Cyberscope

Audit Report

EtherWars

July 2023

EtherWars 0882565c02bd713c064041c3e5dcdc07c9d16299d314a34c611a02a2582957af

EtherWarsQrng 40c4e598a779abc9aa26b36205ac59e502b27427c88dea8e56b93579b9cae3e9

Audited by © cyberscope

Table of Contents

Review

Audit Updates

Initial Audit	18 May 2023 https://github.com/cyberscope-io/audits/blob/main/1-stw/v1/EtherWars.pdf
Corrected Phase 2	27 Jun 2023 https://github.com/cyberscope-io/audits/blob/main/1-stw/v2/EtherWars.pdf
Corrected Phase 3	19 Jul 2023

Source Files

Filename	SHA256
EtherWarsQrng.sol	40c4e598a779abc9aa26b36205ac59e502b27427c88dea8e56b93579b9cae3e9
EtherWars.sol	0882565c02bd713c064041c3e5dcdc07c9d16299d314a34c611a02a2582957af

Overview

The SpinToWin ecosystem consists of various contracts. This audit report focuses on the EtherWars and EtherWarsQrng contracts. EtherWars is a decentralized gaming platform implemented as a smart contract. Users can engage in combat battles and compete for rewards. The contract integrates features such as strength, cooldowns, redemption points, and spin points to create an engaging gameplay experience. The contract relies on the OpenZeppelin library for security and access control. An external QRNGConsumer contract is used for random number generation to determine the outcome of battles. Users can enhance their combat abilities by increasing their power through additional Ether contributions. Spin points earned through the SpinToWin contract can be used to participate in the EtherWars game. The contract offers configurable parameters and supports user withdrawals.

Roles

EtherWars Contract

Owner

The Owner role has authority over the following functions:

- `function ownerWithdrawFees()`
- `function setMaximumStrength(uint256 _strength)`
- `function setMinimumStrength(uint256 _strength)`
- `function setDevFee(uint256 _devFee)`
- `function setRedemptionPercentage(uint256 _percentage)`
- `function setRedemptionPointsCost(uint256 _cost)`
- `function setCooldownReduction(uint256 _time)`
- `function setCooldownTime(uint256 _time)`
- `function setQRNGConsumer(address _qrngConsumer)`
- `function setAttackMultiplier(uint256 _multiplier)`
- `function setSpinToWinContract(address _address)`
- `function setMaxRandomNum(uint256 _num)`
- `function winChanceToggle(bool _attacker, bool _defender)`
- `function enableArena()`
- `function disableArena()`
- `function setSpinPoints(uint256 _winner, uint256 _loser)`

QRNGConsumer

The QRNGConsumer role has authority over the following functions:

- `function beginCombat(address _attacker, string memory _username, uint256 _faction, uint256[] calldata _randomWords)`

SpinToWin

The SpinToWin role has authority over the following functions:

- `function deductSpinPoints(address _user, uint256 _points)`

User

The User role can interact with the following functions:

- `function enterArena()`
- `function attack()`
- `function reduceCooldown()`
- `function userWithdraw(uint256 _amount)`
- `function activateWithdrawCooldown()`
- `function increasePower(address _contender)`
- `function changeUsername(string calldata _newName)`
- `function getFightList()`
- `function numberOfContenders()`

EtherWarsQrng Contract

Owner

The Owner role has authority over the following functions:

- `function setRequestParameters(address _airnode, bytes32 _endpointIdUint256Array, address _sponsorWallet)`
- `function setEtherWarsAddress(address _address)`
- `function setAirnodeAddress(address _address)`
- `function setSponsorWalletAddress(address _address)`
- `function setAirnodeAddress(bytes32 _endpoint)`

AirnodeRrp

The AirnodeRrp role has authority over the following functions:

- `function fulfillUint256Array(bytes32 _requestId, bytes calldata _data)`

EtherWars

The EtherWars role has authority over the following functions:

- `function makeRequestUint256Array(uint256 _size, address _attacker)`

Test Deployments

Contract	Explorer	Address
EtherWars	https://testnet.bscscan.com/address/0x7d009d35f23a9826c9197F0a8ED4079C2D08B977	0x7d009d35f23a9826c9197F0a8ED4079C2D08B977
EtherWarsQrng	https://testnet.bscscan.com/address/0x23f855D8AE3E753Eb8aBDA8266a10571399Cbe99	0x23f855D8AE3E753Eb8aBDA8266a10571399Cbe99

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	4

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	4	0	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	DKO	Delete Keyword Optimization	Unresolved
●	MU	Modifiers Usage	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L13	Divide before Multiply Operation	Unresolved

DKO - Delete Keyword Optimization

Criticality	Minor / Informative
Location	EtherWars.sol#L235,297,306,341,461
Status	Unresolved

Description

The contract resets variables to the default state by setting the initial values. Setting values to state variables increases the gas cost.

```
nameList[contenderProfiles[defender].name] = false
withdrawActive[user] = false
contenderStrength[user] = 0
nameList[previousName] = false
nameList[username] = false
```

Recommendation

The team is advised to use the `delete` keyword instead of setting variables. This can be more efficient than setting the variable to a new value, using delete can reduce the gas cost associated with storing data on the blockchain.

MU - Modifiers Usage

Criticality	Minor / Informative
Location	EtherWars.sol#L139,163
Status	Unresolved

Description

The contract is using repetitive statements on some methods to validate some preconditions. In Solidity, the form of preconditions is usually represented by the modifiers. Modifiers allow you to define a piece of code that can be reused across multiple functions within a contract. This can be particularly useful when you have several functions that require the same checks to be performed before executing the logic within the function.

```
if (contenderStrength[contender] + msg.value < minimumStrength)
revert NotEnoughStrength();
if (contenderStrength[attacker] < minimumStrength) revert
NotEnoughStrength();
```

Recommendation

The team is advised to use modifiers since it is a useful tool for reducing code duplication and improving the readability of smart contracts. By using modifiers to perform these checks, it reduces the amount of code that is needed to write, which can make the smart contract more efficient and easier to maintain.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	EtherWarsQrng.sol#L53,54,55,56,80,102,103,104,115,120,125,130EtherWars.sol#L136,172,173,174,175,288,325,333,347,358,365,372,378,384,389,394,400,406,411,417,422,438,452,466,480,486,487,488,489
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
uint256 _size
address _attacker
string memory _name
uint256 _faction
bytes32 _requestId
bytes calldata _data
address _airnode
bytes32 _endpointIdUint256Array
address _sponsorWallet
address _address
bytes32 _endpoint
string calldata _username
string memory _username
uint256[] calldata _randomWords

...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	EtherWars.sol#L191,194,244
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
uint256 attackCalculation = (attackStrength * maxRandomNum) /
(attackStrength + defenseStrength)
uint256 attackChance = (attackCalculation * (100 +
attackMultiplier)) / 100
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

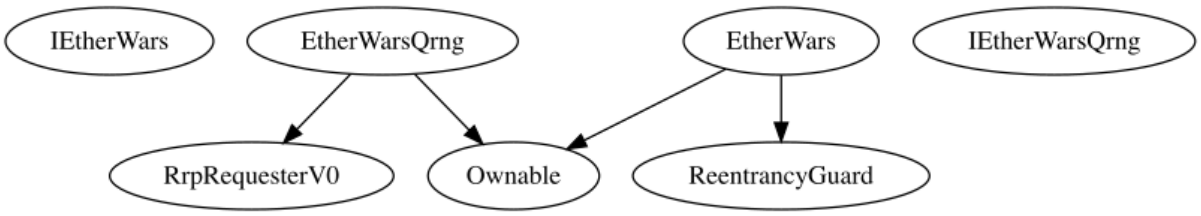
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IEtherWars	Interface			
	beginCombat	External	✓	-
EtherWarsQrng	Implementation	RrpRequesterV0, Ownable		
		Public	✓	RrpRequesterV0
	makeRequestUint256Array	External	✓	onlyEtherWars
	fulfillUint256Array	External	✓	onlyAirnodeRrp
	setRequestParameters	External	✓	onlyOwner
	setEtherWarsAddress	External	✓	onlyOwner
	setAirnodeAddress	External	✓	onlyOwner
	setSponsorWalletAddress	External	✓	onlyOwner
	setAirnodeAddress	External	✓	onlyOwner
IEtherWarsQrng	Interface			
	makeRequestUint256Array	External	✓	-

EtherWars	Implementation	Ownable, ReentrancyGuard		
		Public	✓	-
	enterArena	External	Payable	checkArena
	attack	External	✓	nonReentrant checkArena
	beginCombat	External	✓	onlyQRNGConsumer nonReentrant
	reduceCooldown	External	✓	nonReentrant checkArena
	userWithdraw	External	✓	nonReentrant
	activateWithdrawCooldown	External	✓	-
	increasePower	External	Payable	checkArena
	changeUsername	External	✓	-
	deductSpinPoints	External	✓	onlySpinToWin
	ownerWithdrawFees	External	✓	onlyOwner
	setMaximumStrength	External	✓	onlyOwner
	setMinimumStrength	External	✓	onlyOwner
	setDevFee	External	✓	onlyOwner
	setRedemptionPercentage	External	✓	onlyOwner
	setRedemptionPointsCost	External	✓	onlyOwner
	setCooldownReduction	External	✓	onlyOwner
	setCooldownTime	External	✓	onlyOwner
	setQRNGConsumer	External	✓	onlyOwner
	setAttackMultiplier	External	✓	onlyOwner
	setSpinToWinContract	External	✓	onlyOwner

	setMaxRandomNum	External	✓	onlyOwner
	winChanceToggle	External	✓	onlyOwner
	enableArena	External	✓	onlyOwner
	disableArena	External	✓	onlyOwner
	setSpinPoints	External	✓	onlyOwner
	getFightList	External		-
	numberOfContenders	Public		-
	removeFromList	Private	✓	
	increaseWinnerStrength	Private	✓	
	sendViaCall	Private	✓	
	emitCombatResult	Private	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
EtherWars
<Constructor>
onlyQRNGConsumer
onlySpinToWin
checkArena
enterArena
changeUsername
increasePower
attack
beginCombat
reduceCooldown
userWithdraw
ownerWithdrawFees
deductSpinPoints
activateWithdrawCooldown
setMaximumStrength
setMinimumStrength
setDevFee
setRedemptionPercentage
setRedemptionPointsCost
setCooldownReduction
setCooldownTime
setQRNGConsumer
setAttackMultiplier
setSpinToWinContract
setMaxRandomNum
winChanceToggle
enableArena
disableArena
setSpinPoints
getFightList
NotQRNGConsumer
NotSpinToWinContract
ArenaOffline
AlreadyInArena
StrengthTooHigh
EmptyName
NameAlreadyExists
NotEnoughStrength
SenderNotContender
NotInList
numberOfContenders
EmptyList
emitCombatResult
removeFromList
increaseWinnerStrength
AttackerIsDefender
AttackerTooWeak
NotEnoughRedemptionPoints
RedemptionNotActive
NotEnoughCooldownLeft
WithdrawNotActive
NotEnoughFunds
StillInCooldown
NoAmountIncluded
sendViaCall
SendEthFailed
payable
owner
WithdrawAlreadyActivated
IEtherWarsQrng (iface)
makeRequestUint256Array
EtherWarsQrng
<Constructor>
onlyEtherWars
setAirnodeAddress
setRequestParameters
setEtherWarsAddress
setSponsorWalletAddress
makeRequestUint256Array
fulfillUint256Array
NotEtherWarsContract
BadEndpoint
ZeroAddressNotAllowed
airnodeRrp
makeFullRequest
IEtherWars (iface)
beginCombat

Summary

EtherWars contract implements a game and rewards mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

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