Security Audit Report

SIX Network Bridge

PREPARED FOR:

SIX Network

ARCADIA CONTACT INFO
Email: audits@arcadiamgroup.com
Telegram: https://t.me/thearcadiagroup

Revision history

Date	Commit
5/15/2023	#95881a748f1b2382c28c1b758653288c3f0acee7
6/14/2023	#d14775763bfa105e76c95f2acb7f592ceaeadd3d
	#53d7af08eaf93e1f65367f1cdaeedd85913ea54b
	#3700e76a1f8a076a55749c9b25b190fd1d2d461f

Table of Contents

Executive Summary

Introduction

SIX Network engaged Arcadia to perform a security audit of their bridge smart contracts within the SIX Network organization. Our review of their codebase occurred on the commit hash #**95881a748f1b2382c28c1b758653288c3f0acee7**

Review Team

1. Tuan “Anhnt” Nguyen - Security Researcher and Engineer
2. Joel Farris - Project Manager

Project Background

The SIX Protocol is a purpose-built blockchain infrastructure tailored for real-world businesses and enterprises. It offers a comprehensive suite of components designed to harness the capabilities of Web3 for businesses. These components include a dynamic data layer for secure storage and exchange of NFTs, the decentralized wallet called SIX Vault, the on-chain solution NFT Gen 2 to enhance NFT utility, the SIX ZONE marketplace, the SIX Bridge for seamless connectivity, and the Definix investment platform.

Coverage

For this audit, we performed research, test coverage, investigation, and review of SIX followed by issue reporting, along with mitigation and remediation instructions as outlined in this report. The following code repositories, files, and/or libraries are considered in scope for the review.

Files
contracts-token/SIXErc20.sol
contracts/Bridge.sol
contracts/libraries/LibDiamond.sol

contracts/facets/AccessFacet.sol
contracts/facets/BridgeManagerFacet.sol
contracts/facets/BridgeSwapInFacet.sol
contracts/facets/BridgeSwapOutFacet.sol
contracts/facets/DiamondCutFacet.sol
contracts/facets/DiamondLoupeFacet.sol
contracts/facets/FeeManagerFacet.sol
contracts/facets/OwnershipFacet.sol

Methodology

Arcadia completed this security review using various methods, primarily consisting of dynamic and static analysis. This process included a line-by-line analysis of the in-scope contracts, optimization analysis, analysis of key functionalities and limiters, and reference against intended functionality.

The followings are the steps we have performed while auditing the smart contracts:

- Investigating the project and its technical architecture overview through its documentation
- Understanding the overview of the smart contracts, the functions of the contracts, the inheritance, and how the contracts interface with each others thanks to the graph created by Solidity Visual Developer
- Manual smart contract audit:
  - Review the code to find any issue that could be exploited by known attacks listed by Consensys
  - Identifying which existing projects the smart contracts are built upon and what are the known vulnerabilities and remediations to the existing projects
  - Line-by-line manual review of the code to find any algorithmic and arithmetic related vulnerabilities compared to what should be done based on the project's documentation
  - Find any potential code that could be refactored to save gas
  - Run through the unit-tests and test-coverage if exists
- Static Analysis:
  - Scanning for vulnerabilities in the smart contracts using Static Code Analysis Software
  - Making a static analysis of the smart contracts using Slither
- Fuzzing

    - Arcadia assisted in writing and ensuring full coverage of fuzzing implementations
- Additional review: a follow-up review is done when the smart contracts have any new update. The follow-up is done by reviewing all changes compared to the audited commit revision and its impact to the existing source code and found issues.

Summary

The project is development based on EIP-2535 architecture and is a Trusted model.

There were **11** issues found, **0** of which were deemed to be ‘critical’, and **0** of which were rated as ‘high’. At the end of these issues were found throughout the review of a rapidly changing codebase and not a final static point in time.

Severity Rating	Number of Original Occurrences	Number of Remaining Occurrences
CRITICAL	0	0
HIGH	0	0
MEDIUM	3	3
LOW	3	2
INFORMATIONAL	5	0

Findings in Manual Audit

(SX-1) `BridgeIn` - too much power in operator's hand.

Status

Unresolved

Reason

One of the most common issues of centralized bridge, the team has acknowledged the problem and taken steps to mitigate it by maintaining a low balance in the hot wallet. The solution Arcadia proposed either the increasing gas cost (**solution 1**) or the extended development time (**solution 2**).

Risk Level

Severity: Medium

Code Segment

Description

If an operator has control over ***bridgeIn***, they have the ability to submit and drain all bridge's balance. Since SIX's bridge is based on a trusted model, operations must be executed with extreme caution.

Code Location

```
contracts/facets/BridgeSwapInFacet.sol
```

Proof of Concept

–

Recommendation

Solution 1: Implement a two-step model that delegates the submission of bridgeIn transactions to specific operators, while designating other operators to validate and process these transactions.

Solution 2: Whenever a user deposits funds to the bridge's address on the source chain, SIX's bridge issues the user a signature to facilitate the retrieval of equivalent assets on the destination chain. This approach offers substantial cost savings, particularly when the destination chain is Ethereum, as users are relieved from the burden of paying for the asset claim process themselves.

(SX-2) No test code is present whatsoever.

Status

Unresolved

Reason

The team has acknowledged the problem but has decided not to address it.

Risk Level

Severity: Medium

Code Segment

Description

Bridges have been implicated in numerous major blockchain hacks. The absence of any test code poses a significant risk for potential issues in the future.

Code Location

Proof of Concept

-

Recommendation

To ensure comprehensive coverage, it is strongly recommended to cover at least 90% of the code. Additionally, we suggest migrating the project to *Hardhat* or *Foundry* for enhanced development and testing capabilities..

(SX-3) `BridgeIn` - lacking original deposit transaction info.

Status

Unresolved

Reason

The team has decided to use the off-chain solutions.

Risk Level

Severity: Medium

Code Segment

```
function bridgeIn(
        address _tokenAddr,
        address _toAddr,
        string memory _toMemo,
        uint256 _amount,
        string memory _ticket,
        uint256 _sourceChain
   ) external payable override returns (bool){
```

Description

No original transaction information is available to verify the correctness of this bridgeIn transaction. Without this information, it is not possible for a third party to verify that your bridge asset maintains a 1:1 ratio with the user's deposited asset.

Code Location

```
contracts/facets/BridgeSwapInFacet.sol
```

Proof of Concept

-

Recommendation

Incorporate the original transaction information to enable verification, auditing, and accounting of the bridge-in assets.

(SX-4) `checkPermisionBridgeControlByToken` should return value.

Status

Resolved

Risk Level

Severity: Low

Code Segment

```
function checkPermisionBridgeControlByToken(address _callerAddr,address _tokenAddress)
public view override {
         address bridgeManagerAddr =
IAccessFacet(address(this)).getAddressForRole(_BRIDGE_MANAGER_ROLE_KEY);
         if(_callerAddr == bridgeManagerAddr || _callerAddr == LibDiamond.contractOwner()){
                return;
         }else if(getAllowAccessForTokenOwner(_tokenAddress)){
              if(_callerAddr == getBridgeTokenManager(_tokenAddress)){
                   return;
              }
         }
         revert("BridgeManagerFacet: Caller don't have permission");
  }
```

Description

Since this function is utilized both internally and externally, it is recommended to have a return value to enhance its comprehensiveness and usability

Code Location

```
contracts/facets/BridgeManagerFacet.sol
```

Proof of Concept

–

Recommendation

Return value for function.

(SX-5) Multiple functions are required to enable a token on the bridge.

Status

Unresolved

Reason

The team has acknowledged the problem but has decided not to address it.

Risk Level

Severity: Low

Code Segment

Description

To enable a token on bridge requires multiple functions called, *setMaximumAmountLimitPerTrans, setAllowDestChain, setAllowSourceChain, setMinimumAmountLimitPerTrans, setHotWalletAddr, setFeeNormalReceiverAddr, setBridgeFeeNormalByChain.*Some functions requires chainID, some do not. However, managing a growing number of tokens, such as 10, 20, and beyond, with these functions can lead to management complexities.

Code Location

```
contracts/facets/BridgeManagerFacet.sol
contracts/facets/FeeManagerFacet.sol
```

Proof of Concept

–

Recommendation

Consider utilizing one or two functions to enable or disable a token on the bridge. Additionally, when no fee is explicitly set for a token, it is recommended to use the default fee.

(SX-5) `bridgeOut` - *`ticket`* parameter should be generated inside the code.

Status

Unresolved

Reason

The team has acknowledged the problem but has decided not to address it.

Risk Level

Severity: Low

Code Segment

```
function bridgeOut(
        address _tokenAddr,
        string memory _toAddr,
        string memory _toMemo,
        uint256 _amount,
        string memory _ticket,
        uint256 _destChain
   ) external payable override returns (bool) {
```

Description

This function enables users to withdraw funds from the SIX Network. The _ticket parameter is intended for unique identification purposes only and should not be user-selectable.

Code Location

```
contracts/facets/BridgeSwapOutFacet.sol
```

Proof of Concept

–

Recommendation

Generate the `_ticket` inside the solidity code.

(SX-7) *`SwapInTicketStruct`* - lacking the *`tokenAddress`* info.

Status

Resolved

Risk Level

Severity: Informational

Code Segment

Description

The struct does not store the tokenAddress, which is crucial information for a bridgeIn transaction.

Code Location

```
contracts/libraries/LibDiamond.sol
```

Proof of Concept

–

Recommendation

Store `tokenAddress` in `SwapInTicketStruct.`

(SX-8) The flag is not being used.

Status

Resolved

Risk Level

Severity: Informational

Code Segment

```
function setBridgeFeeNormalByChain(address _tokenAddr,uint256 _chainId,uint256
_fee) external override onlyTokenManager{
LibDiamond.setUint(keccak256(abi.encodePacked("FEE_NORMAL_AMOUNT",_tokenAddr,_chain
Id)),_fee);
LibDiamond.setBool(keccak256(abi.encodePacked("IS_FEE_NORMAL_SET",_tokenAddr,_chain
Id)),true);

    }
```

Description

`IS_FEE_NORMAL_SET` is never used.

Code Location

```
contracts/facets/FeeManagerFacet.sol
```

Proof of Concept

–

Recommendation

Should be removed.

(SX-9) Emitting more events.

Status

Resolved

Risk Level

Severity: Informational

Code Segment

Description

To validate the proper deployment and initialization of the contracts, it's a good practice to emit events, also any important state transaction can be logged, which is beneficial for monitoring the contract and tracking eventual bugs.

Code Location

```
contracts/facets/BridgeManagerFacet.sol
contracts/facets/OwnershipFacet.sol
contracts/facets/FeeManagerFacet.sol
```

Proof of Concept

–

Recommendation

Emit events.

(SX-10) Messy deployment script.

Status

Unresolved

Risk Level

Severity: Informational

Code Segment

Description

At the time we received the source code, there was no direct script to deploy the code. Validating the accurate deployment and initialization of the contracts has been challenging. It is strongly recommended to include test/deploy scripts in the package.json file to streamline the process.

Code Location

Proof of Concept

–

Recommendation

Add the script to compile/deploy/test

(SX-11) `userTotalRecive` variable contains a typo.

Status

Resolved

Risk Level

Severity: Informational

Code Segment

Description

The variable name includes a misspelling.

Code Location

Proof of Concept

–

Recommendation

Automated Tests and Tooling

Static Analysis with Slither

As a part of our engagement with SIX, we ran a static analysis against the source code using Slither, which is a Solidity static analysis framework written in Python. Slither runs a suite of vulnerability detectors and prints visual information about contract details. Slither enables developers to find vulnerabilities, enhance their code comprehension, and quickly prototype custom analyses.

While Slither is not the primary element of Arcadia's offering, in some cases, it can be useful. The following shows the results found by the static analysis by Slither. We reviewed the results, and all of the issues found by Slither were at that point in time false positives.

```
→  evm-bridge-contract-review git:(master) ✗ slither contracts
INFO:Detectors:
Function Bridge.constructor(address,address) (contracts/Bridge.sol#16-30) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#91-96) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function LibDiamond.setSwapOutTicket(string,LibDiamond.SwapOutTicketStruct) (contracts/libraries/LibDiamond.sol#263-266) is a strange setter. Nothing is set in constructor or set in a function without usi
ng function parameters
Function LibDiamond.setSwapInTicket(string,LibDiamond.SwapInTicketStruct) (contracts/libraries/LibDiamond.sol#267-270) is a strange setter. Nothing is set in constructor or set in a function without using
 function parameters
Function LibDiamond.setUint(bytes32,uint256) (contracts/libraries/LibDiamond.sol#272-275) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function LibDiamond.setAddress(bytes32,address) (contracts/libraries/LibDiamond.sol#277-280) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function LibDiamond.setString(bytes32,string) (contracts/libraries/LibDiamond.sol#282-285) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function LibDiamond.setBool(bytes32,bool) (contracts/libraries/LibDiamond.sol#287-290) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function LibDiamond.setArrayUint(bytes32,uint256[]) (contracts/libraries/LibDiamond.sol#291-294) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md
INFO:Detectors:
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#187-205) is an unprotected initializer.
Function LibDiamond.initialToken() (contracts/libraries/LibDiamond.sol#295-297) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
Token contract has a Bridge.fallback() (contracts/Bridge.sol#34-61) which might be dangerous to implement
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/token_fallback.md
INFO:Detectors:
LibDiamond.replaceFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#151) is a local variable never initialized
LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes).facetIndex (contracts/libraries/LibDiamond.sol#114) is a local variable never initialized
LibDiamond.addFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#136) is a local variable never initialized
LibDiamond.removeFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#168) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#187-205) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
INFO:Detectors:
Setter function LibDiamond.setSwapOutTicket(string,LibDiamond.SwapOutTicketStruct) (contracts/libraries/LibDiamond.sol#263-266) does not emit an event
Setter function LibDiamond.setSwapInTicket(string,LibDiamond.SwapInTicketStruct) (contracts/libraries/LibDiamond.sol#267-270) does not emit an event
Setter function LibDiamond.setUint(bytes32,uint256) (contracts/libraries/LibDiamond.sol#272-275) does not emit an event
Setter function LibDiamond.setAddress(bytes32,address) (contracts/libraries/LibDiamond.sol#277-280) does not emit an event
Setter function LibDiamond.setString(bytes32,string) (contracts/libraries/LibDiamond.sol#282-285) does not emit an event
Setter function LibDiamond.setBool(bytes32,bool) (contracts/libraries/LibDiamond.sol#287-290) does not emit an event
Setter function LibDiamond.setArrayUint(bytes32,uint256[]) (contracts/libraries/LibDiamond.sol#291-294) does not emit an event
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/event_setter.md
INFO:Detectors:
Bridge.fallback() (contracts/Bridge.sol#34-61) uses assembly
        - INLINE ASM (contracts/Bridge.sol#38-40)
        - INLINE ASM (contracts/Bridge.sol#45-60)
LibDiamond.diamondStorage() (contracts/libraries/LibDiamond.sol#82-87) uses assembly
        - INLINE ASM (contracts/libraries/LibDiamond.sol#84-86)
LibDiamond.enforceHasContractCode(address,string) (contracts/libraries/LibDiamond.sol#207-213) uses assembly
        - INLINE ASM (contracts/libraries/LibDiamond.sol#209-211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
LibDiamond.contractOwner() (contracts/libraries/LibDiamond.sol#98-100) is never used and should be removed
LibDiamond.enforceIsContractOwner() (contracts/libraries/LibDiamond.sol#102-104) is never used and should be removed
LibDiamond.getAddress(bytes32) (contracts/libraries/LibDiamond.sol#243-246) is never used and should be removed
LibDiamond.getArrayUint(bytes32) (contracts/libraries/LibDiamond.sol#257-260) is never used and should be removed
LibDiamond.getBool(bytes32) (contracts/libraries/LibDiamond.sol#253-256) is never used and should be removed
LibDiamond.getString(bytes32) (contracts/libraries/LibDiamond.sol#248-251) is never used and should be removed
LibDiamond.getSwapInTicket(string) (contracts/libraries/LibDiamond.sol#233-236) is never used and should be removed
LibDiamond.getSwapOutTicket(string) (contracts/libraries/LibDiamond.sol#229-232) is never used and should be removed
LibDiamond.getUint(bytes32) (contracts/libraries/LibDiamond.sol#238-241) is never used and should be removed
LibDiamond.initialToken() (contracts/libraries/LibDiamond.sol#295-297) is never used and should be removed
LibDiamond.registerInterface(bytes4) (contracts/libraries/LibDiamond.sol#220-224) is never used and should be removed
```

Conclusion

Arcadia identified issues that occurred at hash **#95881a748f1b2382c28c1b758653288c3f0acee7**.

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, The Arcadia Group and the Authors assume no responsibility for errors, omissions, or damages resulting from the use of the provided information. Additionally, Arcadia would like to emphasize that the use of Arcadia's services does not guarantee the security of a smart contract or set of smart contracts and does not guarantee against attacks. One audit on its own is not enough for a project to be considered secure; that categorization can only be earned through extensive peer review and battle testing over an extended period.