Cyberscope

Audit Report

SoccerNBet

November 2022

SHA256 40e0848c4967d9a3175819f9f98a4c35bd6f8eaa9e96899292d8fa2059207969

Audited by © cyberscope

Table of Contents

Contract Review

Contract Name	SoccerNBet
Compiler Version	v0.8.14+commit.80d49f37
Explorer	https://testnet.bscscan.com/token/0xF227188048c00B25E3aB791Afe4A6e87A25Ab75F
Domain	https://soccern.xyz

Audit Updates

Initial Audit	18th November 2022
Corrected	

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249aa4df6024fbfaa94f79ab2f44f3231
@openzeppelin/contracts/security/ReentrancyGuard.sol	aa73590d5265031c5bb64b5c0e7f84c44cf5f8539e6d8606b763adac784e8b2e
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol	fc0e6c5d7184bd03b8deae6ca9a48a1eaaecf9f5e4703611aabfb63401e6d43f
@openzeppelin/contracts/utils/cryptography/ECDSA.sol	4e45d53327d561848fbcf381262ec5c0ac91b2f1f06432210bf76db55279d945
@openzeppelin/contracts/utils/math/SafeMath.sol	0dc33698a1661b22981abad8e5c6f5ebca0dfe5ec14916369a2935d888ff257a
@openzeppelin/contracts/utils/Strings.sol	34127ad0054df5963b0fd694c1b313d17e9114a2f426b85526d6d976210298ab
@openzeppelin/contracts/utils/structs/EnumerableSet.sol	778d5305652c4eb562b12880cb6cf023d67df24844c15783a0b80fac2e715585
contracts/EIP712Checker.sol	6e088305a9d57c7e97c5b2ff8753a5f9b49485a6196dc1352c3081ffb4ea8db1
contracts/SoccerNBet.sol	40e0848c4967d9a3175819f9f98a4c35bd6f8eaa9e96899292d8fa2059207969

contracts/SoccerNLib.sol	a3d9f32a8e3f63e2302af870a910c96b3f87896f5213b0eda20c72b884942aff

Introduction

The contract implements a betting mechanism. The admin set fixtures and the users can place bets. At the end of the round, the owner set the winning bets.

Admin Responsibilities

The admin is responsible for pickering the winners.

The admin is responsible for providing funds to cover the betting rewards. The contract does not keep reserves in relation to the rewarding amount. Hence, if the contract owner does not manually transfer the reward funds to the contract, the users will not be able to claim their rewards. This could happen in the `setBetOddResults()` method, when the rewarded funds are calculated.

Out of Audit Context

The `ISoccerNFT public nft;` and all the related functionality is out of the audit scope.

Roles

The SoccerNBet contract has two roles. They consist of the admin and the owner role.

Owner

The owner has the authority to

- Manage Signers.
- Give Admin privileges.

Admin

The admin has the authority to

- Set Fixtures
- Set Bet status.
- Set Bet odds.

Contract Diagnostics

🔴 Critical 🟠 Medium ● Minor / Informative

Severity	Code	Description	Status
🟠	OCA	Overpriced Claim Amount	Unresolved
●	STC	Succeeded Transfer Check	Unresolved
●	FSI	Fixture Status Inconsistency	Unresolved
●	BOVD	Bet Odd Value Duplication	Unresolved
●	MC	Missing Check	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved

OCA - Overpriced Claim Amount

Criticality	Medium
Location	contract.sol#L285,332,365,490,491
Status	Unresolved

Description

The admin has the ability to set the bet winners. During this phase, the contract calculates the total winning amount using the odd weight from the BetOdd structure.
When the users place a bet, they have the ability to add an arbitrary odd amount different than the BetOdd.odd to the BettingSlip.odd.
During the bet claim phase, the winners have the ability to claim their rewards. The reward is calculated based on the BettingSlip.odd weight. Hence, the users have the ability to claim a greater amount than the expected `(claimedTokenWinningsAmount > totalTokenWinningsAmount)`

```
// setBetOddResults()
tokenWinningsAmount = _betOdd.totalTokenStaked.mul(_betOdd.odd).div(ODD_RATE_BASE);
betResult.totalTokenWinningsAmount += tokenWinningsAmount;


// _bet()
betSlip.odd = betSlipData.odd;

// _claimBetSlip
_getBetSlipAmounts(_betSlip.stakeAmount, _betSlip.odd)
```

Recommendation

The contract should take in account the actual claimable amount.

STC - Succeeded Transfer Check

Criticality	minor / informative
Location	contract.sol#L285,332,365,490,491
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
erc20Token.transfer(DEAD, _burnAmount);

erc20Token.transfer(upInvter.holder, commissionAmount);

erc20Token.transferFrom(msg.sender, address(this), allStakeAmount);

erc20Token.transfer(feeWallet, _betSlip.commissionAmount);
erc20Token.transfer(_betSlip.bettor, _betSlip.claimAmount);
```

Recommendation

The contract should check if the result of the transfer methods is successful.

FSI - Fixture Status Inconsistency

Criticality	minor / informative
Location	contract.sol
Status	Unresolved

Description

The fixture status can be initialized from two methods. The `setFixture()` can set any status on any existing fixture. On the contrary, the `setBetStatus()` prevents the admin to set an invalid status value. For instance, the `setBetStatus()` prevents the admin to set the status from Betting to Auto but this can happen from the `setFixtures()` method. As a result, the contract creates inconsistency between the status values that can be initialized.

```
function setFixtures(SetFixture[] memory data) external onlyAdmin {
    require(flagActive, "Not active");

    for (uint256 index = 0; index < data.length; index++) {
        uint64 fixtureId = data[index].fixtureId;

        Fixture storage _fixtrue = _fixtures[fixtureId];
        if (_fixtureIds.contains(fixtureId) == false) {
            _fixtrue.id = fixtureId;
            _fixtureIds.add(fixtureId);
        }

        _fixtrue.betStatus = BetStatus.Auto;
        _fixtrue.bettingStartTime = data[index].bettingStartTime;
        _fixtrue.bettingStopTime = data[index].bettingStopTime;
        emit FixtureChanged(fixtureId, _fixtrue.bettingStartTime,
_fixtrue.bettingStopTime);
    }
}

function setBetStatus(uint64 fixtureId, BetStatus _status) external onlyAdmin
{
    _setBetStatus(fixtureId, _status);
}
```

Recommendation

The initialization of the structure values should be the same across all the setter methods.

BOVD - Bet Odd Value Duplication

Criticality	minor / informative
Location	contract.sol
Status	Unresolved

Description

The business logic seems peculiar. The implementation may not follow the expected behavior. The contract performs transfer transactions without checking if there is sufficient amount on the contract.

```
betOdd.odd = betSlipData.odd;
//...
betSlip.odd = betSlipData.odd;
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic. The contract could pre-check if there are sufficient resources to perform a transaction.

MC - Missing Check

Criticality	minor / informative
Location	contract.sol#L1
Status	Unresolved

Description

The contract is processing variables that have not properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

```
_fixtrue.bettingStartTime = data[index].bettingStartTime;
_fixtrue.bettingStopTime = data[index].bettingStopTime;

betOdd.id = betSlipData.betOddId;
betOdd.odd = betSlipData.odd;
```

Recommendation

The contract should properly check the variables according to the required specifications.

- bettingStopTime > bettingStartTime

- `uint256 actualWinningsAmount = claimAmount - _betSlip.stakeAmount;` that means that the claimAmount should always be greater than `_betSlip.stakeAmount;`. Hence the `mul(odd).div(ODD_RATE_BASE)` should produce numbers greater than 1. So, `betSlipData.odd > ODD_RATE_BASE`

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contracts/SoccerNBet.sol#L120,123,106,121,116,499,205,122,118,108,107,228
Status	Unresolved

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_lastBetSlipId
_fixturesBetSlips
_fixtures
_betSlips
_betOdds
_wd
_startActive
_bettorAllBetSlips
_betOddIds
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions.

L11 - Unnecessary Boolean equality

Criticality	minor / informative
Location	contracts/SoccerNBet.sol#L209,374
Status	Unresolved

Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
_fixtureIds.contains(fixtureId) == false
_betOddIds[fixtureId].contains(betSlipData.betOddId) == false
```

Recommendation

Remove the equality to the boolean constant.

L13 - Divide before Multiply Operation

Criticality	minor / informative
Location	contracts/SoccerNBet.sol#L506
Status	Unresolved

Description

Performing divisions before multiplications may cause lose of prediction.

```
winningsAmount = stakeAmount.mul(odd).div(ODD_RATE_BASE)
```

Recommendation

The multiplications should be prior to the divisions.

L14 - Uninitialized Variables in Local Scope

Criticality	minor / informative
Location	contracts/SoccerNBet.sol#L264
Status	Unresolved

Description

The are variables that are defined in the local scope and are not initialized.

```
betResult
```

Recommendation

All the local scoped variables should be initialized.

L15 - Local Scope Variable Shadowing

Criticality	minor / informative
Location	contracts/SoccerNBet.sol#L151,446,228,163
Status	Unresolved

Description

The are variables that are defined in the local scope containing the same name from an upper scope.

```
_status
_signers
```

Recommendation

The local variables should have different names from the upper scoped variables.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
ReentrancyGuard	Implementation			
	<Constructor>	Public	✓	-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
EIP712	Implementation			
	<Constructor>	Public	✓	-
	_domainSeparatorV4	Internal		
	_buildDomainSeparator	Private		
	_hashTypedDataV4	Internal		

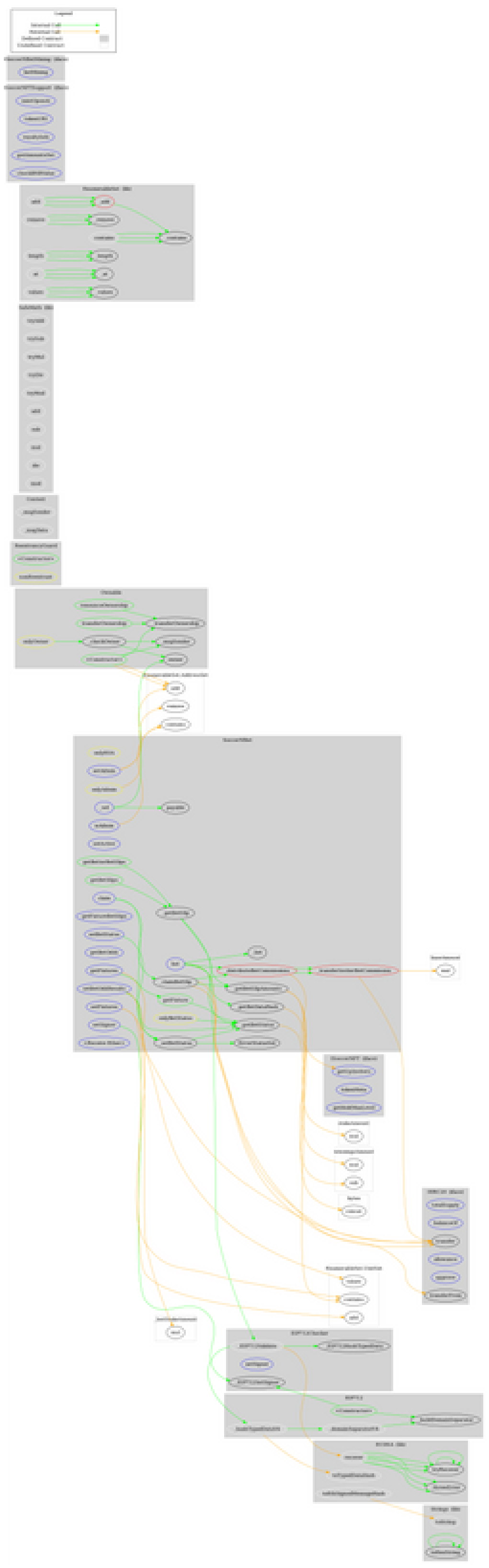
ECDSA	Library			
	_throwError	Private		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	toEthSignedMessageHash	Internal		
	toEthSignedMessageHash	Internal		
	toTypedDataHash	Internal		
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
Strings	Library			
	toString	Internal		
	toHexString	Internal		
	toHexString	Internal		
	toHexString	Internal		
EnumerableSet	Library			

	_add	Private	✓	
	_remove	Private	✓	
	_contains	Private		
	_length	Private		
	_at	Private		
	_values	Private		
	add	Internal	✓	
	remove	Internal	✓	
	contains	Internal		
	length	Internal		
	at	Internal		
	values	Internal		
	add	Internal	✓	
	remove	Internal	✓	
	contains	Internal		
	length	Internal		
	at	Internal		
	values	Internal		
	add	Internal	✓	
	remove	Internal	✓	
	contains	Internal		
	length	Internal		
	at	Internal		
	values	Internal		
EIP712Checker	Implementation	EIP712		
	<Constructor>	Public	✓	EIP712
	_EIP712HashTypedData	Internal		
	_EIP712SetSigner	Internal	✓	
	setSigner	External	✓	-
	_EIP712Validate	Internal	✓	

SoccerNBet	Implementation	EIP712Checker, ReentrancyGuard, Ownable		
	<Constructor>	Public	✓	Ownable ReentrancyGuard EIP712Checker
	setSigner	External	✓	onlyOwner
	isAdmin	External		-
	setAdmin	External	✓	onlyOwner
	setActive	External	✓	onlyAdmin
	setFixtures	External	✓	onlyAdmin
	setBetStatus	External	✓	onlyAdmin
	_setBetStatus	Internal	✓	
	setBetOddResults	External	✓	onlyAdmin onlyBetStatus
	getFixtureBetSlips	External		-
	getFixture	Public		-
	getFixtures	External		-
	getBetOdds	External		-
	_transferInviterBetCommission	Private	✓	
	_distributeBetCommissions	Private	✓	
	bet	External	Payable	onlyEOA nonReentrant
	_bet	Internal	✓	onlyBetStatus
	_getBetStatus	Private		
	getBettorBetSlips	Public		-
	getBetSlips	Public		-
	_getBetSlip	Internal		
	claim	External	✓	onlyEOA nonReentrant
	_claimBetSlip	Private	✓	
	_wd	External	✓	-
	<Receive Ether>	External	Payable	-
	_getBetSlipAmounts	Private		
	_getBetDataHash	Private		

SoccerNLib	Library			
ISoccerNFTSupport	Interface			
	mintOpenAt	External		-
	tokenURI	External		-
	royaltyInfo	External		-
	getAmountsOut	External		-
	checkBNBValue	External		-
ISoccerNFT	Interface			
	getUpInviters	External		-
	tokenMeta	External		-
	getHoldMaxLevel	External		-
ISoccerNBetMining	Interface			
	betMining	External	✓	-

Contract Flow



Domain Info

Domain Name	soccern.xyz
Registry Domain ID	D333457734-CNIC
Creation Date	2022-11-15T06:18:12.0Z
Updated Date	2022-11-15T06:53:42.0Z
Registry Expiry Date	2023-11-15T23:59:59.0Z
Registrar WHOIS Server	whois.godaddy.com
Registrar URL	https://www.godaddy.com/
Registrar	Go Daddy, LLC
Registrar IANA ID	146

The domain was created 3 days before the creation of the audit. It will expire in 12 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

The SoccerNBEt contract operates a betting mechanism. This audit focused on investigating possible security issues, potential improvements and business logic consistency.

Disclaimer

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io