Cyberscope

Audit Report

Galaxy Fox

January 2024

Repository	https://github.com/humanshield89/galaxy-fox-token
Commit	b9de5008ec8d874fb92685bf6e78a06c0277510b
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	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
Medium	IFR	Ineffective Function Restriction	Acknowledged
Minor	MEE	Missing Events Emission	Acknowledged
Minor	PLPI	Potential Liquidity Provision Inadequacy	Acknowledged
Minor	PVC	Price Volatility Concern	Acknowledged
Minor	RML	Redundant Mutex Locking	Acknowledged
Minor	RSW	Redundant Storage Writes	Acknowledged
Minor	RC	Repetitive Calculations	Acknowledged
Minor	L04	Conformance to Solidity Naming Conventions	Acknowledged
Minor	L07	Missing Events Arithmetic	Acknowledged
Minor	L16	Validate Variable Setters	Acknowledged
Minor	L19	Stable Compiler Version	Acknowledged
Minor	L20	Succeeded Transfer Check	Acknowledged

Table of Contents

Review

Contract Name	GalaxyFox
Repository	https://github.com/humanshield89/galaxy-fox-token
Commit	b9de5008ec8d874fb92685bf6e78a06c0277510b
Testing Deploy	https://mumbai.polygonscan.com/address/0x441f6ad303de9fb0bec66aebe436df3d91e1aa51
Symbol	GFOX
Decimals	18
Total Supply	5,000,000,000
Badge Eligibility	Yes

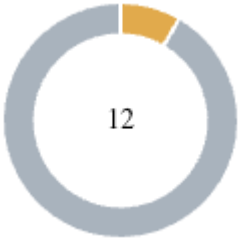
Audit Updates

Initial Audit	23 Jan 2024

Source Files

Filename	SHA256
contracts/GalaxyFoxToken.sol	bcba21d781b573216329a4765f4e633cace119f22cbf38da03c0e0c0d673b3d8
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol	a2900701961cb0b6152fc073856b972564f7c798797a4a044e83d2ab8f0e8d38
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol	0439ffe0fd4a5e1f4e22d71ddbda76d63d61679947d158cba4ee0a1da60cf663
@openzeppelin/contracts/utils/Context.sol	847fda5460fee70f56f4200f59b82ae622bb03c79c77e67af010e31b7e2cc5b6
@openzeppelin/contracts/token/ERC20/IERC20.sol	6f2faae462e286e24e091d7718575179644dc60e79936ef0c92e2d1ab3ca3cee
@openzeppelin/contracts/token/ERC20/ERC20.sol	2d874da1c1478ed22a2d30dcf1a6ec0d09a13f897ca680d55fb49fbcc0e0c5b1
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	1d079c20a192a135308e99fa5515c27acfbb071e6cdb0913b13634e630865939
@openzeppelin/contracts/interfaces/draft-IERC6093.sol	4aea87243e6de38804bf8737bf86f750443d3b5e63dd0fd0b7ad92f77cdbd3e3
@openzeppelin/contracts/access/Ownable.sol	38578bd71c0a909840e67202db527cc6b4e6b437e0f39f0c909da32c1e30cb81

Findings Breakdown



Critical	0
Medium	1
Minor / Informative	11

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	1	0	0
Minor / Informative	0	11	0	0

IFR - Ineffective Function Restriction

Criticality	Medium
Location	contracts/GalaxyFoxToken.sol#L364,368
Status	Acknowledged

Description

The `liquify` function is designed to trigger the token swap functionality manually and can only be called by the contract owner. However, the internal function `_liquify` is marked as public, enabling any user to execute a manual token swap. This discrepancy renders the ownership restriction in the liquify function ineffective, as the internal logic can be accessed by any external user.

Additionally, this discrepancy in access control raises a security concern. It unintentionally grants any user the ability to execute a manual token swap, potentially enabling price arbitrage opportunities when combined with buy or sell transactions.

```
function liquify() external onlyOwner {
    _liquify();
}

function _liquify() public {
    ...
}
```

Recommendation

The team is advised to update the `_liquify` function to have the public visibility modifier only if it is meant to be called by external users. If the intention is to keep `_liquify` as an internal function, the team could change its visibility to internal or private accordingly. The team should ensure that the visibility modifier aligns with the intended access control for the token swap functionality.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	contracts/GalaxyFoxToken.sol#L195,217,226,237,248,259,339
Status	Acknowledged

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.

```
taxEnabled = taxEnabledArg;
isExcludedFromFee[account] = excluded;
isPair[pair] = isPairArg;
ecosystemHolder = _ecosystemHolder;
marketingHolder = _marketingHolder;
liquidityHolder = _liquidityHolder;
isExcludedFromDailyVolume[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.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	contracts/GalaxyFoxToken.sol#L394
Status	Acknowledged

Description

The contract operates under the assumption that liquidity is consistently provided to the pair between the contract's token and the native currency. However, there is a possibility that liquidity is provided to a different pair. This inadequacy in liquidity provision in the main pair could expose the contract to risks. Specifically, during eligible transactions, where the contract attempts to swap tokens with the main pair, a failure may occur if liquidity has been added to a pair other than the primary one. Consequently, transactions triggering the swap functionality will result in a revert.

```
function _swapTokensForEth(uint256 tokenAmount) internal {
    // generate the uniswap pair path of token -> weth
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = weth;

    // make the swap
    uniRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
        tokenAmount,
        0, // accept any amount of ETH
        path,
        address(this),
        block.timestamp
    );
}
```

Recommendation

The team is advised to implement a runtime mechanism to check if the pair has adequate liquidity provisions. This feature allows the contract to omit token swaps if the pair does not have adequate liquidity provisions, significantly minimizing the risk of potential failures.

Furthermore, the team could ensure the contract has the capability to switch its active pair in case liquidity is added to another pair.

Additionally, the contract could be designed to tolerate potential reverts from the swap functionality, especially when it is a part of the main transfer flow. This can be achieved by executing the contract's token swaps in a non-reversible manner, thereby ensuring a more resilient and predictable operation.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	contracts/GalaxyFoxToken.sol#L202
Status	Acknowledged

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `miniBeforeLiquify` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
function setMiniBeforeLiquify(
    uint256 miniBeforeLiquifyArg
) public onlyOwner {
    miniBeforeLiquify = miniBeforeLiquifyArg;
}
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

RML - Redundant Mutex Locking

Criticality	Minor / Informative
Location	contracts/GalaxyFoxToken.sol#L369,370
Status	Acknowledged

Description

The contract employs the `inswap` variable as a mutex to prevent the concurrent execution of the token swap process. However, the functionality inside the mutex is dependent on the value of `liquidityReserves` being greater than `miniBeforeLiquify`. Consequently, when the condition not is met, the contract redundantly locks the swapping process using the mutex.

```
if (inswap == 1) return;
inswap = 1;
if (liquidityReserves > miniBeforeLiquify) {
    ...
}
inswap = 0;
```

Recommendation

The team is advised to move the mutex locking inside the if-statement. That way the contract will active its guarding mechanism only when the condition is met.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	contracts/GalaxyFoxToken.sol#L195,217,226,339
Status	Acknowledged

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.

```
taxEnabled = taxEnabledArg;
isExcludedFromFee[account] = excluded;
isPair[pair] = isPairArg;
isExcludedFromDailyVolume[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.

RC - Repetitive Calculations

Criticality	Minor / Informative
Location	contracts/GalaxyFoxToken.sol#L182,184
Status	Acknowledged

Description

The contract contains segments with multiple occurrences of the same calculation being performed. The calculation is repeated without utilizing a variable to store its result, which leads to redundant code, hinders code readability, and increases gas consumption. Each repetition of the calculation requires computational resources and can impact the performance of the contract, especially if the calculation is resource-intensive.

```
volume[sender][block.timestamp / DAY] += amount;
require(
    volume[sender][block.timestamp / DAY] <= maxDailyVolume ||
        isExcludedFromDailyVolume[sender],
    "GalaxyFox: max daily volume exceeded"
);
```

Recommendation

To address this finding and enhance the efficiency and maintainability of the contract, it is recommended to refactor the code by assigning the calculation result to a variable once and then utilizing that variable throughout the method. By storing the calculation result in a variable, the contract eliminates the need for redundant calculations and optimizes code execution.

Refactoring the code to assign the calculation result to a variable has several benefits. It improves code readability by making the purpose and intent of the calculation explicit. It also reduces code redundancy, making the method more concise, easier to maintain, and gas effective. Additionally, by performing the calculation once and reusing the variable, the contract improves performance by avoiding unnecessary computations.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/GalaxyFoxToken.sol#L234,245,256,269,270,271,287,288,289,368
Status	Acknowledged

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.

```
address payable _ecosystemHolder
address payable _marketingHolder
address payable _liquidityHolder
uint16 _liquidity
uint16 _marketing
uint16 _ecosystem

...
```

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/GalaxyFoxToken.sol#L205,352
Status	Acknowledged

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.

```
miniBeforeLiquify = miniBeforeLiquifyArg
maxDailyVolume = maxDailyVolumeArg
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	contracts/GalaxyFoxToken.sol#L69,70,71
Status	Acknowledged

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.

```
ecosystemHolder = _ecosystemHolder
marketingHolder = _marketingHolder
liquidityHolder = _liquidityHolder
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/GalaxyFoxToken.sol#L2
Status	Acknowledged

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 ^0.8.20;
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	contracts/GalaxyFoxToken.sol#L317
Status	Acknowledged

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
IERC20(tokenAddress).transfer(msg.sender, tokenAmount)
```

Recommendation

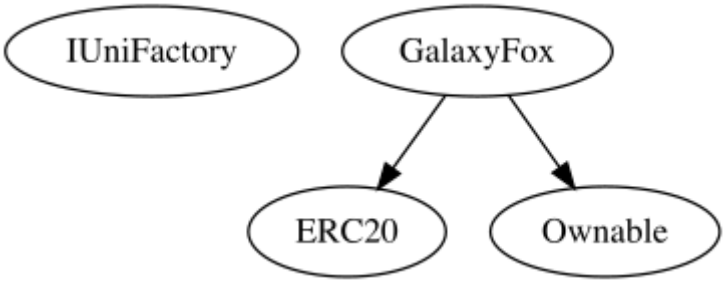
The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

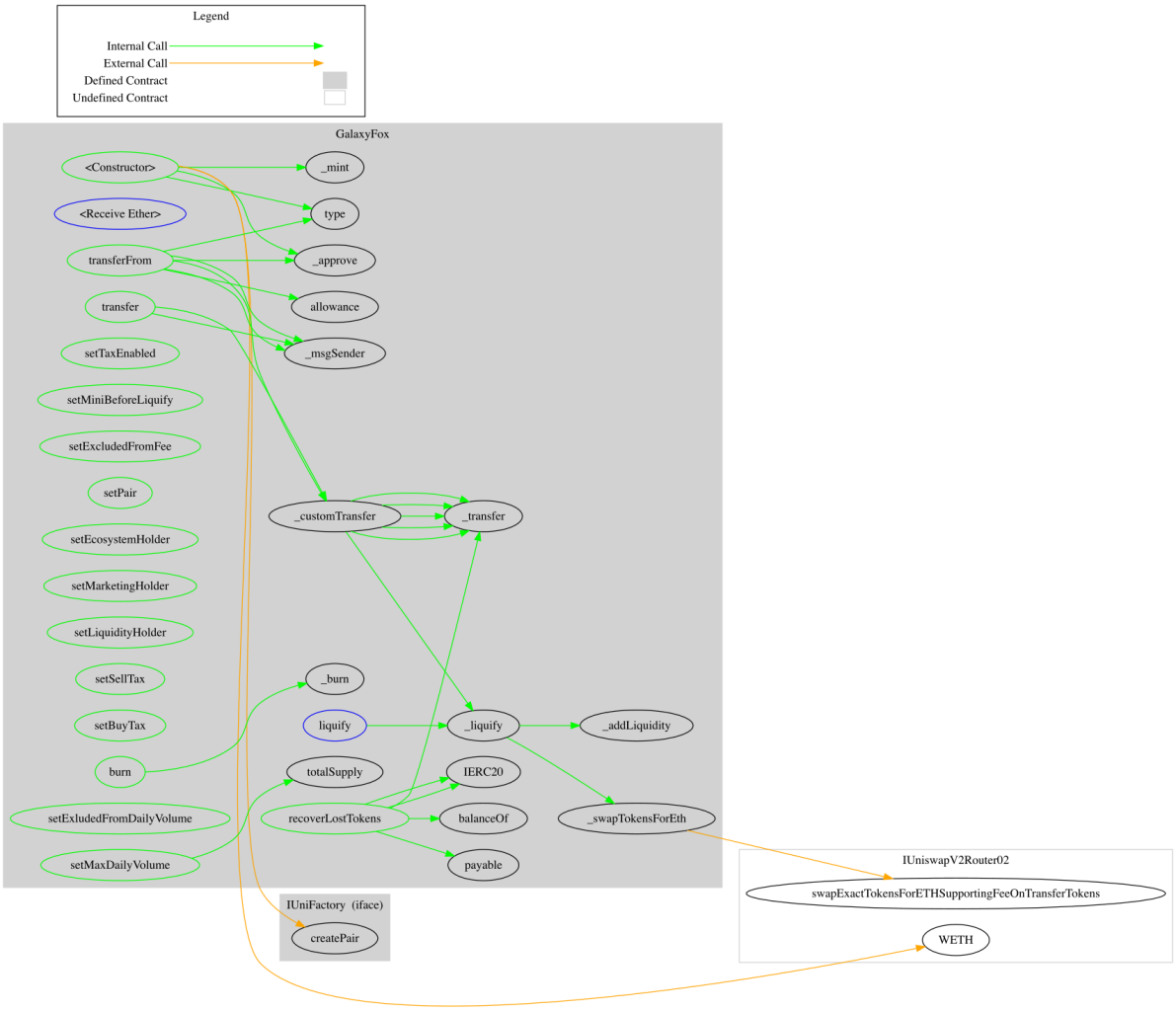
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IUniFactory	Interface			
	createPair	External	✓	-
GalaxyFox	Implementation	ERC20, Ownable		
		Public	✓	ERC20 Ownable
		External	Payable	-
	transfer	Public	✓	-
	transferFrom	Public	✓	-
	_customTransfer	Internal	✓	
	setTaxEnabled	Public	✓	onlyOwner
	setMiniBeforeLiquify	Public	✓	onlyOwner
	setExcludedFromFee	Public	✓	onlyOwner
	setPair	Public	✓	onlyOwner
	setEcosystemHolder	Public	✓	onlyOwner
	setMarketingHolder	Public	✓	onlyOwner
	setLiquidityHolder	Public	✓	onlyOwner
	setSellTax	Public	✓	onlyOwner
	setBuyTax	Public	✓	onlyOwner

	burn	Public	✓	-
	recoverLostTokens	Public	✓	onlyOwner
	setExludedFromDailyVolume	Public	✓	onlyOwner
	setMaxDailyVolume	Public	✓	onlyOwner
	liquify	External	✓	onlyOwner
	_liquify	Public	✓	-
	_swapTokensForEth	Internal	✓	
	_addLiquidity	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
GalaxyFox
<Constructor>
<Receive Ether>
transferFrom
transfer
setTaxEnabled
setMiniBeforeLiquify
setExcludedFromFee
setPair
setEcosystemHolder
setMarketingHolder
setLiquidityHolder
setSellTax
setBuyTax
burn
setExludedFromDailyVolume
setMaxDailyVolume
_mint
type
_approve
allowance
_msgSender
_customTransfer
_transfer
_burn
liquify
_liquify
_addLiquidity
totalSupply
IERC20
recoverLostTokens
balanceOf
_swapTokensForEth
payable
IUniFactory (iface)
createPair
IUniswapV2Router02
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH

Summary

Galaxy Fox contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Galaxy Fox is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 20% buy and sell fees.

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