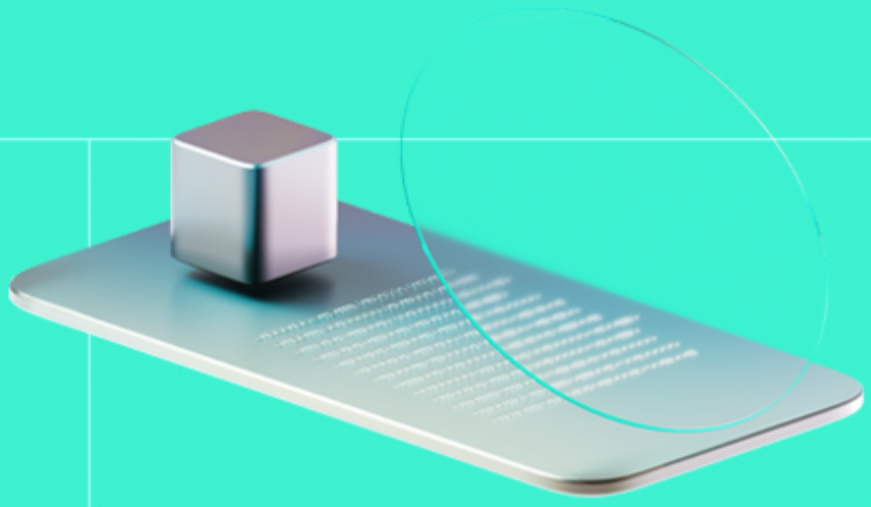
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Dione Protocol

Date: 24/12/2024

We express our gratitude to the Dione Protocol team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

The Dione NFT Bridge is a bridge between Ethereum and Odyssey blockchains for bridging NFTs and ERC1155 tokens within the Dione protocol.

Document

Name	Smart Contract Code Review and Security Analysis Report for Dione Protocol
Audited By	Grzegorz Trawinski
Approved By	Przemyslaw Swiatowiec
Website	https://www.dioneprotocol.com/
Changelog	11/12/2024 - Preliminary Report
	24/12/2024 - Final Report
Platform	Ethereum, Odyssey
Language	Solidity
Tags	Bridge, NFT, Semi-fungible Token, ERC721, ERC1155
Methodology	https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/DioneProtocol/nft-bridge-backend
Commit	3f2a880
Retest Commit	5b1bc23, 67bdd20

Audit Summary

The system users should acknowledge all the risks summed up in the risks section of the report

3	3	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by Severity

Severity	Count
Critical	0
High	0
Medium	1
Low	2

Vulnerability	Severity
F-2024-7542 - Protocol owner has access to the NFTs	Medium
F-2024-7545 - Incorrect assertion within the configDstNftSc function	Low
F-2024-7597 - Tbe bridgeNFT allows to transfer to address 0	Low

Documentation quality

- Sufficient documentation was provided in the project's README.md.

Code quality

- Code represents mature quality.

Test coverage

Code coverage of the project is around **65%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- The tests assumes bridging tokens to it-self, thus, no minting new NFTs is simulated. This part was covered in QA testing.

Table of Contents

System Overview

- DioneNFTBridge - a bridge between Ethereum and Odyssey blockchains for bridging NFTs and ERC1155 tokens within the Dione protocol. This contract is going to be deployed on both Ethereum and Odyssey blockchains.
- Elysium88 - a NFT token to be minted on the Odyssay side of the bridge.
- ICustomERC721Upgradeable - a custom interface to expose `mint` and `uri` functions.
- ICustomERC1155Upgradeable - a custom interface to expose `safeMint` and `tokenURI` functions.

Privileged roles

- The `DEFAULT_ADMIN_ROLE` role of Elysium88 can manage the access to the contract.
- The `MINTER_ROLE` role of Elysium88 can mint new tokens.
- The `DEFAULT_ADMIN_ROLE` role of DioneNFTBridge can manage protocol's configuration: pause, unpause, set Gas limit, set Wmb Gateway, acceptable NFT addresses on both source and destination, retrieve the NFT from the chain.
- The DioneNFTBridge proxy owner can upgrade the protocol.

Potential Risks

- **Interactions with External DeFi Protocols**: Dependence on external DeFi protocols (Wanchain Message Bridge) inherits their risks and vulnerabilities. This might lead to direct financial losses if these protocols are exploited, indirectly affecting the audited project.
- **Centralized Control of Minting Process**: The token contract’s design allows for centralized control over the minting process, posing a risk of unauthorized token issuance, potentially diluting the token value and undermining trust in the project's economic governance.
- **Owner's Unrestricted State Modification**: The absence of restrictions on state variable modifications by the owner leads to arbitrary changes, affecting contract integrity and user trust, especially during critical operations like minting phases.
- **Single Points of Failure and Control**: The project is fully or partially centralized, introducing single points of failure and control. This centralization can lead to vulnerabilities in decision-making and operational processes, making the system more susceptible to targeted attacks or manipulation.
- **Administrative Key Control Risks**: The digital contract architecture relies on administrative keys for critical operations. Centralized control over these keys presents a significant security risk, as compromise or misuse can lead to unauthorized actions or loss of funds.
- **Single Entity Upgrade Authority**: The token ecosystem grants a single entity the authority to implement upgrades or changes. This centralization of power risks unilateral decisions that may not align with the community or stakeholders' interests, undermining trust and security.
- **Flexibility and Risk in Contract Upgrades**: The project's contracts are upgradable, allowing the administrator to update the contract logic at any time. While this provides flexibility in addressing issues and evolving the project, it also introduces risks if upgrade processes are not properly managed or secured, potentially allowing for unauthorized changes that could compromise the project's integrity and security.
- **Absence of Upgrade Window Constraints**: The contract suite allows for immediate upgrades without a mandatory review or waiting period, increasing the risk of rapid deployment of malicious or flawed code, potentially compromising the system's integrity and user assets.

Findings

Vulnerability Details

F-2024-7542 - Protocol owner has access to the NFTs - Medium

Description:

The `DioneNFTBridge` contract allows the privileged user to access all NFTs deposited within the bridge by means of the `adminRetrieveNFT`. This function is meant to retrieve stuck NFT, however, it does not check whether the `user` originally owned the particular NFT upon retrieval. Thus, any token can be transferred to any user. This functionality can be especially dangerous assuming the account with the `DEFAULT_ADMIN_ROLE` is compromised.

```
    function adminRetrieveNFT(
        address user,
        address nftContract,
        uint256 tokenId
    ) external onlyRole(DEFAULT_ADMIN_ROLE) nonReentrant {
        require(user != address(0), "DioneNFTBridge: Invalid user address");
        require(nftContract != address(0), "DioneNFTBridge: Invalid nftContra
ct");

        if (_isERC721(nftContract)) {
            ICustomERC721Upgradeable erc721 = ICustomERC721Upgradeable(
                nftContract
            );
            require(
                erc721.ownerOf(tokenId) == address(this),
                "DioneNFTBridge: NFT not held by bridge"
            );
            erc721.safeTransferFrom(address(this), user, tokenId);
        } else if (_isERC1155(nftContract)) {
            ICustomERC1155Upgradeable erc1155 = ICustomERC1155Upgradeable(
                nftContract
            );
            require(
                erc1155.balanceOf(address(this), tokenId) > 0,
                "DioneNFTBridge: NFT not held by bridge"
            );
            erc1155.safeTransferFrom(address(this), user, tokenId, 1, "");
        } else {
            revert("DioneNFTBridge: Unsupported token standard");
        }
```

```
        emit AdminRetrievedNFT(user, nftContract, tokenId);
    }
```

Assets:

- DioneNFTBridge.sol [https://github.com/DioneProtocol/nft-bridge-backend]

Status: Fixed

Classification

Impact: 5/5

Likelihood: 3/5

Exploitability: Semi-Dependent

Complexity: Simple

Severity: Medium

Recommendations

Remediation: It is recommended to remove aforementioned functionality to remove any associated risk. Alternatively, it is recommended to track addresses who were NFTs owners upon bridging and limit retrieval of the tokens to the saved list of pairs.

Resolution: Fixed in `5b1bc23ec4b2e8c94863c3dfc0d0783e1ce79694`.
The protocol now tracks the owners of ERC721 and ERC1155 tokens who did bridge tokens both directions. The `adminRetrieveAssets` only allows to retrieve tokens belonging to the original owners.

F-2024-7545 - Incorrect assertion within the configDstNftSc function - Low

Description: The `configDstNftSc` configures the source and destination addresses for the bridging operation. It implements assertions that both addresses cannot be 0. However, both assertions check the `srcNftSc` input variable and there is no check for `dstNftSc`. Thus, the incorrect validation increases the risk that `dstNftSc` is going to be set to address 0.

```
    function configDstNftSc(
        address srcNftSc,
        uint dstChainId,
        address dstNftSc
    ) external onlyRole(DEFAULT_ADMIN_ROLE) {
        require( srcNftSc != address(0), "DioneNFTBridge: invalid srcNftSc, z
ero address");
        require( srcNftSc != address(0), "DioneNFTBridge: invalid dstNftSc, z
ero address");
        dstChainNftSC[srcNftSc][dstChainId] = dstNftSc;
    }
```

Assets:

- DioneNFTBridge.sol [https://github.com/DioneProtocol/nft-bridge-backend]

Status: Fixed

Classification

Impact: 4/5

Likelihood: 1/5

Exploitability: Semi-Dependent

Complexity: Simple

Severity: Low

Recommendations

Remediation: It is recommended to fix the assertion so both input addresses are asserted.

Resolution: Fixed in commit `5b1bc23ec4b2e8c94863c3dfc0d0783e1ce79694`.
The `configDstAssetSc` function now has correct assertions.

F-2024-7597 - Tbe bridgeNFT allows to transfer to address 0 - Low

Description:

The `bridgeNFT` allows user to bridge any held NFT token. However, it does not check whether the `to` address is not 0 address. Thus, in the event of human error, the NFT can be transferred to the address 0 without possibility of retrieval.

```
function bridgeNFT(
    address srcNftSC,
    uint dstChainId,
    address to,
    uint256 tokenId
) public payable nonReentrant whenNotPaused {
    // Ensure the destination chain ID is not the current chain
    require(
        dstChainId != IChainID(wmbGateway).chainId(),
        "DioneNFTBridge: wrong dstChainId"
    );
    // Ensure the destination bridge smart contract address exists
    require(
        dstChainBridgeSC[dstChainId] != address(0),
        "DioneNFTBridge: dstChainBridgeSC doesn't exists"
    );
    // Ensure the destination NFT smart contract address exists
    require(
        dstChainNftSC[srcNftSC][dstChainId] != address(0),
        "DioneNFTBridge: dstChainNftSC doesn't exists"
    );
    // Ensure sufficient fee is provided
    uint fee = estimateFee(dstChainId, _gasLimit);
    require(msg.value >= fee, "DioneNFTBridge: insufficient fee");

    // Handle ERC721 and ERC1155 NFTs
    string memory uri;
    if (_isERC721(srcNftSC)) {
    ...
```

Assets:

- DioneNFTBridge.sol [https://github.com/DioneProtocol/nft-bridge-backend]

Status: Fixed

Classification

Impact: 5/5

Likelihood: 1/5

Exploitability: Independent

Complexity: Simple

Severity: Low

Recommendations

Remediation: It is recommended to add an assertion that prevents bridging NFT tokens to the address 0.

Resolution: Fixed in `5b1bc23ec4b2e8c94863c3dfc0d0783e1ce79694`.
The protocol now disallows to bridge to address 0 within the `bridgeAssets` function.

Observation Details

F-2024-7539 - The setupWmbGateway function has duplicate assertion - Info

Description: The `setupWmbGateway` function has duplicate assertion implemented, one time in itself, second time within the `_setupWmbGateway` internal function. Thus, it consumes additional Gas needlessly.

```
    function setupWmbGateway(
        address gateway
    ) public onlyRole(DEFAULT_ADMIN_ROLE) {
        require( gateway != address(0), "DioneNFTBridge: invalid gateway, zer
o address");
        _setupWmbGateway(gateway);
    }
```

```
    function _setupWmbGateway(address gateway) internal {
        require(gateway != address(0), "WmbApp: gateway address zero");
        wmbGateway = gateway;
    }
```

Assets:

- DioneNFTBridge.sol [https://github.com/DioneProtocol/nft-bridge-backend]

Status: Fixed

Recommendations

Remediation: It is recommended to remove redundant assertion to save some Gas during the deployment and transaction processing.

Resolution: Fixed in commit `5b1bc23ec4b2e8c94863c3dfc0d0783e1ce79694`.
The duplication is now removed.

F-2024-7540 - The NFT bridge supports ERC1155 tokens - Info

Description: The `DioneNFTBridge` allows to bridge both ERC721 and ERC1155 tokens via `bridgeNFT` function. The name convention used within the protocol can be misleading however, as the ERC1155 is considered as Semi-fungible Token.

Assets:

- DioneNFTBridge.sol [https://github.com/DioneProtocol/nft-bridge-backend]

Status: Fixed

Recommendations

Remediation: It is recommended to either change the name of the bridge and related functionalities to reflect the actual functionality implement, or split the `bridgeNFT` function into two that handle ERC721 and ERC1155 separately. Alternatively, it is recommended to carefully document the functionality.

Resolution: Fixed in commit `5b1bc23ec4b2e8c94863c3dfc0d0783e1ce79694`.
The bridge name and functions is now renamed to contain `Assets` keyword, as it supports both ERC721 and ERC1155 tokens.

F-2024-7541 - User can bridge ERC1155 tokens with 1 amount only - Info

Description:

The `bridgeNFT` function allows bridging of both ERC721 and ERC1155 tokens. However, there is not configuration allowing to bridge more than one ERC1155 token at once. Thus, the implementation limits the capabilities in this area and the end user is enforced to trigger multiple bridging processes to transfer desired amount of the ERC1155 tokens.

```
function bridgeNFT(
        address srcNftSC,
        uint dstChainId,
        address to,
        uint256 tokenId
    ) public payable nonReentrant whenNotPaused {
...
        if (_isERC721(srcNftSC)) {
...
        } else if (_isERC1155(srcNftSC)) {
            ICustomERC1155Upgradeable erc1155 = ICustomERC1155Upgradeable(
                srcNftSC
            );
            require(
                erc1155.balanceOf(msg.sender, tokenId) > 0,
                "DioneNFTBridge: NFT balance is zero"
            );

            // Ensure the contract is approved to transfer the token
            require(
                erc1155.isApprovedForAll(msg.sender, address(this)),
                "DioneNFTBridge: Contract is not approved"
            );

            erc1155.safeTransferFrom(msg.sender, address(this), tokenId, 1, "
");

            // Retrieve the URI if available
            uri = _getERC1155URI(erc1155, tokenId);
        } else {
            revert("DioneNFTBridge: Unsupported token standard");
        }

        // Dispatches message to gateway
        bytes32 res = _dispatchMessage(
```

```
            dstChainId,
            dstChainBridgeSC[dstChainId],
            abi.encode(dstChainNftSC[srcNftSC][dstChainId], to, tokenId, uri)
,
            fee
        );

        emit NFTBridged(res);
    }
```

Status: Fixed

Recommendations

Remediation: It is recommended to allow users to bridge any amount of ERC1155 tokens in single transaction.

Resolution: Fixed in commit `5b1bc23ec4b2e8c94863c3dfc0d0783e1ce79694`.
The protocol now allows to transfer any amount of ERC1155 tokens.

F-2024-7543 - Elysium88 does not benefit from AccessControlDefaultAdminRulesUpgradeable - Info

Description:

The `Elysium88` contract inherit from the `AccessControlUpgradeable` library. However, there is an alternative extension available from the OpenZepplin provider: `AccessControlDefaultAdminRulesUpgradeable`.

Extension of `AccessControl` that allows specifying special rules to manage the `DEFAULT_ADMIN_ROLE` holder, which is a sensitive role with special permissions over other roles that may potentially have privileged rights in the system. If a specific role does not have an admin role assigned, the holder of the `DEFAULT_ADMIN_ROLE` will have the ability to grant it and revoke it. The extension implements the following risk mitigations on top of `AccessControl`:

- Only one account holds the `DEFAULT_ADMIN_ROLE` since deployment until it is potentially renounced.
- Enforces a 2-step process to transfer the `DEFAULT_ADMIN_ROLE` to another account.
- Enforces a configurable delay between the two steps, with the ability to cancel before the transfer is accepted.
- The delay can be changed by scheduling.
- It is not possible to use another role to manage the `DEFAULT_ADMIN_ROLE`.

```
contract Elysium88 is
    Initializable,
    ERC721URIStorageUpgradeable,
    AccessControlUpgradeable,
    ReentrancyGuardUpgradeable
{
...
```

Additionally, the `WmbAppUpgradeable` implements the `AccessControlDefaultAdminRulesUpgradeable`. The finding is reported as a deviation from leading security practices.

Assets:

- Elysium88.sol [https://github.com/DioneProtocol/nft-bridge-backend]

Status: Fixed

Recommendations

Remediation: It is recommended to review the protocol's business rules and assumption and consider implementing the `AccessControlDefaultAdminRulesUpgradeable` instead of plain `AccessControlUpgradeable`.

Resolution: Fixed in commit `5b1bc23ec4b2e8c94863c3dfc0d0783e1ce79694`.
The `Elysium88` contract now implements `AccessControlDefaultAdminRulesUpgradeable`.

F-2024-7544 - Redundant implementation of the receive function - Info

Description: The `DioneNFTBridge` allows to receive the native tokens. However, it does not process native tokens at all. Thus, this function implementation is redundant.

```
/**
 * @dev Fallback function to receive native token.
 */
receive() external payable {}
```

Assets:

- DioneNFTBridge.sol [https://github.com/DioneProtocol/nft-bridge-backend]

Status: Fixed

Recommendations

Remediation: It is recommended to remove redundant functionality to reduce contract size and Gas consumption during the deployment.

Resolution: Fixed in commit `5b1bc23ec4b2e8c94863c3dfc0d0783e1ce79694`.
The `receive` function is now removed.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Definitions

Severities

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution.

Potential Risks

The "Potential Risks" section identifies issues that are not direct security vulnerabilities but could still affect the project’s performance, reliability, or user trust. These risks arise from design choices, architectural decisions, or operational practices that, while not immediately exploitable, may lead to problems under certain conditions. Additionally, potential risks can impact the quality of the audit itself, as they may involve external factors or components beyond the scope of the audit, leading to incomplete assessments or oversight of key areas. This section aims to provide a broader perspective on factors that could affect the project's long-term security, functionality, and the comprehensiveness of the audit findings.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details	
Repository	https://github.com/DioneProtocol/nft-bridge-backend
Commit	3f2a880
Retest Commit	5b1bc23, 67bdd20
Whitepaper	https://dione-protocol.gitbook.io/dione-whitepaper
Requirements	https://github.com/DioneProtocol/nft-bridge-backend/blob/auditing/README.md
Technical Requirements	https://github.com/DioneProtocol/nft-bridge-backend/blob/auditing/README.md

Asset	Type
DioneNFTBridge.sol [https://github.com/DioneProtocol/nft-bridge-backend]	Smart Contract
Elysium88.sol [https://github.com/DioneProtocol/nft-bridge-backend]	Smart Contract
ICustomERC1155Upgradeable.sol [https://github.com/DioneProtocol/nft-bridge-backend]	Smart Contract
ICustomERC721Upgradeable.sol [https://github.com/DioneProtocol/nft-bridge-backend]	Smart Contract

Appendix 3. Additional Valuables

Verification of System Invariants

During the audit of Dione NFT Bridge, Hacken followed its methodology by performing fuzz-testing on the project's main functions. Foundry a tool used for fuzz-testing, was employed to check how the protocol behaves under various inputs. Due to the complex and dynamic interactions within the protocol, unexpected edge cases might arise. Therefore, it was important to use fuzz-testing to ensure that several system invariants hold true in all situations.

Fuzz-testing allows the input of many random data points into the system, helping to identify issues that regular testing might miss. A specific Echidna fuzzing suite was prepared for this task, and throughout the assessment, 7 invariants were tested over 100,000 runs. This thorough testing ensured that the system works correctly even with unexpected or unusual inputs.

Invariant	Test Result	Run Count
The configDstNftSc allows to configure source and destination addresses for particular chain. No zero address can be set.	Failed	100k
The configDstBridgeSc allows to configure the destination addresses for particular chain. No zero address can be set.	Passed	100k
The setGasLimit allows to set Gas limit with the range defined by the WMB Gateway.	Passed	100k
The bridgeNFT alows to bridge any NFT to any address. The destination address cannot be 0 address.	Failed	100k
The adminRetrieveNFT can retrieve any NFT from the bridge address. The destination address cannot be 0 address.	Passed	100k
The wmbReceive allows to transfer NFT from bridge to the end user.	Passed	100k
The wmbReceive allows to mint NFT to the end user.	Passed	100k

Additional Recommendations

The smart contracts in the scope of this audit could benefit from the introduction of automatic emergency actions for critical activities, such as unauthorized operations like ownership changes or proxy upgrades, as well as unexpected fund manipulations, including large withdrawals or minting events. Adding such mechanisms would enable the protocol to react automatically to unusual activity, ensuring that the contract remains secure and functions as intended.

To improve functionality, these emergency actions could be designed to trigger under specific conditions, such as:

- Detecting changes to ownership or critical permissions.
- Monitoring large or unexpected transactions and minting events.

- Pausing operations when irregularities are identified.

These enhancements would provide an added layer of security, making the contract more robust and better equipped to handle unexpected situations while maintaining smooth operations.