Cyberscope

Audit Report

GBURN

November 2023

Network BSC

Address 0x0f5da998ca0759162fdd0fb38de8b06144ead75d

Audited by © cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OTUT	Transfers User's Tokens	Unresolved
●	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
Critical	USF	Unlocked Swap Functionality	Unresolved
Minor / Informative	MEE	Missing Events Emission	Unresolved
Minor / Informative	RSW	Redundant Storage Writes	Unresolved
Minor / Informative	EIS	Excessively Integer Size	Unresolved
Minor / Informative	IDI	Immutable Declaration Improvement	Unresolved
Minor / Informative	L04	Conformance to Solidity Naming Conventions	Unresolved
Minor / Informative	L14	Uninitialized Variables in Local Scope	Unresolved
Minor / Informative	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	GBURN
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://bscscan.com/address/0x0f5da998ca0759162fdd0fb38de8b06144ead75d
Address	0x0f5da998ca0759162fdd0fb38de8b06144ead75d
Network	BSC
Symbol	GBURN
Decimals	18
Total Supply	100,000,000

Audit Updates

Initial Audit	10 Nov 2023

Source Files

Filename	SHA256
contracts/GburnFixed.sol	ced4204e9f691b7301abe7a8302bf847c543b1fa3fd4757dd52a10ef044ff4e2
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a

@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/token/ERC20/ERC20.sol	bce14c3fd3b1a668529e375f6b70ffdf9cef8c4e410ae99608be5964d98fa701
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249aa4df6024fbfaa94f79ab2f44f3231

Findings Breakdown

Critical	2
Medium	0
Minor / Informative	7

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	2	0	0	0
Medium	0	0	0	0
Minor / Informative	7	0	0	0

OTUT - Transfers User's Tokens

Criticality	Critical
Location	contracts/GburnFixed.sol#L116
Status	Unresolved

Description

The contract contains the `transfer` function which is intended to handle token transfers with an associated fee. This fee, calculated as `feeAmount`, is meant to be transferred to the contract's address as a form of fee deduction. However, the contract deducts the `feeAmount` from the recipient of the transfer, not from the sender. Spefically the `recipient` is incorrectly specified as the source of the fee transfer. As a result, the `recipient` of the transfer, instead of receiving only the `amountAfterFee`, is forced to transfer additionally the `feeAmount` to the contract.

```
 function transfer(address recipient, uint256 amount)
        public
        override
        returns (bool)
    {
        ...
        uint256 feeAmount = (amount * FEE_PERCENTAGE) / 100;
        uint256 amountAfterFee = amount - feeAmount;

        if (isGBURNPair(_msgSender()) ||
isGBURNPair(recipient)) {
            // If buy or sell, apply fee
            super.transfer(recipient, amountAfterFee);
            super._transfer(recipient, address(this),
feeAmount);
            _convertTokenToBNB(feeAmount);
        } else {
        ...
    }
```

Recommendation

It is recommended to alter the fee deduction mechanism so that the fees are sent to the contract from the sender of the transfer, instead of the `recipient`. The corrected logic

should ensure that the `amountAfterFee` is transferred to the `recipient`, and the `feeAmount` is separately transferred from the `sender` to the contract's address. Such an approach will align the contract's functionality, preserve the integrity of transaction values as intended by the senders, and maintain transparency and fairness in fee deductions.

USF - Unlocked Swap Functionality

Criticality	Critical
Location	contracts/GburnFixed.sol#L128
Status	Unresolved

Description

The contract is designed to perform a token swap operation through the `_convertTokenToBNB` function, which is called within the `transfer` function. This function swaps a specified amount of tokens using the Pancake Swap Router. However, the current implementation does not include a locking mechanism during the swap operation. This absence of a lock could lead to an infinite loop, causing the contract to be stuck and consume all available gas.

```
    function transfer(address recipient, uint256 amount)
        public
        override
        returns (bool)
    {
        ...
            _convertTokenToBNB(feeAmount);
        ...
    }

    function _convertTokenToBNB(uint256 tokenAmount) private {
        // Generate the pancakeSwap pair path of token -> WBNB
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = pancakeSwapRouter.WETH();

        _approve(address(this), address(pancakeSwapRouter),
tokenAmount);

        // Make the swap

pancakeSwapRouter.swapExactTokensForETHSupportingFeeOnTransferTo
kens(
            tokenAmount,
            0, // Accept any amount of BNB
            path,
            feeReceiver,
            block.timestamp
        );
    }
```

Recommendation

It is recommended to lock the swap operation during the transaction. This can be achieved by introducing a state variable that locks the swap operation when the `_convertTokenToBNB` function is called and unlocks it once the operation is complete. This would ensure that the swap operation cannot be called again until the first swap operation has finished, preventing potential infinite loops.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	contracts/GburnFixed.sol#L155
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 setExcludedFromFees(address account, bool excluded)
        public
        onlyOwner
    {
        require(
            account != address(0),
            "GBURN: exclude from fees the zero address"
        );
        excludedFromFees[account] = excluded;
    }
```

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.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	contracts/GburnFixed.sol#L155
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
function setExcludedFromFees(address account, bool excluded)
    public
    onlyOwner
{
    require(
        account != address(0),
        "GBURN: exclude from fees the zero address"
    );
    excludedFromFees[account] = excluded;
}
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

EIS - Excessively Integer Size

Criticality	Minor / Informative
Location	contracts/GburnFixed.sol#L62
Status	Unresolved

Description

The contract is using a bigger unsigned integer data type that the maximum size that is required. By using an unsigned integer data type larger than necessary, the smart contract consumes more storage space and requires additional computational resources for calculations and operations involving these variables. This can result in higher transaction costs, longer execution times, and potential scalability bottlenecks.

```
uint256 public constant FEE_PERCENTAGE = 3;
```

Recommendation

To address the inefficiency associated with using an oversized unsigned integer data type, it is recommended to accurately determine the required size based on the range of values the variable needs to represent.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	contracts/GburnFixed.sol#L72
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
feeReceiver
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/GburnFixed.sol#L17,49
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);
function INIT_CODE_PAIR_HASH() external view returns (bytes32);
```

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.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	contracts/GburnFixed.sol#L85,86
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
address token0Address
address token1Address
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/GburnFixed.sol#L3
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.

```
pragma solidity ^0.8.19;
```

Recommendation

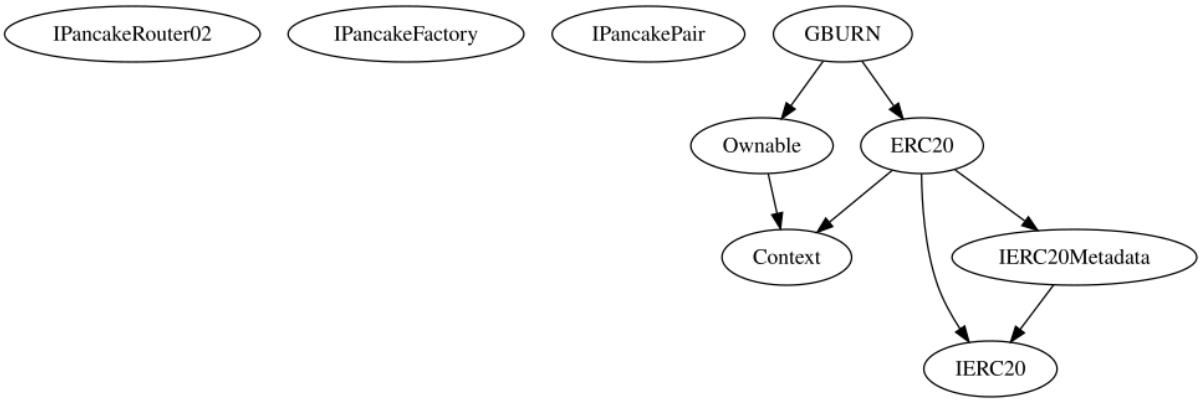
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest 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.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IPancakeRouter02	Interface			
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	WETH	External		-
IPancakeFactory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
	INIT_CODE_PAIR_HASH	External		-
IPancakePair	Interface			
	token0	External		-
	token1	External		-

GBURN	Implementation	ERC20, Ownable		
		Public	✓	ERC20
	isGBURNPair	Internal		
	transfer	Public	✓	-
	_convertTokenToBNB	Private	✓	
		External	Payable	-
	withdrawBNB	Public	✓	onlyOwner
	setExcludedFromFees	Public	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20Metadata (iface)
name
symbol
decimals
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transferFrom
transfer
_spendAllowance
decreaseAllowance
increaseAllowance
approve
_mint
_burn
_transfer
type
allowance
_approve
_msgSender
_beforeTokenTransfer
_afterTokenTransfer
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData
IPancakePair (iface)
token0
token1
IPancakeFactory (iface)
feeTo
feeToSetter
getPair
allPairs
allPairsLength
createPair
setFeeTo
setFeeToSetter
INIT_CODE_PAIR_HASH
GBURN
transferOwnership
<Constructor>
decimals
_mint
transfer
_msgSender
<Receive Ether>
_convertTokenToBNB
withdrawBNB
isGBURNPair
setExcludedFromFees
owner
_approve
IPancakeRouter02 (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
Ownable
_transfer
transfer
onlyOwner
_checkOwner
<Constructor>
renounceOwnership
transferOwnership
owner
_msgSender
_transferOwnership

Summary

GBURN contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused like incorrect user’s tokens transfers. 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 3% fee on buy and sell transactions.

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