Blaize.Security

April 18th, 2022 / V. 1.0

SANR LAYER 2
SMART CONTRACT AUDIT

TABLE OF CONTENTS

AUDIT RATING

SanR Layer 2 contract's source code was taken from the repository provided by the SanR Layer 2 team.

SCORE **9** /10

The scope of the project is **SanR Layer 2** set of contracts**:**

1/ contracts/OwnableCustom.sol

2/ contracts/ERC721Custom/

3/ contracts/competitions/

4/ contracts/followings/

5/ contracts/sanTokens/

6/ contracts/signals/

Contracts repository:
https://gitlab.com/santiopts/sanr-layer-two/-/tree/side-chain

Initial commit:

- 5109bf6be1ccf6e6ce1e1341fca68a1429e0130c

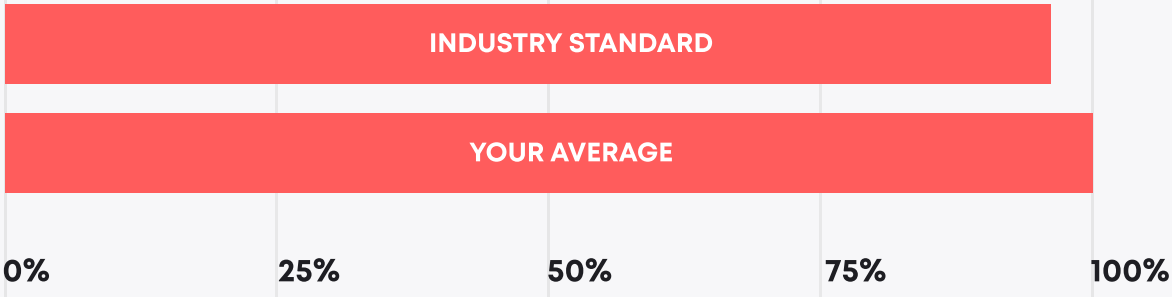
Final commit:

- 7869fd9735526cf6a486d25546bc46e0d55e0a3d

TECHNICAL SUMMARY

In this report, we consider the security of the contracts for SanR Layer 2 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 **SanR Layer 2** smart contracts conducted between **March 16th, 2022 - April 18th, 2022.**

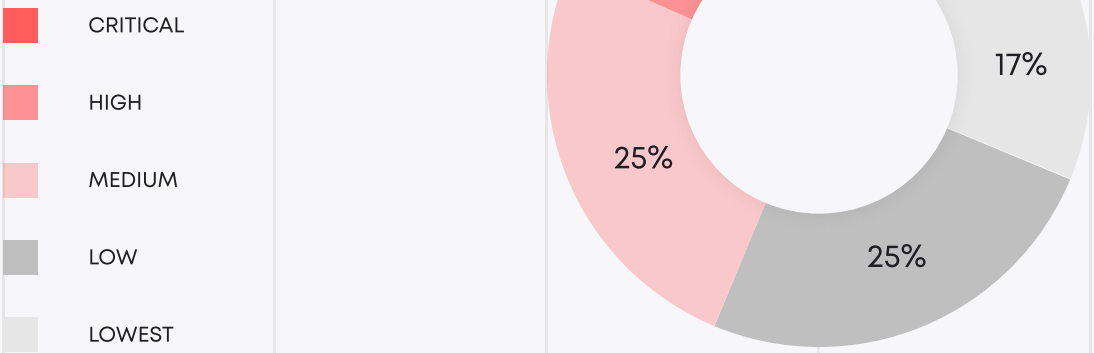
Testable code



The testable code is 97%, which is 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 SanR Layer 2 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 SanR Layer 2 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 12 problems were found. 9 issues were fixed or verified by the SanR Layer 2 team.

	FOUND	FIXED/VERIFIED
Critical	1	1
High	3	3
Medium	3	3
Low	3	2
Lowest	2	0

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 STRATEGYAND 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

The contract set implements complex system for “signals” creation, providing competitions for signals performance and alerts around profits/losses. Though, such a complex system has lack of documentation and descriptions or schemes of any kind. Thus, most of issues were connected to the unclear and unclarified functionality, which required additional comments from the Client and additional verification with tests. All issues connected to the contracts security and stability were successfully resolved by the SanR team. Nevertheless, the code does not follow Solidity best practices in full, is not documented, has commented and/or unfinished functionality (verified by the SanR team as unrelevant though).

	RATING
Security	9
Gas usage and logic optimization	10
Code quality *	5
Test coverage**	3.3
Total	9

* Code quality includes readability, following Solidity best practices, appropriate commenting and documenting, and presence of the documentation. Though most of items are purely followed in the contracts set and documentation is missing.
** Original contracts unit-test coverage is pure - 33% of coverage with the 95% standard. Nevertheless Blaize Security team has written own set of unit-tests in order to verify contracts functionality.

COMPLETE ANALYSIS

CRITICAL ✔ **Verified**

Denial of Service attack when finding winners.

Rewarding.sol, functions: findAvgDESCWinners(), findSumDESCWinners(), findSumASCWinners().
Sorted competitors are set from greatest values to lowest and can be set partially with parameter “partOfSortedCompetitors”. However, in case a competitor with a greater value is set after other competitors with lower value, this competitor would not be set due to assertions on lines 87, 89, 105, 110, 124, 129.
This issue prevents functions setAvgDESCWinners(), setSumDESCWinners(), setSumASCWinners(), since winners can be set only when all participants are set.

Recommendation:

Either restrict functions findAvgDESCWinners(), findSumDESCWinners(), findSumASCWinners(), so that the cannot be called by anyone, or sort participants in one transaction(Might revert to “out of gas” if amount of participants is too big).

Post-audit:

Verified with the client, that the ability for the sorting will disappear only for the address with initial incorrect sorting request. The issue was additionally virified during the contracts testing.

HIGH	✔ Verified

Same reward might be sent multiple times.

Rewarding.sol.
In case, not all signals were closed, but competition expired, winners can be found and rewards can be sent. However, in case additional signals are closed and after that winners are found again, other addresses can become winners and get a reward which was already sent.
Example: Competition has 1 winner and 3 participants.

1. Participant 1 and 2 have closed all their signals but participant 3 hasn't.
2. Participant 1 is set as a winner and collects a reward.
3. Participant 3 closes his open signals.
4. Winners are found again and participant 3 becomes the winner.
5. Participant 3 collects a reward which was already collected by participant

Recommendation:

Add a restriction, so that once winners are set, they cannot be set again.

Post-audit.

The issue was verified after the conversation with the client and additional round of the testing. Though during writing of test-coverage, it was discovered that it is possible to reset winners and change their sequence even after sending rewards, in case there are non-closed signals left. However rewards for specific winner places cannot be sent more than once thanks to the specific check in function sendAvgDESCRewards() that rewards were already sent.

HIGH ✔ Verified

Usage of tx.origin.

L2SanToken.sol, function _isUnlimitedAllowance().
Usage of tx.origin is considered insecure and can lead to potential funds loss. Such a vulnerability allows scummers to create malicious platforms and trick users into it. For example, a malicious contract can use this vulnerability to steal funds from a user who calls a malicious contract's method.

Recommendation:

Do not use tx.origin.

Post-audit.

The issue was verified after the conversation with the Client.
Modifier _isUnlimitedAllowance() checks if tokens are transfered from the transaction originator and that it has come though the authorized contract/account. So, the check itself verifies, that if the transaction comes from the authorized contract, tokens should be transferred from the transaction initiator.
Though, during test-coverage it was discovered that it is possible for malicious actor to use tx.origin, in case approved contract has an interface to pass "_from" parameter instead of using msg.sender, thus interface of approved contract should be verified.
Example of vulnerable interface for approved contract:

```
function transferTokens(address _from, address _to, uint256 _amount) external {
    token.transferFrom(_from, _to, _amount);
}
```

HIGH ✔ Resolved

Incorrect function used for transfer.

FollowingsL2.sol, function finalizeTokenTransferToL2().
Function finalizeTokenTransferToL2() is used to transfer NFT token from contract to “msgSender” which means that token is stored on contract’s balance. However, when the owner is calling this function, the transferFrom() function is called which checks that msg.sender(Which is the owner who originally called the function) is the owner of the token or approved. Owner of contract can’t be owner of token, since token is stored on contract’s balance and there is not function to make owner of contract approved for token. This way NFT tokens are got stuck on contract’s balance.

Recommendation:

Use function _transfer() in order to correctly transfer a token.

MEDIUM ✔ Verified

Use SafeERC20 library.

Rewarding.sol, lines 171, 180, 189
FollowingsSanStorage, lines 71, 104, 157, 158
L1SanGateway.sol: line 27
ERC20 tokens should be transferred with ‘safeTransfer’ and ‘safeTransferFrom’. SafeERC20 performs all the checks that tokens were transferred, including non-standard ERC20 implementation(like in USDT tokens).

Recommendation:

Use ‘safeTransfer’ and ‘safeTransferFrom’ instead.

Post-audit:

Verified after the conversation with the Client - contracts are not designed to work with tokens other than SAN tokens.

MEDIUM ✔ Verified

Contract can't receive ETH.

L1SanGateway.sol.
Contract has a function sendETH() which transfers ETH that has already been on contract's balance, however, there is neither function receive() nor payable fallback implemented, which mean that contract cannot receive and store ETH on its balance.

Recommendation:

Add receive() function.

Post-audit:

The contract is not able to receive ETH with direct transfers, though it is designed to get ETH from coinbase transactions. The behavior was confirmed with the Client.
https://docs.soliditylang.org/en/latest/contracts.html#receive-ether-function

MEDIUM ✔ Resolved

Value from mapping is never updated.

FollowingsCounter.sol
Value "toDate" is never updated in mapping "providerActualFollowersCount". Function parameter "toDate" is passed in function changeProviderMaxFollowingCount(), however its value is not assigned in mapping. Since "toDate" is never updated in mapping, assertions in lines 19, 30 can never fail.

Recommendation:

Update value in mapping.

LOW	✔ Verified

Add validation for time parameters.

Competitions.sol: function addCompetition().
Time parameters should be validated before setting. Parameter “endTimestamp” should be validated to be greater than “startTimestamp” and “startTimestamp” should be validated to be greater than block.timestamp.

Recommendation:

Validate time parameters.

Post-audit:

Marked as verified after the conversation with the Client. Since the method can be called only by the owner, it is agreed, that it will be extra gas usage for extra checks. Though, it is still a point to increase the decentralization, since owner is completely responsible for the correct parameters usage and avoidance of human-based errors.

LOW	Unresolved

Potential loss of accuracy.

FeeCalculator.sol: function calculateFee().
Performing a multiplication on a result of division might lead to accuracy loss.

Recommendation:

Perform division on a result of multiplication. It is the violation of "muldiv" pattern and contradiction to best practices of Solidity development. The accuracy loss is within several wei and the real exploit is hard to reproduce, so the issue is marked as low-risk . Though even such accuracy loss may lead to unexpected behavior.

LOW	✔ Resolved

XOR operation is used instead of exponentiation.

IssuingSettings.sol: function getTariffPlan().
PlatformIssuingSettings.sol: Line 16.
In lines 70, 81, an operator ^ is used which is an exclusive OR bitwise operation. The result of 2 ^ 256 - 1 would be 253 instead of maximum uint256. In case, maximum of uint256 was intended to be used, use type(uint256).max.

Recommendation:

Use type(uint256).max in order to retrieve the value of maximum uint256 or confirm that XOR operation should be used.

LOWEST	Unresolved

Commented code.

FollowingsCounter.sol: line 10
IssuingSettings.sol: line 12
L1SanGateway.sol: line 25
Pre-production contracts should not contain commented code or unfinished logic. There are several places with commented code which shows exactly missing points, or planned and not implemented, so it is important to verify, that there is no missing functionality, which can be crucial.

Recommendation:

Remove all commented code if it is not needed by the contract.

LOWEST	Unresolved

ToDo in code.

Signals.sol: line 85, 109

Pre-production contracts should not contain unfinished logic. There are several todos and commented code which shows exactly missing points, or planned and not implemented, so it is important to verify, that there is no missing functionality, which can be crucial.

Recommendation:

Either remove ToDos or finish contract's logic.

	OwnableCustom.sol	ERC721.sol
✔ Re-entrancy	Pass	Pass
✔ Access Management Hierarchy	Pass	Pass
✔ Arithmetic Over/Under Flows	Pass	Pass
✔ Delegatecall Unexpected Ether	Pass	Pass
✔ Default Public Visibility	Pass	Pass
✔ Hidden Malicious Code	Pass	Pass
✔ Entropy Illusion (Lack of Randomness)	Pass	Pass
✔ External Contract Referencing	Pass	Pass
✔ Short Address/ Parameter Attack	Pass	Pass
✔ Unchecked CALL Return Values	Pass	Pass
✔ Race Conditions / Front Running	Pass	Pass
✔ General Denial Of Service (DOS)	Pass	Pass
✔ Uninitialized Storage Pointers	Pass	Pass
✔ Floating Points and Precision	Pass	Pass
✔ Tx.Origin Authentication	Pass	Pass
✔ Signatures Replay	Pass	Pass
✔ Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	ERC721Enumerable.sol	ERC721Metadata.sol
✔ Re-entrancy	Pass	Pass
✔ Access Management Hierarchy	Pass	Pass
✔ Arithmetic Over/Under Flows	Pass	Pass
✔ Delegatecall Unexpected Ether	Pass	Pass
✔ Default Public Visibility	Pass	Pass
✔ Hidden Malicious Code	Pass	Pass
✔ Entropy Illusion (Lack of Randomness)	Pass	Pass
✔ External Contract Referencing	Pass	Pass
✔ Short Address/ Parameter Attack	Pass	Pass
✔ Unchecked CALL Return Values	Pass	Pass
✔ Race Conditions / Front Running	Pass	Pass
✔ General Denial Of Service (DOS)	Pass	Pass
✔ Uninitialized Storage Pointers	Pass	Pass
✔ Floating Points and Precision	Pass	Pass
✔ Tx.Origin Authentication	Pass	Pass
✔ Signatures Replay	Pass	Pass
✔ Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

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

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

	FollowingsCounter.sol	FollowingsL1.sol
✓ Re-entrancy	Pass	Pass
✓ Access Management Hierarchy	Pass	Pass
✓ Arithmetic Over/Under Flows	Pass	Pass
✓ Delegatecall Unexpected Ether	Pass	Pass
✓ Default Public Visibility	Pass	Pass
✓ Hidden Malicious Code	Pass	Pass
✓ Entropy Illusion (Lack of Randomness)	Pass	Pass
✓ External Contract Referencing	Pass	Pass
✓ Short Address/ Parameter Attack	Pass	Pass
✓ Unchecked CALL Return Values	Pass	Pass
✓ Race Conditions / Front Running	Pass	Pass
✓ General Denial Of Service (DOS)	Pass	Pass
✓ Uninitialized Storage Pointers	Pass	Pass
✓ Floating Points and Precision	Pass	Pass
✓ Tx.Origin Authentication	Pass	Pass
✓ Signatures Replay	Pass	Pass
✓ Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	FollowingsL2.sol	FollowingsSanStorage.sol
✔ Re-entrancy	Pass	Pass
✔ Access Management Hierarchy	Pass	Pass
✔ Arithmetic Over/Under Flows	Pass	Pass
✔ Delegatecall Unexpected Ether	Pass	Pass
✔ Default Public Visibility	Pass	Pass
✔ Hidden Malicious Code	Pass	Pass
✔ Entropy Illusion (Lack of Randomness)	Pass	Pass
✔ External Contract Referencing	Pass	Pass
✔ Short Address/ Parameter Attack	Pass	Pass
✔ Unchecked CALL Return Values	Pass	Pass
✔ Race Conditions / Front Running	Pass	Pass
✔ General Denial Of Service (DOS)	Pass	Pass
✔ Uninitialized Storage Pointers	Pass	Pass
✔ Floating Points and Precision	Pass	Pass
✔ Tx.Origin Authentication	Pass	Pass
✔ Signatures Replay	Pass	Pass
✔ Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

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

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

	L1SanGateway.sol	L2SanToken.sol
✔ Re-entrancy	Pass	Pass
✔ Access Management Hierarchy	Pass	Pass
✔ Arithmetic Over/Under Flows	Pass	Pass
✔ Delegatecall Unexpected Ether	Pass	Pass
✔ Default Public Visibility	Pass	Pass
✔ Hidden Malicious Code	Pass	Pass
✔ Entropy Illusion (Lack of Randomness)	Pass	Pass
✔ External Contract Referencing	Pass	Pass
✔ Short Address/ Parameter Attack	Pass	Pass
✔ Unchecked CALL Return Values	Pass	Pass
✔ Race Conditions / Front Running	Pass	Pass
✔ General Denial Of Service (DOS)	Pass	Pass
✔ Uninitialized Storage Pointers	Pass	Pass
✔ Floating Points and Precision	Pass	Pass
✔ Tx.Origin Authentication	Pass	Fail
✔ Signatures Replay	Pass	Pass
✔ Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

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

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

	SignalsSanStorage.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 (BY SANR TEAM)

Within the scope of the audit, Blaize Security team has checked original unit test coverage provided by SanR team, which appears to cover basic cases but is not sufficient for the security level.

Contract: Following contract

Basic

- ✓ Deployment should set correct total supply of tokens
- ✓ Deployment should set correct default tariff plan
- ✓ Follower should can create free following token (285ms)
- ✓ Signal provider should can create tariff plan (92ms)
- ✓ Follower should can create following token with correct dates (1019ms)
- ✓ Withdraw and withdraw (973ms)
- ✓ Burn and burn (326ms)
- ✓ Withdraw and burn (520ms)
- ✓ Burn and withdraw (442ms)
- ✓ Withdraw and withdraw (228ms)
- ✓ Burn and burn (126ms)
- ✓ Withdraw and burn (458ms)
- ✓ Burn and withdraw (352ms)
- ✓ Withdraw and withdraw (410ms)
- ✓ Burn and burn (166ms)
- ✓ Withdraw and burn (421ms)
- ✓ Burn and withdraw (386ms)
- ✓ Withdraw and withdraw (248ms)
- ✓ Burn and burn (129ms)
- ✓ Withdraw and burn (329ms)
- ✓ Burn and withdraw (337ms)
- ✓ Withdraw and withdraw (459ms)
- ✓ Burn and burn (207ms)
- ✓ Withdraw and burn (601ms)

- ✓ Burn and withdraw (486ms)
- ✓ Withdraw and withdraw (254ms)
- ✓ Burn and burn
- ✓ Withdraw and burn (498ms)
- ✓ Burn and withdraw (658ms)
- ✓ Withdraw and withdraw (309ms)
- ✓ Burn and burn (52ms)
- ✓ Withdraw and burn (417ms)
- ✓ Burn and withdraw (401ms)
- ✓ Withdraw and withdraw (193ms)
- ✓ Burn and burn
- ✓ Withdraw and burn (164ms)
- ✓ Burn and withdraw (248ms)
- ✓ Withdraw and withdraw (339ms)
- ✓ Burn and burn
- ✓ Withdraw and burn (351ms)
- ✓ Burn and withdraw (298ms)
- ✓ On first second (3859ms)
- ✓ On last second (4242ms)

Signals contract

- ✓ Deployment should set correct total supply of tokens
- ✓ Price operator should can set price (70ms)
- ✓ Signal provider should can create signal (149ms)

46 passing (1m)

TEST COVERAGE RESULTS

BY SANR TEAM

FILE	% STMTS	% BRANCH	% FUNCS
OwnableCustom.sol	50	25	50
ERC721.sol	33.9	14.71	34.78
ERC721Enumerable.sol	76.47	50	72.73
ERC721Metadata.sol	44.44	0	20
Competitions.sol	0	0	0
Rewarding.sol	0	0	0
FeeCalculator.sol	100	100	100
FollowingsCounter.sol	20	16.67	66.67
FollowingsL1.sol	0	100	0
FollowingsL2.sol	71.43	100	60

FollowingsSanStorage.sol	91.07	77.78	81.82
IssuingSettings.so	89.66	53.33	100
PlatformIssuingSettings.sol	7.69	100	14.29
ProviderIssuingSettings.sol	25	0	33.33
L1SanGateway.sol	0	0	0
L2SanToken.sol	63.64	33.33	80
UniswapV2ERC20Custom.sol	44.83	0	44.44
PerformanceStorage.sol	0	0	0
Prices.sol	58.82	50	71.43
SignalHashes.sol	0	100	0
Signals.sol	16	6.67	13.33
SignalsSanStorage.sol	55.56	100	66.67
All files	**36.68**	**20.56**	**34.09**

CODE COVERAGE AND TEST RESULTS FOR ALL FILES (BY BLAIZE TEAM)

Contract: FollowingsL1

- ✓ Should finalize token transfer to L1 (62ms)
- ✓ Should finalize token transfer to L1 manually (50ms)
- ✓ Should initiate token transfer to L2 (111ms)
- ✓ Should change gas limit

Contract: FollowingsL2, FollowingsSanStorage, FollowingsCounter

- ✓ Anyone should get amount which can be unlocked for the first following
- ✓ Owner should change the fee recipient (38ms)
- ✓ Owner should change an issuing setting address
- ✓ Should return locked amount for tarrif plan with id 0 (92ms)
- ✓ Should not withdraw daily reward if tarrif plan id is 0 (87ms)
- ✓ Should not withdraw daily reward if caller is not provider (104ms)
- ✓ Should change fee recipient address
- ✓ Should change issuing settings address
- ✓ Should change max following counter for provider (56ms)

Contract: Token transfer

- ✓ Follower should initiate token transfer to L1
- ✓ Owner should finalize token transfer to L2

Contract: IssuingSettings, ProviderIssuingSettings, PlatformIssuingSettings

- ✓ Someone that is not a FollowingL2 contract should not increase the count of tariff plan followers

Getting of a tariff plan

- ✓ Anyone should get a provider tariff plan for a follower
- ✓ Anyone should not get a provider tariff plan for a follower if nonexistent
- ✓ Anyone should not get a tariff plan if nonexistent
- ✓ Anyone should not get a default tariff plan if another tariff

Getting of issuing parameters

- ✓ Anyone should not get issuing parameters if a delay is greater than a maximum
- ✓ Anyone should not get issuing parameters if a duration is greater than a maximum
- ✓ Anyone should not get issuing parameters if a duration is less than a minimum

Platform setters

- ✓ Owner should change the default max following duration
- ✓ Owner should change the default min following duration
- ✓ Owner should change the default max following delay
- ✓ Owner should change the default provider daily reward
- ✓ Owner should change the default max count of followers
- ✓ Owner should change the address of fee calculator

Provider setting functionality

- ✓ Signal Provider should delete a tariff plan
- ✓ Signal Provider should not delete a tariff plan if nonexistent
- ✓ Signal Provider should change a tariff plan for a follower (62ms)

Contract: OwnableCustom

- ✓ Anyone should get an owner address
- ✓ Owner should transfer ownership

- ✓ Owner should not transfer ownership if zero address
- ✓ Someone who is not owner should not transfer ownership

Contract: Competition, Rewarding

- ✓ Should add and set competition (53ms)
- ✓ Should not add competition if end timestamp is 0
- ✓ Should not set competitions if existed competition is not set (70ms)
- ✓ Should not set competitions if competition doesn't exist (81ms)
- ✓ Should not set competitions if competition is ended and wasn't started (76ms)
- ✓ Should not set competitions if competition is competition is already ended (86ms)
- ✓ Should not set competitions if not enough reward tokens on balance (56ms)
- ✓ Should open signals in competition (141ms)
- ✓ Should close signal in competition (88ms)
- ✓ Only signal contract can call
- ✓ Should find and set avg desc winners and send rewards (264ms)
- ✓ Should not find avg desc winners if prev comp has not all signals closed and current has all (193ms)
- ✓ Should not find avg desc winners if wrong sort for last (178ms)
- ✓ Should not find avg desc winners if wrong sort (186ms)
- ✓ Should not find avg desc winners if not competitors (52ms)
- ✓ Should not find avg desc winners if not competitors repeat (233ms)
- ✓ Should not set avg desc winners if not all competitors are sorted (185ms)
- ✓ Owner should change a duration to close signals

- ✓ Should not find avg desc winners if competition is not ended (164ms)
- ✓ Should not find avg desc winners if not all signals is closed (151ms)
- ✓ Should find and set sum desc winners and send rewards (258ms)
- ✓ Should find sum desc winners when open signal count is equal to closed (236ms)
- ✓ Should not find sum desc winners if prev comp has not all signals closed and current has all (175ms)
- ✓ Should not find sum desc winners if wrong sort for last (168ms)
- ✓ Should not find sum desc winners if wrong sort (184ms)
- ✓ Should not set sum desc winners if not all competitors are sorted (185ms)
- ✓ Should find and set sum asc winners and send rewards (250ms)
- ✓ Should find sum asc winners when open signal count is equal to closed (241ms)
- ✓ Should not find sum asc winners if prev comp has not all signals closed and current has all (162ms)
- ✓ Should not find sum asc winners if wrong sort for last (170ms)
- ✓ Should not find sum asc winners if wrong sort (184ms)
- ✓ Should not set sum asc winners if not all competitors are sorted (170ms)

Contract: Signals, SignalsSanStorage, SignalHash PerformanceStorage

- ✓ Should set signal lifetime
- ✓ Should set duration to close signals
- ✓ Should set locking amount
- ✓ Should set performance storage address
- ✓ Should set competition address
- ✓ Should create new signal (84ms)

- ✓ Should close signal with current price (93ms)
- ✓ Should close signal with provided index price (97ms)
- ✓ Should unlock balance (93ms)
- ✓ Should change signal (76ms)
- ✓ Should not let not issuer change signal (70ms)
- ✓ Should not create signal if price doesn't exist (38ms)
- ✓ Should not close signal if called not by issuer or time not ended (62ms)
- ✓ Should not close signal if provided info hash is wrong (67ms)
- ✓ Should not close signal if provided changeable info hash is wrong (70ms)
- ✓ Should not unlock balance if duration is not ended (61ms)
- ✓ Should make profit negative if direction is down (82ms)
- ✓ Should not track profit if duration is ended (79ms)
- ✓ Should not close signal if timestamp of price is incorrect (64ms)
- ✓ Should not close signal if timestamp of price doesn't exist (77ms)
- ✓ Should not close signal if not stop loss or take profit closing (88ms)
- ✓ Should revert if signal is closed (85ms)
- ✓ Should set prices
- ✓ Should not set zero address as operator
- ✓ Should not let not operator set price
- ✓ Should not let set same price
- ✓ Should change performance for issuer after closing signal (107ms)
- ✓ Should set issuer start performance
- ✓ Should not set issuer start performance more than once (130ms)
- ✓ Only signals contract can change performance
- ✓ Should track signal in competition (89ms)

Contract: L1SanGateway, L1SanGateway

- ✓ Should not set zero address as approved contract
- ✓ Should subtract allowance if approval is not unlimited (47ms)
- ✓ Should get ETH
- ✓ Should not send tokens through gateway if insufficient balance
- ✓ Should send tokens through gateway (45ms)
- ✓ Should not send ETH through gateway if insufficient balance
- ✓ Should send ETH through gateway

Additional scenarios tests

- ✓ Should deny one address from finding winners if it provided wrong sequence and let another set winners (184ms)
- ✓ Should not send the same reward more than once (304ms)
- ✓ Malicious contract can use tx.origin (41ms)

107 passing (11s)
2 failing

2) Contracts: L1SanGateway, L1SanGateway
Should send ETH through gateway:
Error: Transaction reverted: function selector was not recognized and there's no fallback nor receive function
at L1SanGateway.<unrecognized-selector> (contracts/sanTokens/L1SanGateway.sol:9)
at HardhatNode._mineBlockWithPendingTxs (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1650:23)
at HardhatNode.mineBlock (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:459:16)
at EthModule._sendTransactionAndReturnHash (node_modules/hardhat/src/internal/hardhat-network/provider/modules/eth.ts:1496:18)
at HardhatNetworkProvider.request (node_modules/hardhat/src/internal/hardhat-network/provider/provider.ts:117:18)
at Web3HTTPProviderAdapter._sendJsonRpcRequest (node_modules/@nomiclabs/hardhat-web3/src/web3-provider-adapter.ts:80:22)

Test failing to the direct Eth transfer to the gateway contract. The verification is provided in the appropriate issues in the manual report.

TEST COVERAGE RESULTS

BY BLAIZE TEAM

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	Uncovered lines
contracts/	100	100	100	100	
OwnableCustom.sol	100	100	100	100	
contracts/ERC721Custom/	99.02	98	100	98.13	
ERC721.sol	98.31	97.06	100	96.72	310,316
ERC721Enumerable.sol	100	100	100	100	
ERC721Metadata.sol	100	100	100	100	
contracts/competitions/	100	100	100	100	
Competitions.sol	100	100	100	100	
Rewarding.sol	100	100	100	100	
contracts/followings/	97.87	83.93	100	97.84	

CrossDomainMock.sol	100	100	100	100	
FeeCalculator.sol	100	100	100	100	
FollowingsCounter.sol	80	33.33	100	85.71	20
FollowingsL1.sol	100	100	100	100	
FollowingsL2.sol	92.86	100	100	93.75	99
FollowingsSanStorage.sol	100	94.44	100	100	
IssuingSettings.sol	96.55	86.67	100	95.65	36
PlatformIssuingSettings.sol	100	100	100	100	
ProviderIssuingSettings.sol	100	100	100	100	
contracts/sanTokens/	84	56.25	90.48	84.91	
AuthorizedContract.sol	100	100	100	100	
L1SanGateway.sol	100	75	100	100	
L2SanToken.sol	100	83.33	100	100	
MaliciousContract.sol	100	100	100	100	
UniswapV2ERC20Custom.sol	72.41	16.67	77.78	73.33	... 90,97,98,99
contracts/signals/	100	100	100	100	
PerformanceStorage.sol	100	100	100	100	
Prices.sol	100	100	100	100	

SignalHashes.sol	100	100	100	100
Signals.sol	100	100	100	100
SignalsSanStorage.sol	100	100	100	100
All files	**97.72**	**93.15**	**98.91**	**97.61**

Also it needs to be mentioned, that SanR team has prepared custom version of ERC721 contract which was carefully checked against the standard OpeZeppelin implementation and covered with tests over the custom functionality.

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.