SPORTX SMART CONTRACTS SECURITY AUDIT

January 19, 2021

sportX beta

MixBytes()

CONTENTS

1. INTRODUCTION

1.1 DISCLAIMER

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of SportX (name of Client). If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 PROJECT OVERVIEW

SportX is a platform that makes peer-to-peer trading on sporting and political events more fair using the Ethereum public blockchain.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> Cross check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level
Provide the client with an interim report

05 Bug fixing & re-check.
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

The audited scope includes SportX token and its staking mechanism including fees and rewards. SportX token supports ERC20 interface and permits signature method. Staking contract allows users to participate in pools, pay fees and handle rewards.

1.5 PROJECT DASHBOARD

Client	SportX
Audit name	Smart Contracts Security Audit
Initial version	02240925d05f7536b5dfbbb931b993e5a09d8891
Final version	ef8a0ecbaa891fd2683c946e3f60bd36dee54e66
SLOC	818
Date	2020-01-04 - 2020-01-19
Auditors engaged	2 auditors

FILES LISTING

SportX.sol	SportX.sol
FeePool.sol	FeePool.sol
SportXStakingRewardsPool.sol	SportXStakingRewardsPool.sol
SportXVault.sol	SportXVault.sol
Staking.sol	Staking.sol
StakingParameters.sol	StakingParameters.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	6
Comment	8

CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit no critical issues were found, 1 quite an interesting issue was found and marked as WRN-1, it could lead to some undesired behavior, but then the client decided that the contract owner reliable behavior would not tend to the described problem, also several warnings and comments were found and discussed with the client. After working on the reported findings all of them were resolved or acknowledged (if the problem was not critical). So, the contracts are assumed as secure to use according to our security criteria.

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	`poolTokens` might be changed between `finalizeEpoch` calls
File	Staking.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At Staking.sol#L120 in the `finalizeEpoch` method `poolTokens` are processed, however the list of these tokens may be changed in `stakingParameters`, so it could lead to the lost reward that was already accumulated for token A while EPOCH if token A had been removed just before EPOCH was finalized.

RECOMMENDATION

Freeze `poolTokens` for every epoch.
Add the required logic on change `poolTokens` in `Staking.sol`.

CLIENT'S COMMENTARY

> We've decided to keep this as it is given that the `poolTokens` array can only be changed by an admin account. We believe it is fine under our security assumptions as we already have emergency withdraw methods from such an admin account in case things go wrong. We will not be removing tokens haphazardly, if ever, and are weary of the additional logic to freeze the tokens. Finally, we do not expect the pool to be drained 100% ever.

WRN-2	Reward is lost if not claimed
File	Staking.sol
Severity	Warning
Status	No issue

DESCRIPTION

At line Staking.sol#L132 `previousEpochClaimedRewards` is reset on the `finalizeEpoch` call. So, if somebody did not claim the rewards, it will be lost.

RECOMMENDATION

Keep the old reward even if it's not claimed for a long time.

CLIENT'S COMMENTARY

> This is a design decision as we did not want the additional complexity of maintaining old rewards.

WRN-3	Use SafeMath everywhere
File	Staking.sol
Severity	Warning
Status	Fixed at ef8a0ecb

DESCRIPTION

At Staking.sol#L166 the usual `+` is used, but two big values are added.

RECOMMENDATION

We suggest to use SafeMath.

WRN-4	ERC20 unsafe `approve` method is used
File	SportXStakingRewardsPool.sol SportX.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At line SportXStakingRewardsPool.sol#L60 and SportX.sol#L70 the ERC20 method `approve` is used, which allows a front-running attack. See more here https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit#heading=h.b32yfk54vyg9.

RECOMMENDATION

Use an atomic compare-and-set approve method instead (as described in the document above).

CLIENT'S COMMENTARY

> We decided against this for now as we don't wish to go against the ERC20 spec yet.

WRN-5	Lack of `safeTransfer` in `emergencyWithdraw`
File	FeePool.sol SportXStakingRewardsPool.sol SportXVault.sol
Severity	Warning
Status	Fixed at **ef8a0ecb**

DESCRIPTION

At line FeePool.sol#L12 and SportXStakingRewardsPool.sol#L13

using of SafeERC20 is defined and at line FeePool.sol#L42 it's used to withdraw the fee.

However, just a `transfer` method is used at lines:

- FeePool.sol#L50
- SportXStakingRewardsPool.sol#L52
- SportXVault.sol#L58

and just a `safeTransfer` method is used at lines

- SportXVault.sol#L67.

RECOMMENDATION

Use `safeTransfer` and `safeTransferFrom` for consistency in using API of tokens with no `transfer` return.

WRN-6	Potentially incorrect epoch information retrieval
File	Staking.sol
Severity	Warning
Status	Fixed at **ef8a0ecb**

DESCRIPTION

This warning is about a potentially incorrect epoch reward claim attempt possibility in here: Staking.sol#L161.

Usage of this function on the 0-th epoch can lead to the rewards retrieval from the epoch `uint256(0) - 1`, which is not something expected by the application logic.

RECOMMENDATION

It is recommended to add the `require(epoch > 0)` check in here: Staking.sol#L141 to avoid an incorrect epoch rewards retrieval. It is also recommended to introduce additional checks whether the previous epoch exists at all in order to avoid incorrect `previousEpochClaimedRewards` and `previousEpochRewards` selections (possibly with `startEpochTime` checked as well).

2.4 COMMENTS

CMT-1	SportX.sol incorrect path
File	SportX.sol
Severity	Comment
Status	Fixed at ef8a0ecb

DESCRIPTION

SportX.sol is placed in the `interfaces` folder at SportX.sol, however it implements the whole contract.

RECOMMENDATION

Place SportX.sol to the `impl` folder.

CMT-2	No reentry protection
File	Staking.sol FeePool.sol
Severity	Comment
Status	Fixed at ef8a0ecb

DESCRIPTION

Some methods such as:

- Staking.sol#L436
- Staking.sol#L446
- Staking.sol#L457
- FeePool.sol#L41

are logically assumed to be called once whithin the same transaction. However, in fact, they could be called several times recursively. It does not cause any issue now (at least now), especially considering the usage of `INSUFFICIENT_TIME_PASSED_SINCE_UNSTAKE` and `onlyStaking` checks, but anyway asserted prohibiting of re-entry makes the code more robust.

RECOMMENDATION

Add re-entrancy protection.

CLIENT'S COMMENTARY

> We added re-entrancy protection to the claim rewards function as that is the only one that calls `withdrawFee`, which calls an external contract.

CMT-3	External method defined as `public`
File	SportXVault.sol
Severity	Comment
Status	Fixed at ef8a0ecb

DESCRIPTION

Method `openEmergencyHatch` is defined at SportXVault.sol#L79 public, however it's assumed to be used only as an external method.

RECOMMENDATION

Define the method as `external`.

CMT-4	Unused argument of the `onlySuperAdmin` modifier
File	Staking.sol
Severity	Comment
Status	Fixed at ef8a0ecb

DESCRIPTION

At line Staking.sol#L96 the argument `operator` is not used.

RECOMMENDATION

Remove the argument.

CMT-5	Use `>=` instead of `==` for `ALL_REWARDS_CLAIMED`
File	Staking.sol
Severity	Comment
Status	Fixed at ef8a0ecb

DESCRIPTION

At line Staking.sol#L149 the impossible condition of `previousEpochClaimedRewards[token] > previousEpochRewards[token]` is not properly checked anywhere.

RECOMMENDATION

Use `previousEpochClaimedRewards[token] >= previousEpochRewards[token]` instead of `==`.

CMT-6	Use `<=` for `NEW_REWARD_MULTIPLIER_TOO_HIGH` check
File	StakingParameters.sol
Severity	Comment
Status	Fixed at ef8a0ecb

DESCRIPTION

At line StakingParameters.sol#L58 the reward multiplier may be `== 10**20` by the logic.

RECOMMENDATION

Use `newRewardMultiplier <= FRACTION_PRECISION`.

CMT-7	An incorrect method `getRewardMultipliers` name
File	StakingParameters.sol
Severity	Comment
Status	Fixed at ef8a0ecb

DESCRIPTION

At line StakingParameters.sol#L88 the method returns one multiplier for one specific token, so it's better to rename the method.

RECOMMENDATION

Rename to `getRewardMultiplier`.

CMT-8	Unnecessary computations
File	Staking.sol SportX.sol
Severity	Comment
Status	Fixed at ef8a0ecb

DESCRIPTION

This comment is about unnecessary computations being performed even in case the function has already received an incorrect input in here: Staking.sol#L187, in here: SportX.sol#L38, and in here: Staking.sol#L274.

The function input parameter set can be determined as incorrect without recovering the meta-transaction signature or computing the digest. A staker's address, expiration timestamp, nonce, amount and block timestamp being incorrect, can be determined earlier than the incorrectness of the signature, which actually requires to be restored.

RECOMMENDATION

It is recommended to move these: Staking.sol#L311, Staking.sol#L307, Staking.sol#L306, Staking.sol#L302, Staking.sol#L300, these: Staking.sol#L254, Staking.sol#L253, Staking.sol#L252, Staking.sol#L250, and these: SportX.sol#L69, SportX.sol#L68, SportX.sol#L66 checks to the very beginning of their functions respectively in order to avoid unnecessary computations being performed even with the function parameter set input already known to be incorrect.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

BLOCKCHAINS

 Ethereum

 Cosmos

 EOS

 Substrate

TECH STACK

 Python

 Solidity

 Rust

 C++

CONTACTS

 https://github.com/mixbytes/audits_public

 https://mixbytes.io/

 hello@mixbytes.io

 https://t.me/MixBytes

 https://twitter.com/mixbytes