SECURITY AUDIT OF CROSS CHAIN

Wormhole: Cross Chain Bridge

Manmeet Singh Brar
20UCS112

Introduction

This audit report highlights the overall security of the Wormhole Cross Chain
With this report, I have tried to ensure the reliability of the smart contract by completing the assessment of their system's architecture and smart contract codebase.

Auditing approach and Methodologies applied

In this audit, I consider the following crucial features of the code.

- Whether the implementation of protocol standards.
- Whether the code is secure.
- Whether the code meets the best coding practices.
- Whether the code meets the SWC Registry issue.

The audit has been performed according to the following procedure:

• Manual audit

1. Inspecting the code line by line and revert the initial algorithms of the protocol and then compare them with the specification
2. Manually analyzing the code for security vulnerabilities.
3. Assessing the overall project structure, complexity & quality.
4. Checking SWC Registry issues in the code.
5. Unit testing by writing custom unit testing for each function.
6. Checking whether all the libraries used in the code of the latest version.
7. Analysis of security on-chain data.
8. Analysis of the failure preparations to check how the smart contract performs in case of bugs and vulnerability.

• Automated analysis

1. Scanning the project's code base with Slither Echidna,

2. Manually verifying (reject or confirm) all the issues found by tools.
3. Performing Unit testing.
4. Manual Security Testing (SWC-Registry, Overflow)
5. Running the tests and checking their coverage.

Report: All the gathered information is described in this report.

Audit details

Project Name: BFSC Audit Report

Language: Solidity

Platform and tools: Remix, VScode, securify and other tools mentioned in the automated analysis section.

Audit Goals

The focus of this audit was to verify whether the smart contract is secure, resilient, and working properly according to the specs. The audit activity can be grouped in three categories.

Security: Identifying the security-related issue within each contract and system of contracts.

Sound architecture: Evaluating the architect of a system through the lens of established smart contract best practice and general software practice.

Code correctness and quality: A full review of contract source code. The primary area of focus includes.

- Correctness.
- Section of code with high complexity.
- Readability.
- Quantity and quality of test coverage.

Security

Every issue in this report was assigned a severity level from the following:

High severity issues

Issues mentioned here are critical to smart contract performance and functionality and should be fixed before moving to mainnet.

Medium severity issues
This could potentially bring the problem in the future and should be fixed.

Low severity issues
These are minor details and warnings that can remain unfixed but would be better if it got fixed in the future.

No. of issue per severity

Severity	High	Medium	Low
Open	0	0	0

Manual audit
Following are the reports from our manual analysis.

High severity issues

No High Severity Issue found.

Medium severity issues

No Medium Severity Issue found.

Low Severity Issues :

No Low Severity Issue found.

1. Function Default Visibility (SWC - 100)

The SWC-100 vulnerability pertains to the lack of explicitly defined visibility for functions in Solidity contracts. By omitting the visibility modifier, functions are set to the default visibility level, which is often public. This can lead to unintended consequences and potential security risks.

No instances of functions with default visibility were detected.

2. Integer Overflow and Underflow (SWC - 101)

SWC-101 refers to the potential vulnerabilities arising from improper handling of integer arithmetic operations, such as addition, subtraction, multiplication, and division.

No instances of Integer overflow and underflow were found.

3. Outdated Compiler Version (SWC - 102)

SWC-102 pertains to the use of outdated or insecure compiler versions in the development of smart contracts. Using an outdated compiler version may introduce known vulnerabilities or issues that have been addressed in later compiler releases.

Solidity version 0.8.0 is used in all contracts (0.8.20 is the latest version).

4. Floating Pragma (SWC - 103)

SWC-103 relates to the absence of a fixed compiler version pragma statement in the Solidity source code. The lack of a fixed compiler version pragma may result in unintended behaviors or vulnerabilities when the contract is compiled with different compiler versions.

No cases of floating pragma were found.

5. Unchecked Call and Return Value (SWC - 104)

SWC-104 pertains to the unchecked usage of return values from external function calls in Solidity contracts. Failing to properly handle and validate the return values can lead to unexpected behavior or vulnerabilities, such as reentrancy attacks or incorrect state changes.

No cases of unchecked call and return value found.

6. Unprotected Ether Withdrawal (SWC - 105)

SWC-105 refers to the potential vulnerability arising from the lack of proper protection mechanisms for ether withdrawal functions.

No cases of Unprotected Ether Withdrawal found.

7. Unprotected SELFDESTRUCT Instruction (SWC - 106)

SWC-106 pertains to the potential vulnerability arising from the improper handling of the SELFDESTRUCT instruction in Solidity contracts.

No cases of Unprotected SELFDESTRUCT Instruction found.

8. Reentrancy (SWC - 107)

SWC-107 refers to the potential vulnerability arising from improper handling of external contract calls that can lead to reentrancy attacks.

No cases of Reentrancy found.

9. Reentrancy (SWC - 108)

SWC-108 relates to the absence of explicit visibility modifiers for state variables in Solidity contracts.

No cases of Reentrancy found.

10. Uninitialized Storage Pointer (SWC - 109)

SWC-109 refers to the potential vulnerability arising from the use of uninitialized storage pointers in Solidity contracts.

No cases of Uninitialized Storage Pointer found.

11. Assert Violation (SWC - 110)

SWC-110 refers to potential vulnerabilities arising from incorrect usage of assert statements in Solidity contracts.

No cases of Assert Violation found.

12. Use of Deprecated Solidity Functions (SWC - 111)

SWC-111 pertains to potential vulnerabilities arising from the usage of deprecated Solidity functions or features that may have security risks or suboptimal behaviors.

No cases of Use of Deprecated Solidity Functions found.

13. Delegatecall to Untrusted Callee (SWC - 112)

SWC-112 relates to the potential security risks associated with using delegatecall to invoke external contracts without proper trust verification.

No cases of Delegatecall to Untrusted Callee found.

14. DoS with Failed Call (SWC - 113)

SWC-113 refers to potential vulnerabilities arising from the misuse or mishandling of failed external function calls.

No cases of DoS with Failed Call found.

15. Transaction Order Dependence (SWC - 114)

SWC-114 pertains to potential vulnerabilities arising from the reliance on the order of transactions in the Ethereum blockchain.

No cases of Transaction Order Dependence found.

16. Authorization through tx.origin (SWC - 115)

SWC-115 refers to potential vulnerabilities arising from the misuse or reliance on the tx.origin global variable for authorization purposes.

No cases of Authorization through tx.origin found.

17. Block values as a proxy for time (SWC - 116)

SWC-116 pertains to potential vulnerabilities arising from the misuse of block values as a substitute for accurate time measurements in smart contracts.

No cases of Block values as a proxy for time found.

18. Incorrect Constructor Name (SWC - 118)

SWC-118 pertains to potential vulnerabilities arising from the incorrect naming of constructors in Solidity contracts.

No cases of Incorrect Constructor Name found.

19. Shadowing State Variables (SWC - 119)

SWC-119 refers to potential vulnerabilities arising from the shadowing of state variables within Solidity contracts..

No cases of Shadowing State Variables found.

20. Weak Sources of Randomness from Chain Attributes (SWC - 120)

SWC-120 pertains to potential vulnerabilities arising from the reliance on weak sources of randomness derived from chain attributes in smart contracts.

No cases of Weak Sources of Randomness from Chain Attributes found.

21. Requirement Violation (SWC - 123)

SWC-123 refers to potential vulnerabilities arising from violations of contract requirements, specifications, or assumptions.

No cases of Requirement Violation found.

22. Write to Arbitrary Storage Location (SWC - 124)

SWC-124 refers to potential vulnerabilities arising from the ability to write to arbitrary storage locations within smart contracts.

No cases of Write to Arbitrary Storage Location found.

23. Incorrect Inheritance Order (SWC - 125)

SWC-125 pertains to vulnerabilities that arise from an incorrect order of inheritance within smart contracts.

No cases of Incorrect Inheritance Order found.

24. Insufficient Gas Griefing (SWC - 126)

SWC-126 refers to vulnerabilities that arise from insufficient gas limits during contract execution.

No cases of Insufficient Gas Griefing found.

25. Arbitrary Jump with Function Type Variable (SWC - 127)

SWC-127 refers to vulnerabilities that arise from the use of function type variables to perform arbitrary jumps in the contract's control flow.

No cases of Arbitrary Jump with Function Type Variable found.

26. DoS With Block Gas Limit (SWC - 128)

SWC-128 pertains to vulnerabilities that arise from the misuse or improper handling of the block gas limit within smart contracts.

No cases of DoS With Block Gas Limit found.

27. Typographical Error (SWC - 129)

SWC-129 refers to vulnerabilities that arise from typographical errors or mistakes in the code, leading to unintended behaviors or vulnerabilities.

No cases of Typographical Error found.

28. Right-To-Left-Override control character (U+202E) (SWC - 130)

SWC-130 refers to vulnerabilities that can arise from the malicious use of the Right-To-Left-Override (RLO) control character (U+202E) within the code.

No cases of Right-To-Left-Override control character (U+202E) found.

29. Presence of unused variables (SWC - 131)

SWC-131 refers to vulnerabilities that arise from the presence of unused variables within the code.

No cases of Presence of unused variables found.

30. Unexpected Ether balance (SWC - 132)

SWC-132 refers to vulnerabilities that arise from unexpected Ether balances within contracts. This can occur when Ether is unintentionally trapped in a contract or when the contract's balance handling is i nconsistent with the intended design.

No cases of Unexpected Ether balance found.

31. Message call with hardcoded gas amount (SWC - 134)

SWC-134 refers to vulnerabilities that arise when a contract performs a message call to an external contract with a hardcoded gas amount.

No cases of Message call with hardcoded gas amount found.

32. Code With No Effects (SWC - 135)

SWC-135 refers to vulnerabilities that arise from code that has no effect on the contract's state or behavior.

No cases of Code With No Effects found.

33. Unencrypted Private Data On-Chain (SWC - 136)

SWC-136 refers to vulnerabilities that arise when private or sensitive data is stored on the blockchain without appropriate encryption or obfuscation.

No cases of Unencrpyted Private Data On-Chain found.

Automated test :

We have used Slither and Echinda automated testing frameworks. This makes code more secure Against common attacks. The results are below.

Slither:

Slither is a Solidity static analysis framework which runs a suite of vulnerability detectors, prints visual information about contract details, and provides an API to write custom analyses quickly. Slither enables developers to find vulnerabilities, enhance their code comprehension, and promptly prototype custom analyses. Each solidity file and project together has been analyzed. We got a report with a few warnings and errors.

Slither results are below:

```
INFO:Detectors:
Reentrancy in Bridge._wrapAndTransferETH(uint256) (bridge/Bridge.sol#116-149):
        External calls:
        - WETH().deposit{value: amount - dust}() (bridge/Bridge.sol#135-137)
        External calls sending eth:
        - address(msg.sender).transfer(dust) (bridge/Bridge.sol#131)
        - WETH().deposit{value: amount - dust}() (bridge/Bridge.sol#135-137)
        State variables written after the call(s):
        - bridgeOut(address(WETH()),normalizedAmount) (bridge/Bridge.sol#140)
                - _state.outstandingBridged[token] = outstanding (bridge/BridgeSetters.sol#56)
        BridgeState._state (bridge/BridgeState.sol#56) can be used in cross function reentrancies:
        - BridgeGetters.WETH() (bridge/BridgeGetters.sol#62-64)
        - BridgeGetters.bridgeContracts(uint16) (bridge/BridgeGetters.sol#54-56)
        - BridgeGetters.chainId() (bridge/BridgeGetters.sol#30-32)
        - BridgeGetters.evmChainId() (bridge/BridgeGetters.sol#34-36)
        - BridgeGetters.finality() (bridge/BridgeGetters.sol#74-76)
        - BridgeGetters.governanceActionIsConsumed(bytes32) (bridge/BridgeGetters.sol#14-16)
        - BridgeGetters.governanceChainId() (bridge/BridgeGetters.sol#42-44)
        - BridgeGetters.governanceContract() (bridge/BridgeGetters.sol#46-48)
        - BridgeGetters.isInitialized(address) (bridge/BridgeGetters.sol#18-20)
        - BridgeGetters.isTransferCompleted(bytes32) (bridge/BridgeGetters.sol#22-24)
        - BridgeGetters.isWrappedAsset(address) (bridge/BridgeGetters.sol#70-72)
        - BridgeGetters.outstandingBridged(address) (bridge/BridgeGetters.sol#66-68)
        - BridgeSetters.setBridgeImplementation(uint16,bytes32) (bridge/BridgeSetters.sol#33-35)
        - BridgeSetters.setChainId(uint16) (bridge/BridgeSetters.sol#21-23)
        - BridgeSetters.setEvmChainId(uint256) (bridge/BridgeSetters.sol#63-66)
        - BridgeSetters.setGovernanceActionConsumed(bytes32) (bridge/BridgeSetters.sol#13-15)
        - BridgeSetters.setOutstandingBridged(address,uint256) (bridge/BridgeSetters.sol#55-57)
        - BridgeSetters.setTransferCompleted(bytes32) (bridge/BridgeSetters.sol#17-19)
        - BridgeSetters.setWrappedAsset(uint16,bytes32,address) (bridge/BridgeSetters.sol#50-53)
        - BridgeGetters.tokenImplementation() (bridge/BridgeGetters.sol#58-60)
        - BridgeGetters.wormhole() (bridge/BridgeGetters.sol#26-28)
        - BridgeGetters.wrappedAsset(uint16,bytes32) (bridge/BridgeGetters.sol#50-52)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
```

```
INFO:Detectors:
BytesLib.concatStorage(bytes,bytes) (libraries/external/BytesLib.sol#91-226) performs a multiplication on the result of a division:
        - sstore(uint256,uint256)(_preBytes,fslot_concatStorage_asm_0 + mload(uint256)(_postBytes + 0x20) / 0x100 ** 32 - mlength_concatStorage_asm_0 * 0x100 ** 32 - newlength_concatStorage_asm_0 + mlength_concatStorage_asm_0 * 2) (libraries/external/BytesLib.sol#115-140)
BytesLib.concatStorage(bytes,bytes) (libraries/external/BytesLib.sol#91-226) performs a multiplication on the result of a division:
        - sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_concatStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (libraries/external/BytesLib.sol#189)
BytesLib.concatStorage(bytes,bytes) (libraries/external/BytesLib.sol#91-226) performs a multiplication on the result of a division:
        - sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_concatStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (libraries/external/BytesLib.sol#223)
BytesLib.equalStorage(bytes,bytes) (libraries/external/BytesLib.sol#439-509) performs a multiplication on the result of a division:
        - fslot_equalStorage_asm_0 = fslot_equalStorage_asm_0 / 0x100 * 0x100 (libraries/external/BytesLib.sol#466)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
```

```
INFO:Detectors:
Reentrancy in Bridge._transferTokens(address,uint256,uint256) (bridge/Bridge.sol#219-274):
        External calls:
        - SafeERC20.safeTransferFrom(IERC20(token),msg.sender,address(this),amount) (bridge/Bridge.sol#244)
        - SafeERC20.safeTransferFrom(IERC20(token),msg.sender,address(this),amount) (bridge/Bridge.sol#253)
        - TokenImplementation(token).burn(address(this),amount) (bridge/Bridge.sol#255)
        State variables written after the call(s):
        - bridgeOut(token,normalizedAmount) (bridge/Bridge.sol#264)
                - _state.outstandingBridged[token] = outstanding (bridge/BridgeSetters.sol#56)
        BridgeState._state (bridge/BridgeState.sol#56) can be used in cross function reentrancies:
        - BridgeGetters.WETH() (bridge/BridgeGetters.sol#62-64)
        - BridgeGetters.bridgeContracts(uint16) (bridge/BridgeGetters.sol#54-56)
        - BridgeGetters.chainId() (bridge/BridgeGetters.sol#30-32)
        - BridgeGetters.evmChainId() (bridge/BridgeGetters.sol#34-36)
        - BridgeGetters.finality() (bridge/BridgeGetters.sol#74-76)
        - BridgeGetters.governanceActionIsConsumed(bytes32) (bridge/BridgeGetters.sol#14-16)
        - BridgeGetters.governanceChainId() (bridge/BridgeGetters.sol#42-44)
        - BridgeGetters.governanceContract() (bridge/BridgeGetters.sol#46-48)
        - BridgeGetters.isInitialized(address) (bridge/BridgeGetters.sol#18-20)
        - BridgeGetters.isTransferCompleted(bytes32) (bridge/BridgeGetters.sol#22-24)
        - BridgeGetters.isWrappedAsset(address) (bridge/BridgeGetters.sol#70-72)
        - BridgeGetters.outstandingBridged(address) (bridge/BridgeGetters.sol#66-68)
        - BridgeSetters.setBridgeImplementation(uint16,bytes32) (bridge/BridgeSetters.sol#33-35)
        - BridgeSetters.setChainId(uint16) (bridge/BridgeSetters.sol#21-23)
        - BridgeSetters.setEvmChainId(uint256) (bridge/BridgeSetters.sol#63-66)
        - BridgeSetters.setGovernanceActionConsumed(bytes32) (bridge/BridgeSetters.sol#13-15)
        - BridgeSetters.setOutstandingBridged(address,uint256) (bridge/BridgeSetters.sol#55-57)
        - BridgeSetters.setTransferCompleted(bytes32) (bridge/BridgeSetters.sol#17-19)
        - BridgeSetters.setWrappedAsset(uint16,bytes32,address) (bridge/BridgeSetters.sol#50-53)
        - BridgeGetters.tokenImplementation() (bridge/BridgeGetters.sol#58-60)
        - BridgeGetters.wormhole() (bridge/BridgeGetters.sol#26-28)
        - BridgeGetters.wrappedAsset(uint16,bytes32) (bridge/BridgeGetters.sol#50-52)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
NFTBridgeImplementation.initialize().evmChainId (nft/NFTBridgeImplementation.sol#21) is a local variable never initialized
NFTBridge.bytes32ToString(bytes32).i (nft/NFTBridge.sol#272) is a local variable never initialized
Bridge.bytes32ToString(bytes32).i (bridge/Bridge.sol#748) is a local variable never initialized
Implementation.initialize().evmChainId (Implementation.sol#36) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

```
INFO:Detectors:
NFTImplementation._checkOnERC721Received(address,address,uint256,bytes) (nft/token/NFTImplementation.sol#216-237) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (nft/token/NFTImplementation.sol#223-233)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

```
INFO:Detectors:
MockBridgeImplementation.testUpdateWETHAddress(address).WETH (bridge/mock/MockBridgeImplementation.sol#17) shadows:
        - BridgeGetters.WETH() (bridge/BridgeGetters.sol#62-64) (function)
NFTBridgeImplementation.initialize().evmChainId (nft/NFTBridgeImplementation.sol#21) shadows:
        - NFTBridgeGetters.evmChainId() (nft/NFTBridgeGetters.sol#33-35) (function)
Implementation.initialize().evmChainId (Implementation.sol#36) shadows:
        - Getters.evmChainId() (Getters.sol#33-35) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Bridge.attestToken(address,uint32).tokenAddress (bridge/Bridge.sol#26) lacks a zero-check on :
                - (queriedDecimals) = tokenAddress.staticcall(abi.encodeWithSignature(decimals())) (bridge/Bridge.sol#28)
                - (queriedSymbol) = tokenAddress.staticcall(abi.encodeWithSignature(symbol())) (bridge/Bridge.sol#29)
                - (queriedName) = tokenAddress.staticcall(abi.encodeWithSignature(name())) (bridge/Bridge.sol#30)
MockTokenBridgeIntegration.setup(address)._tokenBridge (bridge/mock/MockTokenBridgeIntegration.sol#45) lacks a zero-check on :
                - tokenBridgeAddress = _tokenBridge (bridge/mock/MockTokenBridgeIntegration.sol#46)
MockBatchedVAASender.setup(address)._wormholeCore (mock/MockBatchedVAASender.sol#50) lacks a zero-check on :
                - wormholeCoreAddress = _wormholeCore (mock/MockBatchedVAASender.sol#51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Migrator.add(uint256) (bridge/utils/Migrator.sol#29-34):
        External calls:
        - SafeERC20.safeTransferFrom(toAsset,msg.sender,address(this),_amount) (bridge/utils/Migrator.sol#31)
        State variables written after the call(s):
        - _mint(msg.sender,_amount) (bridge/utils/Migrator.sol#33)
                - _balances[account] += amount (@openzeppelin/contracts/token/ERC20/ERC20.sol#257)
        - _mint(msg.sender,_amount) (bridge/utils/Migrator.sol#33)
                - _totalSupply += amount (@openzeppelin/contracts/token/ERC20/ERC20.sol#256)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
INFO:Detectors:
Reentrancy in ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79
-107):
        External calls:
        - Address.functionDelegateCall(newImplementation,data) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
        - Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (@openzeppeli
n/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
        Event emitted after the call(s):
        - Upgraded(newImplementation) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#55)
                - _upgradeTo(newImplementation) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#105)
Reentrancy in Migrator.add(uint256) (bridge/utils/Migrator.sol#29-34):
        External calls:
        - SafeERC20.safeTransferFrom(toAsset,msg.sender,address(this),_amount) (bridge/utils/Migrator.sol#31)
        Event emitted after the call(s):
        - Transfer(address(0),account,amount) (@openzeppelin/contracts/token/ERC20/ERC20.sol#258)
                - _mint(msg.sender,_amount) (bridge/utils/Migrator.sol#33)
Reentrancy in Governance.upgradeImplementation(address) (Governance.sol#174-185):
        External calls:
        - (success,reason) = newImplementation.delegatecall(abi.encodeWithSignature(initialize())) (Governance.sol#180)
        Event emitted after the call(s):
        - ContractUpgraded(currentImplementation,newImplementation) (Governance.sol#184)
Reentrancy in BridgeGovernance.upgradeImplementation(address) (bridge/BridgeGovernance.sol#102-113):
        External calls:
        - (success,reason) = newImplementation.delegatecall(abi.encodeWithSignature(initialize())) (bridge/BridgeGovernance.sol#108)
        Event emitted after the call(s):
        - ContractUpgraded(currentImplementation,newImplementation) (bridge/BridgeGovernance.sol#112)
Reentrancy in NFTBridgeGovernance.upgradeImplementation(address) (nft/NFTBridgeGovernance.sol#99-110):
        External calls:
        - (success,reason) = newImplementation.delegatecall(abi.encodeWithSignature(initialize())) (nft/NFTBridgeGovernance.sol#105)
        Event emitted after the call(s):
        - ContractUpgraded(currentImplementation,newImplementation) (nft/NFTBridgeGovernance.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
Messages.verifyVMInternal(Structs.VM,bool) (Messages.sol#40-102) uses timestamp for comparisons
        Dangerous comparisons:
        - vm.guardianSetIndex != getCurrentGuardianSetIndex() && guardianSet.expirationTime < block.timestamp (Messages.sol#80)
TokenImplementation.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (bridge/token/TokenImplementation.sol#274-310) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline_,ERC20Permit: expired deadline) (bridge/token/TokenImplementation.sol#288)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
INFO:Detectors:
Proxy._delegate(address) (@openzeppelin/contracts/proxy/Proxy.sol#21-44) uses assembly
        - INLINE ASM (@openzeppelin/contracts/proxy/Proxy.sol#22-43)
NFTBridge.transferNFT(address,uint256,uint16,bytes32,uint32) (nft/NFTBridge.sol#23-90) uses assembly
        - INLINE ASM (nft/NFTBridge.sol#62-72)
NFTBridge._createWrapped(uint16,bytes32,bytes32,bytes32) (nft/NFTBridge.sol#152-193) uses assembly
        - INLINE ASM (nft/NFTBridge.sol#184-190)
NFTImplementation._checkOnERC721Received(address,address,uint256,bytes) (nft/token/NFTImplementation.sol#216-237) uses assembly
        - INLINE ASM (nft/token/NFTImplementation.sol#229-231)
Address.isContract(address) (@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/Address.sol#207-210)
StorageSlot.getAddressSlot(bytes32) (@openzeppelin/contracts/utils/StorageSlot.sol#51-55) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/StorageSlot.sol#52-54)
StorageSlot.getBooleanSlot(bytes32) (@openzeppelin/contracts/utils/StorageSlot.sol#60-64) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/StorageSlot.sol#61-63)
StorageSlot.getBytes32Slot(bytes32) (@openzeppelin/contracts/utils/StorageSlot.sol#69-73) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/StorageSlot.sol#70-72)
StorageSlot.getUint256Slot(bytes32) (@openzeppelin/contracts/utils/StorageSlot.sol#78-82) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/StorageSlot.sol#79-81)
BytesLib.concat(bytes,bytes) (libraries/external/BytesLib.sol#13-89) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#23-86)
BytesLib.concatStorage(bytes,bytes) (libraries/external/BytesLib.sol#91-226) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#92-225)
BytesLib.slice(bytes,uint256,uint256) (libraries/external/BytesLib.sol#228-295) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#242-292)
BytesLib.toAddress(bytes,uint256) (libraries/external/BytesLib.sol#297-306) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#301-303)
BytesLib.toUint8(bytes,uint256) (libraries/external/BytesLib.sol#308-317) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#312-314)
BytesLib.toUint16(bytes,uint256) (libraries/external/BytesLib.sol#319-328) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#323-325)
BytesLib.toUint32(bytes,uint256) (libraries/external/BytesLib.sol#330-339) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#334-336)
BytesLib.toUint64(bytes,uint256) (libraries/external/BytesLib.sol#341-350) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#345-347)
BytesLib.toUint96(bytes,uint256) (libraries/external/BytesLib.sol#352-361) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#356-358)
BytesLib.toUint128(bytes,uint256) (libraries/external/BytesLib.sol#363-372) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#367-369)
BytesLib.toUint256(bytes,uint256) (libraries/external/BytesLib.sol#374-383) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#378-380)
BytesLib.toBytes32(bytes,uint256) (libraries/external/BytesLib.sol#385-394) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#389-391)
BytesLib.equal(bytes,bytes) (libraries/external/BytesLib.sol#396-437) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#399-434)
BytesLib.equalStorage(bytes,bytes) (libraries/external/BytesLib.sol#439-509) uses assembly
        - INLINE ASM (libraries/external/BytesLib.sol#449-506)
ECDSA.tryRecover(bytes32,bytes) (@openzeppelin/contracts/utils/cryptography/ECDSA.sol#54-83) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/cryptography/ECDSA.sol#64-68)
        - INLINE ASM (@openzeppelin/contracts/utils/cryptography/ECDSA.sol#75-78)
ECDSA.tryRecover(bytes32,bytes32,bytes32) (@openzeppelin/contracts/utils/cryptography/ECDSA.sol#112-124) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/cryptography/ECDSA.sol#119-122)
Bridge.attestToken(address,uint32) (bridge/Bridge.sol#26-59) uses assembly
        - INLINE ASM (bridge/Bridge.sol#39-43)
ECDSA.tryRecover(bytes32,bytes32,bytes32) (@openzeppelin/contracts/utils/cryptography/ECDSA.sol#112-124) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/cryptography/ECDSA.sol#119-122)
Bridge.attestToken(address,uint32) (bridge/Bridge.sol#26-59) uses assembly
        - INLINE ASM (bridge/Bridge.sol#39-43)
Bridge._createWrapped(BridgeStructs.AssetMeta,uint64) (bridge/Bridge.sol#384-419) uses assembly
        - INLINE ASM (bridge/Bridge.sol#410-416)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Bridge._completeTransfer(bytes,bool) (bridge/Bridge.sol#483-563) compares to a boolean constant:
        -require(bool,string)(unwrapWETH == false || address(transferToken) == address(WETH()),invalid token, can only unwrap WETH) (bridge/Bridge.sol#515)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
```

```
INFO:Detectors:
Different versions of Solidity are used:
        - Version used: ['>=0.4.22<0.9.0', '>=0.8.0<0.9.0', '^0.8.0', '^0.8.2']
        - >=0.4.22<0.9.0 (Migrations.sol#2)
        - >=0.8.0<0.9.0 (libraries/external/BytesLib.sol#9)
        - ABIEncoderV2 (Shutdown.sol#5)
        - ABIEncoderV2 (bridge/BridgeSetup.sol#5)
        - ABIEncoderV2 (Messages.sol#5)
        - ABIEncoderV2 (nft/NFTBridgeImplementation.sol#5)
        - ABIEncoderV2 (bridge/utils/Migrator.sol#5)
        - ABIEncoderV2 (Implementation.sol#5)
        - ABIEncoderV2 (Setup.sol#5)
        - ABIEncoderV2 (nft/NFTBridgeSetup.sol#5)
        - ABIEncoderV2 (bridge/BridgeImplementation.sol#5)
        - ^0.8.0 (bridge/token/TokenState.sol#4)
        - ^0.8.0 (bridge/mock/MockBridgeImplementation.sol#4)
        - ^0.8.0 (nft/mock/MockNFTImplementation.sol#4)
        - ^0.8.0 (mock/MockImplementation.sol#4)
        - ^0.8.0 (nft/NFTBridgeState.sol#4)
        - ^0.8.0 (bridge/interfaces/IWETH.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/proxy/Proxy.sol#3)
        - ^0.8.0 (nft/NFTBridgeStructs.sol#4)
        - ^0.8.0 (Shutdown.sol#4)
        - ^0.8.0 (nft/mock/MockNFTBridgeImplementation.sol#4)
        - ^0.8.0 (Setters.sol#4)
        - ^0.8.0 (bridge/BridgeSetup.sol#4)
        - ^0.8.0 (nft/NFTBridgeGetters.sol#4)
        - ^0.8.0 (interfaces/IWormhole.sol#4)
        - ^0.8.0 (nft/token/NFT.sol#4)
        - ^0.8.0 (Messages.sol#4)
        - ^0.8.0 (nft/NFTBridgeImplementation.sol#4)
        - ^0.8.0 (nft/token/NFTState.sol#4)
        - ^0.8.0 (nft/interfaces/INFTBridge.sol#4)
        - ^0.8.0 (nft/NFTBridge.sol#4)
        - ^0.8.0 (bridge/BridgeState.sol#4)
        - ^0.8.0 (nft/NFTBridgeSetters.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/Context.sol#3)
        - ^0.8.0 (Governance.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/introspection/ERC165.sol#3)
        - ^0.8.0 (@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
        - ^0.8.0 (Wormhole.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/security/ReentrancyGuard.sol#3)
        - ^0.8.0 (nft/token/NFTImplementation.sol#4)
        - ^0.8.0 (nft/NFTBridgeShutdown.sol#4)
        - ^0.8.0 (bridge/TokenBridge.sol#4)
        - ^0.8.0 (bridge/BridgeShutdown.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (bridge/mock/MockTokenImplementation.sol#4)
        - ^0.8.0 (bridge/mock/MockTokenBridgeIntegration.sol#4)
        - ^0.8.0 (bridge/BridgeGetters.sol#4)
        - ^0.8.0 (bridge/utils/Migrator.sol#4)
        - ^0.8.0 (Implementation.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#3)
```

```
        - ^0.8.0 (Implementation.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#3)
        - ^0.8.0 (Setup.sol#4)
        - ^0.8.0 (nft/NFTBridgeSetup.sol#4)
        - ^0.8.0 (bridge/interfaces/ITokenBridge.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/Counters.sol#3)
        - ^0.8.0 (State.sol#4)
        - ^0.8.0 (bridge/token/Token.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/StorageSlot.sol#3)
        - ^0.8.0 (Structs.sol#4)
        - ^0.8.0 (bridge/BridgeSetters.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#3)
        - ^0.8.0 (bridge/BridgeGovernance.sol#4)
        - ^0.8.0 (nft/NFTBridgeGovernance.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/proxy/beacon/IBeacon.sol#3)
        - ^0.8.0 (bridge/BridgeImplementation.sol#4)
        - ^0.8.0 (bridge/mock/MockWETH9.sol#16)
        - ^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (@openzeppelin/contracts/utils/cryptography/ECDSA.sol#3)
        - ^0.8.0 (@openzeppelin/contracts/utils/Strings.sol#3)
        - ^0.8.0 (bridge/token/TokenImplementation.sol#4)
        - ^0.8.0 (nft/NFTBridgeEntrypoint.sol#4)
        - ^0.8.0 (mock/MockBatchedVAASender.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#3)
        - ^0.8.0 (@openzeppelin/contracts/proxy/ERC1967/ERC1967Proxy.sol#3)
        - ^0.8.0 (bridge/mock/MockFeeToken.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/proxy/beacon/BeaconProxy.sol#3)
        - ^0.8.0 (bridge/BridgeStructs.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3)
        - ^0.8.0 (Getters.sol#4)
        - ^0.8.0 (bridge/Bridge.sol#4)
        - ^0.8.0 (GovernanceStructs.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721.sol#3)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3)
        - ^0.8.2 (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
NFTBridgeImplementation.initialize() (nft/NFTBridgeImplementation.sol#18-47) has a high cyclomatic complexity (18).
Implementation.initialize() (Implementation.sol#33-62) has a high cyclomatic complexity (18).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
BytesLib.concat(bytes,bytes) (libraries/external/BytesLib.sol#13-89) is never used and should be removed
BytesLib.concatStorage(bytes,bytes) (libraries/external/BytesLib.sol#91-226) is never used and should be removed
BytesLib.equal(bytes,bytes) (libraries/external/BytesLib.sol#396-437) is never used and should be removed
BytesLib.equalStorage(bytes,bytes) (libraries/external/BytesLib.sol#439-509) is never used and should be removed
BytesLib.toUint128(bytes,uint256) (libraries/external/BytesLib.sol#363-372) is never used and should be removed
BytesLib.toUint96(bytes,uint256) (libraries/external/BytesLib.sol#352-361) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
```

```
INFO:Detectors:
Pragma version^0.8.0 (bridge/token/TokenState.sol#4) allows old versions
Pragma version^0.8.0 (bridge/mock/MockBridgeImplementation.sol#4) allows old versions
Pragma version^0.8.0 (nft/mock/MockNFTImplementation.sol#4) allows old versions
Pragma version^0.8.0 (mock/MockImplementation.sol#4) allows old versions
Pragma version^0.8.0 (nft/NFTBridgeState.sol#4) allows old versions
Pragma version^0.8.0 (bridge/interfaces/IWETH.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/proxy/Proxy.sol#3) allows old versions
Pragma version^0.8.0 (nft/NFTBridgeStructs.sol#4) allows old versions
Pragma version^0.8.0 (Shutdown.sol#4) allows old versions
Pragma version^0.8.0 (nft/mock/MockNFTBridgeImplementation.sol#4) allows old versions
Pragma version^0.8.0 (Setters.sol#4) allows old versions
Pragma version^0.8.0 (bridge/BridgeSetup.sol#4) allows old versions
Pragma version^0.8.0 (nft/NFTBridgeGetters.sol#4) allows old versions
Pragma version^0.8.0 (interfaces/IWormhole.sol#4) allows old versions
Pragma version^0.8.0 (nft/token/NFT.sol#4) allows old versions
Pragma version^0.8.0 (Messages.sol#4) allows old versions
Pragma version^0.8.0 (nft/NFTBridgeImplementation.sol#4) allows old versions
Pragma version^0.8.0 (nft/token/NFTState.sol#4) allows old versions
Pragma version^0.8.0 (nft/interfaces/INFTBridge.sol#4) allows old versions
Pragma version^0.8.0 (nft/NFTBridge.sol#4) allows old versions
Pragma version^0.8.0 (bridge/BridgeState.sol#4) allows old versions
Pragma version^0.8.0 (nft/NFTBridgeSetters.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#3) allows old versions
Pragma version^0.8.0 (Governance.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/utils/introspection/ERC165.sol#3) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/utils/introspection/IERC165.sol#3) allows old versions
Pragma version^0.8.0 (Wormhole.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/security/ReentrancyGuard.sol#3) allows old versions
Pragma version^0.8.0 (nft/token/NFTImplementation.sol#4) allows old versions
Pragma version^0.8.0 (nft/NFTBridgeShutdown.sol#4) allows old versions
Pragma version^0.8.0 (bridge/TokenBridge.sol#4) allows old versions
Pragma version^0.8.0 (bridge/BridgeShutdown.sol#4) allows old versions
Pragma version^0.8.2 (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#3) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Address.sol#3) allows old versions
Pragma version^0.8.0 (bridge/mock/MockTokenImplementation.sol#4) allows old versions
Pragma version^0.8.0 (bridge/mock/MockTokenBridgeIntegration.sol#4) allows old versions
Pragma version^0.8.0 (bridge/BridgeGetters.sol#4) allows old versions
Pragma version^0.8.0 (bridge/utils/Migrator.sol#4) allows old versions
Pragma version^0.8.0 (Implementation.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#3) allows old versions
Pragma version^0.8.0 (Setup.sol#4) allows old versions
Pragma version^0.8.0 (nft/NFTBridgeSetup.sol#4) allows old versions
Pragma version>=0.4.22<0.9.0 (Migrations.sol#2) is too complex
Pragma version^0.8.0 (bridge/interfaces/ITokenBridge.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Counters.sol#3) allows old versions
Pragma version^0.8.0 (State.sol#4) allows old versions
Pragma version^0.8.0 (bridge/token/TokenImplementation.sol#4) allows old versions
Pragma version^0.8.0 (nft/NFTBridgeEntrypoint.sol#4) allows old versions
Pragma version^0.8.0 (mock/MockBatchedVAASender.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#3) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/proxy/ERC1967/ERC1967Proxy.sol#3) allows old versions
Pragma version^0.8.0 (bridge/mock/MockFeeToken.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/proxy/beacon/BeaconProxy.sol#3) allows old versions
Pragma version^0.8.0 (bridge/BridgeStructs.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#3) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3) allows old versions
Pragma version^0.8.0 (Getters.sol#4) allows old versions
Pragma version^0.8.0 (bridge/Bridge.sol#4) allows old versions
Pragma version^0.8.0 (GovernanceStructs.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721.sol#3) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
```

```
INFO:Detectors:
Low level call in NFTBridge.transferNFT(address,uint256,uint16,bytes32,uint32) (nft/NFTBridge.sol#23-90):
        - (queriedSymbol) = token.staticcall(abi.encodeWithSignature(symbol())) (nft/NFTBridge.sol#43)
        - (queriedName) = token.staticcall(abi.encodeWithSignature(name())) (nft/NFTBridge.sol#44)
        - (queriedURI) = token.staticcall(abi.encodeWithSignature(tokenURI(uint256),tokenID)) (nft/NFTBridge.sol#49)
Low level call in Governance.upgradeImplementation(address) (Governance.sol#174-185):
        - (success,reason) = newImplementation.delegatecall(abi.encodeWithSignature(initialize())) (Governance.sol#180)
Low level call in Address.sendValue(address,uint256) (@openzeppelin/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (@openzeppelin/contracts/utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (@openzeppelin/contracts/utils/Address.sol#185)
Low level call in BridgeGovernance.upgradeImplementation(address) (bridge/BridgeGovernance.sol#102-113):
        - (success,reason) = newImplementation.delegatecall(abi.encodeWithSignature(initialize())) (bridge/BridgeGovernance.sol#108)
Low level call in NFTBridgeGovernance.upgradeImplementation(address) (nft/NFTBridgeGovernance.sol#99-110):
        - (success,reason) = newImplementation.delegatecall(abi.encodeWithSignature(initialize())) (nft/NFTBridgeGovernance.sol#105)
Low level call in Bridge.attestToken(address,uint32) (bridge/Bridge.sol#26-59):
        - (queriedDecimals) = tokenAddress.staticcall(abi.encodeWithSignature(decimals())) (bridge/Bridge.sol#28)
        - (queriedSymbol) = tokenAddress.staticcall(abi.encodeWithSignature(symbol())) (bridge/Bridge.sol#29)
        - (queriedName) = tokenAddress.staticcall(abi.encodeWithSignature(name())) (bridge/Bridge.sol#30)
Low level call in Bridge._transferTokens(address,uint256,uint256) (bridge/Bridge.sol#219-274):
        - (queriedDecimals) = token.staticcall(abi.encodeWithSignature(decimals())) (bridge/Bridge.sol#232)
        - (queriedBalanceBefore) = token.staticcall(abi.encodeWithSelector(IERC20.balanceOf.selector,address(this))) (bridge/Bridge.sol#240)
        - (queriedBalanceAfter) = token.staticcall(abi.encodeWithSelector(IERC20.balanceOf.selector,address(this))) (bridge/Bridge.sol#247)
Low level call in Bridge._completeTransfer(bytes,bool) (bridge/Bridge.sol#483-563):
        - (queriedDecimals) = address(transferToken).staticcall(abi.encodeWithSignature(decimals())) (bridge/Bridge.sol#518)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
FeeToken (bridge/mock/MockFeeToken.sol#12-177) should inherit from IERC20Metadata (@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#12-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
INFO:Detectors:
Variable TokenState._state (bridge/token/TokenState.sol#47) is not in mixedCase
Parameter MockBridgeImplementation.testUpdateWETHAddress(address).WETH (bridge/mock/MockBridgeImplementation.sol#17) is not in mixedCase
Variable NFTBridgeState._state (nft/NFTBridgeState.sol#60) is not in mixedCase
Parameter BridgeSetup.setup(address,uint16,address,uint16,bytes32,address,address,uint8,uint256).WETH (bridge/BridgeSetup.sol#19) is not in mixedCase
Variable NFTState._state (nft/token/NFTState.sol#40) is not in mixedCase
Variable BridgeState._state (bridge/BridgeState.sol#56) is not in mixedCase
Parameter Governance.submitContractUpgrade(bytes)._vm (Governance.sol#27) is not in mixedCase
Parameter Governance.submitSetMessageFee(bytes)._vm (Governance.sol#54) is not in mixedCase
Parameter Governance.submitNewGuardianSet(bytes)._vm (Governance.sol#79) is not in mixedCase
Parameter Governance.submitTransferFees(bytes)._vm (Governance.sol#117) is not in mixedCase
Parameter Governance.submitRecoverChainId(bytes)._vm (Governance.sol#146) is not in mixedCase
Constant Governance.module (Governance.sol#22) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter NFTImplementation.safeTransferFrom(address,address,uint256,bytes)._data (nft/token/NFTImplementation.sol#136) is not in mixedCase
Variable Migrations.last_completed_migration (Migrations.sol#6) is not in mixedCase
Function ITokenBridge._parseTransferCommon(bytes) (bridge/interfaces/ITokenBridge.sol#67) is not in mixedCase
Function ITokenBridge.WETH() (bridge/interfaces/ITokenBridge.sol#129) is not in mixedCase
Variable State._state (State.sol#51) is not in mixedCase
Parameter BytesLib.concat(bytes,bytes)._preBytes (libraries/external/BytesLib.sol#14) is not in mixedCase
Parameter BytesLib.concat(bytes,bytes)._postBytes (libraries/external/BytesLib.sol#15) is not in mixedCase
Parameter BytesLib.concatStorage(bytes,bytes)._preBytes (libraries/external/BytesLib.sol#91) is not in mixedCase
Parameter BytesLib.concatStorage(bytes,bytes)._postBytes (libraries/external/BytesLib.sol#91) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._bytes (libraries/external/BytesLib.sol#229) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._start (libraries/external/BytesLib.sol#230) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._length (libraries/external/BytesLib.sol#231) is not in mixedCase
Parameter BytesLib.toAddress(bytes,uint256)._bytes (libraries/external/BytesLib.sol#297) is not in mixedCase
Parameter BytesLib.toAddress(bytes,uint256)._start (libraries/external/BytesLib.sol#297) is not in mixedCase
Parameter BytesLib.toUint8(bytes,uint256)._bytes (libraries/external/BytesLib.sol#308) is not in mixedCase
Parameter BytesLib.toUint8(bytes,uint256)._start (libraries/external/BytesLib.sol#308) is not in mixedCase
Parameter BytesLib.toUint16(bytes,uint256)._bytes (libraries/external/BytesLib.sol#319) is not in mixedCase
Parameter BytesLib.toUint16(bytes,uint256)._start (libraries/external/BytesLib.sol#319) is not in mixedCase
Parameter BytesLib.toUint32(bytes,uint256)._bytes (libraries/external/BytesLib.sol#330) is not in mixedCase
Parameter BytesLib.toUint32(bytes,uint256)._start (libraries/external/BytesLib.sol#330) is not in mixedCase
Parameter BytesLib.toUint64(bytes,uint256)._bytes (libraries/external/BytesLib.sol#341) is not in mixedCase
Parameter BytesLib.toUint64(bytes,uint256)._start (libraries/external/BytesLib.sol#341) is not in mixedCase
Parameter BytesLib.toUint96(bytes,uint256)._bytes (libraries/external/BytesLib.sol#352) is not in mixedCase
Parameter BytesLib.toUint96(bytes,uint256)._start (libraries/external/BytesLib.sol#352) is not in mixedCase
Parameter BytesLib.toUint128(bytes,uint256)._bytes (libraries/external/BytesLib.sol#363) is not in mixedCase
Parameter BytesLib.toUint128(bytes,uint256)._start (libraries/external/BytesLib.sol#363) is not in mixedCase
Parameter BytesLib.toUint256(bytes,uint256)._bytes (libraries/external/BytesLib.sol#374) is not in mixedCase
Parameter BytesLib.toUint256(bytes,uint256)._start (libraries/external/BytesLib.sol#374) is not in mixedCase
Parameter BytesLib.toBytes32(bytes,uint256)._bytes (libraries/external/BytesLib.sol#385) is not in mixedCase
Parameter BytesLib.toBytes32(bytes,uint256)._start (libraries/external/BytesLib.sol#385) is not in mixedCase
Parameter BytesLib.equal(bytes,bytes)._preBytes (libraries/external/BytesLib.sol#396) is not in mixedCase
Parameter BytesLib.equal(bytes,bytes)._postBytes (libraries/external/BytesLib.sol#396) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._preBytes (libraries/external/BytesLib.sol#440) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._postBytes (libraries/external/BytesLib.sol#441) is not in mixedCase
Constant BridgeGovernance.module (bridge/BridgeGovernance.sol#25) is not in UPPER_CASE_WITH_UNDERSCORES
Constant NFTBridgeGovernance.module (nft/NFTBridgeGovernance.sol#24) is not in UPPER_CASE_WITH_UNDERSCORES
Function TokenImplementation.DOMAIN_SEPARATOR() (bridge/token/TokenImplementation.sol#316-318) is not in mixedCase
Parameter MockBatchedVAASender.setup(address)._wormholeCore (mock/MockBatchedVAASender.sol#50) is not in mixedCase
Function Bridge._parseTransferCommon(bytes) (bridge/Bridge.sol#727-745) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
Reentrancy in Bridge._wrapAndTransferETH(uint256) (bridge/Bridge.sol#116-149):
        External calls:
        - address(msg.sender).transfer(dust) (bridge/Bridge.sol#131)
        External calls sending eth:
        - address(msg.sender).transfer(dust) (bridge/Bridge.sol#131)
        - WETH().deposit{value: amount - dust}() (bridge/Bridge.sol#135-137)
        State variables written after the call(s):
        - bridgeOut(address(WETH()),normalizedAmount) (bridge/Bridge.sol#140)
                - _state.outstandingBridged[token] = outstanding (bridge/BridgeSetters.sol#56)
Reentrancy in MockWETH9.withdraw(uint256) (bridge/mock/MockWETH9.sol#42-47):
        External calls:
        - address(msg.sender).transfer(wad) (bridge/mock/MockWETH9.sol#45)
        Event emitted after the call(s):
        - Withdrawal(msg.sender,wad) (bridge/mock/MockWETH9.sol#46)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable BridgeSetters.setInitialized(address).implementation (bridge/BridgeSetters.sol#9) is too similar to BridgeGovernance.upgrade(bytes).implementation (bridge/BridgeGovernance.sol#51)
Variable Setters.setInitialized(address).implementation (Setters.sol#25) is too similar to Shutdown.initialize().implementation (Shutdown.sol#23)
Variable NFTBridgeSetters.setInitialized(address).implementation (nft/NFTBridgeSetters.sol#9) is too similar to NFTBridgeGovernance.upgrade(bytes).implementation (nft/NFTBridgeGovernance.sol#49)
Variable BridgeSetters.setInitialized(address).implementation (bridge/BridgeSetters.sol#9) is too similar to BridgeSetup.setup(address,uint16,address,uint16,bytes32,address,address,uint8,uint256).implementation
(bridge/BridgeSetup.sol#13)
Variable NFTBridgeSetters.setInitialized(address).implementation (nft/NFTBridgeSetters.sol#9) is too similar to NFTBridgeShutdown.initialize().implementation (nft/NFTBridgeShutdown.sol#22)
Variable BridgeSetters.setInitialized(address).implementation (bridge/BridgeSetters.sol#9) is too similar to BridgeShutdown.initialize().implementation (bridge/BridgeShutdown.sol#28)
Variable Setters.setInitialized(address).implementation (Setters.sol#25) is too similar to Setup.setup(address,address[],uint16,uint16,bytes32,uint256).implementation (Setup.sol#13)
Variable NFTBridgeSetters.setInitialized(address).implementation (nft/NFTBridgeSetters.sol#9) is too similar to NFTBridgeSetup.setup(address,uint16,address,uint16,bytes32,address,uint8,uint256).implementation (n
ft/NFTBridgeSetup.sol#13)
Variable MockBatchedVAASender.sendMultipleMessages(uint32,bytes,uint8).messageSequence0 (mock/MockBatchedVAASender.sol#22) is too similar to MockBatchedVAASender.sendMultipleMessages(uint32,bytes,uint8).messageS
equence1 (mock/MockBatchedVAASender.sol#23)
Variable MockBatchedVAASender.sendMultipleMessages(uint32,bytes,uint8).messageSequence0 (mock/MockBatchedVAASender.sol#22) is too similar to MockBatchedVAASender.sendMultipleMessages(uint32,bytes,uint8).messageS
equence2 (mock/MockBatchedVAASender.sol#24)
Variable MockBatchedVAASender.sendMultipleMessages(uint32,bytes,uint8).messageSequence1 (mock/MockBatchedVAASender.sol#23) is too similar to MockBatchedVAASender.sendMultipleMessages(uint32,bytes,uint8).messageS
equence2 (mock/MockBatchedVAASender.sol#24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
```

```
INFO:Detectors:
MockBridgeImplementation.slitherConstructorConstantVariables() (bridge/mock/MockBridgeImplementation.sol#8-25) uses literals with too many digits:
        - module = 0x000000000000000000000000000000000000000000000000546f6b656e427269646765 (bridge/BridgeGovernance.sol#25)
MockImplementation.slitherConstructorConstantVariables() (mock/MockImplementation.sol#8-17) uses literals with too many digits:
        - module = 0x000000000000000000000000000000000000000000000000000000436f7265 (Governance.sol#22)
Shutdown.slitherConstructorConstantVariables() (Shutdown.sol#20-31) uses literals with too many digits:
        - module = 0x000000000000000000000000000000000000000000000000000000436f7265 (Governance.sol#22)
NFTBridge._createWrapped(uint16,bytes32,bytes32,bytes32) (nft/NFTBridge.sol#152-193) uses literals with too many digits:
        - name = 0x576f726d686f6c65204272696467656420536f6c616e612d4e46540000000000 (nft/NFTBridge.sol#159)
NFTBridge._createWrapped(uint16,bytes32,bytes32,bytes32) (nft/NFTBridge.sol#152-193) uses literals with too many digits:
        - symbol = 0x574f524d53504c4e4654000000000000000000000000000000000000000000 (nft/NFTBridge.sol#161)
MockNFTBridgeImplementation.slitherConstructorConstantVariables() (nft/mock/MockNFTBridgeImplementation.sol#8-21) uses literals with too many digits:
        - module = 0x000000000000000000000000000000000000000000000000e4654427269646765 (nft/NFTBridgeGovernance.sol#24)
NFTBridgeShutdown.slitherConstructorConstantVariables() (nft/NFTBridgeShutdown.sol#19-30) uses literals with too many digits:
        - module = 0x000000000000000000000000000000000000000000000000e4654427269646765 (nft/NFTBridgeGovernance.sol#24)
BridgeShutdown.slitherConstructorConstantVariables() (bridge/BridgeShutdown.sol#25-36) uses literals with too many digits:
        - module = 0x000000000000000000000000000000000000000000000000546f6b656e427269646765 (bridge/BridgeGovernance.sol#25)
BytesLib.toAddress(bytes,uint256) (libraries/external/BytesLib.sol#297-306) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (libraries/external/BytesLib.sol#302)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
MockWETH9.decimals (bridge/mock/MockWETH9.sol#21) should be constant
MockWETH9.name (bridge/mock/MockWETH9.sol#19) should be constant
MockWETH9.symbol (bridge/mock/MockWETH9.sol#20) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
Migrations.owner (Migrations.sol#5) should be immutable
Migrator.fromAsset (bridge/utils/Migrator.sol#11) should be immutable
Migrator.fromDecimals (bridge/utils/Migrator.sol#13) should be immutable
Migrator.toAsset (bridge/utils/Migrator.sol#12) should be immutable
Migrator.toDecimals (bridge/utils/Migrator.sol#14) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:. analyzed (81 contracts with 85 detectors), 245 result(s) found
```

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of the code. Besides, a security audit, please don't consider this report as investment advice.

Summary

The use of smart contracts is simple and the code is easy to understandl.Altogether the code is written and demonstrates effective use of abstraction, separation of concern, and modularity. But there are a few issues/vulnerabilities to be tackled at various security levels, it is recommended to go through them before deploying the contract