Security Bitkubnext Vault Audit Report

PREPARED FOR:

SIX Network

ARCADIA CONTACT INFO
Email: audits@arcadiamgroup.com

Telegram: https://t.me/thearcadiagroup

Revision history

Date	Reason	Commit
8/28/2023	Initial Audit Scope	#2947260c4a68ad6f19af126c097f29d75cf925b0
9/13/2023	Review Of Remediations	#f102b3073c0a1064af0660c3a955de84f3efffd8

Table of Contents

Executive Summary

Introduction

SIX Network engaged Arcadia to perform a security audit of their Bitkubnext vault smart contracts within the SIX Network organization. Our review of their codebase occurred on the commit hash #2947260c4a68ad6f19af126c097f29d75cf925b0

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/SixKubVaultSigner.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
- 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

There were **4** 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	1	0
LOW	2	1
INFORMATIONAL	1	0

Findings in Manual Audit

(SXV-1) Incorrect approaches to validating state-altering functions.

Status

Resolved

Risk Level

Severity: Medium

Code Segment

```
if (msg.sender == address(0) || _expireEpoch <= block.timestamp) {
     return false;
}
```

Description

In all state-altering functions like ***setDelegationSigner*** and ***removeDelegationSigner***, validating input using a '*return false*' approach can lead to users incurring gas fees for invalidated inputs.

Recommendation

Consider using 'require' statements to validate input and address this issue.

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

Status

Acknowledged

Reason

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

Risk Level

Severity: Low

Description

The absence of any test code poses a significant risk for potential issues in the future.

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..

(SXV-3) The code for removing elements from an array is not gas-efficient.

Status

Resolved

Risk Level

Severity: Informational

Code Segment

```
for (uint256 i = 0; i < binedSigner[msg.sender].signers.length; i++) {
     if (binedSigner[msg.sender].signers[i].actor_address == _actor) {
        delete binedSigner[msg.sender].signers[i];
      }
}
binedSigner[msg.sender].actorCount -= 1;
```

Recommendation

Here's an enhanced version that significantly improves gas efficiency.

```
for (uint256 i = 0; i < binedSigner[msg.sender].signers.length; i++) {
      if (binedSigner[msg.sender].signers[i].actor_address == _actor) {
          // Swap the element to delete with the last element.
          binedSigner[msg.sender].signers[i] =
binedSigner[msg.sender].signers[binedSigner[msg.sender].signers.length - 1];
           // Reduce the array's length by one to "remove" the last element.
           binedSigner[msg.sender].signers.pop();
          // Exit the loop since we found and removed the element.
          break;
      }
}
```

(SXV-4) Poor Code Quality Assessment

Status

Resolved

Risk Level

Severity: Low

Code Segment

```
event EventTypeSetActionSigner(
    address actor,
    address owner,
    uint256 expired_at
);
event EventTypeRemoveActionSigner(address actor, address owner);
```

Description

Key Issues Identified:

- Absence of **'expired_epoch'** Validation.

- Verbosity in Event Naming and Missing Parameter Indexing.
- Non-Informative Copy/Paste Comments.
- Absence of the **'immutable'** Keyword for Fields with Single Assignments.

Recommendation

Revise the indexing for an event's parameters; change the name of them (e.g. to **DelegationSignerSet**/**DelegationSignerRemove**). Utilize the **immutable** keyword for fields that are set only once. Remove some copy/paste comments that cause confusion.

Conclusion

Most of the issues are addressed at commit #f102b3073c0a1064af0660c3a955de84f3efffd8.

Disclaimer