Cyberscope

Audit Report

Pepe Halloween Floki

October 2023

Network	ETH
Address	0xd1348c9Ad42644fF3301A87A378842Bc35985931
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
●	CCC	Contradictory Condition Checks	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	RRS	Redundant Require Statement	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved

Table of Contents

Review

Contract Name	PepeHalloweenFloki
Compiler Version	v0.8.21+commit.d9974bed
Optimization	200 runs
Explorer	https://etherscan.io/address/0xd1348c9ad42644ff3301a87a378842bc35985931
Address	0xd1348c9ad42644ff3301a87a378842bc35985931
Network	ETH
Symbol	WITCH
Decimals	18
Total Supply	666,000,000,000

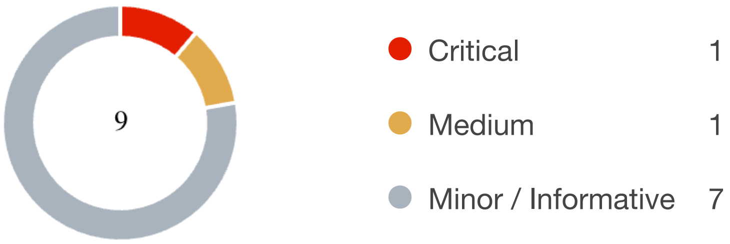
Audit Updates

Initial Audit	23 Oct 2023

Source Files

Filename	SHA256
PepeHalloweenFloki.sol	de29e0d0ef867e6f17625c5bfe6020220038a02e71b1652081519481bf53b721

Findings Breakdown



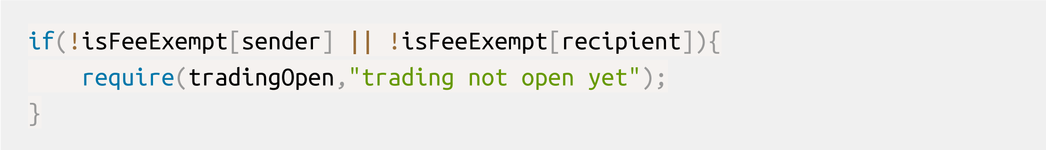
Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	1	0	0	0
●	Medium	1	0	0	0
●	Minor / Informative	7	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	PepeHalloweenFloki.sol#L199
Status	Unresolved

Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. Once the transactions are enabled the owner will not be able to disable them again.



```
if(!isFeeExempt[sender] || !isFeeExempt[recipient]){
    require(tradingOpen,"trading not open yet");
}
```

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 multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

CCC - Contradictory Condition Checks

Criticality	Medium
Location	PepeHalloweenFloki.sol#L295,296
Status	Unresolved

Description

The `setSwapBackSettings` function includes contradictory condition checks for the `_denominator` argument. Specifically, it requires `_denominator` to be both less than or equal to 100 and greater than or equal to 10,000 simultaneously. These conditions cannot be met together, leading to a situation where the function cannot be called with any valid value of `_denominator`. This is contradictory and the contract owner will not be able to modify the `swapEnabled` or `swapThreshold` variables.

```
function setSwapBackSettings(bool _enabled, uint256 _denominator) external
onlyOwner {
    require(_denominator <= 100, "CA sell amount too high");
    require(_denominator >= 10000, "CA sell amount too low");
    swapEnabled = _enabled;
    swapThreshold = totalSupply / _denominator;
}
```

Recommendation

To resolve this issue, the team should carefully review the intended range for `_denominator` and ensure the conditions align with the contract's requirements. If the intention is to limit the range, choose a specific range that makes sense for the contract's functionality and update the conditions accordingly. If a wide range is acceptable, the team may remove one of the conflicting conditions. The choice depends on the contract's requirements and objectives.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	PepeHalloweenFloki.sol#L192,260,289,290,291
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.

```
_maxWalletToken = (totalSupply * _newmaxwallet ) / 100;
tradingOpen = true;
sellMultiplier = _sell;
buyMultiplier = _buy;
transferMultiplier = _trans;
```

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.

RRS - Redundant Require Statement

Criticality	Minor / Informative
Location	PepeHalloweenFloki.sol#L16
Status	Unresolved

Description

The contract utilizes a `require` statement within the `add` function aiming to prevent overflow errors. This function is designed based on the SafeMath library's principles. In Solidity version 0.8.0 and later, arithmetic operations revert on overflow and underflow, making the overflow check within the function redundant. This redundancy could lead to extra gas costs and increased complexity without providing additional security.

```
function add(uint256 a, uint256 b) internal pure returns (uint256) {
      uint256 c = a + b;
      require(c >= a, "SafeMath: addition overflow");

      return c;
}
```

Recommendation

It is recommended to remove the `require` statement from the add function since the contract is using a Solidity pragma version equal to or greater than 0.8.0. By doing so, the contract will leverage the built-in overflow and underflow checks provided by the Solidity language itself, simplifying the code and reducing gas consumption. This change will uphold the contract's integrity in handling arithmetic operations while optimizing for efficiency and cost-effectiveness.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	PepeHalloweenFloki.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.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	PepeHalloweenFloki.sol#L143,149
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.

```
router
marketingFeeReceiver
```

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.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	PepeHalloweenFloki.sol#L122
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.

```
public totalFee = 1;
```

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	PepeHalloweenFloki.sol#L90,104,114,117,190,284,294
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);
address immutable WETH;
uint256 public _maxWalletToken = totalSupply / 100;
mapping (address => mapping (address => uint256)) _allowances;
uint256 _newmaxwallet
...
```

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	PepeHalloweenFloki.sol#L289,298
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.

```
sellMultiplier = _sell;
swapThreshold = totalSupply / _denominator;
```

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.

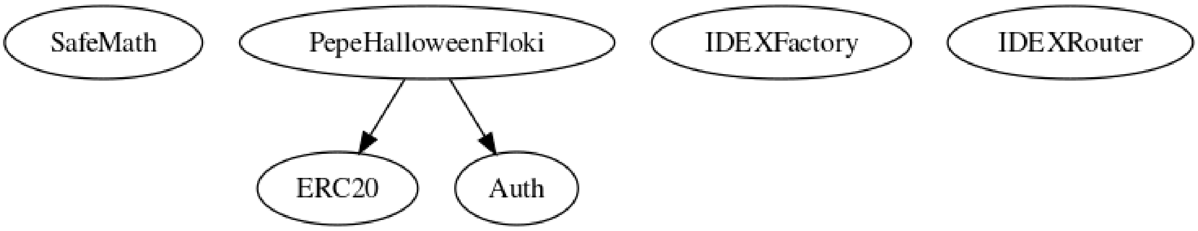
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
ERC20	Interface			
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Auth	Implementation			
		Public	✓	-

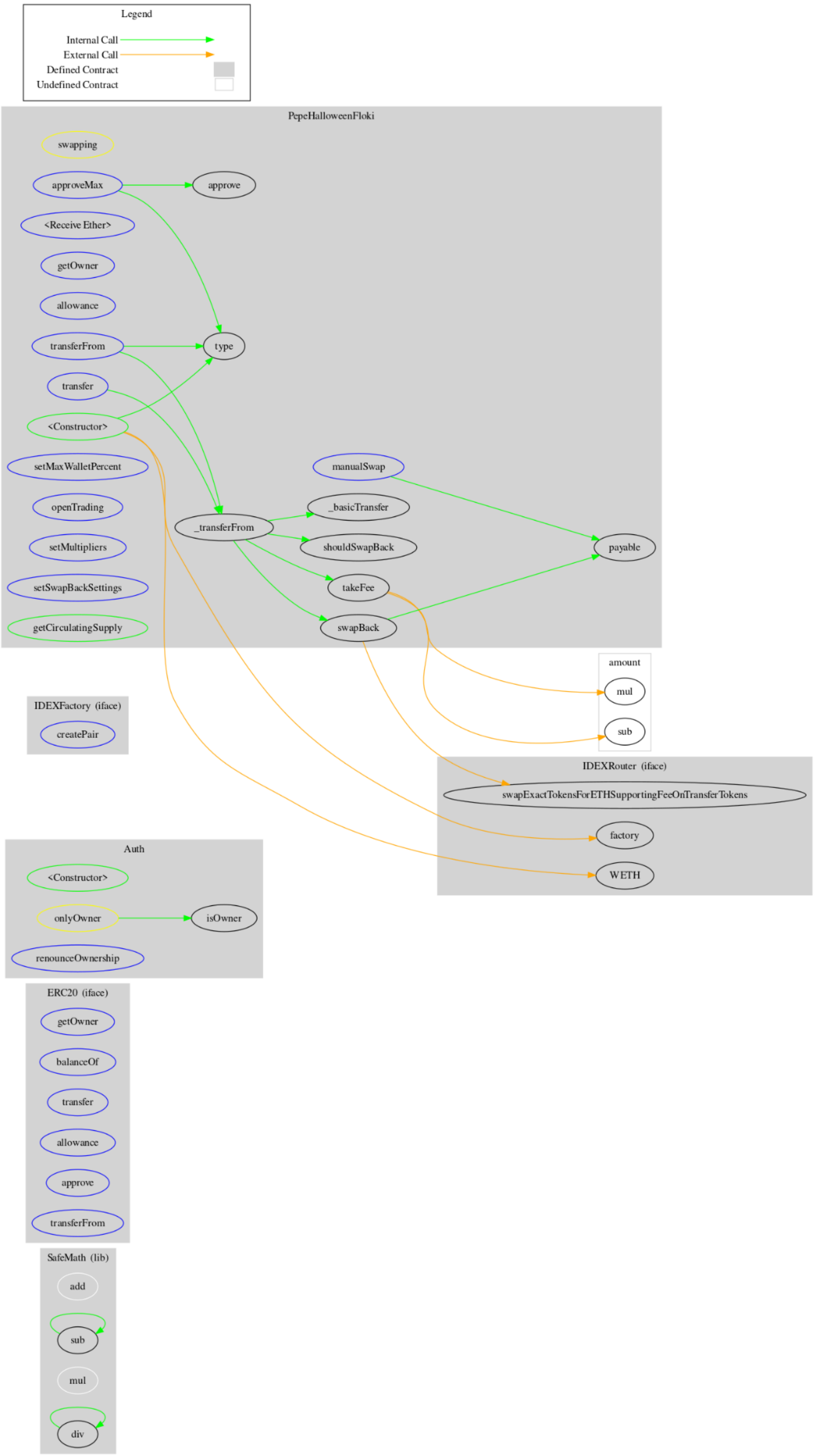
	isOwner	Public		-
	renounceOwnership	External	✓	onlyOwner
IDEXFactory	Interface			
	createPair	External	✓	-
IDEXRouter	Interface			
	factory	External		-
	WETH	External		-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
PepeHalloweenFloki	Implementation	ERC20, Auth		
		Public	✓	Auth
		External	Payable	-
	getOwner	External		-
	allowance	External		-
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	setMaxWalletPercent	External	✓	onlyOwner
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	

	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	manualSwap	External	✓	-
	openTrading	External	✓	onlyOwner
	swapBack	Internal	✓	swapping
	setMultipliers	External	✓	onlyOwner
	setSwapBackSettings	External	✓	onlyOwner
	getCirculatingSupply	Public		-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
PepeHalloweenFloki
swapping
approveMax
approve
<Receive Ether>
getOwner
allowance
transferFrom
type
transfer
<Constructor>
setMaxWalletPercent
manualSwap
openTrading
_basicTransfer
_transferFrom
setMultipliers
shouldSwapBack
payable
setSwapBackSettings
takeFee
getCirculatingSupply
swapBack
amount
mul
sub
IDEXFactory (iface)
createPair
IDEXRouter (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
factory
WETH
Auth
<Constructor>
onlyOwner
isOwner
renounceOwnership
ERC20 (iface)
getOwner
balanceOf
transfer
allowance
approve
transferFrom
SafeMath (lib)
add
sub
mul
div

Summary

Pepe Halloween Floki 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. A multi-wallet signing pattern will provide security against potential hacks. There is also a limit of max 5% 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