Cyberscope

Audit Report

Arcadify

July 2023

SHA256 09dc84f141c9758bf372eab7c0b22bb1d0cb7202573532606bb6af15bc3c2149

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	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	DKO	Delete Keyword Optimization	Unresolved
●	FSA	Fixed Swap Address	Unresolved
●	FRV	Fee Restoration Vulnerability	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved

Table of Contents

Review

Testing Deploy	https://testnet.bscscan.com/address/0x2e70a06686ed84fb5468b01cecace0f53f95eb96

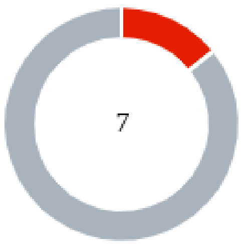
Audit Updates

Initial Audit	12 Jul 2023 https://github.com/cyberscope-io/audits/blob/main/7-arc/v1/audit.pdf
Corrected Phase 2	14 Jul 2023

Source Files

Filename	SHA256
contracts/arcadify1.sol	09dc84f141c9758bf372eab7c0b22bb1d0cb7202573532606bb6af15bc3c2149

Findings Breakdown



●	Critical	1
●	Medium	0
●	Minor / Informative	6

	Severity	Unresolved	Acknowledged	Resolved	Other
●	Critical	1	0	0	0
●	Medium	0	0	0	0
●	Minor / Informative	6	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	contracts/arcadify1.sol#L294,384,542,548
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 setting the `_maxTxAmount` or `_maxWalletSize` to a value close to `1000` which is a very small value compared with the total supply (`_tTotal = 100000000 * 10 ** 8`).

Additionally, the transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. However the owner can stop the transactions for all users by modifying the `tradingOpen` variable using the `setTrading` function.

As a result, the contract may operate as a honeypot, where funds are trapped and cannot be withdrawn by other participants.

```
 function _transfer(address from, address to, uint256 amount)
private {
  ...
  require(amount <= _maxTxAmount, "TOKEN: Max Transaction
Limit");
  ...
  if (to != uniswapV2Pair) {
      require(
         balanceOf(to) + amount < _maxWalletSize,
         "TOKEN: Balance exceeds wallet size!"
      );
  }
  ...
}
```

```
    function setMaxTxnAmount(uint256 maxTxAmount) public
onlyOwner {
        require(_maxTxAmount > 1000, "Max TX Amount needs to be
larger than 1000 tokens of the total supply");
        _maxTxAmount = maxTxAmount;
    }

    function setMaxWalletSize(uint256 maxWalletSize) public
onlyOwner {
        require(_maxWalletSize > 1000, "Max wallet size needs
to be larger than 1000 tokens of the total supply");
        _maxWalletSize = maxWalletSize;
    }
```

```
function setTrading(bool _tradingOpen) public onlyOwner {
   tradingOpen = _tradingOpen;
   launchBlock = block.number;
}
```

```
if (!tradingOpen) {
    require(
      from == owner(),
      "TOKEN: This account cannot send tokens until trading is
enabled"
      );
}
```

Recommendation

The contract could be adjusted to enhance the accurate check that prevents the setting of `_maxTxAmount` and `_maxWalletSize`. This could be achieved by not allowing these parameters to be set lower than 0.001% of the total supply. Specifically, the contract currently use a standard value of `1000` for the `require` check, which does not accurately represent 0.001% of the total supply.

Additionally, by modifying the `tradingOpen` variable through the `setTrading` function, the owner currently has the power to halt transactions for all users, potentially turning the contract into a honeypot. The team should consider implementing a more transparent and decentralized control mechanism over transactions. This could involve limiting the owner's ability to disable and enable transactions arbitrarily.

Furthermore, 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.

DKO - Delete Keyword Optimization

Criticality	Minor / Informative
Location	contracts/arcadify1.sol#L265
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.

```
function removeAllFee() private {
    if (_redisFee == 0 && _taxFee == 0) return;

    _previousredisFee = _redisFee;
    _previoustaxFee = _taxFee;

    _redisFee = 0;
    _taxFee = 0;
}
```

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.

FSA - Fixed Swap Address

Criticality	Minor / Informative
Location	contracts/arcadify1.sol#L181
Status	Unresolved

Description

The swap address is assigned once and it can not be changed. It is a common practice in decentralized exchanges to create new swap versions. A contract that cannot change the swap address may not be able to catch up to the upgrade. As a result, the contract will not be able to migrate to a new liquidity pool pair or decentralized exchange.

```
constructor() {
    ...
    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(
        0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D // uniswap
    );
    uniswapV2Router = _uniswapV2Router;
    uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());
    ...
}
```

Recommendation

The team is advised to add the ability to change the pair and router address in order to cover potential liquidity pool migrations. It would be better to support multiple pair addresses so the token will be able to have the same behavior in all the decentralized liquidity pairs.

FRV - Fee Restoration Vulnerability

Criticality	Minor / Informative
Location	contracts/arcadify1.sol#L265,275,413
Status	Unresolved

Description

The contract demonstrates a potential vulnerability upon removing and restoring the fees. This vulnerability can occur when the fees have been set to zero. During a transaction, if the fees have been set to zero, then both remove fees and restore fees functions will be executed. The remove fees function is executed to temporarily remove the fees, ensuring the sender is not taxed during the transfer. However, the function prematurely returns without setting the variables that hold the previous fee values.

As a result, when the subsequent restore fees function is called after the transfer, it restores the fees to their previous values. However, since the previous fee values were not properly set to zero, there is a risk that the fees will retain their non-zero values from before the fees were removed. This can lead to unintended consequences, potentially causing incorrect fee calculations or unexpected behavior within the contract.

```
/**
 * The given code segment is just an example of this
vulnerability, where the
 * `removeAllFee()` function does not set the previous fee
variables if both
 * `_redisFee` and `_taxFee` are zero. Then, the
`restoreAllFee()` function
 * will modify the `_redisFee` and `_taxFee` to the values the
previous fees
 * hold, which most likely will not be zero.
 */

function removeAllFee() private {
    if (_redisFee == 0 && _taxFee == 0) return;

    _previousredisFee = _redisFee;
    _previoustaxFee = _taxFee;

    _redisFee = 0;
    _taxFee = 0;
}

function restoreAllFee() private {
    _redisFee = _previousredisFee;
    _taxFee = _previoustaxFee;
}

function _tokenTransfer(
    address sender,
    address recipient,
    uint256 amount,
    bool takeFee
) private {
    if (!takeFee) removeAllFee();
    _transferStandard(sender, recipient, amount);
    if (!takeFee) restoreAllFee();
}
...
```

Recommendation

The team is advised to modify the remove fees function to ensure that the previous fee values are correctly set to zero, regardless of their initial values. A recommended approach would be to remove the early return when both fees are zero.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	contracts/arcadify1.sol#L529,541,547
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
    function setMinSwapTokensThreshold(
        uint256 swapTokensAtAmount
    ) public onlyOwner {
        _swapTokensAtAmount = swapTokensAtAmount;
    }

    function setMaxTxnAmount(uint256 maxTxAmount) public
onlyOwner {
        require(_maxTxAmount > 0, "Max TX Amount needs to be
larger than 0");
        _maxTxAmount = maxTxAmount;
    }

    function setMaxWalletSize(uint256 maxWalletSize) public
onlyOwner {
        require(_maxWalletSize > 0, "Max wallet size needs to
be larger than 0");
        _maxWalletSize = maxWalletSize;
    }
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/arcadify1.sol#L107,124,125,126,132,139,147,154,156,167,168,169,384,536
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);
string private constant _name = "Arcadify"
string private constant _symbol = "ARC"
uint8 private constant _decimals = 8
uint256 private constant _tTotal = 100000000 * 10 ** 8
uint256 public _taxFeeOnBuy = 5
uint256 public _taxFee = _taxFeeOnSell
...
        payable(0x5B10C241ee81fb51241fb7d80586614bd021125B)

address payable private constant _marketingAddress =
        payable(0x9F67D213eCF40D4d339e105F71D6cA47c287FCb6)
uint256 public _maxTxAmount = 100000000 * 10 ** 8
uint256 public _maxWalletSize = 100000000 * 10 ** 8
uint256 public _swapTokensAtAmount = 50000 * 10 ** 8

...
```

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	contracts/arcadify1.sol#L521,532,543,549
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.

```
_redisFeeOnBuy = redisFeeOnBuy
_swapTokensAtAmount = swapTokensAtAmount
_maxTxAmount = maxTxAmount
_maxWalletSize = maxWalletSize
```

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.

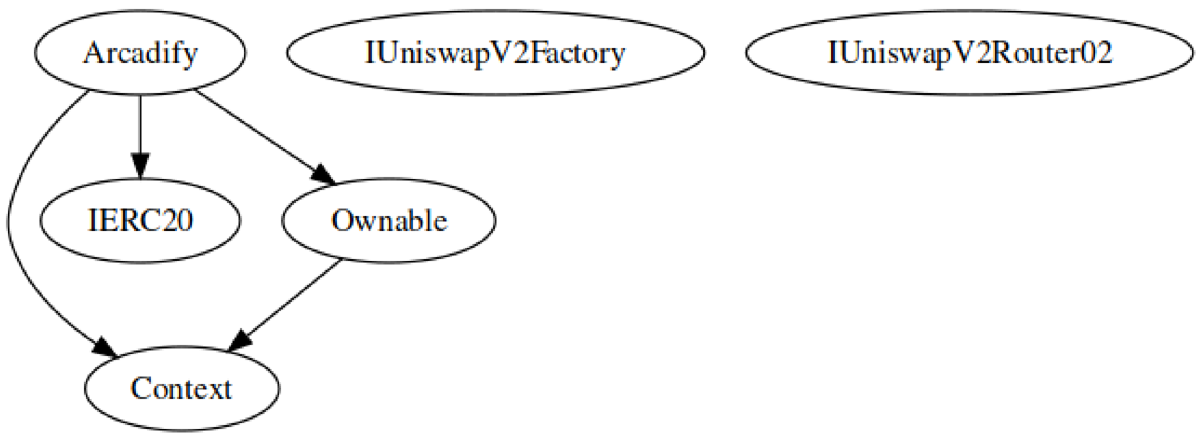
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
IUniswapV2Factory	Interface			

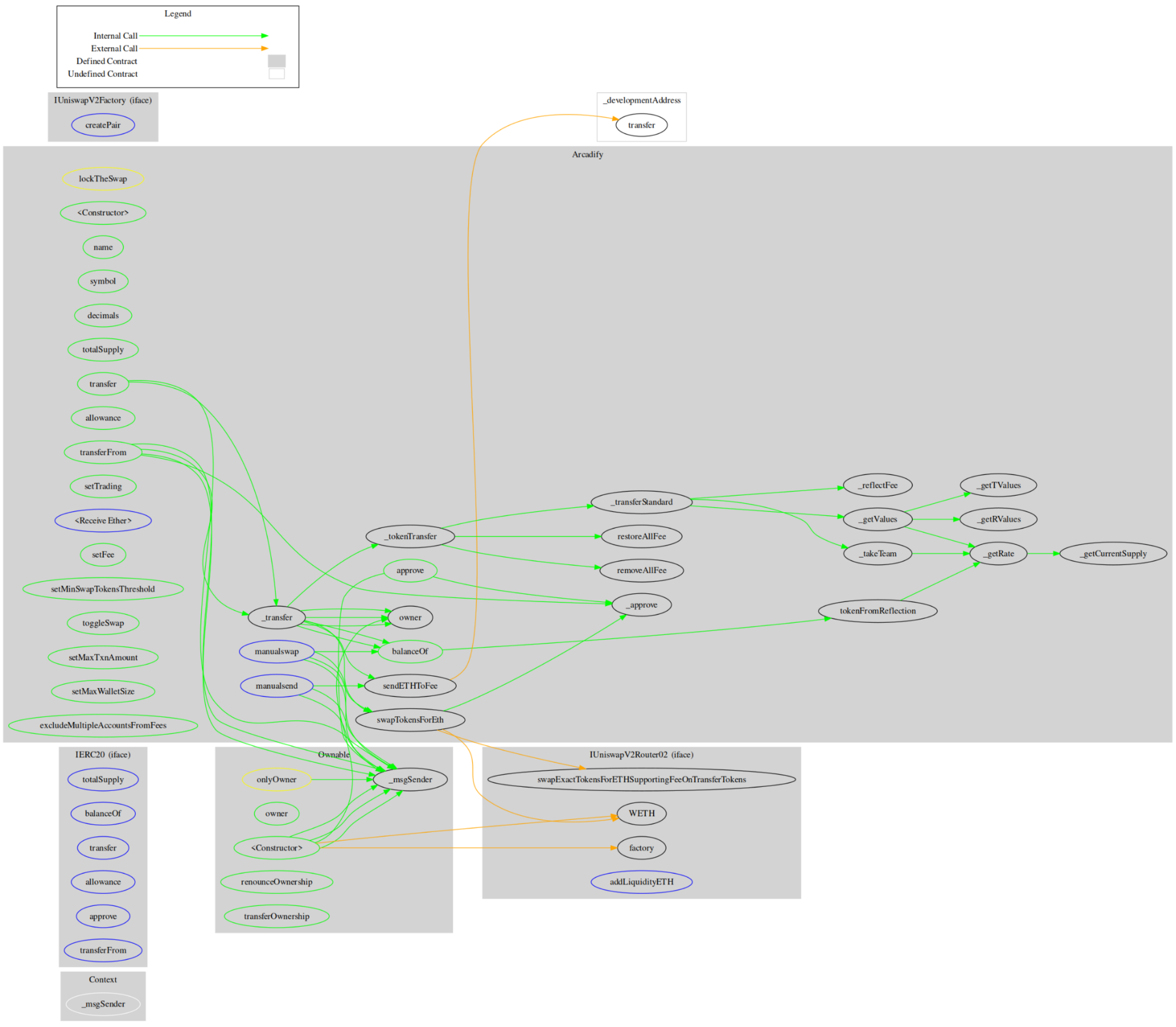
	createPair	External	✓	-
IUniswapV2Router02	Interface			
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
Arcadify	Implementation	Context, IERC20, Ownable		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	tokenFromReflection	Private		
	removeAllFee	Private	✓	
	restoreAllFee	Private	✓	

	_approve	Private	✓	
	_transfer	Private	✓	
	swapTokensForEth	Private	✓	lockTheSwap
	sendETHToFee	Private	✓	
	setTrading	Public	✓	onlyOwner
	manualswap	External	✓	-
	manualsend	External	✓	-
	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_takeTeam	Private	✓	
	_reflectFee	Private	✓	
		External	Payable	-
	_getValues	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	setFee	Public	✓	onlyOwner
	setMinSwapTokensThreshold	Public	✓	onlyOwner
	toggleSwap	Public	✓	onlyOwner
	setMaxTxnAmount	Public	✓	onlyOwner
	setMaxWalletSize	Public	✓	onlyOwner
	excludeMultipleAccountsFromFees	Public	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapV2Factory (iface)
createPair
_developmentAddress
transfer
Arcadify
lockTheSwap
<Constructor>
name
symbol
decimals
totalSupply
transfer
allowance
transferFrom
setTrading
<Receive Ether>
setFee
setMinSwapTokensThreshold
toggleSwap
setMaxTxnAmount
setMaxWalletSize
excludeMultipleAccountsFromFees
_transfer
manualswap
manualsend
_tokenTransfer
approve
owner
balanceOf
sendETHToFee
swapTokensForEth
_transferStandard
restoreAllFee
removeAllFee
_approve
_reflectFee
_getValues
_takeTeam
tokenFromReflection
_getTValues
_getRValues
_getRate
_getCurrentSupply
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
Ownable
onlyOwner
owner
<Constructor>
renounceOwnership
transferOwnership
_msgSender
IUniswapV2Router02 (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory
addLiquidityETH

Summary

Arcadify 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. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract will eliminate all the contract threats. There is also a limit of max 25% fees.

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