Audits BY ICSA

www.icsa.website

Disclaimer

ICSA audits and reports should not be considered as a form of project's "advertisement" and does not cover any interaction and assessment from "project's contract" to
"external contracts" such as Pancakeswap or similar.

ICSA does not provide any warranty on its released reports.
We should not be used as a decision to invest into an audited project please do your own research. ICSA provides transparent reports to all its "clients" and to its "clients participants" and will not claim any guarantee of bug-free code within its Smart Contract.

Each company or project shall be liable for its own security flaws and functionalities.
ICSA presence is to analyze, audit and assess the client's smart contract's code.

Scope of Work

The main focus of this report/audit, is to document an accurate assessment of the condition of
the smart contract and whether it has any security flaws in the implementation of the contract.

Suprime team agreed and provided us with the files that needed to be tested (Through Github, EtherScan, files, etc.). ICSA will be focusing on contract issues and functionalities along with the projects claims from smart contract to their website, white paper and repository where available, which has been provided by the project.
Code is reviewed manually and with the use of software using industry best practices.

Project

Supreme is the first full-service decentralized Web3 accelerator. They are revolutionizing the Web3 landscape by empowering the most talented founders with comprehensive support and essential services for launching and developing their dream projects.

Overview

ICSA was commissioned by Suprime to perform an audit of their smart contract:

0x04cBa9CE81949c76fb8a75Ba8DBf6a489d7F0374*

Blockchain –> Ethereum

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contract Details

Token Name - Suprime

Token Ticker - SUPRIME

Token Description - Utility Token

Decimals - 18

Compiler Version - v0.8.24

LP Lock - N/A

Current Holders - 1 Address

KYCd by - ICSA*

Current Transaction Count - 1

Buy Fee - 0%

Max Supply - 1,000,000,000

Sell Fee - 0%

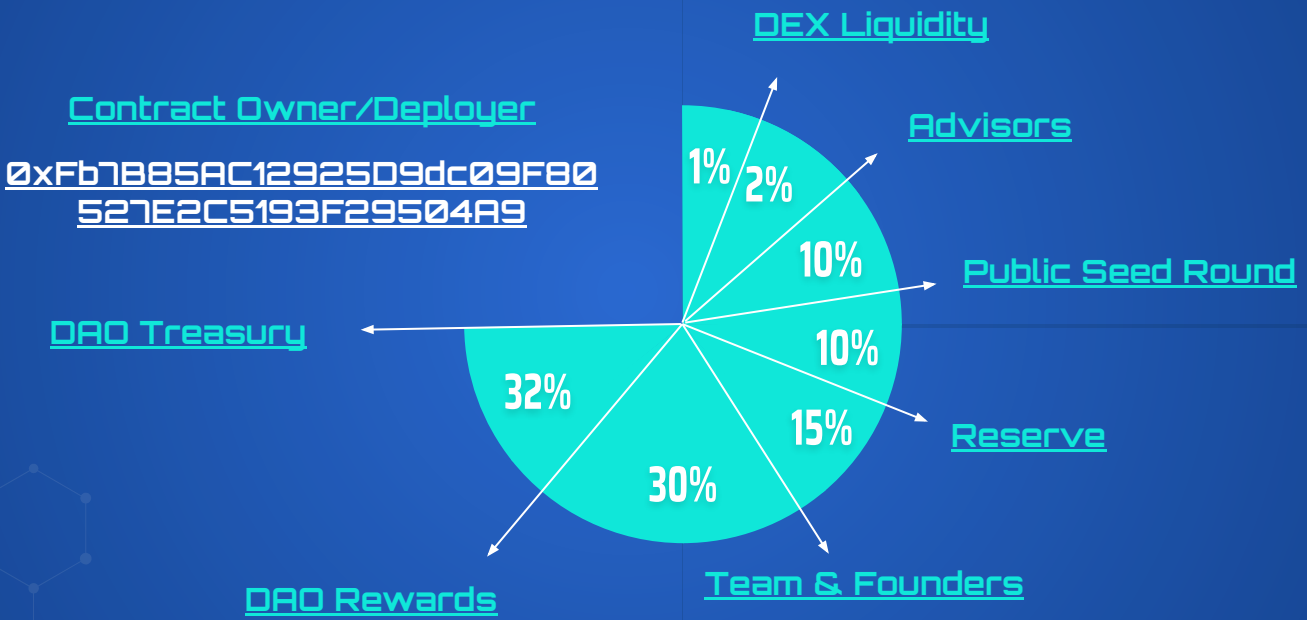
Socials

Suprime Telegram

Suprime Website

Suprime Twitter

Tokenomics


DEX Liquidity
Advisors
Public Seed Round
Reserve
Team & Founders
DAO Rewards
DAO Treasury
Contract Owner/Deployer
0xFb7B85AC12925D9dc09F80527E2C5193F29504A9
1%
2%
10%
10%
15%
30%
32%

Owner Privileges

Notes

- The owner has some privileges/authority to make SOME changes.
- Ownership HAS NOT been renounced.
- The owner can not make changes to fees.

Top 100 Holders

The total supply of 1 Billion tokens are held by the only holder.
The #1 wallet holds 100%% (1,000,000,000) tokens

Adjustable Functions

WRITE FUNCTIONS

1. Approve
2. Permit
3. Transfer
4. Transfer From

Vulnerabilities

Passed = No Issues detected. Code is in good working order

Low Issue = Low-level weakness/vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution.

High Issue = High-level weakness/vulnerabilities

SCAN RESULTS

SWC-100 –> Function Default Visibility = PASSED

SWC-101 –> Integer Overflow and Underflow = HIGH ISSUE

SWC=102 –> Outdated Compiler Version = PASSED

SWC-103 –> Floating Pragma = PASSED

SWC-104 –> Unlocked Call Return Value = PASSED

Vulnerabilities

SCAN RESULTS

SWC-105 –> Unprotected Ether Withdrawal = PASSED

SWC=106 –> Unprotected SELF DESTRUCT Instruction = PASSED

SWC-107 –> Reentrancy = 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 –> Delegatecall to Untrusted Callee = PASSED

Vulnerabilities

SCAN RESULTS

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 Value for Time = LOW ISSUE

SWC-117 –> Signature Malleability = PASSED

SWC-118 –> Incorrect Constructor Name = PASSED

SWC-119 –> Shadowing State Variables = PASSED

SWC-120 –> Weak Source of Randomness From Chain Attributes = PASSED

Vulnerabilities

SCAN RESULTS

SWC-121 –> Missing Protection Against Signature Replay Attacks = PASSED

SWC-122 –> Lack of Proper Signature Verification = PASSED

SWC-123 –> Requirement Violation = PASSED

SWC-124 –> Write to Arbitrary Storage Location = PASSED

SWC-125 –> Incorrect Inheritance Order = PASSED

SWC-126 –> Insufficient Gas Griefing = PASSED

SWC-127 –> Arbitrary Jump with Function Type Variable = PASSED

SWC=128 –> DoS with Block Gas Limit = PASSED

Vulnerabilities

SCAN RESULTS

SWC-129 –> Typographical Error = PASSED

SWC-130 –> Right-to-Left Override Control Character = PASSED

SWC-131 –> Presence of Unused Variables = PASSED

SWC-132 –> Unexpected Ether Balance = PASSED

SWC-133 –> Hash Collisions with Multiple Variable Length Arguments = PASSED

SWC-134 –> Message Call with Hardcoded Gas Amount = PASSED

SWC-135 –> Code with no effects = PASSED

SWC-136 –> Unencrypted Private Data On-Chain = PASSED

Scan Results

SWC-101

An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type. For instance if a number is stored in the uint8 type, it means that the number is stored in a 8 bits unsigned number ranging from 0 to 2^8-1. In computer programming, an integer overflow occurs when an arithmetic operation attempts to create a numeric value that is outside of the range that can be represented with a given number of bits – either larger than the maximum or lower than the minimum representable value.

SWC-116

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp`, and `block.number` can give you a sense of the current time or a time delta, however, they are not safe to use for most purposes.

In the case of `block.timestamp`, developers often attempt to use it to trigger time-dependent events. As Ethereum is decentralized, nodes can synchronize time only to some degree. Moreover, malicious miners can alter the timestamp of their blocks, especially if they can gain advantages by doing so. However, miners can't set a timestamp smaller than the previous one (otherwise the block will be rejected), nor can they set the timestamp too far ahead in the future. Taking all of the above into consideration, developers can't rely on the preciseness of the provided timestamp.

As for `block.number`, considering the block time on Ethereum is generally about 14 seconds, it's possible to predict the time delta between blocks. However, block times are not constant and are subject to change for a variety of reasons, e.g. fork reorganisations and the difficulty bomb. Due to variable block times, `block.number` should also not be relied on for precise calculations of time.

Manual Review

The manually read source code of Suprime
has revealed no issues

NOTES

The contract is feature-rich but complex, we have thorough tested and audited to ensure security and efficiency.

Functions are generally safe due to role restrictions, but misuse of admin roles could be dangerous, team has been KYC through ICSA

Overall Assessment

Un-Satisfactory!

Supime has not successfully passed the ICSA Audit!

October 10th, 2024

Closing Notes

Enhance the security of your crypto smart contracts with ICSA - the company you can trust with your digital assets. Contact us today to schedule an audit and benefit from our cutting-edge expertise in securing your blockchain projects. ICSA: Your gateway to safer, more secure smart contracts.

Whilst there are limitless ownable callable functions that have the potential to be dangerous,. Trust in the team would mitigate many of these risks. Please make sure you do your own research. If in doubt please contact the project team.

Always make sure to inspect all values and variables.
This includes, but is not limited to: · Ownership · Proper Ownership Renouncement (if any) · Taxes · Transaction/Wallet Limits · Token Distributions · Timelocks · Liquidity Locks · Any other owner-adjustable settings or variables.

Thank you for choosing ICSA