot (assigned by the token bridge owner). Additionally, validators have the power to pause the bridge contract.
2. **The `TokenBridge` Owner**: The token bridge owner has a few responsibilities. Aside from initializing the contract and configuring the supported tokens, they can also flag suspicious transactions, blocking them from being withdrawn. Their responsibility expands once a mass exit is involved, requiring them to assign the mass exit block. They also have the power to cancel a mass exit, resuming normal operations, as well as withdraw the contract balance 90 days after the start of a mass exit.
3. **The `TokenMinter` Owner**: Their core responsibility is to set the limit on how many tokens can be withdrawn across the bridge for Chromia Token. They also have the power to change the minter for the Chromia token, authorizing the address to mint any amount of Chromia tokens.
4. **The `Validator` Owner**: They can update the list of validators.

Given the spread of responsibilities, we advise that all of these roles are held by different addresses/entities. While validators are incentivized to act honestly, it is not clear that this holds for the `TokenBridge` owner. Users should be aware that the `TokenBridge` owner is a trusted role. Ideally, the role is held by a DAO.

While many protocols make use of trusted roles, we have listed vulnerabilities where a trusted address can drain the contract (CHR-1, CHR-2, CHR-4, CHR-6). Addressing these issues can help reduce the attack surface and encourage parties to act honestly.

A full list of all guarded functions can be found below:

The Owner of `ChromiaTokenBridge` can call:

- `setMinter()`: Assigns the `tokenMinter` contract.

The Owner of `TokenBridge` can call:

- `setBlockchainRid()`: Sets the blockchain RID used in the contract.
- `unpause()`: Unpauses the contract, resuming normal operations.
- `allowToken()`: Adds a token to the list of allowed tokens for transactions.

- `postponeMassExit()` : Undoes a mass exit, halting the mass exit process and resumes normal operations.
- `pendingWithdraw()` : Marks a withdrawable request as pending. Used to flag a suspicious request. The withdrawal cannot be completed unless the status is returned to `Withdrawable` .
- `unpendingWithdraw()` : Returns a suspicious withdrawal back to status `Withdrawable` .
- `fund()` : Transfers tokens from their address to the contract in case the bridge needs more liquidity.

The Validator of `TokenBridge` can call:

- `pause` : Pauses the contract, stopping certain functions from being executed.

The Owner of `TokenBridgeWithSnapshotWithdraw` can call:

- `triggerMassExit()` : Triggers the mass exit state and assigns the mass exit block from which to perform withdrawals by snapshot.
- `emergencyWithdraw()` : Sends the entire contract balance to any address. *only callable 90 days after a mass exit is triggered.*

The Owner of `TokenMinter` can call:

- `setDayLimit()` : Sets a new daily withdrawal limit, with an optional 2-week delay if increasing the limit.
- `finishSetDayLimit()` : Finalizes the pending day limit after the 2-week delay, applying the new limit.
- `transferMintRole()` : Begins the process of transferring the minter role to a new address after a 2-week delay.
- `finishTransferMintRole()` : Finalizes the transfer of the minter role to a new address after the 2-week delay.
- `transferOwnership()` : Begins the process of transferring ownership to a new owner after a 2-week delay.
- `acceptOwnership()` : Completes the ownership transfer after the 2-week delay.

The Bridge contract of `TokenMinter` can call:

- `mint()` : Mints tokens and transfers them to a specified address, restricted by the daily limit.
- `burn()` : Burns a specified amount of tokens by transferring them to a native address.

The Owner of `Validator` can call:

- `updateValidators()` : Assigns a new set of validators.

The Minter of `ChromiaToken` can call:

- `transferFromChromia()` : Mints tokens to the specified address.
- `changeMinter()` : Reassigns the minter role.

Findings

CHR-1 Anyone Can Forge a Withdrawal in `withdrawBySnapshot()`

● High ⓘ Fixed

Update

Marked as "Fixed" by the client.

Addressed in: `037bf751bedf801bf11696cf96d22ec7a04da62d` .

File(s) affected: `TokenBridgeWithSnapshotWithdraw.sol`

Description: In `withdrawBySnapshot()` , several merkle proofs are provided to validate the `snapshot` . The merkle root containing all of the snapshots is in `extraProof.leaf` . The `stateProof` is a merkle proof that validates that the `snapshot` is in the `extraProof.leaf` .

However, there is no validation on `extraProof.root` to confirm that it is part of the `massExitBlock.blockRid` . While `TokenBridge._withdrawRequest()` calls `verifyBlockHeader()` , this step is skipped in `withdrawBySnapshot()` . If any `extraProof.root` can be forged, this would allow anyone to drain the contract during a mass exit. The relevant code has been reproduced below:

```
function withdrawBySnapshot(
    bytes calldata snapshot,
    Data.Proof memory stateProof,
    Data.ExtraProofData memory extraProof
) public virtual whenMassExit whenNotPaused nonReentrant {
    require(_snapshots[stateProof.leaf] == false, "TokenBridge: snapshot already used");
    require(stateProof.leaf == keccak256(snapshot), "TokenBridge: snapshot data is not correct");
    require(Hash.hashGtvBytes64Leaf(extraProof.leaf) == extraProof.hashedLeaf, "Postchain: invalid EIF
extra data"); // Internal integrity of the `extraProof` is validated.
    bytes32 stateRoot = _bytesToBytes32(extraProof.leaf, 32); // The final reference to `extraProof`
    if (!MerkleProof.verify(stateProof.merkleProofs, stateProof.leaf, stateProof.position, stateRoot)) //
The final reference to `stateRoot`. No validation against the `massExitBlock`.
        revert("TokenBridge: invalid merkle proof");
    ....
}
```

Recommendation: Validate the `extraProof` against the `massExitBlock`.

CHR-2 The `TokenBridge` Owner Can Drain All Funds if the Contract Is Paused

● Medium ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `5c488f8d981e52844501ea6588e68cea51889108`.

File(s) affected: `TokenBridgeWithSnapshotWithdraw.sol`

Description: The token bridge owner can trigger a mass exit if the validators are suspected of being compromised. This allows all users to withdraw their tokens according to a snapshot at the last valid block. The owner is allowed to withdraw the remaining contract balance after 90 days.

Additionally, the ability to pause and unpause the contract is spread between the contract owner and the validators. The validators can pause the contract and the owner can unpause it.

Should the contract be paused for any reason, a malicious owner is incentivized to keep all withdrawals halted and guarantee the full contract balance for themselves. They can call `triggerMassExit()`, regardless of whether the validators have been compromised, and withdraw the full contract balance after 90 days.

Exploit Scenario:
1. The token bridge accumulates a large number of tokens through normal user activity.
2. The validators pause the contract. No further calls to deposit or withdraw are possible.
3. Once the concerns have been cleared, rather than unpausing the contract, the owner calls `triggerMassExit()` with any valid `blockHeader`. Users will still be unable to execute their withdrawals.
4. After 90 days, the owner can call `emergencyWithdraw()` to transfer the full contract balance to any address.

Recommendation: Remove the `whenNotPaused` modifier from `withdrawBySnapshot()` so that users always have the power to reclaim their funds. Given that pausing is meant to halt token flow due to suspicious validators, it should be safe to resume token flow with the approved `massExitBlock`.

CHR-3 The `TokenBridge` Owner Can Double-Spend Withdrawals

● Medium ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `3fa949a2e9a903c54637fcd39abe8ea70d20f8bd`.

File(s) affected: `TokenBridge.sol`, `TokenBridgeWithSnapshotWithdraw.sol`

Description: The bridge owner has the power to both call `triggerMassExit()` and `postponeMassExit()`. Despite its name, `postponeMassExit()` does not delay the mass exit, but instead returns the contract to normal operations.

Additionally, conventional bridge withdrawals and withdrawals by snapshot are tracked differently. Completed withdrawals done by `withdrawRequest()` are tracked in the `_events` mapping and the withdrawals done by `withdrawBySnapshot()` are tracked in the `_snapshots` mapping.

The token bridge owner can batch their transactions such that they trigger a mass exit, withdraw tokens by snapshot, and postpone the mass exit to return the contract to normal operations. They could then withdraw the same tokens normally at their convenience with the usual withdrawal method.

Exploit Scenario:
1. The bridge owner deposits 100 ETH into the bridge.
2. The bridge owner batches the following transactions:
   1. `triggerMassExit()`, which enables `withdrawBySnapshot()`.
   2. `withdrawBySnapshot()` to withdraw their 100 ETH.
   3. `postponeMassExit()` to return the contract to normal operations.
3. The bridge owner makes a `withdrawRequest()` for 100 ETH. Since the `_events[]` mapping indicates that the withdrawal has not yet been performed, the withdrawal request goes through.
4. So long as the contract has sufficient liquidity, the bridge owner can eventually redeem their withdrawal by calling `withdraw()`.

Recommendation: Never allow the contract to resume normal operations after a mass exit is triggered.

CHR-4 Malicious Validators Can Forge Old Blocks During a Mass Exit

● Medium ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `9b8a8fe7f2089a8568d4274a63764dd2574c1ad8`.

File(s) affected: `TokenBridge.sol`

Description: `_withdrawRequest()` allows withdrawals during a mass exit so long as the block is older than the `massExitBlock`. This allows users to execute withdrawals that they've already initiated on the Chromia side, as they would not appear in the token snapshot for the `massExitBlock`.

Though the old block will predate the `massExitBlock`, signifying that it was mined before the validators were compromised, it is still the compromised validators who are signing off on the submitted `blockRid` in the withdrawal request. The compromised validators can approve a forged `blockRid` that claims to be older than `massExitBlock`.

Recommendation: When validating old blocks during a mass exit, require a hash chain proof linking the old block to the mass exit block.

CHR-5
Withdrawal by Snapshot Can Be Double-Spent Across Different Bridges

● Medium Acknowledged

Update
Marked as "Fixed" by the client. No code changes were made, instead the client updated the documentation to warn of this vulnerability. Thus the status has been marked as Acknowledged.
Addressed in: `07959260996dad19474aa71906e0d27dbf28dd94`.

File(s) affected: `TokenBridgeWithdrawBySnapshot.sol`, `hbridge/snapshot.rell`

Description: In `withdrawBySnapshot()`, the `ERC20StateHeader` is validated to confirm that the withdrawal snapshot is intended for the bridge contract. A snapshot can either be authorized for any bridge on a given network, or a specific bridge on the network. The relevant code is reproduced below:

```
// assume networkId must fit in 96 bits
uint256 allowedDiscriminator1 = networkId << 160; // discriminator allows any bridge contract on the
network
uint256 allowedDiscriminator2 = allowedDiscriminator1 + uint160(address(this));

require((header.discriminator == allowedDiscriminator1)
            || (header.discriminator == allowedDiscriminator2),
            "TokenBridge: invalid bridge contract");
```

`hbridge/snapshot.rell` indicates that the open discriminator is used as a default:

```
function network_id_to_discriminator(network_id: integer): big_integer {
    return big_integer(network_id) * big_integer.from_hex("1" + "00".repeat(20));
}

function maybe_create_local_erc20_state_slot(account: ft4.accounts.account, erc20_asset): state_slot? {
    ....
    if (not exists(existing_slot)) {
        existing_slot = create state_slot (
            id = get_next_state_slot_id(),
            network_id,
            recipient_address = account_address,
            type = state_slot_type.erc20,
            discriminator = network_id_to_discriminator(network_id), // Open discriminator
            local = true
        );
    }
    ....
}
```

If a mass exit is triggered across multiple bridges on the same network, then `header.discriminator == allowedDiscriminator1` would allow users to call `withdrawBySnapshot()` across different bridges.

Recommendation: Avoid the use of `header.discriminator == allowedDiscriminator1`.

CHR-6 Signature Format for Linking EVM Accounts May Lead to Stolen Deposits

● Medium ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `69981a4ffa77d758a75b6c88d7c264bdc78e42e6`.

File(s) affected: `hbridge/operations.rell`

Description: In the operation `link_evm_eoa_account()`, two authentications occur:

1. `authenticate()` is called to get the account associated with the operation call.
2. `ft4.auth.verify_signers([address]);` ensures that the EVM address has signed the message permitting the link.

However, the message signed by the EVM address does not appear to have any reference to the FT4 account. An example message format is as follows:

```
Blockchain:
0000000000000000000000000000000000000000000000000000000000000000

Please sign the message to call
operation:
- eif.hbridge.link_evm_eoa_account

with arguments:
- E1FAE9B4FAB2F5726677ECFA912D96B0B683E6A9

Nonce: 165436D801A7942A5D454666B91070AAFF5FE724CCEB74BA99830D380E17D399
```

In the above, `E1FAE9B4FAB2F5726677ECFA912D96B0B683E6A9` corresponds to the EVM address. The `Nonce` field does not contain any reference to the FT4 address except for the `auth_descriptor.ctr` (as seen in `utils.derive_nonce(op, auth_descriptor.ctr)` in `authentication.rell`). As long as the attacker has an `auth_descriptor` with the same `ctr` as the honest owner of the EVM address, they may be able to front-run linking the EVM address, potentially stealing deposited funds.

Exploit Scenario:
1. Alice makes a deposit on the EVM side, but has not yet linked to an FT4 account.
2. Alice signs a message that they would like to link.
3. Bob becomes aware of Alice's signature (possibly through a man-in-the-middle attack).
4. Bob front-runs a call to `link_evm_eoa_account()`, linking their own FT4 account to Alice's EVM address.
5. Bob steals the deposit.

Recommendation: Include the FT4 account in the signed message.

CHR-7 The `TokenBridge` Owner Can Lock All Funds

● Low ⓘ Fixed

Update
Marked as "Fixed" by the client. A function was added to finalize the `blockchainRid`.
Addressed in: `421c433318ce1ee62c6405cbdc2279d86bc4bf40`.

File(s) affected: `TokenBridge.sol`

Description: To initially configure the bridge, the owner can call `setBlockchainRid()` to point to the chain from which the bridge can receive withdrawals. However, the owner can reassign this value at any time, making it impossible to withdraw previous deposits.

Recommendation: Ideally the `blockchainRid` should be immutable and set in the constructor. If this flexibility is important, consider adding a flag to permanently disable the `setBlockchainRid()` function to make this feature trustless.

CHR-8 `triggerMassExit()` Does Not Scale with `withdrawOffset`

● Low ⓘ Acknowledged

Update

The client provided the following explanation:

> We don't consider this fix a priority because currently it is possible to configure withdrawOffset to be ~3 days. Also we trust the owner to select mass exit block in a meaningful way.
>
> In future we plan to refactor it - it would be easiest to define withdrawOffset in seconds, not blocks. Then we can use it instead of the 3 day constant

File(s) affected: `TokenBridgeWithSnapshotWithdraw.sol`, `TokenBridge.sol`

Description: In `TokenBridgeWithSnapshotWithdraw.triggerMassExit()`, the owner assigns the `massExitBlock` as the last valid block from which users can call `withdrawBySnapshot()`. The block is required to have occurred within the last three days:

```
require(header.timestamp >= (block.timestamp - 3 days) * 1000, "TokenBridge: mass exit block is too old");
```

Given that the requirement is hardcoded to 3 days, this duration may be longer than the dispute period if `withdrawOffset` is shorter than 3 days. This would allow users who executed their withdrawals to withdraw additional funds via the older snapshot.

Exploit Scenario:
1. Suppose it is time `N`, the `massExitBlock` is chosen to be at time `N - 3 days`, and `withdrawOffset = 1 day`.
2. Suppose a user initiated a withdraw at time `N - 2 days`, and the user completed the withdraw at `N - 1` day.
3. This withdraw will not be captured by the snapshot, and the user will be allowed to withdraw their funds a second time.

Recommendation: Require that the dispute period is larger than the age of the `massExitBlock`.

CHR-9 `TokenMinter` May Be Unable to Transfer Roles

● Low ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `e49f520a03585da504f2c483facfa1a76f1a8bae`.

File(s) affected: `TokenMinter.sol`

Description: The `transferMintRole()` and `transferOwnership()` functions use a two-week delay. The delay is reset if a new pending address is called. The relevant code is reproduced below:

```
function transferMintRole(address newMinter) external onlyOwner {
    if (pendingNewMinter != address(0)) resetDelayForFunction(this.transferMintRole.selector);
    startDelayedAction(this.transferMintRole.selector);
    pendingNewMinter = newMinter;
}

function transferOwnership(address newOwner) public override onlyOwner {
    if (pendingOwner() != address(0)) resetDelayForFunction(this.transferOwnership.selector);
    startDelayedAction(this.transferOwnership.selector);
    super.transferOwnership(newOwner);
}
```

`startDelayAction()` will revert if a delay is active. This means that these functions will always revert if the pending address is `address(0)` and there is an active delay. This could result in a cumbersome transfer of addresses as the owner waits for the two week delay to complete before reassigning the addresses.

If a transfer of the minter role is needed, it could also lead to the `ChromiaToken_Base()` not having an assigned minter for a period of time. Depending on the requirements of the contract, having no minter assigned may be prohibitive, forcing a launch of a new contract or keeping the current minter.

Exploit Scenario:
The following scenario is for transferring the mint role, though the same logic applies for transferring ownership:
1. The contract has not had any transfers initiated and the `pendingNewMinter = address(0)` as it is uninitialized.
2. The owner calls `transferMintRole(address(0x1))` to assign a new minter. A two-week delay triggers and `pendingNewMinter = address(0x1)`.
3. Realizing that `address(0x1)` is not the desired minter, the owner cancels the transfer by calling `transferMintRole(address(0))`. The delay is reset and `pendingNewMinter` is back to `address(0)`. **However, there is currently an active delay.**
4. After more deliberation, the owner discovers the true desired minter and calls `transferMintRole(address(0x2))`. However, the previous delay is not cleared as `pendingNewMinter == address(0)` does not trigger the reset. This forces `startDelayedAction()` to revert and makes the function impossible to call.

5. The only way to reset the delay is by calling `finishTransferMintRole()`. With the delay finally cleared, the owner calls `transferMintRole(address(0x2))`, but must wait another two-week delay before finalizing.
6. The `ChromiaToken_Base` contract is left with no minter assigned for two weeks.

Recommendation: Always reset the delay function.

CHR-10 The `dayLimit` Can Quickly Increase Despite a Recent Update

• Low ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `7c33332301b184539e38ba5901f2026d47ff479e`.

File(s) affected: `TokenMinter.sol`

Description: The `TokenMinter` contract enforces a two-week delay for increasing the `dayLimit` and no delay for decreasing it. If a new decreased `dayLimit` is assigned, it will not clear any `pendingDayLimit` increases. This can result in the `dayLimit` being decreased, quickly followed by the previous `pendingDayLimit` completing its two-week delay and being assigned. The code has been reproduced below:

```
function setDayLimit(uint _newDayLimit) external onlyOwner {
    if (_newDayLimit > dayLimit) {
        // if we already have a pending change, reset it
        if (pendingDayLimit != 0) resetDelayForFunction(this.setDayLimit.selector);
        // Set pending day limit and start the two-week delay
        pendingDayLimit = _newDayLimit;
        startDelayedAction(this.setDayLimit.selector);
    } else {
        // If the new limit is lower, apply immediately
        dayLimit = _newDayLimit;
        emit DayLimitChanged(_newDayLimit);
    }
}

function finishSetDayLimit() external onlyOwner {
    finishDelayedAction(this.setDayLimit.selector);
    dayLimit = pendingDayLimit;
    delete pendingDayLimit;
    emit DayLimitChanged(dayLimit);
}
```

Exploit Scenario:
1. The current `dayLimit` is `10`.
2. The owner calls `setDayLimit(15)` and a two-week delay triggers.
3. One week later, the owner calls `setDayLimit(5)`. The `dayLimit` is immediately reassigned to `5`, but `pendingDayLimit` remains `15`.
4. Another week later, the two-week delay has elapsed, and the owner calls `finishSetDayLimit()`.
5. The `dayLimit` increased from `5` to `15`, despite it only being a week since the last change.

Recommendation: Call `delete pendingDayLimit;` after reassigning the `dayLimit` and reset the delay.

CHR-11 Withdrawals May Accidentally Resume Despite Suspicious Validators

• Low ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `3fa949a2e9a903c54637fcd39abe8ea70d20f8bd`.
The client provided the following explanation:

```
Same fix as for CHR-3
```

File(s) affected: `TokenBridge.sol`

Description: A mass exit is triggered once it is suspected that the validators cannot be trusted. The bridge owner can do this by calling `triggerMassExit()` and assign the `massExitBlock`. If the `massExitBlock` is incorrectly assigned, perhaps due to an incorrect assessment of the last block signed by uncompromised validators, the `postponeMassExit()` allows the owner to undo this action.

However, `postponeMassExit()` immediately returns the contract to normal functionality. This would allow further withdrawals to be performed, some of which may be signed by compromised validators.

Recommendation: Update `postponeMassExit()` to pause the contract. This should give the owner time to reassign the `massExitBlock` or to determine whether the mass exit was a false alarm.

CHR-12 `setBlockchainRid()` Is Front-Runnable by Anyone

• Low ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `04df3127eeca2943cfb5c3c1026461b8791c620b`.

File(s) affected: `ManagedValidator.sol`

Description: Function `ManagedValidator.setBlockchainRid()` lacks any access controls and may be called by anyone. They can set an arbitrary `blockchainRid` value, impacting any validations being performed on that contract.

Recommendation: We recommend declaring the variable `immutable` and making it only settable once during the constructor. Alternatively, add the `onlyOwner` modifier to `setBlockchainRid()`.

CHR-13 `process_deposit_erc20_event` Incorrectly Assumes Deposit Origin

• Low ⓘ Mitigated

Update
Marked as "Fixed" by the client.
Addressed in: `f894f2c47e9d4d7cdc5c7dfaddefb44e12937870`. The function `TokenBridge.withdrawToPostchain()` was not updated, thus the status has been marked as Mitigated.
The client provided the following explanation:

```
1. We will improve events for SC accounts in  withdrawToPostchain  later since it requires some
   refactoring of the Rell code.
2. We believe that after the fix, the 2nd recommendation might be skipped.
```

File(s) affected: `hbridge/accounts.rell`, `TokenBridge.sol`, `hbridge/operations.rell`

Description: Function `process_deposit_erc20_event()` may falsely assume the deposit origin. Consider the following code segment:

```
function process_deposit_erc20_event(network_id: integer, event: deposited_erc20_event) {
    var beneficiary = event.beneficiary;
    val deposit_from_smart_contract = beneficiary != zero_beneficiary;
    ...
}
```

A `beneficiary == zero_beneficiary` may incorrectly assign `deposit_from_smart_contract` due to the following:

1. `TokenBridge.deposit()` being callable not only from EOAs but also from smart contracts.
2. `TokenBridge.depositToAccountID()` allowing for `accountID` (on RELL side `beneficiary`) being passed in as zero.
3. `TokenBridge.withdrawToPostchain()` defaulting the `accountID` to `0x0`.

While most of the deposit functionality will still work with an incorrect `deposit_from_smart_contract`, we identified the following impacts:

1. Smart contracts do not require an explicit link to an FT4 account to process a deposit.
2. Completed deposits may be incorrectly marked as pending.

Smart Contracts Do Not Require an Explicit Link to a FT4 Account

The function `process_deposit_erc20_event()` will require an explicit link between a smart contract and a FT4 account to prevent unwanted deposits, as indicated by the following code:

```
if (deposit_from_smart_contract) {
    ft4_account = ft4.accounts.account @? { .id == beneficiary };

    if (not empty(ft4_account)) {
        // we require explicit link to avoid unwanted deposits
        if (need_consent_on_deposit and empty(evm_sc_account_link @? { event.sender, network_id,
ft4_account })) {
            bounce_deposit_event(network_id, event.token_address, event.sender, event.amount);
```

```
                return;
            }
        }

    }
```

If `deposit_from_smart_contract` is incorrectly assigend to `false`, then this requirement will be skipped and FT4 accounts may receive unwanted deposits.

Completed Deposits May Be Incorrectly Marked as Pending

Pending deposits made to Chromia can be collected once an FT4 account is registered.

```
function _link_evm_sc_account(address: byte_array, network_id: integer, account: ft4.accounts.account) {
    require(empty(evm_sc_account_link @? { address, network_id }), "EVM address %s on EVM chain ID %d is already linked to FT4 account".format(address, network_id));
    create evm_sc_account_link(address, network_id, account);

    // there's a possibility that the account_id is different from address hash
    // collect funds from the pool account here
    ft4.transfer_strategy.collect_pooled_assets(account, recipient_id = address.hash());

    val pending_deposits = erc20_deposit @* { .sender_address == address, deposit_state.pending, .from_smart_contract == true };
    for (deposit in pending_deposits) {
        update_deposit_state(deposit, deposit_state.completed);
    }

    after_evm_account_linked(address, account, network_id, true);
}
```

If `deposit_from_smart_contract` is incorrectly assigend to `false`, then pending deposits will not be found with the query `val pending_deposits = erc20_deposit @* { .sender_address == address, deposit_state.pending, .from_smart_contract == true };`. They will remain in status `pending` despite having been collected.

`hbridge/operations.recall_pending_deposit()` would still prevent the recall of such a transfer in `ft4.transfer_strategy.recall_transfer()` and we could not find an exploit. However, future functionality that queries the `.deposit_state.pending` field may not work as intended.

Recommendation:

1. Update `TokenBridge.deposit()`, `TokenBridge.depositToAccountID()`, and `TokenBridge.withdrawToPostchain()` such that smart contracts cannot submit a beneficiary of `0x0`.
2. Consider updating the filter in `_link_evm_sc_account()` to accept any boolean for `.from_smart_contract`.

CHR-14 An `evm_block` May Be Associated with Multiple Hashes

● Low ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
In the Event Receiver chain, we allow multiple blocks with the same (network_id, evm_block_height) but different evm_block_hash. Theoretically, it might happen due to a potential EVM chain re-org. It is then up to the Event Receiver client to decide how to handle it.
  To mitigate this:
  we log a message: https://gitlab.com/chromaway/core/postchain-eif/-/blob/hbridge_2/postchain-eif-rell/rell/src/eif_event_receiver/module.rell?ref_type=heads#L30
  we recommend setting the eif.chains..evm_read_offset property to be large enough, e.g. 10 min. for Ethereum, 5 min. for BSC.
```

File(s) affected: `eif_event_receiver/model/module.rell`

Description: The entity `evm_block` has three keys:

1. `network_id`
2. `evm_block_height`
3. `evm_block_hash`

This would allow a given `<network_id, evm_block_height>` pair to be associated with multiple `evm_block_hashes`, which could cause inconsistencies when processing blocks.

Recommendation: Remove the key for `evm_block_hash`.

CHR-15 `postponeMassExit()` May Create a Deficit

● Informational ⓘ Fixed

Update

The client provided the following explanation:

`We removed postponeMassExit(), see CHR-3 and CHR-12`

File(s) affected: `TokenBridge.sol`

Description: A call to `postponeMassExit()` returns the contract back to normal functionality after a mass exit has been triggered. If any withdrawals have occurred, this would lead to a deficit in the bridge, as the contract will not have enough balance to honor all withdrawals made with `withdrawRequest()`. Given that this possibility has been mentioned in the documentation, the issue has been assigned Informational severity.

Recommendation: Either remove the `postponeMassExit()` function or require a call to `fund()` the deficit.

CHR-16 Potential Storage Collisions

● Informational ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

`We'll warn people that it won't be possible to upgrade existing contracts to a new version.`

File(s) affected: `TokenBridge.sol`, `TokenBridgeWithSnapshotWithdraw.sol`, `ChromiaTokenBridge.sol`

Description: The `TokenBridge` contract has had its storage layout refactored since the previous audit, with some variables removed/added across the `ChromiaTokenBridge` and `TokenBridgeWithSnapshotWithdraw` contracts. This makes the new `TokenBridge` incompatible for upgrading previous versions. After speaking with the Chromia team, none of the deployed instances of the `TokenBridge` contract are scheduled for upgrades, and thus the severity of the issue has been marked as Informational.

Recommendation: Either update the storage layout to be compatible with future upgrades, or avoid upgrading any previous deployments of the contract.

Auditor Suggestions

S1 `ChromiaTokenBridge.fund()` May Result in Locked Tokens

Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `2c0feb08ef447aa83a2bed06e1fc87deeef9668f`.

File(s) affected: `ChromiaTokenBridge.sol`, `TokenBridge.sol`

Description: The `fund()` function allows the contract owner to help fund any deficits by transferring tokens into the bridge contract. However, the `ChromiaTokenBridge` holds funds in the `TokenMinter` contract rather than the bridge contract itself. Transferring funds into this contract would likely get them locked.

Recommendation: Disable `fund()` in `ChromiaTokenBridge`.

S2 Contract Owner May Create Empty Withdrawable Entries

Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `7ff13fba841bc9ec2543b2902635442d01b2664a`.

File(s) affected: `TokenBridge.sol`

Description: In the function `unpendingWithdraw()`, when an unused `_hash` is provided the returned struct `_withdraw[_hash]` will be zero-initialized. This means that the unused hash will pass the check `wd.status == Status.Pending`, since `Status.Pending` occupies the zero-th entry in the `Status` enum. This allows the contract owner may create empty `_withdraw[]` entries with status `Status.Withdrawable` by calling `unpendingWithdraw()`.

While this may create false entries and falsely emit events, we did not find an exploitable path.

Recommendation: We recommend prepending the `Status` enum with an unused `UNINITIALIZED` entry at the zero position, before `Pending`, to prevent the above. This will also lower the bug/attack surface as the contract undergoes future development.

S3 Magic Constants and Undocumented Variables

Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `49aecb7e4db6de1e015fe08f7dc6b3122864fa7b`.

File(s) affected: `TokenBridgeWithSnapshotWithdraw.sol`

Description: The following constants and variables do not appear to have corresponding documentation:
1. `ERC20_STATE_TAG_V1 = 0x6862726964467653a65726332303a763101010101010101010101010101010101;`

Recommendation: Add inline comments and/or external documentation to describe each of these variables.

S4 `TokenBridge` Contract Should Be Declared `abstract`

Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
We use the TokenBridge contract in integration and deployment tests and in the chromia-bridge-demo
repository.
```

File(s) affected: `TokenBridge.sol`

Description: Contract `TokenBridge` contains logic to act on the `isMassExit` flag being set and logic to re-set it, but does not implement triggering a mass exit, resulting in partially dead code if deployed as is.

Recommendation: Consider changing the contract to `abstract` type, to prevent accidental deployments in the future that may have incomplete functionality.

S5 Unnecessary Function Parameter

Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Yes, it is unnecessary, but we think it is useful to have it for consistency with the withdrawal object.
  The hash and beneficiary might be stored together off-chain for convenience.
```

File(s) affected: `TokenBridge.sol`

Description: It is not clear why `beneficiary` is needed as a parameter to `withdraw()` as the `_hash` already encapsulates this data.

Recommendation: Remove the parameter or document why it is included.

S6 Missing Input Validation

Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Good point. We will add data validation but have to skip this for now due to time constraints.
```

File(s) affected: `BaseManagedValidator.sol` , `ChromiaToken.sol` , `ChromiaTokenBridge.sol` , `ManagedValidator.sol` , `TokenBridge.sol` , `TokenMinterBase.sol` , `Validator.sol`

Related Issue(s): SWC-123

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error:

- `BaseManagedValidator._updateValidators()` : Validate that `_validators` has no duplicate addresses.
- `ChromiaToken._setMinter()` : Validate that `account` is not the zero address.
- `ChromiaTokenBridge` :
  - Add validation to confirm that only the supported tokens by the `TokenMinter` are added to the allow list.
  - `setTokenMinter()` : Validate that `_tokenMinter` is not the zero address.
- `ManagedValidator.constructor()` : Validate that `directoryChainValidator` is not the zero address.
- `TokenBridge.initialize()` : Validate that `withdrawOffset > 0` .
- `TokenMinterBase.constructor()` : Validate that `tokenContractAddress` and `bridgeContractAddress` are not the zero address.
- `Validator`
  - `constructor()` : Require that the all of the validators are not the zero address and non-duplicate addresses.
  - `updateValidators()` : Require that there are no duplicate addresses.

Recommendation: We recommend adding the relevant checks.

S7 Application Monitoring Can Be Improved by Emitting More Events

Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Good point. We will add more events.
```

File(s) affected: `TokenBridgeWithSnapshotWithdraw.sol` , `TokenMinter.sol` , `ChromiaTokenBridge.sol`

Description: In order to validate the proper deployment and initialization of the contracts, it is a good practice to emit events. Also, any important state transitions can be logged, which is beneficial for monitoring the contract, and also tracking eventual bugs or hacks. Below we present a non-exhaustive list of events that could be emitted to improve application management:

1. `TokenBridgeWithSnapshotWithdraw.emergencyWithdraw()` .
2. `TokenMinter.finishTransferMintRole()` .
3. `ChromiaTokenBridge.setTokenMinter()` .

Recommendation: Consider emitting the events.

S8 Remove Out-Commented Code

Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
We will continue working on the commented code after the audit.
```

File(s) affected: `hbridge/snapshot.rell`

Description: The code in `snapshot.rell#L210-253` is out-commented and unused.

Recommendation: We recommend removing the mentioned lines for improved maintainability.

S9 Remove Deprecated Functions

Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Some of them are still used in tests. We will remove the deprecated functions.
```

File(s) affected: `hbridge/queries.rell`

Description:
1. The function `get_erc20_withdrawal()` has been annotated as deprecated. If there is no current production code using this function, it can be removed.
2. The function `bridge_ft4_token_to_evm()` was intended for a use-case where the same asset may share multiple different addresses and shares nearly identical code with `bridge_ft4_token_to_evm_contract()`.

Recommendation: Remove any deprecated functions.

S10
State Slot Creation May Be Inefficient if Many Assets Have Snapshots Enabled

Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `e45774614bf280a8fa3b6ae4d85dbb0a82af89e6`.
The client provided the following explanation:

```
Fixed in CHR-6
```

File(s) affected: `hbridge/snapshot.rell`

Description: In the `@extend(after_evm_account_linked)` function, when the account does not have existing slots, we potentially create a state slot for all assets that match this expression:

```
val assets = (b:ft4.assets.balance, e: erc20_asset_snapshots_enabled) @* {
                b.asset == e.asset
            } (.erc20_asset);
```

If there are many assets with snapshots enabled, this may become expensive to compute. Additionally, the function will add slots for all tokens, regardless of whether the user has any in possession.

Also, as noted by the Chromia team, there is a missing conditional `b.account == account` in the `WHERE` component.

Recommendation:
1. Ensure that the number of tokens that use snapshots is not unreasonably high.
2. Add the missing conditional.
3. Consider only creating slots for tokens with a non-zero balance.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Files

- `a87...a84 ./module.rell`
- `f32...dd9 ./operations.rell`
- `579...1ff ./withdrawals.rell`
- `826...019 ./assets.rell`
- `14e...13d ./accounts.rell`
- `5a4...7eb ./events.rell`
- `611...1e5 ./queries.rell`
- `ff3...b88 ./snapshot.rell`
- `406...08e ./deposits.rell`
- `69f...5c2 ./module.rell`
- `bfc...a6c ./module.rell`
- `9a6...dee ./module.rell`
- `67c...5b9 ./queries_impl.rell`
- `cdb...323 ./queries_test.rell`
- `919...769 ./helpers.rell`
- `a96...292 ./module.rell`
- `50c...cf5 ./version.rell`
- `408...0ec ./module.rell`
- `2ab...c07 ./module.rell`
- `8ff...019 ./module.rell`
- `5fb...378 ./contracts/Validator.sol`
- `eb5...f28 ./contracts/TokenMinter.sol`
- `68a...cae ./contracts/ChromiaTokenBridge.sol`
- `845...681 ./contracts/IValidator.sol`
- `430...fca ./contracts/Postchain.sol`
- `75d...986 ./contracts/Data.sol`
- `d0f...2db ./contracts/TokenBridgeWithSnapshotWithdraw.sol`
- `29f...d36 ./contracts/TokenBridge.sol`
- `450...6d1 ./contracts/utils/TwoWeekDelay.sol`
- `852...f17 ./contracts/validatorupdate/DirectoryChainValidator.sol`
- `b85...ded ./contracts/validatorupdate/IManagedValidator.sol`
- `2c4...24a ./contracts/validatorupdate/ValidatorUpdateUtils.sol`
- `9e8...fc4 ./contracts/validatorupdate/BaseManagedValidator.sol`
- `fc7...cbc ./contracts/validatorupdate/ManagedValidator.sol`
- `2b3...26f ./contracts/token/ChromiaToken.sol`

Tests

- `043...ab2 ./chromiabridge.test.ts`
- `49c...8fd ./erc20.test.ts`
- `07f...4c8 ./utils.ts`
- `d1e...b41 ./bridge.test.ts`
- `fa2...f82 ./tokenminter.test.ts`
- `eca...e89 ./utility.test.ts`
- `341...613 ./tests/hbridge_it/module.rell`
- `b24...359 ./tests/utils/module.rell`
- `378...613 ./tests/utils/to_32_byte_array.rell`
- `f1a...48e ./tests/hbridge/helpers/module.rell`
- `680...a9d ./tests/hbridge/hbridge/module.rell`

- `22b...5e4 ./tests/hbridge/hbridge/deposit_queries.rell`
- `dce...87b ./tests/hbridge/hbridge/withdrawal_queries.rell`
- `ccc...f8c ./tests/hbridge/hbridge/withdraw_to_evm.rell`
- `972...ca5 ./tests/hbridge/hbridge/deposit_from_evm.rell`

Automated Analysis

N/A

Test Suite Results

The test suite for solidity was run with `yarn test`. The test suite for rell was run with `chr test --settings chromia.yml`.

```
SOLIDITY
Token Bridge Test
    Validators
      ✔ Admin can update validator(s) successfully
    Deposit
      ✔ User can deposit ERC20 token to target smartcontract
    Withdraw by normal user
      ✔ User can request withdraw by providing properly proof data
    Withdraw via smart contract
      ✔ Integrate with smart contract
    Mass Exit
      ✔ Emergency withdrawal can be performed after mass exit
      ✔ only admin can manage mass exit
    Ownership
      ✔ renounce ownership is not allowed

  ChromiaToken Bridge Test
    Validators
      ✔ Admin can update validator(s) successfully
    Deposit
      ✔ User can deposit ERC20 token to target smartcontract
    Withdraw by normal user
      ✔ User can request withdraw by providing properly proof data
    Withdraw via smart contract
      ✔ Integrate with smart contract

  Token
    Mint
      ✔ Should mint some tokens
    Transfer
      ✔ Should transfer tokens between users
      ✔ Should fail to transfer with low balance

  TokenMinter test
    ChromiaToken
      ✔ Admin can change minter
      ✔ Admin can change owner
      ✔ Admin can set daily limit
      ✔ Cant mint more than daily limit

  Utility Test
    Utility
      hash
        ✔ Non-empty node sha3 hash function
        ✔ Right empty node sha3 hash function
        ✔ Left empty node sha3 hash function
        ✔ All empty node
        ✔ hash gtv integer leaf 0
        ✔ hash gtv integer leaf 1
        ✔ hash gtv integer leaf 127
        ✔ hash gtv integer leaf 128
        ✔ hash gtv integer leaf 168
        ✔ hash gtv integer leaf 255
        ✔ hash gtv integer leaf 256
        ✔ hash gtv integer leaf 1023
        ✔ hash gtv integer leaf 1024
        ✔ hash gtv integer leaf 32769
        ✔ hash gtv integer leaf 1234567890
        ✔ hash gtv bytes 64 leaf incorrect
      Merkle Proof
        ✔ Verify valid merkle proof properly
        ✔ Invalid merkle proof
      SHA256 Merkle Proof
        ✔ Verify valid SHA256 merkle proof properly
        ✔ Invalid SHA256 merkle proof due to incorrect merkle root

RELL
OK
tests.hbridge.hbridge:test_deposit_foreign_token_from_evm_when_account_is_registered_and_erc20_token_is_not
_registered
OK
tests.hbridge.hbridge:test_deposit_foreign_token_from_evm_when_both_account_and_erc20_token_are_not_registe
red
OK tests.hbridge.hbridge:test_deposit_foreign_token_from_evm_with_already_created_and_linked_account
OK tests.hbridge.hbridge:test_deposit_foreign_token_from_evm_before_account_creation
OK
tests.hbridge.hbridge:test_deposit_foreign_token_from_evm_will_get_pending_assets_when_registering_account_
with_evm_address
OK tests.hbridge.hbridge:test_can_recall_unclaimed_deposit_from_evm_after_timeout_once_but_not_twice
OK tests.hbridge.hbridge:test_get_erc20_deposits
OK tests.hbridge.hbridge:test_withdraw_to_ethereum_with_already_linked_account
OK tests.hbridge.hbridge:test_withdraw_to_ethereum_using_most_recent_erc20_token_address_by_default
OK tests.hbridge.hbridge:test_withdraw_to_ethereum_using_specific_erc20_token_address
OK tests.hbridge.hbridge:test_can_not_withdraw_to_ethereum_without_linked_token
OK tests.hbridge.hbridge:test_can_not_withdraw_to_ethereum_without_linked_account
OK tests.hbridge.hbridge:test_can_not_withdraw_to_ethereum_with_insufficient_balance
OK tests.hbridge.hbridge:test_can_not_withdraw_negative_or_zero_amount_to_ethereum
OK tests.hbridge.hbridge:test_get_erc20_withdrawals
OK tests.utils:test_evm_address_to_32_byte_array
OK transaction_submitter.anchoring.test.anchoring_test:test_anchoring_operation_enqueues_tx
OK transaction_submitter.signer_update.test.signer_update_test:test_directory_signer_update
OK transaction_submitter.signer_update.test.signer_update_test:test_directory_signer_update_failure
OK transaction_submitter.signer_update.test.signer_update_test:test_directory_parked_signer_update
OK
transaction_submitter.signer_update.test.signer_update_test:test_directory_cancelled_signer_update_at_same_
height
OK
transaction_submitter.signer_update.test.signer_update_test:test_adding_bridge_mapping_triggers_signer_upda
te
OK transaction_submitter.test.bridge_mapping_test:test_bridge_mapping
OK transaction_submitter.test.external_api_test:test_query_transaction_status
OK transaction_submitter.test.external_api_test:test_query_transaction_receipt
OK transaction_submitter.test.external_api_test:test_get_blockchains_with_bridge_and_anomaly_detection
OK transaction_submitter.test.module_test:test_create_evm_submit_transaction
OK transaction_submitter.test.module_test:test_not_whitelisted_contracts_are_ignored
OK transaction_submitter.test.module_test:test_fetch_oldest_queued_transactions_per_contract
OK transaction_submitter.test.module_test:test_fetch_oldest_queued_transactions_per_contract_and_taken_by
OK transaction_submitter.test.module_test:test_fetch_oldest_queued_transactions_per_contract_and_rate_limit
OK transaction_submitter.test.module_test:test_fetch_oldest_queued_transactions_per_contract_and_network
OK transaction_submitter.test.module_test:test_fail_transactions_with_enough_retries_strategy_all
OK transaction_submitter.test.module_test:test_fail_transactions_with_enough_retries_strategy_supermajority
OK transaction_submitter.test.module_test:test_is_supermajority
OK transaction_submitter.test.module_test:test_timeout_evm_transactions
OK transaction_submitter.test.module_test:test_timeout_verification_evm_transactions
OK transaction_submitter.test.module_test:test_timeout_submit_evm_transaction
OK transaction_submitter.test.module_test:test_get_currency_symbol
OK transaction_submitter.test.module_test:test_update_evm_transaction_status_taken_to_pending
OK transaction_submitter.test.module_test:test_update_evm_transaction_status_taken_to_queued
OK transaction_submitter.test.module_test:test_update_evm_transaction_status_taken_only_once
OK transaction_submitter.test.module_test:test_update_evm_transaction_status_rejects
OK transaction_submitter.test.module_test:test_update_evm_transaction_status_tx_hash
OK transaction_submitter.test.module_test:test_update_evm_transaction_status_SUCCESS
OK transaction_submitter.test.module_test:test_get_transaction_taken_by
OK transaction_submitter.test.module_test:test_get_transactions
OK transaction_submitter.test.system_chains_test:test_system_chain_functions
OK transaction_submitter.test.system_chains_test:test_text_without_0x

SUMMARY: 0 FAILED / 49 PASSED / 49 TOTAL (13.072s)

***** OK *****
```

Code Coverage

The code coverage was ran with `yarn coverage`. While most files have adequate testing, the test coverage is poor for `TokenBridgeWithSnapshotWithdraw`. There are also no tests for the new validators. We strongly recommend that tests be written for all contracts before deployment.

We were unable to assess the test coverage for rell.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	76.04	57.21	83.05	81.17	
ChromiaTokenBridge.sol	100	50	100	100	
Data.sol	100	100	100	100	
IValidator.sol	100	100	100	100	
Postchain.sol	100	100	100	100	
TokenBridge.sol	77.78	56.36	77.78	80.56	... 293,294,295
TokenBridgeWithSnapshotWithdraw.sol	39.39	34.78	66.67	43.24	... 76,77,81,96
TokenMinter.sol	79.31	69.44	80	92.68	145,146,150
Validator.sol	91.67	72.22	100	100	
contracts/anchoring/	0	0	0	0	
Anchoring.sol	0	0	0	0	... 29,30,31,38
contracts/token/	59.7	44.12	62.86	65.82	
AliceToken.sol	0	0	0	0	14,15,16,20,21
ChromiaToken.sol	62.07	46.67	64.52	68.57	... 320,327,334
TestToken.sol	100	50	100	100	
contracts/utils/	100	66.67	66.67	88.89	
TwoWeekDelay.sol	100	66.67	66.67	88.89	48
contracts/utils/cryptography/	93.62	76.67	100	94.74	
ECDSA.sol	90	62.5	100	90.91	22
Hash.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
MerkleProof.sol	88.24	66.67	100	91.3	12,32
contracts/utils/test/	90	100	90.91	90.91	
TestDelegator.sol	83.33	100	83.33	83.33	30
TokenBridgeDelegator.sol	100	100	100	100	
contracts/validatorupdate/	0	0	0	0	
BaseManagedValidator.sol	0	0	0	0	... 109,110,111
DirectoryChainValidator.sol	0	100	0	0	9,13,17
IManagedValidator.sol	100	100	100	100	
ManagedValidator.sol	0	0	0	0	... 16,18,22,26
ValidatorUpdateUtils.sol	0	100	0	0	... 19,20,24,25
All files	63.14	50.6	66.42	68.53	

Changelog

- 2024-09-16 - Initial report
- 2024-10-14 - Final Report

About Quantstamp

Timeliness of content

Notice of confidentiality

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites&aspo; owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and and may not be represented as such. No third party is entitled to rely on the report in any any way, including for the purpose of making any decisions to buy or sell a product, product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or or any open source or third-party software, code, libraries, materials, or information to, to, called by, referenced by or accessible through the report, its content, or any related related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

Quantstamp