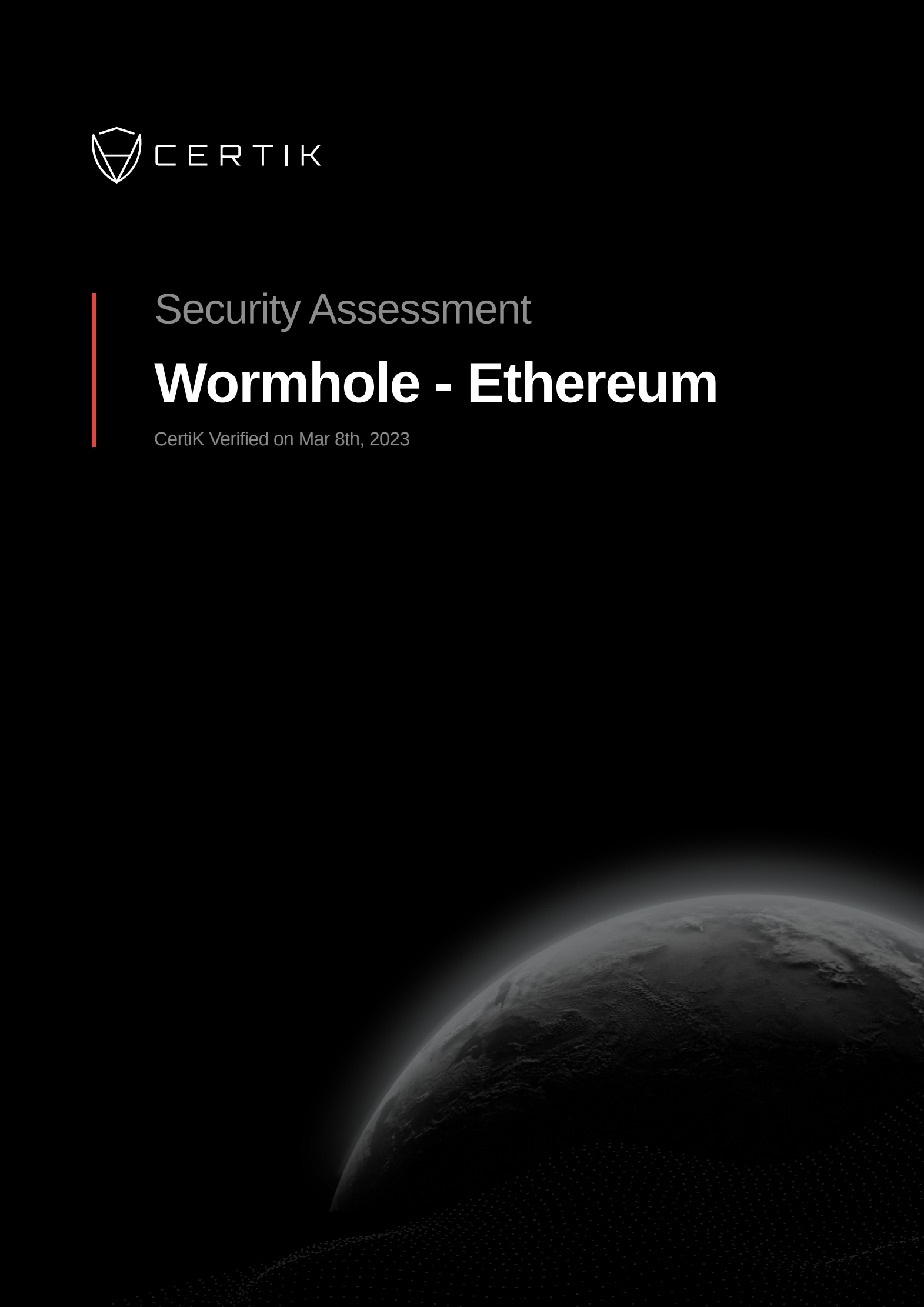
CERTIK

Security Assessment

Wormhole - Ethereum

CertiK Verified on Mar 8th, 2023



CertiK Verified on Mar 8th, 2023

Wormhole - Ethereum

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
Bridge	Ethereum	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 03/08/2023	N/A

CODEBASE

https://github.com/certusone/wormhole

...View All

COMMITS

- 545f35ed3b9ef15cb189936848b6f6578458466f
- 58cd031ea878c44359d4db244f6936d92b47ab7a

...View All

Vulnerability Summary

12	1	0	0	11	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
0	Major		Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
1	Medium	1 Acknowledged	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
4	Minor	1 Resolved, 3 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
7	Informational	7 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | WORMHOLE - ETHEREUM

Disclaimer

CODEBASE | WORMHOLE - ETHEREUM

Repository

https://github.com/certusone/wormhole

Commit

- 545f35ed3b9ef15cb189936848b6f6578458466f
- 58cd031ea878c44359d4db244f6936d92b47ab7a

AUDIT SCOPE | WORMHOLE - ETHEREUM

38 files audited ● 38 files with Acknowledged findings

ID	File	SHA256 Checksum
IWB	contracts/interfaces/IWormhole.sol	7307fccee8d2f9fbe51e95d10822d3e386fa60cd1d721561ac58d2ade5df750b
BLB	contracts/libraries/external/BytesLib.sol	1b6f2ba238f9af311f917ddbf412edc565cfde02398d08727e8bbb98ad14d819
GET	contracts/Getters.sol	91d24680fc1885a1004de52b0f4a28501a2d630713c056cb9b83a1f2e92c44dd
GOV	contracts/Governance.sol	fec9ef082f1a655060bacb9ee1151dcd698bdeaaeb6880e58a40213f9e822cbc
GSB	contracts/GovernanceStructs.sol	ea357d4da8221fda40832faa5bbef4dbbf967455fb9b2b5757584e1cc092a73c
IMP	contracts/Implementation.sol	cf5bb644f3c5644a3fa34c6e605f8e069e220ebf265782bf7404c25444d933bc
MES	contracts/Messages.sol	cee1f1afebdff839c223a2932e5e973adb8c6495d0fa86282ac2ece33914de27
MIG	contracts/Migrations.sol	0b5adea0a2aac87b2a6df5c2cc62761dcaa33f6a42d72e1df6be46e9448d874b
SET	contracts/Setters.sol	5ddca9c7addeea7e4c95459b3125ffc4456ef942dd4929bd0ed82d1fe54335e9
SEU	contracts/Setup.sol	8602d05c8d48dce15f9788dff708a8c78035553e2b2f134af4b9815bf78a6bb4
STA	contracts/State.sol	ab237ec95c2e4dc6ca650ea4f3d8874111fdd53b452406578e14e15313b634bd
STR	contracts/Structs.sol	d6da02e4ddf08e94417e007863b4b8904084481e83587acfe6b134061ee1a98a
WOR	contracts/Wormhole.sol	5e57e8d9cf7cf0738e1404e57e18cd3f21e81b703eafaaa18cbad3ed57b7e9f2
TON	contracts/bridge/token/Token.sol	ae3b3827585188c27bb6a35a43ed268e03b67ff58f3c2d90ed5176d430393a34

ID	File	SHA256 Checksum
TIB	contracts/bridge/token/TokenImplementation.sol	dc15fda91189bb6b7b4bf6211006824531 32f8050280583630c30eeefb82009c
TSB	contracts/bridge/token/TokenState.sol	fbd432ef2cd30a508acd845a822533a502421b7cfb2496432f8acbb1c258f27b
MIR	contracts/bridge/utils/Migrator.sol	42919a9b7fe93b7ca493ab705bb37bb8622496c7d2516eee8f36db95ab94dccf
BRI	contracts/bridge/Bridge.sol	a6f7d45ee2cb24761789207d4f66b185d5cb0e4994a4f8353728ec47294b2378
BGB	contracts/bridge/BridgeGetters.sol	19b0d56634dc58d53a010b721723a7c4ace5ccaa9f037164e724cdaf793e9158
BGU	contracts/bridge/BridgeGovernance.sol	420551c9ee4ae8d4abefe0fda157b1522cac9ee7c8307451c7f091820ae9ba65
BIB	contracts/bridge/BridgeImplementation.sol	ed6c4ed1cd7b63e1deaa2ea947643f8b86c19667ca5aabc4dc72bcb0d6351e02
BSB	contracts/bridge/BridgeSetters.sol	376e35c5bd4ebeda1b4283a512211f6dcd49abeab9a73a6171b3553935ac5ed7
BSU	contracts/bridge/BridgeSetup.sol	72402994988e8caaebf2a825543e15e6523b5ad24d2ededc12bc7fa448369d2e
BSH	contracts/bridge/BridgeState.sol	2654d563a5aff82c8ba131e5cc1d6ec8aa21f4b762265707081ea62c057e3073
BST	contracts/bridge/BridgeStructs.sol	1c194d6f396415d2245ba3357ed0709b67687d0cdc29d15177b6cd15dd82a0d2
TBB	contracts/bridge/TokenBridge.sol	8f5e8bafb11f447b0e2455a8ecd6d00afc5ba57f4c7eec516f445584ea0bce21
NFO	contracts/nft/token/NFT.sol	6cb40d0fa1709a5b68a8e76f6511213c6a50cd00095afb0ba305bf04f4f51b7c
NTI	contracts/nft/token/NFTImplementation.sol	4108781e6d6cc19b7f6a4f97ee01c9bb6064f9e1bac66b87caa66c4a33bc9bc9
FTS	contracts/nft/token/NFTState.sol	03810d3d45d33ac801fc78367ec92ec1a7de0e9e80f1238ceea2a27c1630b35e
NFT	contracts/nft/NFTBridge.sol	bbeb744ba59042a0be5595476003cd72747087a9811a5935823bdbc49d05e9e6
NFB	contracts/nft/NFTBridgeEntrypoint.sol	967f4151c6e47ff6aa80157b013f5eb9ba90ff867fba8f1c7c822371cffce343

ID	File	SHA256 Checksum
NFG	contracts/nft/NFTBridgeGetters.sol	f012f6b8b4dea0b22d6488800d0be46b25faa3bfaac0904141574971d694c930
NTB	contracts/nft/NFTBridgeGovernance.sol	337f6484eb9868c3a6fcd7e1ea55d3474d5baa762ee6baf1c4db56ebf7d2f3ef
NFI	contracts/nft/NFTBridgeImplementation.sol	612f851d8edc2d652fb860e67ca542e2946928cde4ddd54bfcdf1e9ee9a9df01
NFS	contracts/nft/NFTBridgeSetters.sol	586fe4312c62d73c736a34497029d0bca22eaf568aef573f9af22269530461e0
NTS	contracts/nft/NFTBridgeSetup.sol	15381c24904a2d2b94cc10c9e2e1d5414d40f5230992c82d4b6ef80e2e5a25b1
NBS	contracts/nft/NFTBridgeState.sol	60870694131adbf3f6f2084978615705afb6cf4098961ebbea2a1a1d6fd4d132
FTB	contracts/nft/NFTBridgeStructs.sol	cb5a60b4bcf3f1518762f811fc37a599cfa11385a9a2509079c68418159a3046

APPROACH & METHODS | WORMHOLE - ETHEREUM

This report has been prepared for Wormhole to discover issues and vulnerabilities in the source code of the Wormhole - Ethereum project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

REVIEW NOTES | WORMHOLE - ETHEREUM

Overview

The `Wormhole` is a generic message-passing protocol that connects to multiple chains including Ethereum, Solana, Terra, Binance Smart Chain, Polygon, Avalanche, Oasis, Fantom, Karura, Celo, and Aurora.

The Wormhole Ethereum project concerns the Wormhole bridge on many EVM-compatible blockchains, such as Ethereum, BSC, Polygon, and so on. It includes three components:

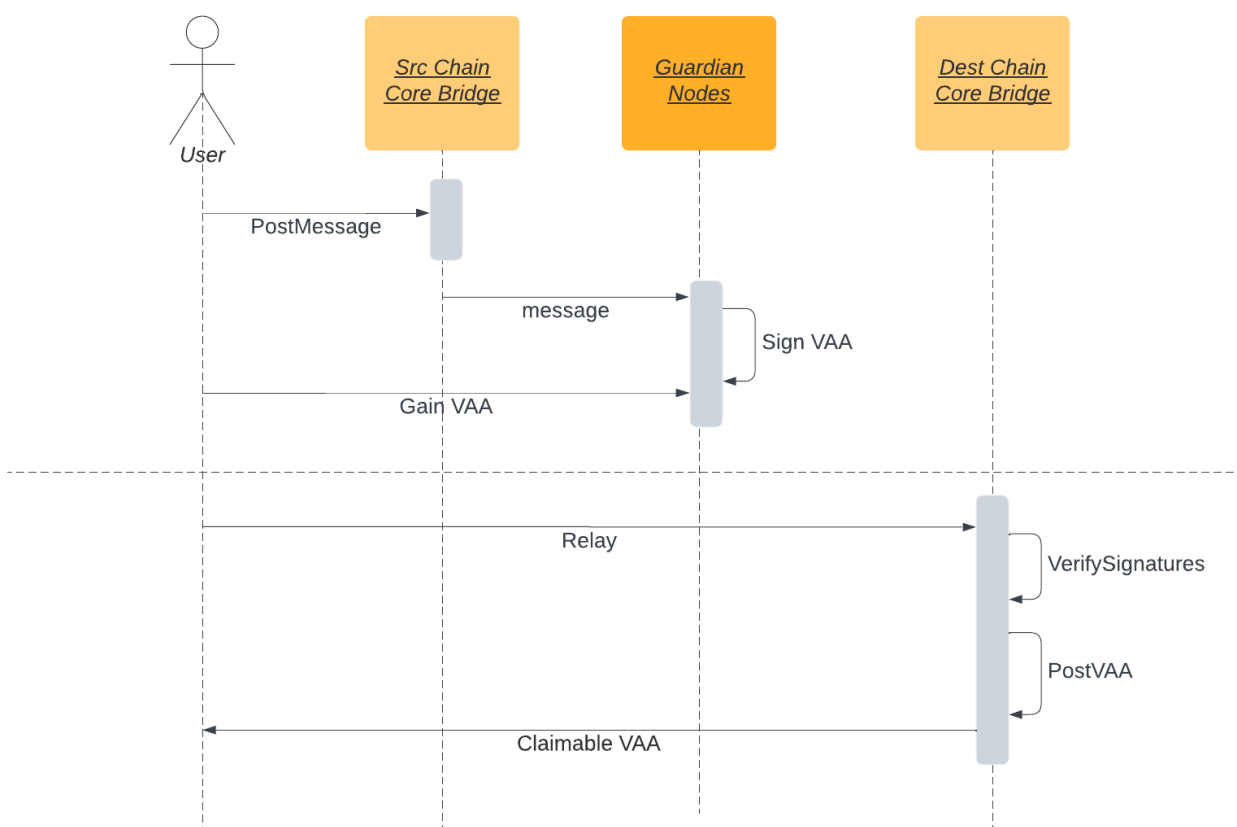
- Core Bridge
- Token Bridge
- NFT Bridge

Core Bridge

The core bridge is the backbone component for the Wormhole protocol on each chain. It mainly maintains two important functionalities:

- `post message`: post messages and allow the guardian network to observe and verify;
- `verify VAA`: verify signed VAAs from the relayer.

Here is an example of the workflow from one chain to another:



Governance Flows

The core bridge contains the following governance flows:

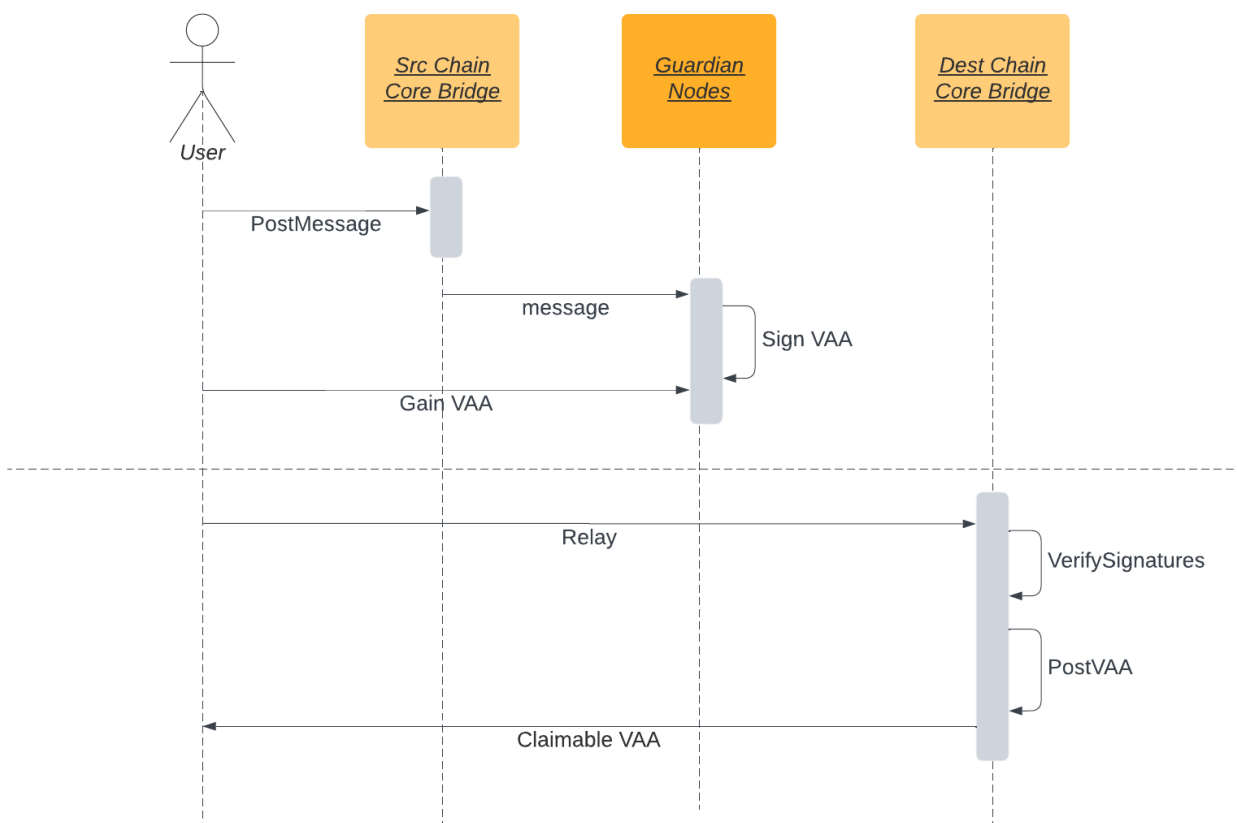
- Upgrade the core contract
- Config message fee
- Upgrade the new guardian set
- Withdraw transfer fees

Token Bridge

The token bridge is an application built on top of the core bridge. It has the following functionalities:

- Attest a token to the target chain
- Create/Update a wrapped token
- Transfer native or ERC20 tokens
- Complete native or ERC20 tokens transfers

Here is an example of the workflow from one chain to another:



Governance Flows

The token bridge contains the following governance flows:

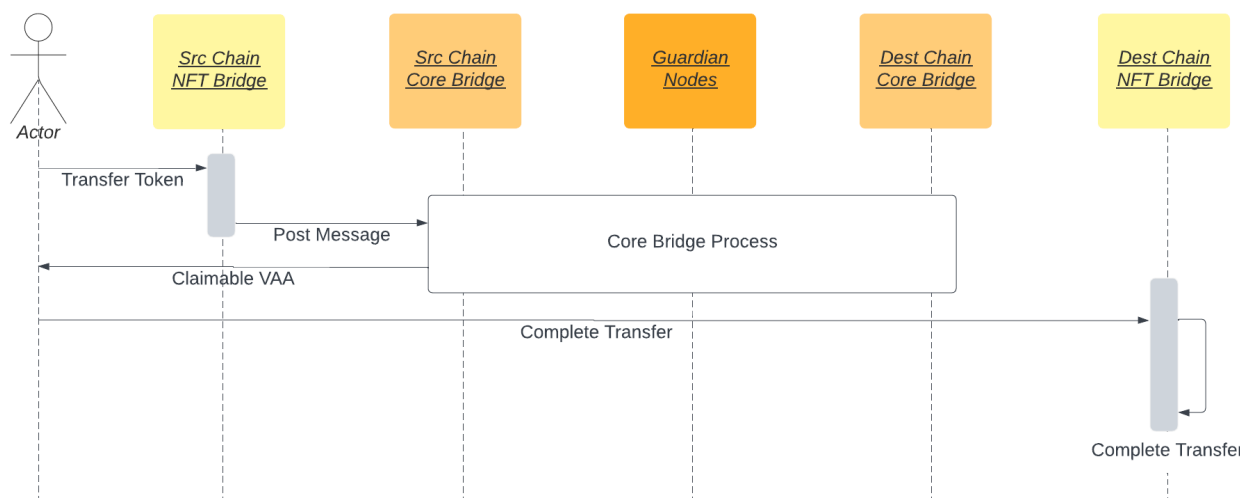
- Register a new blockchain
- Upgrade the token bridge contract

NFT Bridge

The NFT bridge is another application built on top of the core bridge. It has the following functionalities:

- Transfer an NFT
- Complete an NFT transfer

Here is an example workflow:



Governance Flows

The NFT bridge contains the same governance flows as the token bridge.

Privileged Functions

The core bridge, token bridge, and NFT bridge highly rely on the Wormhole network, especially the guardian nodes.

Once compromised guardian nodes reach the threshold for consensus (2/3 of all guardian nodes) the whole bridge will be compromised. Hence, it is important for all guardians to manage their keypairs.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified as soon as possible to the community.

FINDINGS | WORMHOLE - ETHEREUM

12	0	0	1	4	7
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Wormhole - Ethereum. Through this audit, we have uncovered 12 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
BRI-01	Contract Gains Non-Withdrawable Tokens Via The Function `_transferTokens()`	Logical Issue	Medium	Acknowledged
BRI-02	Lack Of Input Validation For `payload`	Volatile Code	Minor	Acknowledged
CON-01	Usage Of `transfer()` For Sending Ether	Volatile Code	Minor	Acknowledged
CON-08	Lack Of Access Control	Control Flow	Minor	Resolved
GSB-01	Not Checking Duplicated Addresses	Volatile Code	Minor	Acknowledged
CON-05	Redundant Statements	Volatile Code	Informational	Acknowledged
CON-06	Declaration Naming Convention	Coding Style	Informational	Acknowledged
CON-07	Different Solidity Versions	Language Specific	Informational	Acknowledged
MES-01	Missing Check For `v` And `s`	Volatile Code	Informational	Acknowledged
MIR-02	Incompatibility With Deflationary Tokens	Volatile Code	Informational	Acknowledged
NBS-01	Mismatch Between Comment And Code	Coding Style	Informational	Acknowledged

ID	Title	Category	Severity	Status
SET-01	Guardian Expiration Time	Inconsistency	Informational	Acknowledged

BRI-01 CONTRACT GAINS NON-WITHDRAWABLE TOKENS VIA THE FUNCTION `_transferTokens()`

Category	Severity	Location	Status
Logical Issue	Medium	contracts/bridge/Bridge.sol (2022/06/15 - 545f3): 160	Acknowledged

Description

The `_transferTokens()` function normalizes the transferred token amount by setting the last 8 digit as zero via `deNormalizeAmount()` and `normalizeAmount()`.

```
136  // don't deposit dust that can not be bridged due to the decimal shift
137  amount = deNormalizeAmount(normalizeAmount(amount, decimals), decimals);
```

When transferring a standard ERC20 token, the normalization in L137 will ensure the transferred amount (calculated in L152) aligns with the `transferResult` specified in L171.

```
152              amount = balanceAfter - balanceBefore;
```

```
168          transferResult = BridgeStructs.TransferResult({
169              tokenChain : tokenChain,
170              tokenAddress : tokenAddress,
171              normalizedAmount : normalizedAmount,
172              normalizedArbiterFee : normalizedArbiterFee,
173              wormholeFee : msg.value
174          });
```

However, when transferring tokens that charge fees during transfer (i.e., deflationary tokens), the normalizing process in L137 cannot ensure the amount calculated in L152 is normalized because of the fee charged. Therefore, normalizing the result in L160 could lead to a potential precision loss.

```
160          uint256 normalizedAmount = normalizeAmount(amount, decimals);
```

In this case, a small number of deflationary tokens will be left in the contract. Since the contract does not provide a method to withdraw these locked tokens, these tokens will be locked in the contract forever.

Recommendation

It is not ideal that more and more deflationary tokens are locked in the contract over time. The simplest solution is to add a `withdraw` function in the contract to withdraw corresponding tokens periodically.

Alleviation

[Wormhole Team, 08/28/2022]:

The team stated the fee-burning / rebasing / non-standard tokens are explicitly unsupported by the token bridge.

Reference: https://github.com/certusone/wormhole/blob/dev.v2/whitepapers/0003_token_bridge.md

BRI-02 | LACK OF INPUT VALIDATION FOR `payload`

Category	Severity	Location	Status
Volatile Code	Minor	contracts/bridge/Bridge.sol (2022/06/15 - 545f3): 69~73, 114~115	Acknowledged

Description

In the Wormhole guardian node, `Unmarshal(data []byte)` from `structs.go` will get the first 1000 bytes from the payload.

```
func Unmarshal(data []byte) (*VAA, error) {
//...
    payload := make([]byte, 1000)
    n, err := reader.Read(payload)
    if err != nil || n == 0 {
        return nil, fmt.Errorf("failed to read payload [%d]: %w", n, err)
    }

    v.Payload = payload[:n]

    return v, nil
}
```

In addition, there is also a size limit of 1232 bytes for transactions containing the `payload` on Solana, see Solana Facts. However, there is no size limit for the input `payload` in `transferTokensWithPayload` or `wrapAndTransferETHWithPayload` and thus part of the `payload` info may be lost when doing a cross-chain transaction.

Recommendation

The auditing team recommends adding proper size checks to avoid unexpected errors.

Alleviation

[Wormhole Team, 03/08/2023]:

The team acknowledged this issue and decided not to change the codebase in the current stage.

CON-01 | USAGE OF `transfer()` FOR SENDING ETHER

Category	Severity	Location	Status
Volatile Code	Minor	contracts/Governance.sol (2022/06/15 - 545f3): 88; contracts/bridge/Bridge.sol (2022/06/15 - 545f3): 89, 366, 386	Acknowledged

Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

Recommendation

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

Alleviation

[Wormhole Team, 03/08/2023]:
The team acknowledged this issue and decided not to change the codebase in the current stage.

CON-08 | LACK OF ACCESS CONTROL

Category	Severity	Location	Status
Control Flow	Minor	contracts/Setup.sol (2022/06/15 - 545f3): 12~35; contracts/bridge/BridgeSetup.sol (2022/06/15 - 545f3): 12~33; contracts/nft/NFTBridgeSetup.sol (2022/06/15 - 545f3): 12~19	Resolved

Description

The `setup` function from `Setup`, `BridgeSetup`, and `NFTBridgeSetup` is a public function and lacks of access control, which can be called many times by anyone.

Recommendation

The auditing team would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

[Wormhole Team, 03/03/2023]:

The Wormhole team declares that these setup functions post-initial deployment are no longer accessible by design.

GSB-01 | NOT CHECKING DUPLICATED ADDRESSES

Category	Severity	Location	Status
Volatile Code	Minor	contracts/GovernanceStructs.sol (2022/06/15 - 545f3): 96~99	Acknowledged

Description

The `parseGuardianSetUpgrade()` function from `GovernanceStructs.sol` decodes `encodedUpgrade` to acquire a new guardian set.

```
96          for(uint i = 0; i < guardianLength; i++) {
97              gsu.newGuardianSet.keys[i] = encodedUpgrade.toAddress(index);
98              index += 20;
99          }
```

However, if many duplicate guardian addresses exist in the new guardian set and the contract successfully upgrades to the new guardian set, it might increase the risk of an attacker compromising the multi-signature validation in the `verifySignatures` function.

Recommendation

The auditing team recommends checking the duplicate addresses when upgrading to a new guardian set.

Alleviation

[Wormhole Team, 03/03/2023]:

The Wormhole team declares that governance messages are generated by the Guardian, which does ensure this behavior in the `verifySignatures` code.

CON-05 | REDUNDANT STATEMENTS

Category	Severity	Location	Status
Volatile Code	Informational	contracts/Governance.sol (2022/06/15 - 545f3): 15; contracts/State.sol (2022/06/15 - 545f3): 8~19; contracts/bridge/BridgeState.sol (2022/06/15 - 545f3): 16~19; contracts/nft/NFTBridgeState.sol (2022/06/15 - 545f3): 15~18	Acknowledged

Description

```
14      event LogMessagePublished(
15          address emitter_address,
16          uint32 nonce,
17          bytes payload
18      );
```

- `LogMessagePublished` is declared in `Events` but never emitted.

```
15      event GuardianSetAdded(uint32 indexed index);
```

- `GuardianSetAdded` is declared in `Governance` but never emitted.

```
 9      event LogGuardianSetChanged(
10          uint32 oldGuardianIndex,
11          uint32 newGuardianIndex
12      );
```

- `LogGuardianSetChanged` is declared in `Events` but never emitted.

```
    struct Asset {
        uint16 chainId;
        bytes32 assetAddress;
    }
```

The linked statements above do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

Recommendation

The auditing team recommends removing the redundant statements.

Alleviation

[Wormhole Team, 03/08/2023]:
The team acknowledged this issue and decided not to change the codebase in the current stage.

CON-06 | DECLARATION NAMING CONVENTION

Category	Severity	Location	Status
Coding Style	Informational	contracts/Governance.sol (2022/06/15 - 545f3): 18; contracts/bridge/BridgeGovernance.sol (2022/06/15 - 545f3): 25; contracts/nft/NFTBridgeGovernance.sol (2022/06/15 - 545f3): 24	Acknowledged

Description

One or more declarations do not conform to the Solidity style guide with regards to its naming convention.

Particularly:

- `UPPER_CASE` : Should be applied to `constant` variables

```
18      bytes32 constant module =
0x00000000000000000000000000000000000000000000000000000000436f7265;
```

- Constant variable `module` is not in `UPPER_CASE` .

```
24      bytes32 constant module =
0x000000000000000000000000000000000000000000000004e4654427269646765;
```

- Constant variable `module` is not in `UPPER_CASE` .

```
25      bytes32 constant module =
0x00000000000000000000000000000000000000000000546f6b656e427269646765;
```

- Constant variable `module` is not in `UPPER_CASE` .

Recommendation

The auditing team recommends adjusting those variable and function names to properly conform to Solidity's naming convention.

Alleviation

[Wormhole Team, 03/08/2023]:

The team acknowledged this issue and decided not to change the codebase in the current stage.

CON-07 | DIFFERENT SOLIDITY VERSIONS

Category	Severity	Location	Status
Language Specific	Informational	contracts/Getters.sol (2022/06/15 - 545f3): 4; contracts/Governance.sol (2022/06/15 - 545f3): 4; contracts/GovernanceStructs.sol (2022/06/15 - 545f3): 4; contracts/Implementation.sol (2022/06/15 - 545f3): 4, 5; contracts/Messages.sol (2022/06/15 - 545f3): 4, 5; contracts/Migrations.sol (2022/06/15 - 545f3): 2; contracts/Setters.sol (2022/06/15 - 545f3): 4; contracts/Setup.sol (2022/06/15 - 545f3): 4, 5; contracts/State.sol (2022/06/15 - 545f3): 4; contracts/Structs.sol (2022/06/15 - 545f3): 4; contracts/Wormhole.sol (2022/06/15 - 545f3): 4; contracts/bridge/Bridge.sol (2022/06/15 - 545f3): 4; contracts/bridge/BridgeGetters.sol (2022/06/15 - 545f3): 4; contracts/bridge/BridgeGovernance.sol (2022/06/15 - 545f3): 4; contracts/bridge/BridgeImplementation.sol (2022/06/15 - 545f3): 4, 5; contracts/bridge/BridgeSetters.sol (2022/06/15 - 545f3): 4; contracts/bridge/BridgeSetup.sol (2022/06/15 - 545f3): 4, 5; contracts/bridge/BridgeState.sol (2022/06/15 - 545f3): 4; contracts/bridge/BridgeStructs.sol (2022/06/15 - 545f3): 4; contracts/bridge/TokenBridge.sol (2022/06/15 - 545f3): 4; contracts/bridge/token/Token.sol (2022/06/15 - 545f3): 4; contracts/bridge/token/TokenImplementation.sol (2022/06/15 - 545f3): 4; contracts/bridge/token/TokenState.sol (2022/06/15 - 545f3): 4; contracts/bridge/utils/Migrator.sol (2022/06/15 - 545f3): 4, 5; contracts/interfaces/IWormhole.sol (2022/06/15 - 545f3): 4; contracts/libraries/external/BytesLib.sol (2022/06/15 - 545f3): 9; contracts/nft/NFTBridge.sol (2022/06/15 - 545f3): 4; contracts/nft/NFTBridgeEntrypoint.sol (2022/06/15 - 545f3): 4; contracts/nft/NFTBridgeGetters.sol (2022/06/15 - 545f3): 4; contracts/nft/NFTBridgeGovernance.sol (2022/06/15 - 545f3): 4; contracts/nft/NFTBridgeImplementation.sol (2022/06/15 - 545f3): 4, 5; contracts/nft/NFTBridgeSetters.sol (2022/06/15 - 545f3): 4; contracts/nft/NFTBridgeSetup.sol (2022/06/15 - 545f3): 4, 5; contracts/nft/NFTBridgeState.sol (2022/06/15 - 545f3): 4; contracts/nft/NFTBridgeStructs.sol (2022/06/15 - 545f3): 4; contracts/nft/token/NFT.sol (2022/06/15 - 545f3): 4; contracts/nft/token/NFTImplementation.sol (2022/06/15 - 545f3): 4; contracts/nft/token/NFTState.sol (2022/06/15 - 545f3): 4	Acknowledged

Description

Multiple Solidity versions are used in the codebase.

Versions used: `^0.8.0`, `^0.8.2`, `>=0.8.0<0.9.0`, `>=0.4.22<0.9.0`, `^0.8.1`

Other directives used: `ABIEncoderV2`

`^0.8.0` is used in ethereum/contracts/bridge/BridgeGovernance.sol file.

```
4  pragma solidity ^0.8.0;
```

`^0.8.2` is used in ethereum/node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol file.

```
4  pragma solidity ^0.8.2;
```

`>=0.8.0<0.9.0` is used in ethereum/contracts/libraries/external/BytesLib.sol file.

```
9  pragma solidity >=0.8.0 <0.9.0;
```

`>=0.4.22<0.9.0` is used in ethereum/contracts/Migrations.sol file.

```
2  pragma solidity >=0.4.22 <0.9.0;
```

`^0.8.1` is used in ethereum/node_modules/@openzeppelin/contracts/utils/Address.sol file.

```
4  pragma solidity ^0.8.1;
```

`ABIEncoderV2` is used in ethereum/contracts/Messages.sol file.

```
5  pragma experimental ABIEncoderV2;
```

Recommendation

The auditing team recommends using one Solidity version.

Alleviation

[Wormhole Team, 03/08/2023]:

The team acknowledged this issue and decided not to change the codebase in the current stage.

MES-01 | MISSING CHECK FOR `v` AND `s`

Category	Severity	Location	Status
Volatile Code	Informational	contracts/Messages.sol (2022/06/15 - 545f3): 115~120	Acknowledged

Description

EIP-2 still allows signature malleability for `ecrecover()`. Remove this possibility and make the signature unique. Appendix F in the Ethereum Yellow paper, defines the valid range for `s` in (281): `0 < s < secp256k1n ÷ 2 + 1`, and for `v` in (282): `v ∈ {27, 28}`. Most signatures from current libraries generate a unique signature with an s-value in the lower half order.

If your library generates malleable signatures, such as s-values in the upper range, calculate a new s-value with `0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141 - s1` and flip v from 27 to 28 or vice versa. If your library also generates signatures with 0/1 for v instead of 27/28, add 27 to v to accept these malleable signatures as well.

Recommendation

The auditing team recommends refactoring the linked statement as below:

```
  65  require(uint256(vm.signatures[i].s) <=
0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0, "ECDSA: invalid
signature 's' value");
  66  require(vm.signatures[i].v == 27 || vm.signatures[i].v == 28, "ECDSA: invalid
signature 'v' value");
```

Alleviation

[Wormhole Team, 03/08/2023]:

The team acknowledged this issue and decided not to change the codebase in the current stage.

MIR-02 | INCOMPATIBILITY WITH DEFLATIONARY TOKENS

Category	Severity	Location	Status
Volatile Code	Informational	contracts/bridge/utils/Migrator.sol (2022/06/15 - 545f3): 29~42	Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. For example, if `toAsset` is a deflationary token and a user adds 100 deflationary tokens (with a 10% transaction fee) to the contract, 100 LP tokens are minted but only 90 tokens actually arrived in the contract. However, when the user wants to burn 100 LP tokens and withdraw 100 tokens, the program blocks due to a lack of `toAsset` tokens.

Recommendation

The auditing team recommends regulating `toAsset` and `fromAsset` and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

[Wormhole Team, 03/08/2023]:
The team acknowledged this issue and decided not to change the codebase in the current stage.

NBS-01 | MISMATCH BETWEEN COMMENT AND CODE

Category	Severity	Location	Status
Coding Style	Informational	contracts/nft/NFTBridgeState.sol (2022/06/15 - 545f3): 49~50	Acknowledged

Description

The comment on `splCache` describes that `splCache` is a nested `mapping`, while the code implies otherwise.

Recommendation

The auditing team recommends correcting either the comment or the code.

Alleviation

[Wormhole Team, 03/08/2023]:

The team acknowledged this issue and decided not to change the codebase in the current stage.

SET-01 | GUARDIAN EXPIRATION TIME

Category	Severity	Location	Status
Inconsistency	Informational	contracts/Setters.sol (2022/06/15 - 545f3): 14	Acknowledged

Description

When expiring a guardian set, the guardian set is set to expire after a day.

```
13      function expireGuardianSet(uint32 index) internal {
14          _state.guardianSets[index].expirationTime = uint32(block.timestamp) +
86400;
15      }
```

However, the `WormholeState` struct has a `guardianSetExpiry` field, which decides how long it takes for guardian sets to expire. Since the value in this field may be different from a day, guardian sets may expire earlier or later than expected.

Recommendation

We recommend fixing the `guardianSetExpiry` to 86400 if it is not meant to be changed, otherwise changing the `expireGuardianSet()` function to use the value in `guardianSetExpiry`.

Alleviation

[Wormhole Team, 03/08/2023]:

The team acknowledged this issue and decided not to change the codebase in the current stage.

OPTIMIZATIONS | WORMHOLE - ETHEREUM

ID	Title	Category	Severity	Status
CON-02	Logical Issue Of The Check Of `messageFee` In Function `publishMessage()`	Logical Issue, Gas Optimization	Optimization	Acknowledged
CON-03	Function Should Be Declared External	Gas Optimization	Optimization	Acknowledged
CON-04	Inefficient Require Statement Location	Gas Optimization	Optimization	Acknowledged
MIR-01	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	Acknowledged

CON-02 LOGICAL ISSUE OF THE CHECK OF `messageFee` IN FUNCTION `publishMessage()`

Category	Severity	Location	Status
Logical Issue, Gas Optimization	Optimization	contracts/Implementation.sol (2022/06/15 - 545f3): 15; contracts/bridge/Bridge.sol (2022/06/15 - 545f3): 59, 206, 227; contracts/nft/NFTBridge.sol (2022/06/15 - 545f3): 95	Acknowledged

Description

According to the following code, the check on `messageFee` in the function `publishMessage()` uses `==` to check whether the `msg.value` is equal to the `messageFee`.

```
    function publishMessage(
        uint32 nonce,
        bytes memory payload,
        uint8 consistencyLevel
    ) public payable returns (uint64 sequence) {
        // check fee
        require(msg.value == messageFee(), "invalid fee");

        sequence = useSequence(msg.sender);
        // emit log
        emit LogMessagePublished(msg.sender, sequence, nonce, payload,
consistencyLevel);
    }
```

The function `publishMessage()` is called in the functions `Bridge.attestToken()`, `Bridge.logTransfer()`, `Bridge.logTransferWithPayload()` and `NFTBridge.logTransfer()`. In these calls, the `msg.value` is not checked or limited. As a result, if `msg.value` is not equal to `messageFee`, the call will be rolled back and all processes will be invalidated.

We would like to confirm with the client if the current implementation aligns with the original project design.

Recommendation

The auditing team recommends checking the `msg.value` in these calls or using `>=` to check `msg.value` and refunding the excess amount.

Alleviation

[Wormhole Team, 03/08/2023]:

The team acknowledged this issue and decided not to change the codebase in the current stage.

CON-03 | FUNCTION SHOULD BE DECLARED EXTERNAL

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/Getters.sol (2022/06/15 - 545f3): 17; contracts/Governance.sol (2022/06/15 - 545f3): 20, 36, 52, 73; contracts/Implementation.sol (2022/06/15 - 545f3): 15; contracts/Messages.sol (2022/06/15 - 545f3): 16; contracts/Migrations.sol (2022/06/15 - 545f3): 16; contracts/Setup.sol (2022/06/15 - 545f3): 12; contracts/bridge/Bridge.sol (2022/06/15 - 545f3): 24, 64, 69, 109, 114, 300, 304, 308, 312; contracts/bridge/BridgeGovernance.sol (2022/06/15 - 545f3): 28, 43; contracts/bridge/BridgeImplementation.sol (2022/06/15 - 545f3): 14, 18; contracts/bridge/BridgeSetup.sol (2022/06/15 - 545f3): 12; contracts/bridge/token/TokenImplementation.sol (2022/06/15 - 545f3): 16, 38, 42, 50, 54, 58, 62, 66, 70, 75, 79, 84, 94, 99, 119, 131, 154; contracts/nft/NFTBridge.sol (2022/06/15 - 545f3): 23, 100; contracts/nft/NFTBridgeGovernance.sol (2022/06/15 - 545f3): 27, 41; contracts/nft/NFTBridgeImplementation.sol (2022/06/15 - 545f3): 14, 18; contracts/nft/NFTBridgeSetup.sol (2022/06/15 - 545f3): 12; contracts/nft/token/NFTImplementation.sol (2022/06/15 - 545f3): 22, 47, 58, 62, 66, 72, 76, 84, 102, 113, 124, 162, 177	Acknowledged

Description

The functions which are never called internally within the contract should have external visibility for gas optimization.

```
100     function completeTransfer(bytes memory encodedVm) public {
```

```
102     function setApprovalForAll(address operator, bool approved) public override
{
```

```
109     function transferTokens(address token, uint256 amount, uint16
recipientChain, bytes32 recipient, uint256 arbiterFee, uint32 nonce) public payable
nonReentrant returns (uint64 sequence) {
```

```
113     function transferFrom(
```

```
114     function transferTokensWithPayload(address token, uint256 amount, uint16 recipientChain, bytes32 recipient, uint256 arbiterFee, uint32 nonce, bytes memory payload) public payable nonReentrant returns (uint64 sequence) {
```

```
119     function mint(address account_, uint256 amount_) public onlyOwner {
```

```
12     function setup(
```

```
12     function setup(
```

```
12     function setup(
```

```
124     function safeTransferFrom(
```

```
131     function burn(address account_, uint256 amount_) public onlyOwner {
```

```
14     function implementation() public view returns (address) {
```

```
14     function implementation() public view returns (address) {
```

```
15     function publishMessage(
```

```
154     function updateDetails(string memory name_, string memory symbol_, uint64 sequence_) public onlyOwner {
```

```
16     function initialize(
```

```
16     function parseAndVerifyVM(bytes calldata encodedVM) public view returns (Structs.VM memory vm, bool valid, string memory reason) {
```

```
16     function setCompleted(uint completed) public restricted {
```

```
162     function mint(address to, uint256 tokenId, string memory uri) public
onlyOwner {
```

```
17     function getGuardianSetExpiry() public view returns (uint32) {
```

```
177     function burn(uint256 tokenId) public onlyOwner {
```

```
18     function initialize() initializer public virtual {
```

```
18     function initialize() initializer public virtual {
```

```
20     function submitContractUpgrade(bytes memory _vm) public {
```

```
22     function initialize(
```

```
23     function transferNFT(address token, uint256 tokenID, uint16 recipientChain,
bytes32 recipient, uint32 nonce) public payable returns (uint64 sequence) {
```

```
24     function attestToken(address tokenAddress, uint32 nonce) public payable
returns (uint64 sequence){
```

```
27     function registerChain(bytes memory encodedVM) public {
```

```
28     function registerChain(bytes memory encodedVM) public {
```

```
300     function completeTransferWithPayload(bytes memory encodedVm, address
feeRecipient) public returns (bytes memory) {
```

```
304     function completeTransferAndUnwrapETHWithPayload(bytes memory encodedVm,
address feeRecipient) public returns (bytes memory) {
```

```
308     function completeTransfer(bytes memory encodedVm) public {
```

```
312     function completeTransferAndUnwrapETH(bytes memory encodedVm) public {
```

```
36      function submitSetMessageFee(bytes memory _vm) public {
```

```
38      function name() public view returns (string memory) {
```

```
41      function upgrade(bytes memory encodedVM) public {
```

```
42      function symbol() public view returns (string memory) {
```

```
43      function upgrade(bytes memory encodedVM) public {
```

```
47      function balanceOf(address owner_) public view override returns (uint256) {
```

```
50      function decimals() public view returns (uint8) {
```

```
52      function submitNewGuardianSet(bytes memory _vm) public {
```

```
54      function totalSupply() public view returns (uint256) {
```

```
58      function chainId() public view returns (uint16) {
```

```
58      function name() public view override returns (string memory) {
```

```
62      function nativeContract() public view returns (bytes32) {
```

```
62      function symbol() public view override returns (string memory) {
```

```
64      function wrapAndTransferETH(uint16 recipientChain, bytes32 recipient,
uint256 arbiterFee, uint32 nonce) public payable returns (uint64 sequence) {
```

```
66      function balanceOf(address account_) public view returns (uint256) {
```

```
66      function tokenURI(uint256 tokenId) public view override returns (string
memory) {
```

```
 69      function wrapAndTransferETHWithPayload(uint16 recipientChain, bytes32
recipient, uint256 arbiterFee, uint32 nonce, bytes memory payload) public payable
returns (uint64 sequence) {
```

```
70      function transfer(address recipient_, uint256 amount_) public returns (bool)
{
```

```
72      function chainId() public view returns (uint16) {
```

```
73      function submitTransferFees(bytes memory _vm) public {
```

```
75      function allowance(address owner_, address spender_) public view returns
(uint256) {
```

```
76      function nativeContract() public view returns (bytes32) {
```

```
 79      function approve(address spender_, uint256 amount_) public returns (bool) {
```

```
 84      function approve(address to, uint256 tokenId) public override {
```

```
 84      function transferFrom(address sender_, address recipient_, uint256 amount_)
public returns (bool) {
```

```
 94      function increaseAllowance(address spender_, uint256 addedValue_) public
returns (bool) {
```

```
 99      function decreaseAllowance(address spender_, uint256 subtractedValue_)
public returns (bool) {
```

Recommendation

The auditing team recommends changing the visibility of the aforementioned functions to `external`.

Alleviation

[Wormhole Team, 03/08/2023]:
The team acknowledged this issue and decided not to change the codebase in the current stage.

CON-04 | INEFFICIENT REQUIRE STATEMENT LOCATION

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/bridge/Bridge.sol (2022/06/15 - 545f3): 323~332; contracts/nft/NFTBridge.sol (2022/06/15 - 545f3): 111~114	Acknowledged

Description

The `parseAndVerifyVM()` method from the core bridge of Wormhole decodes `encodedVm` and returns `vm` as the output. Then `_completeTransfer` will decode `vm.payload` and check whether `vm` is completed or not by its hash attribute.

```
        (IWormhole.VM memory vm, bool valid, string memory reason) =
wormhole().parseAndVerifyVM(encodedVm);

        require(valid, reason);
        require(verifyBridgeVM(vm), "invalid emitter");

        BridgeStructs.Transfer memory transfer = parseTransfer(vm.payload);

        // payload 3 must be redeemed by the designated proxy contract
        address transferRecipient = address(uint160(uint256(transfer.to)));
        if (transfer.payloadID == 3) {
            require(msg.sender == transferRecipient, "invalid sender");
        }

        require(!isTransferCompleted(vm.hash), "transfer already completed");
        setTransferCompleted(vm.hash);
```

If the transfer is already completed, the invocation will revert in the require statement `require(!isTransferCompleted(vm.hash), "transfer already completed");`. However, considering the gas consumption, it is recommended to perform an early revert to save gas.

Recommendation

The auditing team recommends adjusting the inspection sequence to save gas. As an example:

```
        require(!isTransferCompleted(vm.hash), "transfer already completed");
        setTransferCompleted(vm.hash);

        BridgeStructs.Transfer memory transfer = parseTransfer(vm.payload);

        // payload 3 must be redeemed by the designated proxy contract
        address transferRecipient = address(uint160(uint256(transfer.to)));
        if (transfer.payloadID == 3) {
            require(msg.sender == transferRecipient, "invalid sender");
        }
```

Alleviation

[Wormhole Team, 03/08/2023]:

The team acknowledged this issue and decided not to change the codebase in the current stage.

MIR-01 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/bridge/utils/Migrator.sol (2022/06/15 - 545f3): 13, 14	Acknowledged

Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as immutable. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

Alleviation

[Wormhole Team, 03/08/2023]:
The team acknowledged this issue and decided not to change the codebase in the current stage.

APPENDIX | WORMHOLE - ETHEREUM

Finding Categories

Categories	Description
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Control Flow	Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Language Specific	Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.
Inconsistency	Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.