SMART CONTRACT AUDIT

ZOKYO.

March 7th, 2022 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

TECHNICAL SUMMARY

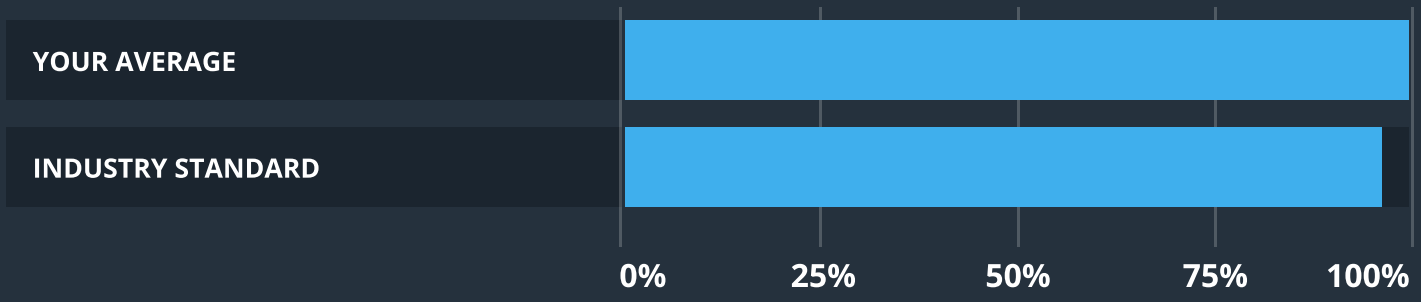
This document outlines the overall security of the SXNetwork smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the SXNetwork smart contract codebase for quality, security, and correctness.

Contract Status

There were no critical issues found during the audit.

Testable Code



The testable code is 98%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the SXNetwork team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

Auditing Strategy and Techniques Applied 3

Executive Summary 4

Structure and Organization of Document 5

Complete Analysis 6

Code Coverage and Test Results for all files 11

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the SXNetwork repository.

Repository - https://github.com/sx-network/token-contracts

Last commit - b7dd2982b502a2cdfd2466304831f59bbb4591ae

Within the scope of this audit Zokyo auditors have reviewed the following contract(s):

- ChildERC20.sol
- WSX.sol
- ContextMixin.sol
- EIP712Base.sol
- Initializable.sol
- NativeMetaTransaction.sol
- SXVault.sol

Throughout the review process, care was taken to ensure that the contract:

- Implements and adheres to existing standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of resources, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of SXNetwork smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

EXECUTIVE SUMMARY

There were no critical issues found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner.

Contracts are well written and structured. The findings during the audit have no impact on contract performance or security, so it is fully production-ready.

Despite the fact, the expected logic is managing all vestings by the owner, it should be careful with parameters to avoid mistakes during the vesting process.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

LOW | UNRESOLVED

Contract WSX have a centralization risk in the adminWithdrawal function, the admin can withdraw any amount of native tokens that are available without the burning of the underlying erc20 tokens.

Recommendation:
Implement a multi-sig mechanism for the admin address.

Team Response:
Will be implemented after initial launch

LOW | UNRESOLVED

Contract SXVault, possible to have a centralization risk in the withdraw function, the admin can withdraw any balance from the vault.

Recommendation:
Implement a multi-sign mechanism for the admin address.

Team Response:
Will be implemented after initial launch

INFORMATIONAL | RESOLVED

Contract EIP712Base contains a typo at line 46 in function name “getDomainSeperator”.

Recommendation:
Rename the function to ““getDomainSeparator”

INFORMATIONAL | RESOLVED

Contract EIP712Base contains a type at line 20 in internal variable named ""domainSeperator"".

Recommendation:
Rename the internal variable name to "domainSeparator".

INFORMATIONAL | RESOLVED

In contract WSX, at line 38 and line 54, the Deposit and Withdrawal events are emitted without using the 'emit' keywords.
The emit keywords was introduced since solidity v0.4.21- 8.3.2018, and is the recommend way of explicitly calling events to make them stand out from regular function calls.

Recommendation:
Call the events using the recommend way through the emit keyword.

INFORMATIONAL | RESOLVED

In contract WSX, function adminWithdraw uses the modifier onlyAdmin which use the global variable msg.sender, but inside the function at line 65, is used the _msgSender internal function.

Recommendation:
Use the global variable msg.sender in both places to have the same code context.

	ChildERC20.sol	WSX.sol	ContextMixin.sol
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

	EIP712Base.sol	Initializable.sol	SXVault.sol
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

	NativeMetaTransaction.sol
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	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

Tests written by Zokyo Secured team

As part of our work assisting SXNetwork in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the SXNetwork contract requirements for details about issuance amounts and how the system handles these.

Contract: **Child Erc20**
- ✓ Should be set up properly (94ms)

Contract: **NativeMetaTransaction**
- ✓ Should be able to execute meta transactions (642ms)
- ✓ Should ensure EIP712 is initialized only once (103ms)

Contract: **SX VAULT**
- ✓ Should be set up properly
- ✓ Should be able to set handler (55ms)
- ✓ Should allow users to fund contract
- ✓ Should only allow bridge exit from handler (152ms)
- ✓ Should only allow withdrawals from admin account (96ms)
- ✓ Should prevent withdrawals on failed ETH transfer (194ms)
- ✓ Should be funded via receive function (53ms)

Contract: **WSX**
- ✓ Should be set up properly
- ✓ Should take in deposits
- ✓ Should prevent withdrawals on failed ETH transfer (126ms)
- ✓ Should allow withdrawals (79ms)
- ✓ Should prevent admin withdrawals on failed ETH transfers (166ms)
- ✓ Should be able to do admin withdrawals (87ms)
- ✓ Should all deposits via receive function (51ms)

17 passing (7s)

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	Uncovered Lines
ChildERC20.sol	100	100	100	100	
WSX.sol	100	100	100	100	
ContextMixin.sol	100	100	100	100	
EIP712Base.sol	100	100	100	100	
Initializable.sol	100	100	100	100	
NativeMetaTransaction.sol	100	100	100	100	
SXVault.sol	100	100	100	100	
All files	**100**	**100**	**100**	**100**	

We are grateful to have been given the opportunity to work with the SXNetwork team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the SXNetwork team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.