Gnosis Hashi

Final Audit Report

June 27, 2024

Team Omega

teamomega.eth.limo

Summary 2
Scope of the Audit 3
Resolution 5
Methods Used 5
Disclaimer 5
Severity definitions 6
Findings 6
General 6
G1. Anyone can mark any messageId as approved by Hashi in the bridges [high] [resolved] 6
G2. Test are failing, and test coverage is incomplete [low] [not resolved] 7
G3. Pin solidity dependencies [low] [resolved] 8
G4. Ether can remain locked in contracts [low] [not resolved] 8
G5. ShoyuBashi is unused [info] [not resolved] 9
G6. Consider following the ERC-5164 standard [info] [not resolved] 9
Hashi.sol 9
H1. Code duplication between checkHashWithThresholdFromAdapters and getHashesFromAdapters [info] [resolved] 9
Yaho.sol 10
YH1. Avoid unnecessary writing to storage on _dispatchMessage [info] [resolved] 10
ShuSo.sol 10
SS1. Init pattern for setHashi is overcomplicated [info] [resolved] 10
SS2. Remove unused function _setDomainThreshold [info] [resolved] 11
SS3. _setThreshold does not verify new threshold is valid compared to count [info] [not resolved] 11
SS4. Declare functions as external instead of public when possible [info] [resolved] 11
SS5. _getThresholdHash may choose arbitrarily between multiple valid hashes [info] [resolved] 11
DendrETHAdapter.sol 12
DEA1 Anyone can set the chainId for the hash stored by the adapter [low] [resolved] 12
XDai Branch - BasicForeignBridge.sol 12
XBFB1. Unused functions can be removed [info] [resolved] 12

Summary

Gnosis has asked Team Omega to audit the update of their bridge contracts.

We found **1 high severity issue** - these are issues that can lead to a loss of funds, and are essential to fix. We classified **no** issues as "medium" - this is an issue we believe you should definitely address. In addition, **4** issues were classified as "low", and **10** issues were classified as "info" - we believe the code would improve if these issues were addressed as well.

After we submitted a preliminary report, the team fixed the majority of issues. We marked their resolution below. Two "low" severity issues and two "info" issues were left unresolved. These issues, in our opinion, do not constitute a security risk.

Severity	**Number of issues**	**Number of resolved issues**
High	1	1
Medium	0	
Low	4	2
Info	10	8

Scope of the Audit

Scope of the Audit
The audit concerns Solidity files in the following repository and branch:

```
https://github.com/gnosis/hashi/tree/feat/v0.2.0
```

The commit we audited was `6f5bf9e15e37901965c169e40a7ef907f9019eca`

Specifically, the audit concerns all the files in the `packages/evm/contracts` directory, except for the `GiriGiriBashi.sol` file (which is not under development and not compatible with the current code base).
The code in the `adapters` directory is excluded from the audit, except for the `Dendreth`, `Electron` and `Telepathy` adapters.

So the audit concerned the following files::

```
.
    ├── Hashi.sol
    ├── Yaho.sol
    ├── Yaru.sol
    ├── adapters
    │   ├── Adapter.sol
    │   ├── BlockHashAdapter.sol
    │   ├── DendrETH
    │   │   ├── DendrETHAdapter.sol
    │   │   └── interfaces
    │   │       └── IDendrETH.sol
    │   ├── Electron
    │   │   ├── ElectronAdapter.sol
    │   │   ├── interfaces
    │   │   │   └── ILightClient.sol
    │   │   └── lib
    │   │       ├── Merkle.sol
    │   │       └── Receipt.sol
    │   ├── Telepathy
    │   │   ├── TelepathyAdapter.sol
    │   │   ├── interfaces
    │   │   │   └── ITelepathy.sol
    │   │   └── libraries
    │   │       └── SimpleSerialize.sol
    │   ├── Reporter.sol
    ├── interfaces
    │   ├── IAdapter.sol
    │   ├── IBlockHashAdapter.sol
    │   ├── IHashi.sol
    │   ├── IHeaderStorage.sol
    │   ├── IJushin.sol
    │   ├── IMessage.sol
    │   ├── IMessageHashCalculator.sol
    │   ├── IMessageIdCalculator.sol
    │   ├── IReporter.sol
    │   ├── IShoyuBashi.sol
    │   ├── IShuSho.sol
    │   ├── IYaho.sol
    │   └── IYaru.sol
    ├── ownable
    │   ├── ShoyuBashi.sol
    │   └── ShuSo.sol
    └── utils
        ├── HeaderStorage.sol
        ├── MessageHashCalculator.sol
        └── MessageIdCalculator.sol
```

In addition, the audit includes the update of the tokenbridge-contracts developed in the following repository:

```
https://github.com/crosschain-alliance/tokenbridge-contracts
```

We audited the following changes:

1. The AMB upgrade, which is the diff between `feat/hashi-integration-amb` at commit `0cef7054be1be91203b09333aac8f7621b07afd5` and the `master` branch
2. The XDAI bridge upgrade, which is the diff between `feat/hashi-integration-xdai-bridge` at commit `7e60b0c46e168d1b73e766877c0d24cedbad6db6` and `xdaibridge-upgrade-sdai`

Resolution

After we submitted a preliminary report, most of the issues were addressed by the team.

We audited the fixes in the following commits:

- For the Hashi code: commit `0976475c04dd75c74b1a05e23aad1eafe2aa1258`
- For the AMB upgrade: commit `14e330ed2705539147b3bb7ac106589a8ae1473c`
- For the XDAI bridge upgrade: commit `fb6bae7589a102613b48c12addb425b72836574e`

Methods Used

Code Review

We manually inspected the source code to identify potential security flaws.

The contracts were compiled, deployed, and tested in a test environment.

Automatic analysis

We have used static analysis tools to detect common potential vulnerabilities. The tools have detected a number of low severity issues, concerning mostly the variables naming and external calls, were found. We have included any relevant issues below in the appropriate parts of the report.

Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

Severity definitions

High	Vulnerabilities that can lead to loss of assets or data manipulations.
Medium	Vulnerabilities that are essential to fix, but that do not lead to assets loss or data manipulations
Low	Issues that do not represent direct exploit, such as poor implementations, deviations from best practice, high gas costs, etc
Info	Matters of opinion

Findings

General

G1. Anyone can mark any messageId as approved by Hashi in the bridges [high] [resolved]

The bridges now implement the `IJushin` interface which defines an `onMessage` function. This function is called by Yaru as part of the `executeMessages` function. An attacker can call Yaru's `executeMessages` with a `message` of their choosing and mark any messageId as "approved by Hashi" in the bridge.

Specifically, an attacker can call Yaru `executeMessages` function with a `message` that has a single adapter that is controlled by the attacker, and a threshold set to 1. The malicious adapter is crafted in such a way that it will return any arbitrary hash on the `getHash` function. Because there is no validation on the adapters and threshold used, this is enough to pass the Hashi validation. Yaru will then call the `onMessage` function on the bridge. This function validates that the message was sent by the registered Yaru instance, and that the target chain ID and `message.sender` match those defined in the Hashi Manager contract.

The bridge then constructs a message ID from the `message.data`, and marks that ID as approved by Hashi.
Since there is no restriction at all on the `message.data` that is passed with the message, this means anyone can mark *any* `messageId` on the bridge as approved by Hashi. In the AMB bridge upgrade, `mesageId` is simply read as the first 32 bytes of `message.data`, so it is very easy to approve any message ID. In the XDAI bridge, the `messageId` is the hash of `message.data`, so it is a bit more difficult to construct an arbitrary ID, but an attacker could approve any known hashes.
All this renders the Hashi integration completely ineffective in adding security to the bridge process.

This vulnerability applies to the the following bridge implementation:

- In the XDAI upgrade, to both `BasicForeignBride.onMessage` and `BasicHomeBridge.onMessage`
- In the AMB upgrade, in `BasicHomeAMB.onMessage` and `BasicForeignAMB.onMessage`

Recommendation: It is necessary for the bridges to validate the message properly. For that, the bridge must try to reconstruct the message with the parameters it expected to be used, like the correct threshold and adapters, and ensure that the messageId of the reconstructed message matches the `messageId` passed in the `onMessage` function. But for bridges to reconstruct the message, it is necessary to also pass the message nonce in the `onMessage` function called from Yaru. The Yaho address is also needed, but can be either pre-saved or read from the Yaru contract.
Severity: High
Resolution: The issue was resolved. Yaru now also sends the adapters and threshold used in the message, and the bridge validates they match the threshold and adapters expected in the HashiManager.

G2. Test are failing, and test coverage is incomplete [low] [not resolved]

Coverage of the Hashi code is very incomplete: `npm run coverage` outputs that:

46.95% Statements `277/590`
45.21% Branches `198/438`
38.66% Functions `75/194`
46.67% Lines `386/827`

The missing test coverage in Hashi regards almost exclusively the adapters.
We also note that the CI process on github is failing.

With regard to the bridge contracts, we executed the tests locally on the AMB branch and got the same results as the github CI process: there are 43 out of 410 tests that fai.

Recommendation: Fix the tests and the CI processes. Try to have a complete coverage of all the code.
Severity: Low
Resolution: The issue was not resolved.

G3. Pin solidity dependencies [low] [resolved]

In `package.json` in the Hashi repository, the Solidity dependencies are defined as a range, and so do not specify that a particular release needs to be used by anyone who installs the project and compiles the contracts.

```
"@openzeppelin/contracts-upgradeable": "^4.8.1",
"@axelar-network/axelar-gmp-sdk-solidity": "^5.6.2",
"@chainlink/contracts-ccip": "^0.7.6",
"@connext/interfaces": "^2.0.0",
"@hyperlane-xyz/core": "^3.1.10",
"@polytope-labs/solidity-merkle-trees": "^0.2.1",
"@routerprotocol/evm-gateway-contracts": "^1.1.13",
"solidity-rlp": "^2.0.8"
```

With these settings, the building and compiling process is not completely deterministic, as `npm` or `yarn` can, and will if not explicitly instructed to do otherwise, use the latest versions. It is recommended to provide specific versions of the solidity dependencies. This will make it easier for future developers who want to build on the contracts to verify their deployment on-chain.
In addition, we recommend reorganizing `package.json` and put the Solidity dependencies that are compiled into the contracts under “`dependencies`”, and the other packages under “`devDependencies`”.
Recommendation: Pin the version of Solidity dependencies, and move non-Solidity dependencies to the `devDependencies.`
Severity: Low
Resolution: The issue was resolved as recommended.

G4. Ether can remain locked in contracts [low] [not resolved]

A number of contracts have payable functions, but no way to withdraw the ether. Any ether sent with these functions will effectively be lost.

The contracts affected are:

```
Yaho
```

`CelerAdapter`
`HyperlaneAdapter`
and all the contracts that implement `IReporter`

Recommendation: Remove the payable marking from the functions.
Severity: Low
Resolution: The issue was not resolved. The developers acknowledged the issue.

G5. ShoyuBashi is unused [info] [not resolved]

The bridges are using a new contract called `HashiManager` in order to manage parameters such as trusted adapters and a required threshold. This contract seems to be a less sophisticated, minimal version of the `ShoyuBashi` contract, and it lacks safety checks that are implemented in `ShoyuBashi`, such as ensuring no duplication of adapters occurs, and a valid threshold is given.
The `ShoyuBashi` contract is not used in any part of the code we audited.
Recommendation: Use the `ShoyuBashi` contract instead of the `HashiManager` contract.
Severity: Info
Resolution: The issue was not resolved.

G6. Consider following the ERC-5164 standard [info] [not resolved]

The ERC-5164 standard for cross-chain execution defines a standard for sending cross chain messages, which could be utilized by Hashi to improve future interoperability.
Recommendation: As Hashi aims to be a generic, modular, system, it makes sense to try and follow the ERC-5164 standard for dispatching and executing messages. Cf. https://eips.ethereum.org/EIPS/eip-5164
Severity: Info
Resolution: The issue was not resolved.

Hashi.sol

H1. Code duplication between checkHashWithThresholdFromAdapters and getHashesFromAdapters [info] [resolved]

The function `checkHashWithThresholdFromAdapters` checks that the adapters array length is not 0 in line 47, then in lines 50 to 56 it gets the hashes from all the adapters. All this code is duplicated in the public function `getHashesFromAdapters`, which is a helper function that does the same exact thing.

Recommendation: Use the getHashesFromAdapters instead of the code in lines 47, 50 - 56 to simplify your code
Severity: Info
Resolution: The issue was resolved as recommended.

Yaho.sol

YH1. Avoid unnecessary writing to storage on _dispatchMessage [info] [resolved]

The _dispatchMessage function is used 3 times in the code. Among other things, the function saves the messageHash in the _pendingMessageHashes mapping.
Out of these 3 times it is used, in 2 of them the messageHash is immediately deleted from the _pendingMessageHashes mapping, which means there was no reason to save it in the first place.
Instead saving the messageHash to the mapping could be moved to the dispatchMessage function, which will allow removing the costly write and delete pattern in the 2 other instances where _dispatchMessage is used.
Recommendation: Move the writing of the messageHash from the _dispatchMessage to the dispatchMessage function, and remove the _resetPendingMessageHashesByMessageIds function entirely from the code.
Severity: Info
Resolution: The issue was resolved as recommended.

ShuSo.sol

SS1. Init pattern for setHashi is overcomplicated [info] [resolved]

The init function calls the setHashi function, which is not implemented in the contract, but is defined virtually. The ShoyuBashi contract which implements it simply calls the _setHashi function of ShuSo. It is not clear why this pattern is used, since there seems to be no need for customizing the functionality of setting the Hashi address which is expected to always just call the _setHashi function, and so the could could be simplified to call the _setHashi directly.
Recommendation: In the init function call _setHashi, and remove the setHashi function from the ShuSo and ShoyuBashi code.
Severity: Info
Resolution: The issue was resolved as recommended. The setHashi function was removed from ShuSo and is now only callable externally in ShoyuBashi.

SS2. Remove unused function _setDomainThreshold [info] [resolved]

The `_setDomainThreshold` function is not used anywhere in the codebase, and performs the same functionality as the `_setThreshold` function, but without doing any of the safety checks, which could be dangerous in the future.
Recommendation: Remove the `_setDomainThreshold` function.
Severity: Info
Resolution: The issue was resolved as recommended.

SS3. _setThreshold does not verify new threshold is valid compared to count [info] [not resolved]

The `_setThreshold` function verifies that the new threshold is bigger than half of the count of adapters, but it does not verify that the new threshold is not bigger than the count, which should also be invalid.
Recommendation: Add a check to the `_setThreshold` function to ensure the new threshold is not bigger than the count of adapters. This will also allow you to safely remove the checks that the threshold is not bigger than the adapters count in lines 157 and 194.
Severity: Info
Resolution: The issue was not resolved. The developers indicated that this is by design,

SS4. Declare functions as external instead of public when possible [info] [resolved]

The `getAdapterLink` and `getDomain` functions are marked as `public`, but are not used within the contract and can be declared `external`.
Recommendation: Mark `getAdapterLink` and `getDomain` as `external`.
Severity: Info
Resolution: The issue was resolved as recommended.

SS5. _getThresholdHash may choose arbitrarily between multiple valid hashes [info] [resolved]

The `_getThresholdHash` function returns the first hash that is agreed upon by a threshold of the adapters enabled, yet in case the threshold is not higher than half the count of the adapters, there could be multiple hashes that would qualify, and the one returned will essentially be arbitrary.

Recommendation: The `_setThreshold` function checks that the threshold must be higher than half the count of the adapters, but this is not enforced when new adapters are enabled. We recommend ensuring this requirement holds when the adapters list is updated. Alternatively, consider making the function revert if multiple hashes superate the threshold.
Severity: Info
Resolution: The issue was resolved as recommended. The `_enableAdapters` function now takes a `threshold` argument and validates it is more than half of the adapters count.

DendrETHAdapter.sol

DEA1 Anyone can set the chainId for the hash stored by the adapter [low] [resolved]

The `storeBlockHeader` functions receive a `_chainId` parameter which is used as the chainId for storing the hash. Yet there are no limits or checks on the value of the `_chainId` parameter, which means anyone calling the function could set it to any value they choose, allowing them to save data into the wrong domain.
This can lead to confusion, as the adapter will misreport a blockhash when queried with an `_chainId` that is different from the chain that `dendrETHAddress` is reporting from.
Recommendation: Hardcode the `chainId` to be stored like in the Electron adapter (or read it from the contract at `dendrETHAddress`)
Severity: Low
Resolution: The issue was resolved as recommended.

XDai Branch - BasicForeignBridge.sol

XBFB1. Unused functions can be removed [info] [resolved]

The `canBeExecuted` and `_setMessageToExecute` functions were added to the contract but are not used, and it is not clear what is their purpose.
Recommendation: Remove the used `canBeExecuted` and `_setMessageToExecute` functions.
Severity: Info
Resolution: The issue was resolved as recommended.