SecureDapp

DApp Developers and Smart Contract Auditors

SMART CONTRACT SECURITY AUDIT

of

BLITS ESTATES CONTRACTS

Smart Contract Audit of Blits Estates

July 20th, 2023 | v. 1.0

TABLE OF CONTENT

AUDIT INTRODUCTION

Auditing Firm	SecureDApp Auditors
Audit Architecture	SecureDApp Auditing Standard
Language	Solidity
Client Firm	Blits Estates
Website	Blits Estates
Twitter	https://twitter.com/BlitsEstates
Linkedin	https://www.linkedin.com/company/blits-estates/
Report Date	July 20th, 2023

About Blits Estates

Blits Estates DAO platform allows investors to invest in a tangible real estate asset and gives them better ROI year-on-year(YoY). Investors will be able to own a percentage of property with all legal compliances handled within a click of a button.

AUDIT DOCUMENT

Name	Smart Contract Code Review and Security Analysis Report for Blits Estates
Approved By	Himanshu Gautam| CTO at SecureDApp
Type	Real Estates Tokenized DAO
Platform	EVM
Language	Solidity
Changelog	20.07.2023 – Initial Review

AUDIT SCOPE

The scope of this report is to audit the smart contract source code of Blits Estates.
Our client provided us with three smart contracts.

- BlitsToken.sol
- FactoryCloneContract.sol
- Shares.sol

All the contracts were written in Solidity and based on the OpenZeppelin library.
Smart contracts were to be deployed to the Polygon network. BlitsToken contract was for deploying Blits Token with anti bot mechanism (Transfer Fee) based on ERC-20 standard. Blits token will be used for purchasing shares of properties tokenized on the platform. FactoryCloneContract will be used to deploy tokenized property shares contract on chain by the contract owner. Shares contract is used to store property info, its share's price and functionality for investors to purchase property shares.

After initial research, we agreed to perform the following tests and analyses as part of our well-rounded audit:

- Smart contract behavioral consistency analysis
- Test coverage analysis
- Penetration testing: checking against our database of vulnerabilities and simulating manual attacks against the contracts
- Static analysis
- Manual code review and evaluation of code quality
- Analysis of GAS usage
- Contract analysis with regards to the host network

Initial Review Scope

Repository	https://github.com/securedapp-github/Blits_Estates_Audit
Commit	87c6b75f0b0d5f9cf5fa66586c0f86e4ed3911ad
Functional Requirements	Partial documentation provided. README.md
Technical Requirements	Partial documentation provided. README.md
Contracts Addresses	Blits Token: 0xB2d4Cc77f233e1e9442EBd11E22D9A286f2D35C3 Factory Contract: 0xeb531309Be832a407CbFbF08A70F602123f5b099 Factory Implementation: 0x305d1c5eeceacbcb65945255a0d7e85fc2dd5fdb
Contracts	File: ./contracts/BlitsToken.sol SHA3: aead98c060fac03f407008573b1720cc2783f5ff6d0355918ed3bb4db0ed60c3 File: ./contracts/FactoryCloneContract.sol SHA3: c51c85c7d583a45c75793f9dcc06bbf2dc8936652c9f3540671bc8566e2c12e8 File: ./contracts/Shares.sol SHA3: c51c85c7d583a45c75793f9dcc06bbf2dc8936652c9f3540671bc8566e2c12e8

Final Review Scope

Repository	https://github.com/securedapp-github/Blits_Estates_Audit
Commit	029eb7b97b3631fbd832659fa811dae633a52d30
Functional Requirements	Partial documentation provided. README.md
Technical Requirements	Partial documentation provided. README.md
Contracts Addresses	Blits Token: 0xB2d4Cc77f233e1e9442EBd11E22D9A286f2D35C3 Factory Contract: 0xd5151881c97c53de667214560b3232b8f47665d6
Contracts	File: ./contracts/BlitsToken.sol SHA3: d931b12ad0231f6cc64214bed8cfa48a309a38fc91ec04e43d76c7551b7cdcdb File: ./contracts/FactoryCloneContract.sol SHA3: e6cc8c832fc6f0278e1a792d47dbd6f368e482b0ae54ec9b0bf8ac080133a5ab File: ./contracts/Shares.sol SHA3: dd5b7ba922038f3f0c007fb49294cf2438fad624e6850e7159a24427148328bd

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to asset loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are important to fix; however, they cannot lead to asset loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution.
Informational	Issue listed to improve understanding, readability and quality of code

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.

AUDIT SUMMARY

The SecureDApp team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

Status	Critical	High	Medium	Low	Informative
Open	0	0	0	0	0
Acknowledged	2	0	0	1	0
Resolved	1	1	0	1	1

AUDIT METHODOLOGY

SecureDApp scans contracts and reviews codes for common vulnerabilities, exploits, hacks and back- doors. Mentioned are the steps used by SecureDApp to audit smart contracts:

a. Smart contract source code reviewal:
    i. Review of the specifications, sources, and instructions provided to SecureDApp to make sure we understand the audit scope, intended business behavior, overall architecture, and project's goal.
    ii. Manual review of code, which is the process of reading source code line-by-line to identify potential vulnerabilities.
b. Test coverage analysis: (Unit testing)
    i. Test coverage analysis is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
c. Static analysis:
    i. Run a suite of vulnerability detectors to find security concerns in smart contracts with different impact levels.
d. Symbolically executed tests: (SMTChecker testing) (Taint analysis)
    i. Symbolic execution is analyzing a program to determine what inputs cause each part of a program to execute.
    ii. Check for security vulnerabilities using static and dynamic analysis
e. Property based analysis (Fuzz tests)(Invariant testing)
    i. Run the execution flow multiple times by generating random sequences of calls to the contract.
    ii. Asserts that all the invariants hold true for all scenarios.
f. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
g. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Automated 5S frameworks used to assess the smart contract vulnerabilities

- Consensys Tools
- SWC Registry
- Solidity Coverage
- Open Zeppelin Code Analyzer
- Solidity Code Compiler

We have audited the smart contracts for commonly known and more specific vulnerabilities.
Below is the list of smart contract tests, vulnerabilities, exploits, and hacks:

ID	Description	Status
EEA 3.3	Oracle Manipulation	N/A
EEA 3.3	Bad Randomness - VRF	N/A
S60	Assembly Usage	Passed
S59	Dangerous usage of block.timestamp	Passed
EEA 3.7	Front-Running Attacks	N/A
EEA 3.7	Back-Running Attacks	N/A
EEA 3.7	Sandwich Attacks	N/A
DASP	Gas Griefing Attacks	Passed
DASP	Force Feeding	Passed
SCSVS V2	Access Control	Passed
DASP	Short Address Attack	Passed
DASP	Checks Effects Interactions	Passed
EEA 4.1	No Self-destruct	Passed
SCSVS V14	Decentralized Finance Checks	Passed

Slither Tests	Checks for ERC's conformance	Passed
Coverage	Unit tests with 100% coverage	Passed
Gas Reporter	Gas usage & limitations	Passed
Echidna Tests	Malicious input handling	Passed
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	Floating Pragma	Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELF-DESTRUCT Instruction	Passed
SWC-107	Re-entrancy	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed

SWC-113	DoS with Failed Call	**Passed**
SWC-114	Transaction Order Dependence	**Passed**
SWC-115	Authorization through tx.origin	**Passed**
SWC-116	Block values as a proxy for time	**Passed**
SWC-117	Signature Malleability	**Passed**
SWC-134	Message call with the hardcoded gas amount	**Passed**
SWC-135	Code With No Effects (Irrelevant/Dead Code)	**Informational**
SWC-136/SCSVS V3	Unencrypted Private Data On-Chain	**Passed**

SYSTEM OVERVIEW

BlitsEstate provides investors with a unique opportunity to digitally invest in Real Estate around the world via Fiat. Platform selects the best properties and independently verifies all properties showcased on the marketplace. System architect uses three core contracts. BlitsEstate Token contract is a ERC-20 Contract with Anti-Bot mechanism. It allows investors to buy shares of properties listed on BlitsEstate marketplace.

BlitsEstate FactoryClone contract allows its admin to launch new property shares onchain and list it in the marketplace. Share contract stores all relevant info about the property. The scope of the audit is BlitsToken.sol, FactoryCloneContract.sol and Shares.sol

Privileged roles

1. BlitsToken - DEFAULT_ADMIN_ROLE:
    a. Manage minter role and transfer fee
2. BlitsToken - MINTER_ROLE:
    a. Mint and Burn Blits tokens
3. Factory - OWNER:
    a. Launch Properties contract and list them on marketplace
    b. Assign property shares numbers and value in Blits tokens

Risk

1. The impact of the admin role being compromised would have a huge impact on the protocol.
2. Centralization risk is the most common cause of cryptography asset loss
3. Compromising the DEFAULT_ADMIN_ROLE may lead to all user's asset loss.

FINDINGS

Centralization Risk

Centralization risk is the most common cause of dapp's hacks. When a smart contract has an active contract ownership, the risk related to centralization is elevated. There are some well-intended reasons to be an active contract owner, such as:

- Contract owners can be granted the power to pause() or lock() the contract in case of an external attack.
- Contract owners can use functions like, include(), and exclude() to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale, and to list on an exchange.

Authorizing a full centralized power to a single body can be dangerous. Unfortunately, centralization related risks are higher than common smart contract vulnerabilities. Centralization of ownership creates a risk of rug pull scams, where owners cash out tokens in such quantities that they become valueless. Most important question to ask here is, how to mitigate centralization risk? Here's SecureDApp's recommendation to lower the risks related to centralization hacks:

- Smart contract owner's private key must be carefully secured to avoid any potential hack.
- Smart contract ownership should be shared by multi-signature (multi-sig) wallets.
- Smart contract ownership can be locked in a contract, user voting, or community DAO can be introduced to unlock the ownership.

BlitsEstate's Centralization Status

- BlitsEstate's smart contract has DEFAULT_ADMIN_ROLE, MINTER_ADMIN roles and contract Ownership.

STATIC ANALYSIS REPORT

| Symbol | Meaning |

|:--------:|----------|

| 🛑 | Function can modify state |

| 💵 | Function is payable |

| Contract | Type | Bases |

| **FactoryCloneContract** | Implementation | ||

| **BlitsToken** | Implementation | ERC20, ERC20 Burnable, AccessControl |||

| **Shares** | Implementation | Initializable, ERC20Upgradeable |||

Function Name | **Visibility** | **Mutability** | **Modifiers** |

| **FactoryCloneContract** | Implementation | ||

| └ | <Constructor> | Public ❗ | 🛑 | NO ❗ |

| └ | changeOrganiser | External ❗ | 🛑 | isOrganiser |

| └ | changeBlitsToken | External ❗ | 🛑 | isOrganiser |

| └ | createDAO | External ❗ | 🛑 | isOrganiser |

||||||

| **BlitsToken** | Implementation | ERC20, ERC20 Burnable, AccessControl |||

| └ | <Constructor> | Public ❗ | 🛑 | ERC20 |

| └ | mint | Public ❗ | 🛑 | onlyRole |

| └ | changeFeeStatus | External ❗ | 🛑 | onlyRole |

| └ | transfer | Public ❗ | 🛑 |NO ❗ |

| └ | transferFrom | Public ❗ | 🛑 |NO ❗ |

||||||

| **Shares** | Implementation | Initializable, ERC20Upgradeable |||

| └ | initialize | Public ❗ | 🛑 | initializer |

| └ | changeMaxSupply | Public ❗ | 🛑 | isOwner |

| └ | changeOwner | Public ❗ | 🛑 | isOwner |

| └ | changeBlits | Public ❗ | 🛑 | isOwner |

| └ | changePrice | Public ❗ | 🛑 | isOwner |

| └ | _afterTokenTransfer | Internal 🔒 | 🛑 | |

| └ | _mint | Internal 🔒 | 🛑 | |

| └ | _burn | Internal 🔒 | 🛑 | |

| └ | buyShares | External ❗ | 🛑 |NO ❗ |

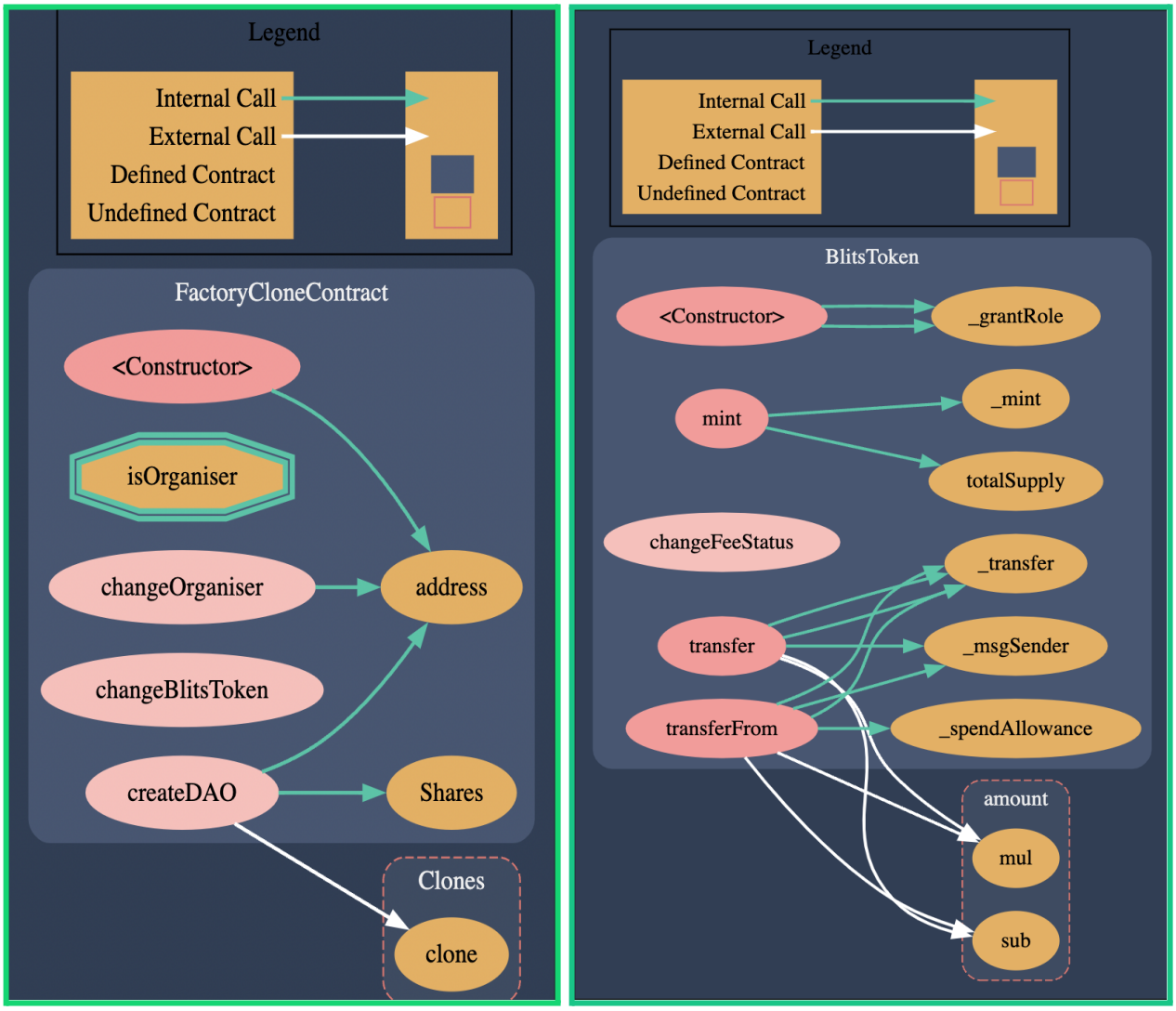
| └ | burnToken | External ❗ | 🛑 | isOwner |

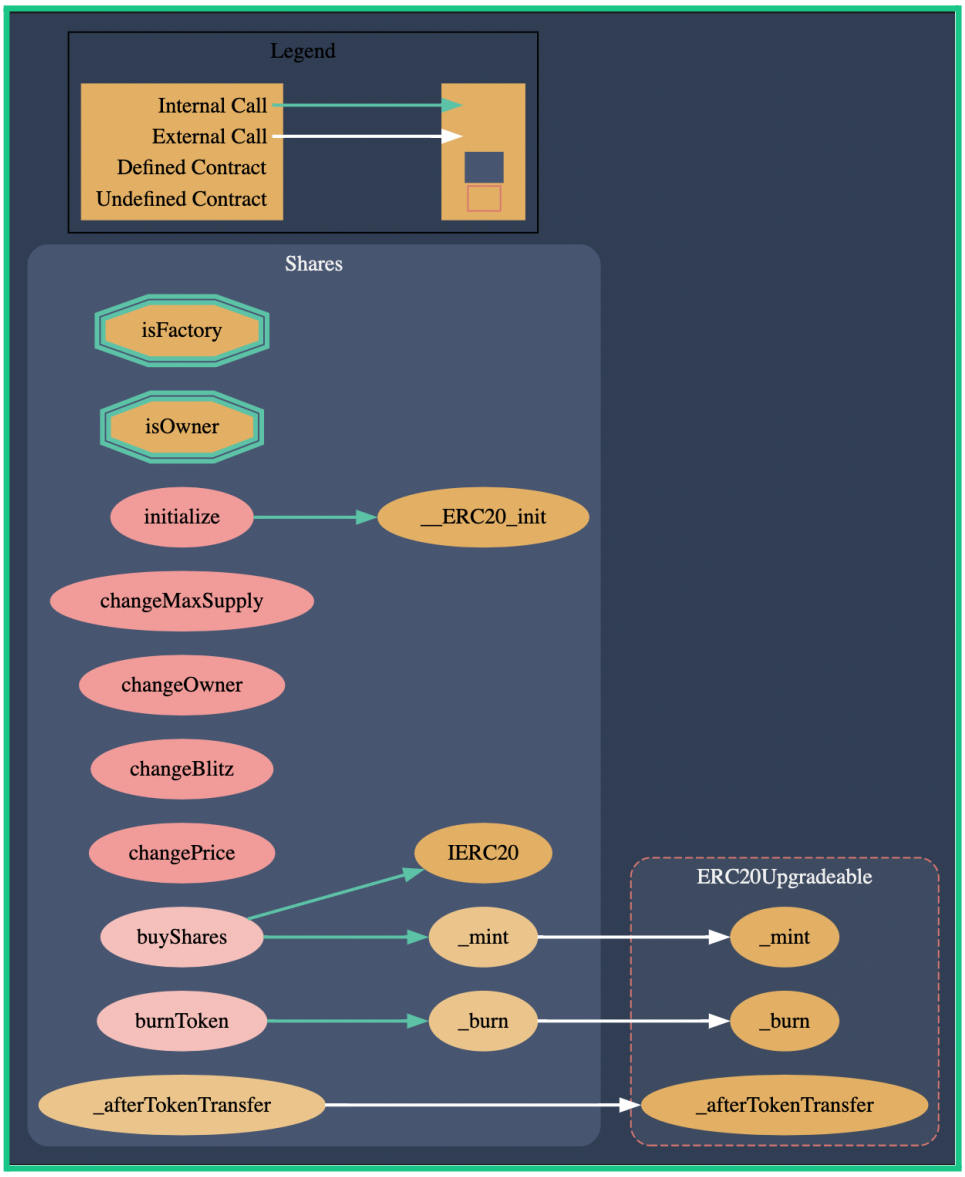
||||||

TRANSACTION GAS CHART

Solc version: 0.8.7	Optimizer enabled: false	Runs: 200	Block limit: 30000000 gas		
Methods					
Contract	**Method**	Min	Max	Avg	**# calls**	**usd (avg)**
BlitsToken	approve	–	–	46939	2	–
BlitsToken	changeFeeStatus	–	–	68751	1	–
BlitsToken	mint	54985	72073	63529	4	–
BlitsToken	transfer	–	–	79749	1	–
FactoryCloneContract	createDAO	–	–	350526	1	–
Shares	burnToken	–	–	39575	1	–
Shares	buyShares	95921	130121	113021	2	–
Shares	changePrice	–	–	31534	1	–
Deployments					**% of limit**	
BlitsToken		–	–	2497625	8.3 %	–
FactoryCloneContract		–	–	3068158	10.2 %	–

INHERITANCE CALL GRAPHS




Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Shares
isFactory
isOwner
initialize
__ERC20_init
changeMaxSupply
changeOwner
changeBlitz
changePrice
IERC20
buyShares
_mint
burnToken
_burn
_afterTokenTransfer
ERC20Upgradeable
_mint
_burn
_afterTokenTransfer

MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralization privileges of BlitsEstate Admins	Critical

Centralized privileges are listed below:

- BlitsToken - DEFAULT_ADMIN_ROLE:
  - Manage minter role and transfer fee
- BlitsToken - MINTER_ROLE:
  - Mint and Burn Blits tokens
- Factory - OWNER:
  - Launch Properties contract and list them on marketplace
  - Assign property shares numbers and value in Blits tokens

RECOMMENDATION

Access control privileges must be authenticated and their private keys should be secured carefully. Usage of Multi-Sig wallet for authorisation is recommended. Please refer to CENTRALIZED PRIVILEGES for a detailed understanding.

Status: Acknowledged

All wallet's sensitive data is handled securely by the team.

Identifier	Definition	Severity
CEN-02	Unchecked-Transfer	Critical

Shares contract function buyShares() ignores return value by Blits transfers. It allows users to purchase property shares without transferring an accurate amount of Blits token.

RECOMMENDATION

Verify the status of Blits token transfer before allotting property shares to investors.

Status: Acknowledged

Transfer status is verified on Blits Token contract

Identifier	Definition	Severity
CEN-03	Property Shares Max Supply not enforced	Critical

Max Supply of Property shares not enforced in contract so Investors can purchase property shares beyond the max supply limit.

RECOMMENDATION

Enforce max supply while minting new property shares, also keep shares without decimal places

Status: Resolved

Added max supply check on Share Contract

Identifier	Definition	Severity
HGH-01	Immutable States	High

Blits Token contract lacks option to change Fee wallet.
Factory Contract lacks the option to change Shares Contract Implementation address.

RECOMMENDATION

Provide an option to the admin to change Fee wallet and Shares implementation contract address to avoid funds being lost due to wallet compromise.

Status: Resolved

Added option to change fee wallet in Blits Token Contract

Identifier	Definition	Severity
LOW-01	Different pragma directives are used	Low

Different versions of Solidity are used:

- Version used: ['^0.8.0', '^0.8.1', '^0.8.2', '^0.8.4', '^0.8.7']
- ^0.8.4 (BlitsToken.sol)
- ^0.8.7 (FactoryCloneContract.sol)
- ^0.8.7 (Shares.sol)

RECOMMENDATION

Use one Solidity version preferably (0.8.7)

Status: Resolved

Updated all version to 0.8.7

Identifier	Definition	Severity
LOW-02	Missing events access control	Low

Detect missing events for critical access control parameters

- FactoryCloneContract - changeOrganiser()
- Shares Contract - changeOwner()

RECOMMENDATION

Emit an event for critical parameter changes.

Status: Acknowledged

Events are logged in DB to monitor changes.

Identifier	Definition	Severity
INF-01	Unused code	Informational

Imported Contract AccessControl.sol not used anywhere inside Factory Clone Contract.

RECOMMENDATION

Remove or comment out the unused code.

Status: Acknowledged

Removed unused code imports

UNIT TEST REPORT

Blits Estates Unit Test Cases
Active Address:0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Blits Address:0x5FbDB2315678afecb367f032d93F642f64180aa3
Factory address:0xDc64a140Aa3E981100a9becA4E685f962f0cF6C9
Share_address 0xb0279Db6a2F1E01fbC8483FCCef0Be2bC6299cC3

- ✔ Deployed Successfully
- ✔ Mint Blits Token
- ✔ Transfer Blits Token
- ✔ Create DAO by Admin
- ✔ Approve Blits to Shares Contract
- ✔ Buy Property Shares using Blits
- ✔ Burn Blits Token by Admin
- ✔ Change Property share price by admin
- ✔ Change Blits Transfer Fee
- ✔ Change Organiser of Factory
- ✔ Grant Roles for Blits Token
- ✔ Change Shares Max Supply

12 passing (1s)

DISCLAIMER

SecureDApp Auditors provides the easy-to-understand audit of solidity source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without SecureDApp's prior written consent.

NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. for avoidance of doubt, services, including any associated audit reports or materials, shall not be considered or relied upon as any form of financial, tax, legal, regulatory, or other advice.

TECHNICAL DISCLAIMER

All services, audit reports, smart contract audits, other materials, or any products or results of the use thereof are provided "as is" and "as available" and with all faults and defects without warranty of any kind. To the maximum extent permitted under applicable law, SecureDApp hereby disclaims all

warranties, whether expressed, implied, statutory, or otherwise with respect to services, audit report, or other materials. Without limiting the foregoing, SecureDApp specifically disclaims all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement, and all warranties arising from the course of dealing, usage, or trade practice. Without limiting the foregoing, SecureDApp makes no warranty of any kind that all services, audit reports, smart contract audits, or other materials, or any products or results of the use thereof, will meet the client's or any other individual's requirements, achieve any intended result, be compatible or work with any software, system, or other services, or be secure, accurate, complete, free of harmful code, or error-free.

TIMELINESS OF CONTENT

The content contained in this audit report is subject to change without any prior notice. SecureDApp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.

LINKS TO OTHER WEBSITES

This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than SecureDApp. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites' and social accounts' owners. You agree that SecureDApp is not responsible for the content or operation of such websites and social accounts and that SecureDApp shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.

ABOUT SECUREDAPP

SecureDApp Auditor provides intelligent blockchain solutions. SecureDapp is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. SecureDapp's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy to use.

SecureDApp is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 6+ core team members, and 10+ casual contributors. SecureDApp provides manual, static, and automatic smart contract analysis, to ensure that the project is checked against known attacks and potential vulnerabilities.

To learn more, visit : https://securedapp.in/

To view our audit portfolio, visit : github.securedapp.in

To book an audit, message : securedapp.telegram

Securedapp.in

Securedapp_Linkedin

Securedapp_Telegram