Cyberscope

Audit Report

StarWar

February 2023

Type	ERC20
Network	ARBITRUM
Address	0xf427b1f577fd367ccfb9ec8928b01e01d0446de4
Audited by	© cyberscope

Table of Contents

Review

Contract Name	StarWar
Compiler Version	v0.8.16+commit.07a7930e
Optimization	200 runs
Explorer	https://arbiscan.io/address/0xf427b1f577fd367ccfb9ec8928b01e01d0446de4
Address	0xf427b1f577fd367ccfb9ec8928b01e01d0446de4
Network	ARBITRUM
Symbol	STAR
Decimals	18
Total Supply	100,000,000

Audit Updates

Initial Audit	25 Feb 2023

Source Files

Filename	SHA256
StarWar.sol	c96e0ca9f81b3621c8fee7d6177048a112bd040a9e7f9d83f1cdca9a72d865fd

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

ST - Stops Transactions

Criticality	Medium
Location	StarWar.sol#L440
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users excluding the owner. The owner may take advantage of it by calling the `pauseTrading()`.

```
if(!tradingActive){
    require(_isExcludedFromFees[from] || _isExcludedFromFees[to], "Trading is
not active.");
}
```

Additionally, the contract blacklists the contracts that were bought up to 10 blocks after the opening of the trade. The trades can open multiple times by calling sequentially the `pauseTrading()` and `unpauseTrading()` methods.

If the `transferDelayEnabled` is enabled, each user is limited to one trade per 5 blocks.

Recommendation

The contract could embody a check for not allowing setting the _maxTxAmount less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply. 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	StarWar.sol#L338
Status	Unresolved

Description

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

```
function manageRestrictedWallets(address[] calldata wallets,  bool restricted)
external onlyOwner {
    for(uint256 i = 0; i < wallets.length; i++){
        restrictedWallets[wallets[i]] = restricted;
    }
}
```

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.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	DDP	Decimal Division Precision	Unresolved
●	RSV	Redundant State Variables	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	StarWar.sol#L516
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
tokensForLiquidity += fees * sellLiquidityFee / sellTotalFees;
tokensForOperations += fees * sellOperationsFee / sellTotalFees;
tokensForRewards += fees * sellRewardsFee / sellTotalFees;
tokensForReserve += fees * sellReserveFee / sellTotalFees;
```

Recommendation

The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

RSV - Redundant State Variables

Criticality	Minor / Informative
Location	StarWar.sol#L415
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 state variables `buyLiquidityFee` and `buyTotalFees` contain always the same value. As a result, the deviation will always yield 1.

```
function updateBuyFees(uint256 _liquidityFee) external onlyOwner {
    buyLiquidityFee = _liquidityFee;
    buyTotalFees = buyLiquidityFee;
    require(buyTotalFees <= 2, "Must keep fees at 2% or less");
    emit UpdatedBuyFee(buyTotalFees);
}
...
tokensForLiquidity += fees * buyLiquidityFee / buyTotalFees;
```

Recommendation

The team is advised to take into consideration these segments 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. Some suggestions are:

- Remove the `buyLiquidityFee` since it is always the same value as `buyTotalFees`
- Remove the deviation since it always yields 1.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	StarWar.sol#L179,237,410,417,611,625,631,637,643
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.

```
function WETH() external pure returns (address);
mapping (address => bool) public _isExcludedMaxTransactionAmount
uint256 _liquidityFee
uint256 _operationsFee
uint256 _reserveFee
uint256 _rewardsFee
address _to
address _token
address _operationsAddress
address _rewardsAddress
address _reserveAddress
address _liquidityAddress
```

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	StarWar.sol#L378
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.

```
swapTokensAtAmount = newAmount
```

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.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	StarWar.sol#L506,511,512,513,514,515,520,521
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.

```
fees = amount * buyTotalFees / 100
tokensForLiquidity += fees * buyLiquidityFee / buyTotalFees
```

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.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	StarWar.sol#L271
Status	Unresolved

Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
uint256 totalSupply = 100 * 1e6 * 1e18
```

Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

Functions Analysis

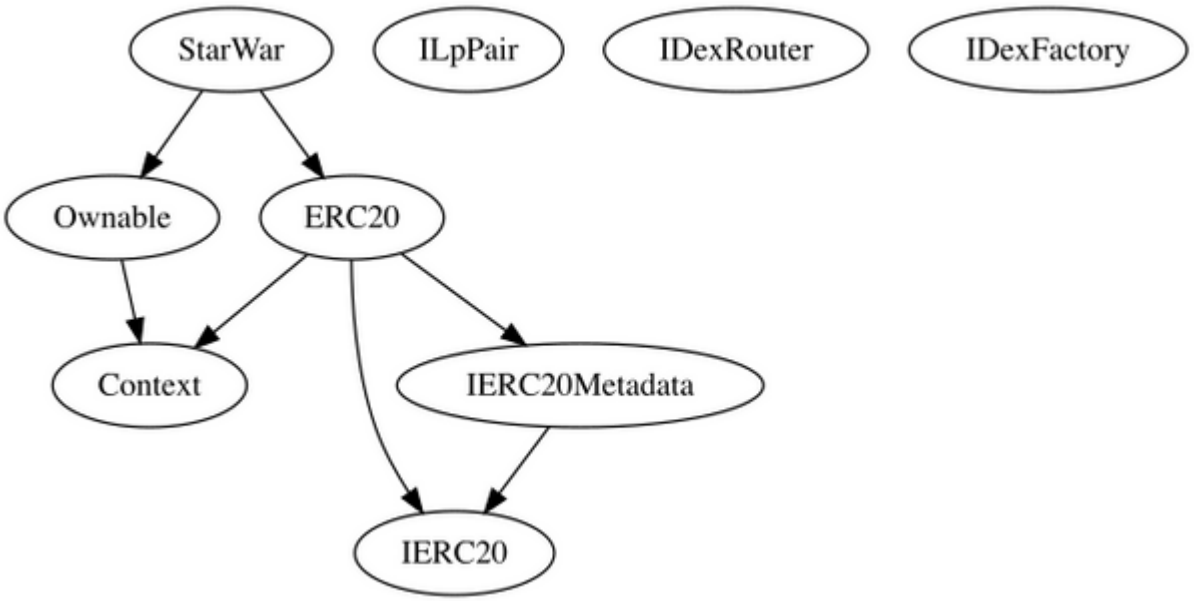
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-

	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_createInitialSupply	Internal	✓	
	_approve	Internal	✓	
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	External	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
ILpPair	Interface			
	sync	External	✓	-
IDexRouter	Interface			
	factory	External		-
	WETH	External		-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	addLiquidityETH	External	Payable	-

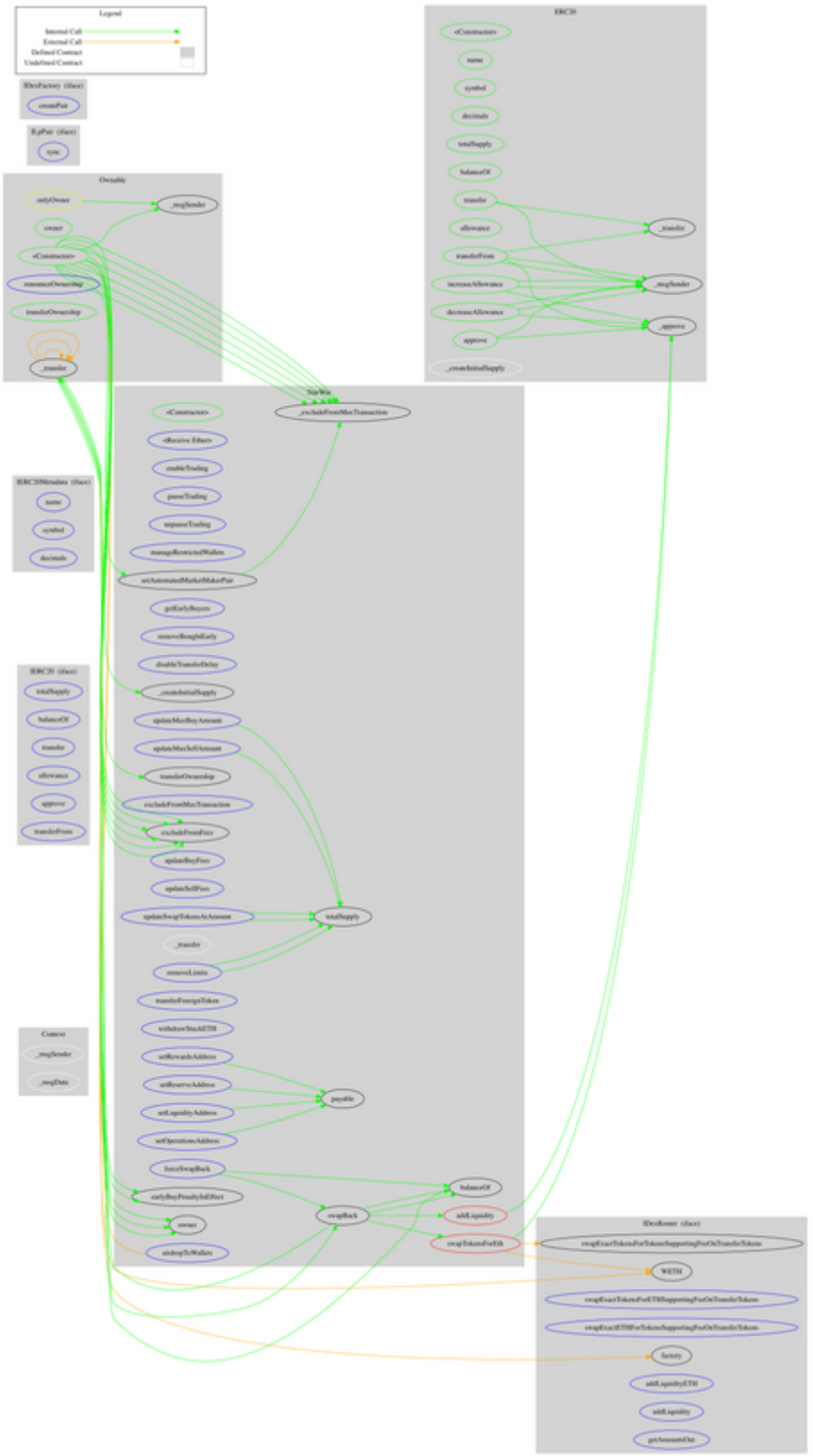
	addLiquidity	External	✓	-
	getAmountsOut	External		-
IDexFactory	Interface			
	createPair	External	✓	-
StarWar	Implementation	ERC20, Ownable		
		Public	✓	ERC20
		External	Payable	-
	enableTrading	External	✓	onlyOwner
	pauseTrading	External	✓	onlyOwner
	unpauseTrading	External	✓	onlyOwner
	manageRestrictedWallets	External	✓	onlyOwner
	removeLimits	External	✓	onlyOwner
	getEarlyBuyers	External		-
	removeBoughtEarly	External	✓	onlyOwner
	disableTransferDelay	External	✓	onlyOwner
	updateMaxBuyAmount	External	✓	onlyOwner
	updateMaxSellAmount	External	✓	onlyOwner
	updateSwapTokensAtAmount	External	✓	onlyOwner
	_excludeFromMaxTransaction	Private	✓	
	airdropToWallets	External	✓	onlyOwner
	excludeFromMaxTransaction	External	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	updateBuyFees	External	✓	onlyOwner
	updateSellFees	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	_transfer	Internal	✓	
	earlyBuyPenaltyInEffect	Public		-

	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	swapBack	Private	✓	
	transferForeignToken	External	✓	onlyOwner
	withdrawStuckETH	External	✓	onlyOwner
	setOperationsAddress	External	✓	onlyOwner
	setRewardsAddress	External	✓	onlyOwner
	setReserveAddress	External	✓	onlyOwner
	setLiquidityAddress	External	✓	onlyOwner
	forceSwapBack	External	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IDexFactory (iface)
createPair
IUpPair (iface)
sync
Ownable
onlyOwner
_msgSender
owner
<Constructor>
renounceOwnership
transferOwnership
_transfer
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
increaseAllowance
decreaseAllowance
approve
_createInitialSupply
_transfer
_msgSender
_approve
StarWar
_excludeFromMaxTransaction
<Constructor>
<Receive Ether>
enableTrading
pauseTrading
unpauseTrading
manageRestrictedWallets
setAutomatedMarketMakerPair
getEarlyBuyers
removeBoughtEarly
disableTransferDelay
_createInitialSupply
updateMaxBuyAmount
updateMaxSellAmount
transferOwnership
excludeFromMaxTransaction
excludeFromFees
updateBuyFees
updateSellFees
updateSwapTokensAtAmount
totalSupply
_transfer
removeLimits
transferForeignToken
withdrawStuckETH
setRewardsAddress
setReserveAddress
setLiquidityAddress
setOperationsAddress
payable
forceSwapBack
earlyBuyPenaltyInEffect
balanceOf
swapBack
addLiquidity
owner
swapTokensForEth
airdropToWallets
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData
IDexRouter (iface)
swapExactTokensForTokensSupportingFeeOnTransferTokens
WETH
swapExactTokensForETHSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
factory
addLiquidityETH
addLiquidity
getAmountsOut

Summary

There are some functions that can be abused by the owner like stop transactions and massively blacklist addresses. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. There is also a limit of max 20% fee is sales and 2% is bought.

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