SecureDapp

DApp Developers and Smart Contract Auditors

SMART CONTRACT SECURITY AUDIT

of

SECUREDAPP TOKEN LAUNCHPAD

Smart Contract Audit of SecureDApp

May 20th, 2023 | v. 1.0

TABLE OF CONTENT

AUDIT INTRODUCTION

Auditing Firm	Secure DApp Auditors
Audit Architecture	Secure DApp Auditing Standard
Language	Solidity
Client Firm	SecureDApp
Website	https://securedapp.in/
Linkedin	https://www.linkedin.com/company/securedapp/
Report Date	May 20th, 2023

About SecureDApp

SecureDApp is Leading Ethereum blockchain software and security company.

SecureDApp is a Startup recognized by the Department For Promotion Of Industry And Internal Trade, Ministry of Commerce and Industry, Government of India. Powered by Vettedcode Technologies India Pvt. Ltd.

AUDIT DOCUMENT

Name	Smart Contract Code Review and Security Analysis Report for SecureDApp
Approved By	Himanshu Gautam| CTO and Co-Founder @SecureDApp
Type	Token Launchpad with multiple services
Language	Solidity
Changelog	20.05.2023 – Initial Review | Final Review

AUDIT SCOPE

The scope of this report is to audit the smart contract source code of SecureDApp's Token Launchpad. Our client provided us with

- SecureToken.sol
- SecurityToken.sol
- SecureSale.sol
- SecureLock.sol
- SecureFactory.sol

The contract was written in Solidity and based on the OpenZeppelin library latest updates.
The smart contracts were to be deployed on Ethereum, Binance, Polygon and Avalanche network. The purpose of the contracts is to allow SecureDApp's clients to launch their token and manage its whole life cycle from token sale with vesting, airdrop to locking mechanism with secure features like AntiBot Mechanism and minimal proxies to optimize deployment gas. 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/LaunchPad_Contracts
Commit	12AC03266C6B29257C190C9465B45898589D5B38
Functional Requirements	Full documentation provided. README.md
Technical Requirements	Full documentation provided. README.md
Contracts Addresses	https://app.gitbook.com/o/SWDk523uWOa8Zsvpdhl5/s/sL9I2rmKlh78kRer586N/

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 Secure DApp 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	0	0	0	0	0
Resolved	0	0	0	3	0

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	Passed
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

SecureDApp's Factory Contract is used to deploy a minimal proxy of Secure Token, Security Token, Token Sale and Token Lock Contracts by providing respective fees set by Admin. Secure Token and Security Token have different useful features like whitelist, Transfer fee, blacklist, forced transfers etc. Token Sale can be configured for any token as payment onchain with different sale parameters like softcap, hardcap, min and max investments.

.

Privileged roles

1. DEFAULT_ADMIN_ROLE Factory Contract
2. ADMIN_ROLE For Token, Sale and Lock Contracts
3. ORACLE_ROLE:
    a. Providing real time prices of Onchain Payment Tokens (USDC, USDT, ETH, BNB)

Risk

1. The impact of ORACLE_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.
4. Contract upgradeability allows privileged roles to change current contract implementation which negatively elevates centralization risk.

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.

SecureDApp's Centralization Status

- SecureDApp's Secure Token and Secure Sale smart contract has separate DEFAULT_ADMIN_ROLE for each proxy to handle centralization risk.

STATIC ANALYSIS REPORT

Legend

| Symbol | Meaning |

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

| 🛑 | Function can modify state |

| 💵 | Function is payable |

| Contract | Type | Bases |

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

ISecureToken | Interface | |||

| └ | initialize | External ❗ | 🛑 | NO ❗ |

||||||

ISecurityToken | Interface | |||

| └ | initialize | External ❗ | 🛑 | NO ❗ |

||||||

ISecureSale | Interface | |||

| └ | initialize | External ❗ | 🛑 | NO ❗ |

||||||

ISecureLock | Interface | |||

| └ | initialize | External ❗ | 🛑 |NO ❗ |

||||||

SecureFactory | Implementation | Initializable, AccessControlUpgradeable, UUPSUpgradeable |||

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

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

| └ | _authorizeUpgrade | Internal 🔒 | 🛑 | onlyRole |

| └ | changeImplementation | External ❗ | 🛑 | onlyRole |

| └ | setFees | External ❗ | 🛑 | onlyRole |

| └ | launchSecureToken | External ❗ | 💵 |NO ❗ |

| └ | launchSecurityToken | External ❗ | 💵 |NO ❗ |

| └ | launchSecureTokenSale | External ❗ | 💵 |NO ❗ |

| └ | launchLock | External ❗ | 💵 |NO ❗ |

| └ | GetUserLocks | External ❗ | |NO ❗ |

| └ | GetUserTokens | External ❗ | |NO ❗ |

| └ | GetUserSecurityTokens | External ❗ | |NO ❗ |

| └ | GetUserSales | External ❗ | |NO ❗ |

| └ | withdraw | External ❗ | 🛑 | onlyRole |

SecureTokenSale | Implementation | Initializable, AccessControlUpgradeable |||

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

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

| └ | modifyWhitelist | External ❗ | 🛑 | onlyRole |

| └ | changeSaleParameters | External ❗ | 🛑 | onlyRole |

| └ | buyToken | Public ❗ | 💵 |NO❗ |

| └ | ClaimLockedToken | External ❗ | 🛑 |NO❗ |

| └ | ClaimInvestments | External ❗ | 🛑 |NO❗ |

| └ | getTokenAmount | Internal 🔒 | | |

| └ | withdraw | External ❗ | 🛑 | onlyRole |

||||||

SecureLock | Implementation | OwnableUpgradeable |||

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

| └ | depositToken | Public ❗ | 🛑 | onlyOwner |

| └ | reLock | Public ❗ | 🛑 | onlyOwner |

| └ | withdraw | Public ❗ | 🛑 | onlyOwner |

|||||

SecureToken | Implementation | Initializable, ERC20Upgradeable, ERC20BurnableUpgradeable, PausableUpgradeable, AccessControlUpgradeable |||

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

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

| └ | decimals | Public ❗ | |NO ❗ |

| └ | pause | Public ❗ | 🛑 | onlyRole |

| └ | unpause | Public ❗ | 🛑 | onlyRole |

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

| └ | airDropToken | Public ❗ | 🛑 |NO ❗ |

| └ | _beforeTokenTransfer | Internal 🔒 | 🛑 | whenNotPaused |

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

| └ | transfer | Public ❗ | 🛑 |NO ❗ |

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

|||||

SecurityToken | Implementation | Initializable, ERC20Upgradeable, ERC20BurnableUpgradeable, PausableUpgradeable, AccessControlUpgradeable |||

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

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

| └ | decimals | Public ❗ | | |NO ❗ |

| └ | modifyWhitelist | External ❗ | 🛑 | onlyRole |

| └ | setKYC | Public ❗ | 🛑 | onlyRole |

| └ | setDocument | Public ❗ | 🛑 | onlyRole |

| └ | pause | Public ❗ | 🛑 | onlyRole |

| └ | unpause | Public ❗ | 🛑 | onlyRole |

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

| └ | airDropToken | Public ❗ | 🛑 |NO ❗ |

| └ | _beforeTokenTransfer | Internal 🔒 | 🛑 | whenNotPaused |

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

| └ | transfer | Public ❗ | 🛑 |NO ❗ |

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

| └ | _spendAllowance | Internal 🔒 | 🛑 | |

TRANSACTION GAS CHART

Solc version: 0.8.12	Optimizer enabled: true	Runs: 1000	Block limit: 30000000 gas		
Methods					
Contract	**Method**	**Min**	**Max**	**Avg**	**# calls**	**usd (avg)**
ERC20Upgradeable	approve	–	–	49003	1	–
SecureFactory	launchLock	–	–	193720	1	–
SecureFactory	launchSecureToken	–	–	265300	1	–
SecureFactory	launchSecureTokenSale	337314	340354	338834	2	–
SecureFactory	launchSecurityToken	–	–	291667	1	–
SecureLock	depositToken	–	–	120646	1	–
SecureLock	withdraw	–	–	53938	1	–
SecureToken	mint	45879	82371	66853	5	–
SecureTokenSale	buyToken	172896	187602	180249	2	–
USDT	approve	–	–	46279	1	–
USDT	mint	–	–	68255	1	–
Deployments					**% of limit**	
SecureFactory		–	–	2168193	7.2 %	–
SecureLock		–	–	708613	2.4 %	–
SecureToken		–	–	1859536	6.2 %	–
SecureTokenSale		–	–	1884398	6.3 %	–
SecurityToken		–	–	2287694	7.6 %	–
USDT		–	–	669131	2.2 %	–

MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralization privileges of SecureDApp Admins	Low

Centralized privileges are listed below:

SecureFactory.sol

- _authorizeUpgrade()
- changeImplementation()
- setFees()
- withdraw()

RECOMMENDATION

Deployer, contract owner, role, and access control privileges must be authenticated and their private keys should be secured carefully. Please refer to CENTRALIZED PRIVILEGES for a detailed understanding.

Status: Acknowledged

According to the project team, Multisig wallets will be used to lower centralization related risks.

Identifier	Definition	Severity
CEN-02	Use of proxy and upgradeable contracts	Low

Contract upgradeability allows privileged roles to change current contract implementation.
Valid only for Secure Factory Contract

RECOMMENDATION

Test and validate the current contract thoroughly before deployment. Future contract upgradeability negatively elevates centralization risk.

Status: ACKNOWLEDGEMENT

According to the project team, SecureDApp is currently in the development phase, hence contract upgradeability is required for future development.

Identifier	Definition	Severity
LOW-01	Timestamp manipulation via block.timestamp	Low

Be aware that the timestamp of the block can be manipulated by a miner. When the contract uses the timestamp to seed a random number, the miner can actually post a timestamp within 15 seconds of the block being validated, effectively allowing the miner to precompute an option more favorable to their chances, this is a critical exploit for contracts calculating random numbers, e.g., lottery.

RECOMMENDATION

To maintain block integrity, follow the 15 seconds rule, and scale time dependent events accordingly.

Status: Acknowledged

block.timestamp is required and kept as-is by the project team.

UNIT TEST REPORT

```
Initiating Tests
  Test Wallet Contract
Deployed USDT Contract: 0x5FbDB2315678afecb367f032d93F642f64180aa3
Deployed Security Token Contract: 0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512
Deployed Sale Contract: 0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0
Deployed Token Contract: 0xCf7Ed3AccA5a467e9e704C703E8D87F634fB0Fc9
Deployed Lock Contract: 0xDc64a140Aa3E981100a9becA4E685f962f0cF6C9
Deployed Factory Contract: 0x0165878A594ca255338adfa4d48449f69242Eb8F
    ✔ Deployments (198ms)
New Token Created =  [ '0x3B02fF1e626Ed7a8fd6eC5299e2C54e1421B626B' ]
    ✔ Create Token
New Sale Created =  [ '0xBA12646CC07ADBe43F8bD25D83FB628D29C8A762' ]
Raised =  BigNumber { value: "5000000000000000000" }
Investment Record User1 =  BigNumber { value: "5000000000000000000" }
USDT Sale Created =  [
  '0xBA12646CC07ADBe43F8bD25D83FB628D29C8A762',
  '0x7ab4C4804197531f7ed6A6bc0f0781f706ff7953'
]
Raised =  BigNumber { value: "5000000000000000000" }
Investment Record User1 =  BigNumber { value: "5000000000000000000" }
    ✔ Create Sale w native token and no vesting (85ms)
    ✔ Create sale purchase request
    ✔ Create sale 2 w ERC20 token as payment with no vesting
    ✔ Create sale 2 purchase request
New Lock Created =  [ '0xc8CB5439c767A63aca1c01862252B2F3495fDcFE' ]
    ✔ Create Lock (57ms)
    ✔ Check lock early release
    ✔ Check lock release after locked timestamp
New Security Token Created =  0xD79aE87F2c003Ec925fB7e9C11585709bfe41473
    ✔ Create Security Token
    ✔ Create sale purchase request for security token with whitelist
    ✔ Create sale 2 w ERC20 token as payment with no vesting
    ✔ Create sale 2 purchase request
  13 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 SECURE DAPP

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