



Cyberscope

Audit Report

President's Squid Game

July 2023

Network BSC

Address 0x2037d5d9E1B184b0C2D2991DFCA303F404A29478

Audited by © cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Unresolved
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	PAV	Pair Address Validation	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	URC	Update Router Concern	Unresolved
●	RES	Redundant Event Statement	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L22	Potential Locked Ether	Unresolved

Table of Contents

Analysis	1
Diagnostics	2
Table of Contents	3
Review	5
Audit Updates	5
Source Files	5
Findings Breakdown	6
ST - Stops Transactions	7
Description	7
Recommendation	7
ELFM - Exceeds Fees Limit	8
Description	8
Recommendation	8
BC - Blacklists Addresses	9
Description	9
Recommendation	9
PAV - Pair Address Validation	10
Description	10
Recommendation	10
RSW - Redundant Storage Writes	11
Description	11
Recommendation	11
URC - Update Router Concern	13
Description	13
Recommendation	13
RES - Redundant Event Statement	14
Description	14
Recommendation	14
RSML - Redundant SafeMath Library	15
Description	15
Recommendation	15
L02 - State Variables could be Declared Constant	16
Description	16
Recommendation	16
L04 - Conformance to Solidity Naming Conventions	17
Description	17
Recommendation	18

L07 - Missing Events Arithmetic	19
Description	19
Recommendation	19
L09 - Dead Code Elimination	20
Description	20
Recommendation	20
L16 - Validate Variable Setters	22
Description	22
Recommendation	22
L17 - Usage of Solidity Assembly	23
Description	23
Recommendation	23
L19 - Stable Compiler Version	24
Description	24
Recommendation	24
L22 - Potential Locked Ether	25
Description	25
Recommendation	25
Functions Analysis	26
Inheritance Graph	33
Flow Graph	34
Summary	35
Disclaimer	36
About Cyberscope	37

Review

Contract Name	PresidentsSquidGame
Compiler Version	v0.8.7+commit.e28d00a7
Optimization	200 runs
Explorer	https://bscscan.com/address/0x2037d5d9e1b184b0c2d2991dfca303f404a29478
Address	0x2037d5d9e1b184b0c2d2991dfca303f404a29478
Network	BSC
Symbol	PSG1000X
Decimals	18
Total Supply	45,600,000,000

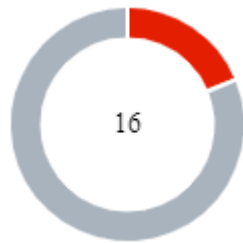
Audit Updates

Initial Audit	05 Jul 2023
---------------	-------------

Source Files

Filename	SHA256
PresidentsSquidGame.sol	26327e31999c3200a50239ff0ac7492c3cada0ca4c1891b172a7fc039c482919

Findings Breakdown



Critical	3
Medium	0
Minor / Informative	13

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	3	0	0	0
Medium	0	0	0	0
Minor / Informative	13	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	PresidentsSquidGame.sol#L522,532
Status	Unresolved

Description

The contract owner has the authority to stop the sales for all users excluding the owner. The owner may take advantage of it by setting the `SellFee` to a high value. As a result, the contract may operate as a honeypot.

```
if (takeFee) {  
    uint256 feePercent = automatedMarketMakerPairs[_to] ? SellFee :  
    BuyFee;
```

The contract owner has the authority to stop the sales for all users excluding the owner. The owner may take advantage of it by setting the `isTradingEnabled` to false.

```
require(isTradingEnabled || _to == owner() || _from == owner(),  
    "Trading is disabled");
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	PresidentsSquidGame.sol#L600,604
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `setBuyFee` or the `setSellFee` function with a high percentage value.

```
function setBuyFee(uint256 _buyFee) public onlyOwner {  
    BuyFee = _buyFee;  
}  
  
function setSellFee(uint256 _sellFee) public onlyOwner {  
    SellFee = _sellFee;  
}
```

Recommendation

The contract could embody a check for the maximum acceptable value. The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

BC - Blacklists Addresses

Criticality	Critical
Location	PresidentsSquidGame.sol#L608
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `blacklistAddress` function.

```
function blacklistAddress(address account, bool value) public onlyOwner
{
    _isBlacklisted[account] = value;
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

PAV - Pair Address Validation

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol#L593
Status	Unresolved

Description

The contract is missing address validation in the pair address argument. The absence of validation reveals a potential vulnerability, as it lacks proper checks to ensure the integrity and validity of the pair address provided as an argument. The pair address is a parameter used in certain methods of decentralized exchanges for functions like token swaps and liquidity provisions.

The absence of address validation in the pair address argument can introduce security risks and potential attacks. Without proper validation, if the owner's address is compromised, the contract may lead to unexpected behavior like loss of funds.

```
function _setAutomatedMarketMakerPair(address pair, bool value) private
{
    require(automatedMarketMakerPairs[pair] != value, "Error: Automated
market maker pair is already set to that value");
    automatedMarketMakerPairs[pair] = value;
}
```

Recommendation

To mitigate the risks associated with the absence of address validation in the pair address argument, it is recommended to implement comprehensive address validation mechanisms. A recommended approach could be to verify pair existence in the decentralized application. Prior to interacting with the pair address contract, perform checks to verify the existence and validity of the contract at the provided address. This can be achieved by querying the provider's contract or utilizing external libraries that provide contract verification services.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol#L579,594,608,616
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract updates the whitelisted status of an account even if its current state is the same as the one passed as an argument. As a result, the contract performs redundant storage writes.

```
function excludeFromFees(address account, bool excluded) public
onlyOwner {
    require(!_isBlacklisted[account], "Account is blacklisted");
    _isExcludedFromFees[account] = excluded;
}

function setRewardAddress(address _rewardAddress) public onlyOwner {
    excludeFromFees(rewardAddress, false);
    rewardAddress = _rewardAddress;
    excludeFromFees(rewardAddress, true);
}

function blacklistAddress(address account, bool value) public onlyOwner
{
    _isBlacklisted[account] = value;
}

function setIsTradingEnabled(bool value) public onlyOwner {
    isTradingEnabled = value;
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

URC - Update Router Concern

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol#L571
Status	Unresolved

Description

The contract allows for the modification of the router address, which subsequently updates the pair address. However, the contract does not add the new pair address to the `automatedMarketMakerPairs` mapping. This omission can lead to unexpected behavior and potential issues with the contract's functionality.

```
function updateUniswapV2Router(address newAddress) public onlyOwner {
    require(newAddress != address(uniswapV2Router), "Error: The router
already has that address");
    uniswapV2Router = IUniswapV2Router02(newAddress);
    address _uniswapV2Pair =
    IUniswapV2Factory(uniswapV2Router.factory())
        .createPair(address(this), uniswapV2Router.WETH());
    uniswapV2Pair = _uniswapV2Pair;
}
```

Recommendation

It is recommended to include the necessary logic to add the new pair address to the `automatedMarketMakerPairs` mapping when updating the router address. This can be achieved by explicitly calling the appropriate function to add the pair address to the mapping after the router address is modified.

RES - Redundant Event Statement

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol#L433
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The `InvestFundAddressChanged` event statement is not used in the contract's implementation.

```
event InvestFundAddressChanged(address _new) ;
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it. It is recommend removing the unused event statement from the contract..

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on

<https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes>.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol#L417,418,419,420
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
function totalSupply() external override view retur  
  
t256) {  
    return _totalSupply;  
}  
function getOwner() external overr  
  
w returns (address) {
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol#L255,256,263,281,422,423,429,430,550,594,600,604
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.

```
IUniswapV2Router01 {  
    function factory() external pure returns (address);  
    function WETH() external pure returns (address);  
    ...  
    uint amountETHMin,  
    address  
    ...  
    function swapExactETHForTokens(uint amountOutMin,  
        address[] memory path, address to, uint deadline)  
        external returns (uint[] memory amounts);  
    ...  
}
```

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>.

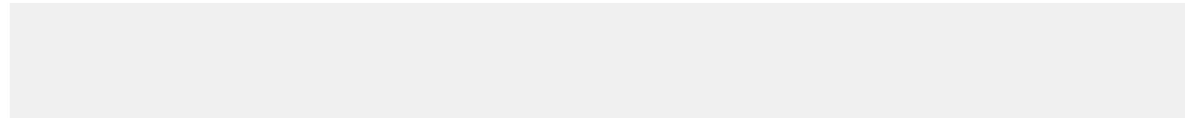
L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol#L601,605
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.



Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol#L117,123,129,133,137,141,148,152,159,163,169
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
(success, returndata, errorMessage);  
    }  
  
    function functionStaticCall(address target, bytes memory data)  
internal view returns (bytes memory) {  
    return f  
  
taticCall(target, data, "Address: low-level static call failed");  
    }  
  
    function functionStaticCall(address target, bytes memory data,  
string memory errorMessage) internal view returns (bytes memory) {  
        require(isContract(target), "Address: static call to  
non-contract");  
        (bool success, b  
  
...
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol#L576,596
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
public onlyOwner {  
    isT
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol#L119,174
Status	Unresolved

Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
bytes memory data) internal view returns (
    uint256);
    function transfer(address recipient, uint256 amount) external
    returns (bool);
    function allowance(address _owner, address spender) external
```

Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol#L14
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
internal pure returns (u
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

L22 - Potential Locked Ether

Criticality	Minor / Informative
Location	PresidentsSquidGame.sol#L452
Status	Unresolved

Description

The contract contains Ether that has been placed into a Solidity contract and is unable to be transferred. Thus, it is impossible to access the locked Ether. This may produce a financial loss for the users that have called the payable method.

```
;
    }

    function incr
```

Recommendation

The team is advised to either remove the payable method or add a withdraw functionality. it is important to carefully consider the risks and potential issues associated with locked Ether.

Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
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		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		

Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	_verifyCallResult	Private		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
IERC20	Interface			
	totalSupply	External		-
	decimals	External		-

	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-

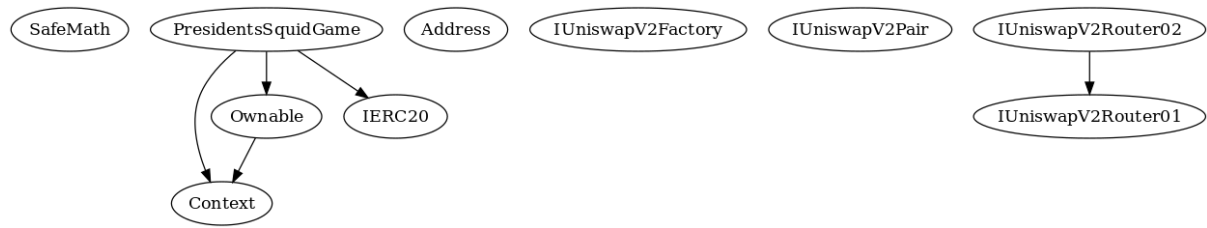
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-

	initialize	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-

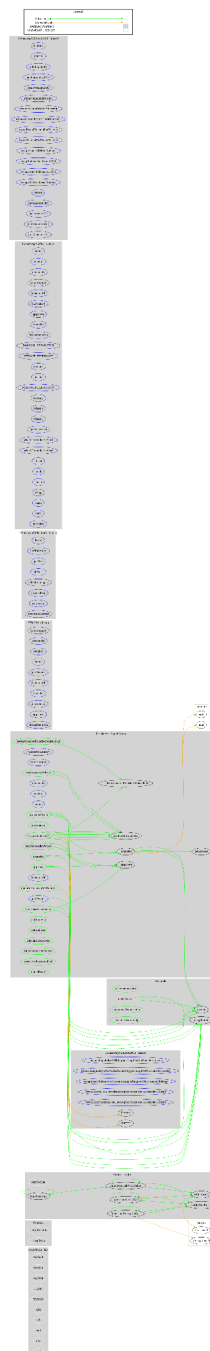
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
PresidentsSquidGame	Implementation	Context, IERC20, Ownable		
		Public	✓	-
		External	Payable	-
	totalSupply	External		-
	getOwner	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-

	_transfer	Internal	✓	
	sendFees	Internal	✓	
	_approve	Internal	✓	
	balanceOf	External		-
	updateUniswapV2Router	Public	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	setRewardAddress	Public	✓	onlyOwner
	setBuyFee	Public	✓	onlyOwner
	setSellFee	Public	✓	onlyOwner
	blacklistAddress	Public	✓	onlyOwner
	isExcludedFromFees	Public		-
	setIsTradingEnabled	Public	✓	onlyOwner
	openTrade	Public	✓	onlyOwner

Inheritance Graph



Flow Graph



Summary

President's Squid Game contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stop transactions, manipulate the fees and massively blacklist addresses. The contract can be converted into a honeypot and prevent users from selling if the owner abuses the admin functions. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

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>