



ONYX LABS

# Smart Contract Audit

Prepared For:  
Dream Domain

Date:  
11/12/22



# Table of Contents

<b>Table of Contents</b>	<b>2</b>
<b>Summary</b>	<b>4</b>
<b>Overview</b>	<b>5</b>
<b>Findings</b>	<b>7</b>
BETA-001   Move to ERC721A	8
BETA-002   Inefficient Whitelisting	9
BETA-003   Restrict setMaxSupply	10
BETA-004   Redundant Logic in Stake Function	11
BETA-005   Unnecessary/Unused Function	12
BETA-006   Improper function declaration	13
BETA-007   Typo in Constructor	14
BETA-008   Inconsistent Code Styling	15
<b>Disclaimer</b>	<b>15</b>
<b>About</b>	<b>16</b>




## Summary

This report was prepared to summarize the findings of the audit performed for Dream Domain of their ERC-721 minting contract. The primary objective of the audit was to identify issues and vulnerabilities in the source code. This audit was performed utilizing Manual Review techniques and Static Analysis.

The audit consisted of:

- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Static Analysis utilizing Slither Static Analyzer.
- Unit testing specific functions.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Line-by-line manual review of the entire codebase by auditors.

The following report contains the results of the audit, and recommendations to guard against potential security threats and improve contract functionality.

A  next to a recommendation indicates that the recommendation was implemented prior to the completion of the audit.

We would like to thank the Dream Domain team for their business and cooperation; their openness and communication made this audit a success.



# Overview

## Project Summary

Project Name	Dream Domain
Description	Dream Domain is an ERC721 NFT mint, consisting of a single WL minting phase and a single public phase. Tokens can be classified as genesis.
Blockchain	Ethereum
Language	Solidity
Codebase	Beta3DDomain.sol
Commit	N/A

## Audit Summary

Delivery Date	11/12/22
Audit Methodology	Static Analysis, Manual Review, Unit Testing

## Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Resolved
Critical	0	0	0	0	0
High	2	0	2	0	0
Medium	2	0	0	2	0
Low	3	0	0	3	0
Informational	1	0	0	1	0



## Audit Scope

ID	File
BETA	Beta3DDomain.sol



## Findings

ID	Title	Category	Severity	Status
BETA-001	Move to ERC721A	Optimization	High	Declined
BETA-002	Inefficient Whitelisting	Optimization	High	Declined
BETA-003	Restrict setMaxSupply	Security	Medium	Acknowledged
BETA-004	Redundant Logic in Stake	Optimization	Medium	Acknowledged
BETA-005	Unnecessary/Unused Method	Optimization	Low	Acknowledged
BETA-006	Improper function declaration	Optimization	Low	Acknowledged
BETA-007	Typo in Constructor	Functional	Low	Acknowledged
BETA-008	Inconsistent Code Styling	Styling	Information	Acknowledged



## BETA-001 | Move to ERC721A

Category	Severity	Location	Status
Optimization	High	N/A	Declined

### Description

The initial contract was built based on the ERC721-T contract written by Squeebo. At this time, it makes sense for most ERC721 minting contracts to inherit ERC721A, developed by Churi Labs. ERC721A offers many optimization advantages, including bulk minting with reduced gas usage.

### Recommendation

Replace the inheritance of ERC721T with ERC721A and rework the necessary functions to fit.

### Action Taken

None.



## BETA-002 | Inefficient Whitelisting

Category	Severity	Location	Status
Optimization	High	N/A	Declined

### Description

The contract utilizes a method of storing whitelisted addresses in a mapping. This is highly inefficient, and requires each address to be written to storage. Switching to a more efficient/flexible method such as a Merkle tree could save thousands of dollars.

### Recommendation

Replace the whitelist mapping with a Merkle tree for efficient and flexible whitelisting.

### Action Taken

None.





## BETA-003 | Restrict setMaxSupply

Category	Severity	Location	Status
Security	Medium	N/A	Acknowledged

### Description

The function `setMaxSupply` allows the contract owner unfettered increases of the maximum supply of tokens. This could lead to tokens beyond the intended/advertised maximum supply being minted.

### Recommendation

Include logic that requires `maxSupply` to be less than `MAX_SUPPLY`.

### Action Taken

As of the completion of this report, action has not been taken.



## BETA-004 | Redundant Logic in Stake Function

Category	Severity	Location	Status
Optimization	Medium	Line 383	Acknowledged

### Description

The function stake includes redundant logic. Checks for token ownership and existence are not required, as they are already checked in ERC721 transfer functions.

### Recommendation

Remove redundant logic.

### Action Taken

As of the completion of this report, action has not been taken.



## BETA-005 | Unnecessary/Unused Function

Category	Severity	Location	Status
Optimization	Low	Line 108	Acknowledged

### Description

tokenByIndex is unused and unnecessary.

### Recommendation

Remove tokenByIndex function.

### Action Taken

As of the completion of this report, action has not been taken.



## BETA-006 | Improper function declaration

Category	Severity	Location	Status
Optimization	Low	N/A	Acknowledged

### Description

Functions that are not called both internally and externally should be declared as external (or internal if appropriate) in lieu of public. In the sample contract, the functions **withdraw** and **getStakedTokens** should be declared as external.

### Recommendation

Declare withdraw and getStakedTokens as external.

### Action Taken

As of the completion of this report, action has not been taken.



## BETA-007 | Typo in Constructor

Category	Severity	Location	Status
Security	Medium	N/A	Acknowledged

### Description

The contracts constructor contains a typo when setting the token name.

### Recommendation

Replace the incorrect string with the correct name.

### Action Taken

As of the completion of this report, action has not been taken.



## BETA-008 | Inconsistent Code Styling

Category	Severity	Location	Status
Styling	Informational	N/A	Acknowledged

### Description

The Solidity Style guide dictates requirements for code styling. These guidelines are not followed in the contract. In particular, CAPITAL names should be reserved for constant variables, and mixedCase should be used for normal variables.

### Recommendation

Review the solidity style guide and make appropriate changes.

### Action Taken

As of the completion of this report, action has not been taken.



## Disclaimer

The audit performed by Onyx Labs is intended to identify function weaknesses, potential security threats, and improve performance. It does not provide a guarantee of contract performance or the absence of any exploitable vulnerabilities. Any alterations to the source code provided could potentially invalidate the analysis provided in this audit report. Onyx Labs is not liable for any losses or damages incurred as a result of contract deployment and the proceeding occurrences.



## About

Onyx Labs is a full-service web3 development firm founded in December 2021. Their scope of expertise ranges from smart contract development for a multitude of applications, to web development and audits. By offering best-in-class service, Onyx Labs has helped many successful projects enter the web3 space, or provided assistance with their ongoing development needs.