Blaize.Security

July 22nd, 2022 / V. 1.0

VIEWPOINT LABS

SMART CONTRACT AUDIT

TABLE OF CONTENTS

AUDIT RATING

Viewpoint Labs contract's source code was taken from the repository provided by the Viewpoint Labs team.

SCORE **9.7** /10

The scope of the project is **Viewpoint Labs** set of contracts**:**

1/ TitleDeeds.sol

Repository:

https://github.com/viewpoint-labs/smart-contracts

Primary commit (audited):

- 88b34ba72b278f956120d3d306f6a2ff2f6db9ae

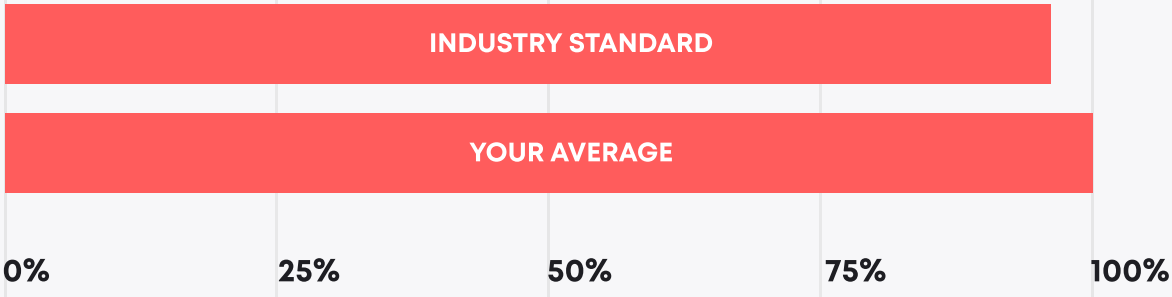
Final commit (post-audit):

- 000e757bfadaf1f01f36fc329f23168af054f5d6

TECHNICAL SUMMARY

In this report, we consider the security of the contracts for Viewpoint Labs protocol. Our task is to find and describe security issues in the smart contracts of the platform. This report presents the findings of the security audit of **Viewpoint Labs** smart contracts conducted between **Juky 13th, 2022 - July 22nd, 2022.**

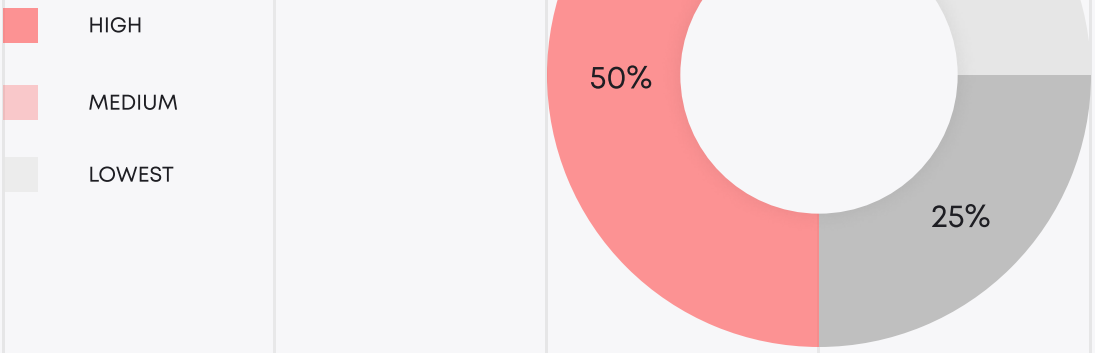
Testable code



The testable code is 100%, which above the industry standard of 95%.

The scope of the audit includes the unit test coverage, that bases on the smart contracts code, documentation and requirements presented by the Viewpoint Labs team. Coverage is calculated based on the set of Truffle framework tests and scripts from additional testing strategies. Though, in order to ensure a security of the contract Blaize.Security team recommends the Viewpoint Labs team put in place a bug bounty program to encourage further and active analysis of the smart contracts.

THE GRAPH OF VULNERABILITIES DISTRIBUTION:



The table below shows the number of found issues and their severity. A total of 4 problems were found. 0 issues were fixed or verified by the Viewpoint Labs team.

	FOUND	FIXED/VERIFIED
Critical	0	0
High	2	2
Medium	1	1
Low	0	0
Lowest	1	1

SEVERITY DEFINITION

Critical

A system contains several issues ranked as very serious and dangerous for users and the secure work of the system. Needs immediate improvements and further checking.

High

A system contains a couple of serious issues, which lead to unreliable work of the system and migh cause a huge information or financial leak. Needs immediate improvements and further checking.

Medium

A system contains issues which may lead to mediumfinancial loss or users' private information leak. Needs immediate improvements and further checking.

Low

A system contains several risks ranked as relatively small with the low impact on the users' information and financial security. Needs improvements.

Lowest

A system does not contain any issue critical to the secure work of the system, yet is relevant for best

AUDITING STRATEGY AND TECHNIQUES APPLIED \ PROCEDURE

We have scanned this smart contract for commonly known and more specific vulnerabilities:

- Unsafe type inference;
- Timestamp Dependence;
- Reentrancy;
- Implicit visibility level;
- Gas Limit and Loops;
- Transaction-Ordering Dependence;
- Unchecked external call - Unchecked math;
- DoS with Block Gas Limit;
- DoS with (unexpected) Throw;
- Byte array vulnerabilities;
- Malicious libraries;
- Style guide violation;
- ERC20 API violation;
- Uninitialized state/storage/ local variables;
- Compile version not fixed.

Procedure

In our report we checked the contract with the following parameters:

- Whether the contract is secure;
- Whether the contract corresponds to the documentation;
- Whether the contract meets best practices in efficient use of gas, code readability;

Automated analysis:

Scanning contract by several public available automated analysis tools such as Mythril, Solhint, Slither and Smartdec. Manual verification of all the issues found with tools.

Manual audit:

Manual analysis of smart contracts for security vulnerabilities. Checking smart contract logic and comparing it with the one described in the documentation.

EXECUTIVE SUMMARY

Audited protocol represents an extended ERC1155 with private and public minting staged, during which users of the protocol are able to buy limited NFTs for Ether. Users than are would be able to redeem their NFTs later. It was verified during audit, that only users can initiate redemption of their tokens.

There were no critical issues found during audit. Nevertheless, all found issues were successfully resolved or verified by Viewpoint Labs team.

The overall security is high enough and the code has a good readability and documentation, which describes the protocol.

	RATING
Security	9.9
Gas usage and logic optimization	9.5
Code quality	9.5
Test coverage**	10
Total	9.7

**Contract has a native coverage, prepared by Viewpoint Labs team, though, Blaize Security has prepared their own set of unit tests to cover the whole code.

PROTOCOL OVERVIEW

TITLEDEEDS

ONLY OWNER ACTIONS

Owner

can withdraw ether from contract

.withdraw()

Owner

can allow and prohibit minting for users

.setAllowlisted()

Owner

can start and end minting for users

.startPrivateMint()
.startPublicMint()
.endMint()

Owner

can mint to user

.mintTo()
.mintBatchTo()

Owner

can set base URI for contract

.setBaseURI()

ALL USERS ACTIONS

User

can read:

supported ERC interfaceIds
.supportsInterface()

URI for concrete tokenId
.uri()

allow for a concrete wallet
.isAllowlisted()

minted count for concrete tokenId
.getMintedCount()

max supply for concrete tokenId
.getMaxSupplyOf()

User

can mint when it allowed and he can mint

.mint() payable

User

can can redeem his own NFTs through additional redeemer contract

.redeemFrom()

DESCRIPTION

TitleDeeds

contract for grant users with title deeds to be exchanged on parcel NFT later on.
Fungible or non-fungible (ERC1155), royalties (ERC2981) token.

Fungible - Common, Rate, Epic, Legendary
Non-fungible - District 1-5, Rockŝ House

Exchange mechanism: burn title deed token after exchange on parce

JSON files will be stored on our servers.

Token re-sale commission: 2% goes to the owner wallet

COMPLETE ANALYSIS

HIGH	✔ Verified

Owner can mint to not allowed users.

TitleDeeds.sol: function .mintBatchFor(), .mintFor().
Owner can mint to not allowed users calling the .mintBatchFor() or .mintFor() function. It is recommended to prevent this possibility. To do this, you can use the existing whenMintAllowed() modifier.

Recommendation:

Add whenMintAllowed() modifier to .mintBatchFor() and .mintFor() functions or verify that the owner is able to mint to any users.

Post audit:

Client verified that owner should be able to mint tokens to not allowed users.

HIGH	✔ Resolved

Deprecated ETH transfer.

TitleDeeds.sol: function withdraw(), line 210.
Due to the Istanbul update there were several changes provided to the EVM, which made .transfer() and .send() methods deprecated for the ETH transfer. Thus it is highly recommended to use .call() functionality with mandatory result check, or the built-in functionality of the Address contract from OpenZeppelin library.

Recommendation:

Correct ETH sending functionality.

MEDIUM ✔ Resolved

Extra ETH is not transferred back to users.

TitleDeeds.sol: function mint().
It is allowed for users to send more ETH than required for purchasing NFTs and extra ETH is not returned back to the user.

Recommendation:

Return extra ETH back to the user if there is any. Be careful and consider the reentrancy vulnerability.

Post audit:

A restriction was added, that an exact amount of ETH should be sent.

LOWEST ✔ Resolved

Redeemer is able to burn tokens from any address without approval.

TitleDeeds.sol: function redeemFrom().
Redeemer is able to burn tokens from any address without approval. Issue is marked as info, since only a trusted redeemer, set by owner can burn, however the redeem mechanism doesn't require user's approval for burning his NFTs and grants redeemer a direct access to users' tokens.

Recommendation:

Add a validation that the user approved his NFTs to redeemer to be burnt or verify that this is an intended functionality.

Post audit:

A modifier was added, which checks that only from address can initiate the redemption.

	TitleDeeds.sol
✔ Re-entrancy	Pass
✔ Access Management Hierarchy	Pass
✔ Arithmetic Over/Under Flows	Pass
✔ Delegatecall Unexpected Ether	Pass
✔ Default Public Visibility	Pass
✔ Hidden Malicious Code	Pass
✔ Entropy Illusion (Lack of Randomness)	Pass
✔ External Contract Referencing	Pass
✔ Short Address/ Parameter Attack	Pass
✔ Unchecked CALL Return Values	Pass
✔ Race Conditions / Front Running	Pass
✔ General Denial Of Service (DOS)	Pass
✔ Uninitialized Storage Pointers	Pass
✔ Floating Points and Precision	Pass
✔ Tx.Origin Authentication	Pass
✔ Signatures Replay	Pass
✔ Pool Asset Security (backdoors in the underlying ERC-20)	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES, PREPARED BY VIEWPOINT LABS TEAM

Contract: CountersPlus

- ✓ starts at zero

incBy

- ✓ increase the counter by 1
- ✓ increase the counter by 5
- ✓ increase the counter by amount multiple times (61ms)
- ✓ reverts on overflow (88ms)

decBy

- ✓ decrease the counter by 1 (71ms)
- ✓ decrease the counter by 5
- ✓ decrease the counter by amount multiple times (50ms)
- ✓ reverts on underflow

Contract: ERC1155Redeemable

- ✓ should have the owner as redeemer by default
- ✓ should be able to set the redeemer
- ✓ should be able to set the redeemer to 0x0
- ✓ should revert if redeeming by non-redeemer (48ms)
- ✓ should revert if trying redeem more than balance

Contract: TitleDeed Exchanger

- ✓ should deploy

Configuration

- ✓ should have a title deeds contract
- ✓ should have a parcel nft contract

exchange

- ✓ should exchange a title deed for a parcel nft (63ms)

TEST COVERAGE RESULTS

VIEWPOINT LABS TEAM

FILE	% STMTS	% BRANCH	% FUNCS
TitleDeed.sol	62.3	22.73	40.91
All files	**62.3**	**22.73**	**40.91**

CODE COVERAGE AND TEST RESULTS FOR ALL FILES, PREPARED BY BLAIZE SECURITY TEAM

Contract: TitleDeeds

OnlyOwner functions

- ✓ Owner should set base uri (55ms)
- ✓ Owner should give allowance
- ✓ Owner should take away allowance
- ✓ Owner should start private minting
- ✓ Owner should start public minting
- ✓ Owner should end minting
- ✓ Owner should mint to user
- ✓ Owner should not mint to zero address (95ms)
- ✓ Owner should mint to user some tokens (58ms)
- ✓ Owner should not mint some tokens with one amount
- ✓ Owner should not mint some tokens to zero address
- ✓ Owner should withdraw Ether from contract (50ms)

Users functions

- ✓ User should get supported interface
- ✓ User should get uri
- ✓ User should not get uri for non-existing token
- ✓ User should get his allowance
- ✓ User should get minted count of token
- ✓ User should get max supply of token
- ✓ User should mint if he sends Ether (39ms)
- ✓ User should not mint if mint is not active
- ✓ User should not mint if mint is ended (42ms)
- ✓ User should not mint if he is nott allowed
- ✓ User should not mint if amount == 0
- ✓ User should not mint if he is trying to mint not enough tokens left
- ✓ User should not mint if he sent not enough eth
- ✓ User should redeem token if he is redeemer (44ms)
- ✓ User should not redeem token if he is not redeemer
- ✓ User should not mint if he sends more then should Ether

28 passing (1s)

TEST COVERAGE RESULTS

BLAIZE SECURITY TEAM

FILE	% STMTS	% BRANCH	% FUNCS
TitleDeed.sol	100	100	100
All files	**100**	**100**	**100**

DISCLAIMER

The information presented in this report is an intellectual property of the customer including all presented documentation, code databases, labels, titles, ways of usage as well as the information about potential vulnerabilities and methods of their exploitation. This audit report does not give any warranties on the absolute security of the code. Blaize.Security is not responsible for how you use this product and does not constitute any investment advice.

Blaize.Security does not provide any warranty that the working product will be compatible with any software, system, protocol or service and operate without interruption. We do not claim the investigated product is able to meet your or anyone else requirements and be fully secure, complete, accurate and free of any errors and code inconsistency.

We are not responsible for all subsequent changes, deletions and relocations of the code within the contracts that are the subjects of this report.

You should perceive Blaize.Security as a tool which helps to investigate and detect the weaknesses and vulnerable parts that may accelerate the technology improvements and faster error elimination.