Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Charity Token
Website: charitytoken.online
Platform: Polygon Network
Language: Solidity
Date: June 14th, 2022

Table of contents

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by Charity Token to perform the Security audit of the Charity Token Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 14th, 2022.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

Charity Token Pty Ltd is Independently owned and operated. Located in beautiful Australia, they have the goal to bridge shortcomings of the Charity and Foreign Aid Sector by streamlining the payments and grant facilitation process to create a "new standard" of issuance. One that is transparent, accountable, public and efficient.

Charity Token, or “ChaT” for short, is a governance token that serves the following purposes within the ecosystem:

- Allows transactions in conjunction with the smart contract.
- Provides governance on the Charity Token platform.
- Makes the “Charity for All” reward pool and reflection mechanism possible.
- Is will be ONLY currency accepted when paying for Charity Token NFTs and NF As.
- ANY smart contract can be programmed to utilize Charity token as the native currency.

Audit scope

Name	Code Review and Security Analysis Report for Charity Token Protocol Smart Contracts
Platform	**Polygon / Solidity**
File 1	CharityFactory.sol
File 1 MD5 Hash	3099B9672CE9AA7B80A348194BED7E65
Updated File 1 MD5 Hash	8A453937F660CB6EA89351A0CFC58F30
File 2	CharityToken.sol
File 2 MD5 Hash	5D5F79DDF49822D8185E22959CF00835
Updated File 2 MD5 Hash	1DF87B062F064B9DC94D05C0D6400851
Audit Date	June 14th,2022
Revise Audit Date	September 23rd,2022

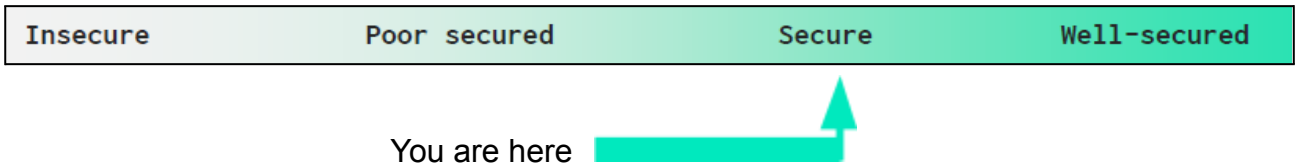
Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 CharityToken.sol • Name: CharityToken • Symbol: CHAT • Decimals: 18 • Maximum Transaction Amount: 1 Trillion • Number Of Tokens To Exchange For Charity: 1 Million • Total supply: 8100 million	**YES, This is valid.**
File 2 CharityFactory.sol • CharityFactory has functions like: createOrganization, allOrganizationsLength, etc.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. Also, these contracts do contain owner control, which does not make them fully decentralized.



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 3 low and some very low level issues.

All the issues have been resolved / acknowledged in the revised code.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 2 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the Charity Token Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Charity Token Protocol.

The Charity Token team has not provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Some code parts are well commented on smart contracts. We suggest using Ethereum's NatSpec style for the commenting.

Documentation

We were given a Charity Token Protocol smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are well commented. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://charitytoken.online which provided rich information about the project architecture.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

CharityToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	lockTheSwap	modifier	Passed	No Issue
8	name	read	Passed	No Issue
9	symbol	read	Passed	No Issue
10	decimals	read	Passed	No Issue
11	totalSupply	read	Passed	No Issue
12	rate	read	Passed	No Issue
13	balanceOf	read	Passed	No Issue
14	transfer	write	Passed	No Issue
15	allowance	read	Passed	No Issue
16	approve	write	Passed	No Issue
17	transferFrom	write	Passed	No Issue
18	increaseAllowance	write	Passed	No Issue
19	decreaseAllowance	write	Passed	No Issue
20	isExcluded	read	Passed	No Issue
21	totalFees	read	Passed	No Issue
22	deliver	write	Passed	No Issue
23	reflectionFromToken	read	Passed	No Issue
24	tokenFromReflection	read	Passed	No Issue
25	excludeAccount	external	access only Owner	No Issue
26	includeAccount	external	access only Owner	No Issue
27	removeAllFee	write	Passed	No Issue
28	restoreAllFee	write	Passed	No Issue
29	isExcludedFromFee	read	Passed	No Issue
30	_approve	write	Passed	No Issue
31	_transfer	write	Passed	No Issue
32	swapTokensForEth	write	lockTheSwap	No Issue
33	sendETHToCharity	write	charityFactory variable check	Refer Audit Findings
34	manualSwap	external	access only Owner	No Issue

35	manualSend	external	access only Owner	No Issue
36	burnTokens	external	access only Owner	No Issue
37	_tokenTransfer	write	Passed	No Issue
38	_transferStandard	write	Passed	No Issue
39	_transferToExcluded	write	Passed	No Issue
40	_transferFromExcluded	write	Passed	No Issue
41	_transferBothExcluded	write	Passed	No Issue
42	_takeCharity	write	Passed	No Issue
43	_reflectFee	write	Passed	No Issue
44	_getValues	read	Passed	No Issue
45	_getTValues	write	Passed	No Issue
46	_getRValues	write	Passed	No Issue
47	_getRate	read	Passed	No Issue
48	_getCurrentSupply	read	Passed	No Issue
49	_getTaxFee	read	Passed	No Issue
50	_getMaxTxAmount	read	Passed	No Issue
51	_getETHBalance	read	Passed	No Issue
52	_setTaxFee	external	access only Owner	No Issue
53	_setCharityFee	external	access only Owner	No Issue
54	_setCharityFactory	external	access only Owner	No Issue
55	_setMaxTxAmount	external	access only Owner	No Issue
56	_setUniswapV2Router	external	access only Owner	No Issue
57	_setUniswapV2Pair	external	access only Owner	No Issue

CharityFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	reinitializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_disableInitializers	internal	Passed	No Issue
6	__Context_init	internal	access only Initializing	No Issue
7	__Context_init_unchained	internal	access only Initializing	No Issue
8	_msgSender	internal	Passed	No Issue
9	_msgData	internal	Passed	No Issue

10	__Ownable_init	internal	access only Initializing	No Issue
11	__Ownable_init_unchained	internal	access only Initializing	No Issue
12	onlyOwner	modifier	Passed	No Issue
13	owner	read	Passed	No Issue
14	_checkOwner	internal	Passed	No Issue
15	renounceOwnership	write	access only Owner	No Issue
16	transferOwnership	write	access only Owner	No Issue
17	_transferOwnership	internal	Passed	No Issue
18	validateSymbol	modifier	Passed	No Issue
19	validateEthAmount	modifier	Passed	No Issue
20	validateAmount	modifier	Passed	No Issue
21	validateOrganization	modifier	Passed	No Issue
22	initialize	write	Passed	No Issue
23	createOrganization	external	access only Owner	No Issue
24	allOrganizationsLength	external	Passed	No Issue
25	donateTokens	external	totalEthDonations not increased by donateTokens	Refer Audit Findings
26	_updateEthBalances	internal	Passed	No Issue
27	_updateTokenBalances	internal	Passed	No Issue
28	ethBalance	external	access only Owner	No Issue
29	tokenBalance	external	access only Owner	No Issue
30	withdrawEth	external	Passed	No Issue
31	withdrawTokens	external	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) charityFactory variable check : **CharityToken.sol**

```
function sendETHToCharity(uint256 amount) private {
    charityFactory.transfer(amount);
}
```

charityFactory variable is set to address(0) by default. sendETHToCharity function is used to send contract balance to charityFactory without checking if it is set to some address or not.

Resolution: We recommend that check the charityFactory variable inside the sendETHToCharity method to ensure that the contract balance does not flow to the address (0).

Status: Acknowledged.

(2) Unnecessary condition check: **CharityToken.sol**

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

    if (_isExcluded[sender] && !_isExcluded[recipient]) {
        _transferFromExcluded(sender, recipient, amount);
    } else if (!_isExcluded[sender] && _isExcluded[recipient]) {
        _transferToExcluded(sender, recipient, amount);
    } else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
        _transferStandard(sender, recipient, amount);
    } else if (_isExcluded[sender] && _isExcluded[recipient]) {
        _transferBothExcluded(sender, recipient, amount);
    } else {
        _transferStandard(sender, recipient, amount);
```

In the transfer function, the else will execute the same functionality.

Resolution: We suggest removing extra else if to reduce the gas fee.

Status: Fixed

(3) totalEthDonations not increased by donateTokens: **CharityFactory.sol**

```
function donateTokens(string calldata _symbol, uint256 _amount)
    external
    payable
    validateSymbol(_symbol)
    validateAmount(_amount)
{
    charityToken.transferFrom(msg.sender, address(this), _amount);
    totalTokenDonations += _amount;
    _updateTokenBalances(_amount);
    emit Donation(msg.sender, _amount, charityToken.symbol(), address(this));
}
```

donateTokens is a payable function, but its msg.value is not added into totalEthDonations. Hence the organizations won't get distribution of the MATIC received by donateTokens.

Resolution: Confirm the logic.

Status: Acknowledged.

Very Low / Informational / Best practices:

(1) Unused events / variable / interface: **CharityToken.sol**

```
bool inSwap = false;
bool public swapEnabled = false;

uint256 private _maxTxAmount = 1000000000000 * 10**18;
uint256 private _numOfTokensToExchangeForCharity = 1 * 10**6 * 10**18;

event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
event SwapEnabledUpdated(bool enabled);
```

```
import "./interfaces/IUniswapV2Factory.sol";
import "./interfaces/IUniswapV2Pair.sol";
import "./interfaces/IUniswapV2Router01.sol";
import "./interfaces/IUniswapV2Router02.sol";
```

MinTokensBeforeSwapUpdated, SwapEnabledUpdated events are defined but not used in code.

swapEnabled variable has been defined and used but it does not change to true ever.

So the use of this variable is meaningless.

IUniswapV2Router02.sol contains IUniswapV2Router01.sol. So no need to import that file in CharityToken.sol.

Resolution: We suggest either removing all these unused variables and events or use them in code.

Status: Fixed

(2) Empty function: **CharityToken.sol**

```
// Exposing a burn method for burning events
function burnTokens(uint256 percentage) external onlyOwner {}
```

burnTokens function has been defined with empty code.

Resolution: We suggest either removing the empty function.

Status: Fixed

(3) All functions which are not called internally, must be declared as external. It is more efficient as sometimes it saves some gas.
https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices
Status: Fixed

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- createOrganization: CharityFactory owners can create new organizations.
- _setUniswapV2Pair: CharityToken owners can set uniswapV2PairAddress.
- _setUniswapV2Router: CharityToken owners can set uniswapV2RouterAddress.
- _setMaxTxAmount: CharityToken owners can set max transaction amount.
- _setCharityFactory: CharityToken owners can set charityFactoryAddress.
- _setCharityFee: CharityToken owners can set charity fees.
- _setTaxFee: CharityToken owners can set tax fees.
- manualSend: CharityToken owners can send manual tokens.
- manualSwap: CharityToken owners can manual swap and send tokens.
- includeAccount: CharityToken owners can include accounts.
- excludeAccount: CharityToken owners can exclude accounts.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We have not observed any major issues in the smart contracts. **So, smart contracts are ready for the mainnet deployment**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

The security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

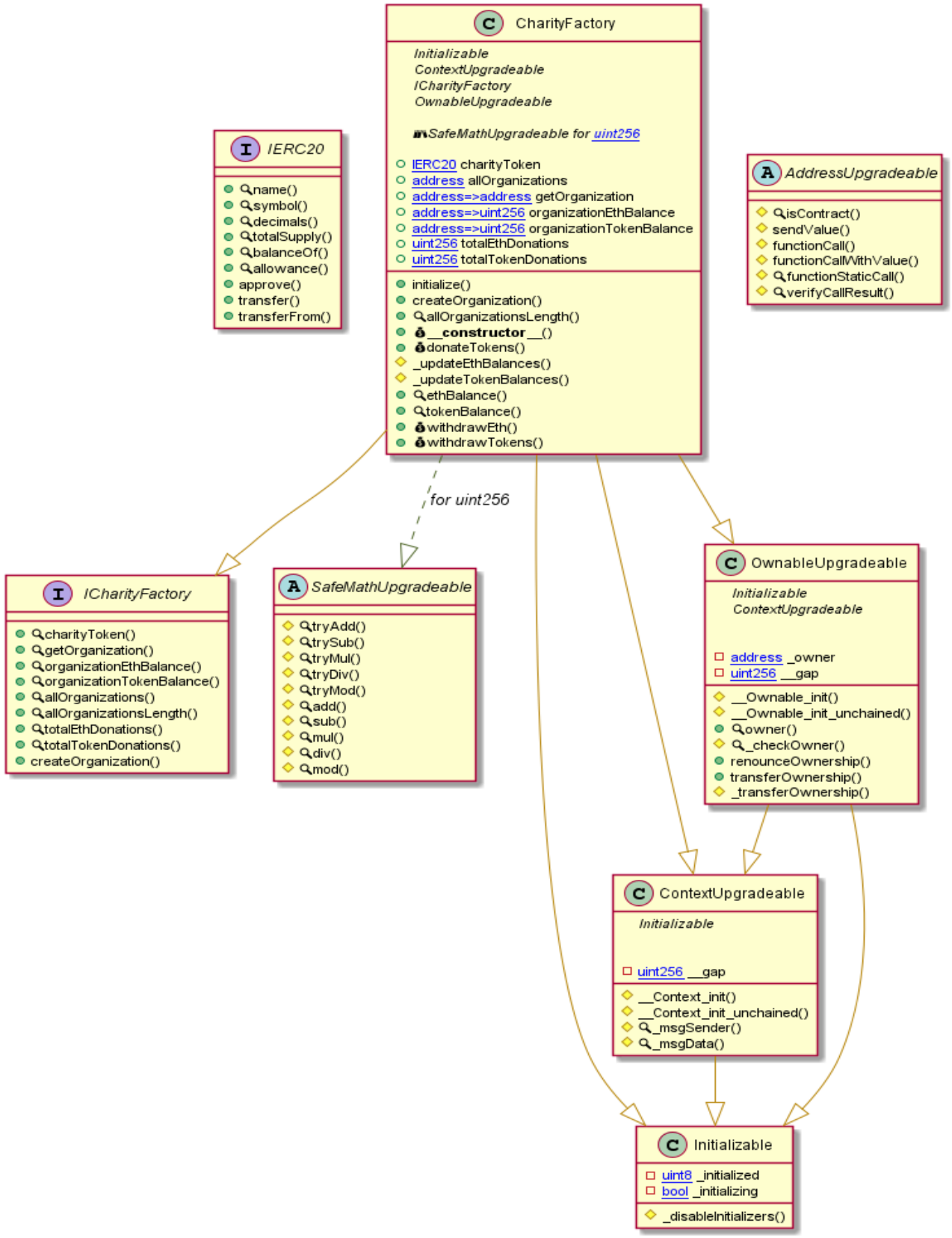
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

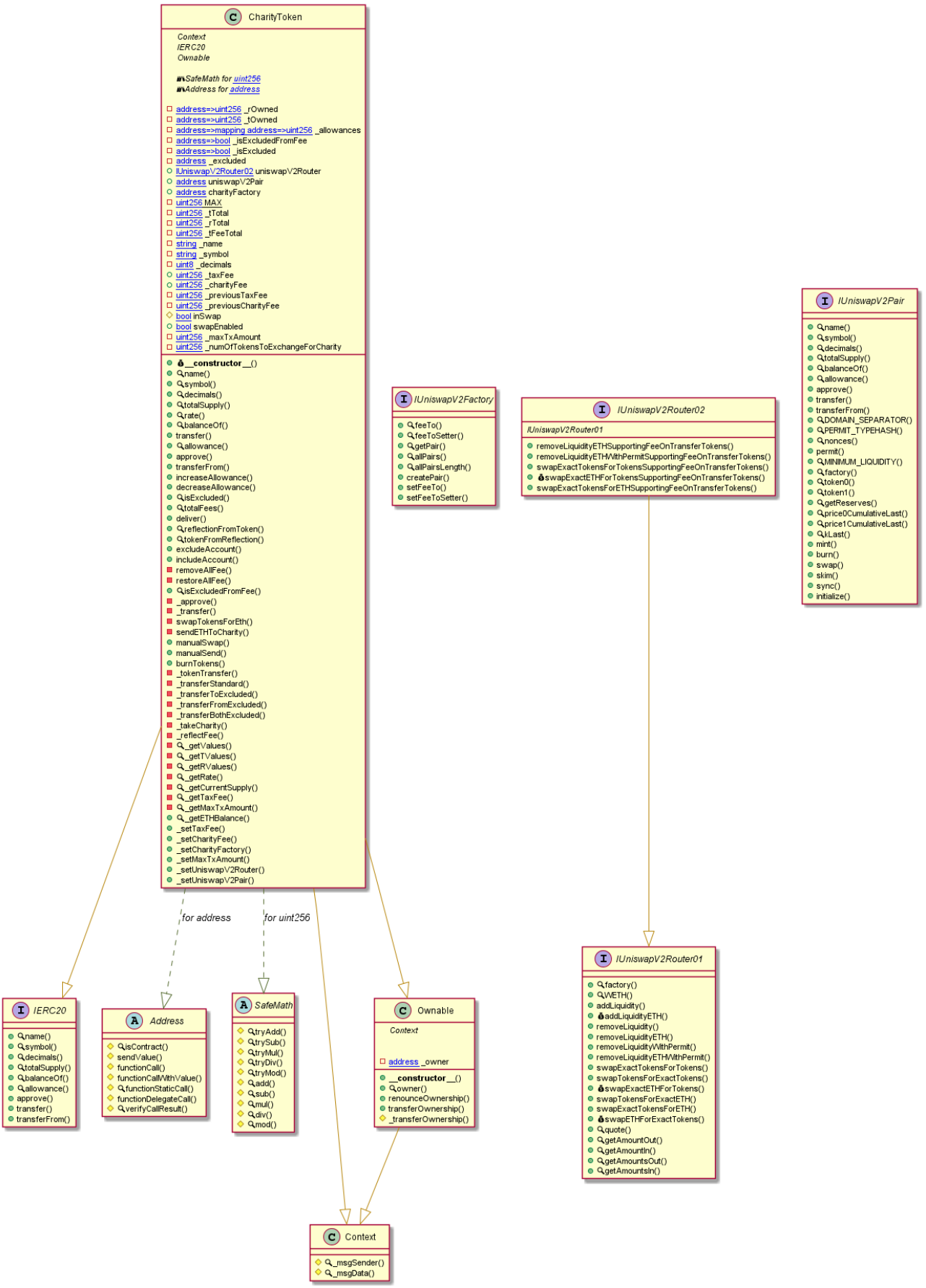
Appendix

Code Flow Diagram - Charity Token Protocol

CharityFactory Diagram



CharityToken Diagram



Slither Results Log

Slither log >> CharityFactory.sol

```
INFO:Detectors:
Reentrancy in CharityFactory.donateTokens(string,uint256) (CharityFactory.sol#710-720):
        External calls:
        - charityToken.transferFrom(msg.sender,address(this),_amount) (CharityFactory.sol#716)
        State variables written after the call(s):
        - _updateTokenBalances(_amount) (CharityFactory.sol#718)
                - organizationTokenBalance[allOrganizations[i]] += _organizationAmount (CharityFactory.sol#732)
        - totalTokenDonations += _amount (CharityFactory.sol#717)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CharityFactory.donateTokens(string,uint256) (CharityFactory.sol#710-720):
        External calls:
        - charityToken.transferFrom(msg.sender,address(this),_amount) (CharityFactory.sol#716)
        Event emitted after the call(s):
        - Donation(msg.sender,_amount,charityToken.symbol(),address(this)) (CharityFactory.sol#719)
Reentrancy in CharityFactory.withdrawTokens(string,uint256) (CharityFactory.sol#755-770):
        External calls:
        - charityToken.transfer(_msgSender(),_amount) (CharityFactory.sol#768)
        Event emitted after the call(s):
        - Withdraw(_msgSender(),_amount,charityToken.symbol(),address(this)) (CharityFactory.sol#769)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
AddressUpgradeable.verifyCallResult(bool,bytes,string) (CharityFactory.sol#438-458) uses assembly
        - INLINE ASM (CharityFactory.sol#450-453)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AddressUpgradeable.functionCall(address,bytes) (CharityFactory.sol#349-351) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (CharityFactory.sol#359-365) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (CharityFactory.sol#378-384) is never used and should be remove
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (CharityFactory.sol#392-403) is never used and should be
emoved
```

```
SafeMathUpgradeable.tryMod(uint256,uint256) (CharityFactory.sol#120-125) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (CharityFactory.sol#91-101) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (CharityFactory.sol#79-84) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.4 (CharityFactory.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (CharityFactory.sol#324-329):
        - (success) = recipient.call{value: amount}() (CharityFactory.sol#327)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (CharityFactory.sol#392-403):
        - (success,returndata) = target.call{value: value}(data) (CharityFactory.sol#401)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (CharityFactory.sol#421-430):
        - (success,returndata) = target.staticcall(data) (CharityFactory.sol#428)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function ContextUpgradeable.__Context_init() (CharityFactory.sol#545-546) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (CharityFactory.sol#548-549) is not in mixedCase
Variable ContextUpgradeable.__gap (CharityFactory.sol#563) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (CharityFactory.sol#574-576) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (CharityFactory.sol#578-580) is not in mixedCase
Variable OwnableUpgradeable.__gap (CharityFactory.sol#639) is not in mixedCase
Parameter CharityFactory.initialize(IERC20)._charityToken (CharityFactory.sol#678) is not in mixedCase
```

```
Function OwnableUpgradeable.__Ownable_init_unchained() (CharityFactory.sol#578-580) is not in mixedCase
Variable OwnableUpgradeable.__gap (CharityFactory.sol#639) is not in mixedCase
Parameter CharityFactory.initialize(IERC20)._charityToken (CharityFactory.sol#678) is not in mixedCase
Parameter CharityFactory.donateTokens(string,uint256)._symbol (CharityFactory.sol#710) is not in mixedCase
Parameter CharityFactory.donateTokens(string,uint256)._amount (CharityFactory.sol#710) is not in mixedCase
Parameter CharityFactory.tokenBalance(string)._symbol (CharityFactory.sol#740) is not in mixedCase
Parameter CharityFactory.withdrawEth(uint256)._amount (CharityFactory.sol#744) is not in mixedCase
Parameter CharityFactory.withdrawTokens(string,uint256)._symbol (CharityFactory.sol#755) is not in mixedCase
Parameter CharityFactory.withdrawTokens(string,uint256)._amount (CharityFactory.sol#755) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Reentrancy in CharityFactory.withdrawEth(uint256) (CharityFactory.sol#744-753):
        External calls:
        - address(_msgSender()).transfer(_amount) (CharityFactory.sol#751)
        Event emitted after the call(s):
        - Withdraw(_msgSender(),_amount,MATIC,address(this)) (CharityFactory.sol#752)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
CharityFactory (CharityFactory.sol#642-771) does not implement functions:
        - ICharityFactory.createOrganization(address,string,string) (CharityFactory.sol#53-57)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
OwnableUpgradeable.__gap (CharityFactory.sol#639) is never used in CharityFactory (CharityFactory.sol#642-771)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (CharityFactory.sol#611-613)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (CharityFactory.sol#619-622)
initialize(IERC20) should be declared external:
        - CharityFactory.initialize(IERC20) (CharityFactory.sol#678-681)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CharityFactory.sol analyzed (8 contracts with 75 detectors), 57 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> CharityToken.sol

```
INFO:Detectors:
CharityToken._setCharityFactory(address).charityFactoryAddress (CharityToken.sol#1348) lacks a zero-check on :
        - charityFactory = charityFactoryAddress (CharityToken.sol#1349)
CharityToken._setUniswapV2Pair(address).uniswapV2PairAddress (CharityToken.sol#1361) lacks a zero-check on :
        - uniswapV2Pair = uniswapV2PairAddress (CharityToken.sol#1362)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CharityToken._transfer(address,address,uint256) (CharityToken.sol#1049-1092):
        External calls:
        - swapTokensForEth(contractTokenBalance) (CharityToken.sol#1074)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.tim
estamp) (CharityToken.sol#1103-1109)
        External calls sending eth:
        - sendETHToCharity(address(this).balance) (CharityToken.sol#1078)
                - charityFactory.transfer(amount) (CharityToken.sol#1113)
        State variables written after the call(s):
        - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
                - _charityFee = _previousCharityFee (CharityToken.sol#1030)
                - _charityFee = 0 (CharityToken.sol#1025)
        - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
                - _previousCharityFee = _charityFee (CharityToken.sol#1022)
        - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
                - _previousTaxFee = _taxFee (CharityToken.sol#1021)
        - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
                - _tFeeTotal = _tFeeTotal.add(tFee) (CharityToken.sol#1247)
        - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
                - _taxFee = _previousTaxFee (CharityToken.sol#1029)
                - _taxFee = 0 (CharityToken.sol#1024)
Reentrancy in CharityToken.transferFrom(address,address,uint256) (CharityToken.sol#933-945):
        External calls:
        - _transfer(sender,recipient,amount) (CharityToken.sol#938)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.tim
estamp) (CharityToken.sol#1103-1109)
```

```
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance))
(CharityToken.sol#939-943)
                - _allowances[owner_][spender] = amount (CharityToken.sol#1045)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CharityToken._transfer(address,address,uint256) (CharityToken.sol#1049-1092):
        External calls:
        - swapTokensForEth(contractTokenBalance) (CharityToken.sol#1074)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.tim
estamp) (CharityToken.sol#1103-1109)
        External calls sending eth:
        - sendETHToCharity(address(this).balance) (CharityToken.sol#1078)
                - charityFactory.transfer(amount) (CharityToken.sol#1113)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tTransferAmount) (CharityToken.sol#1171)
                - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
        - Transfer(sender,recipient,tTransferAmount) (CharityToken.sol#1192)
                - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
        - Transfer(sender,recipient,tTransferAmount) (CharityToken.sol#1213)
                - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
        - Transfer(sender,recipient,tTransferAmount) (CharityToken.sol#1235)
                - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
Reentrancy in CharityToken.transferFrom(address,address,uint256) (CharityToken.sol#933-945):
        External calls:
        - _transfer(sender,recipient,amount) (CharityToken.sol#938)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.tim
estamp) (CharityToken.sol#1103-1109)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (CharityToken.sol#938)
                - charityFactory.transfer(amount) (CharityToken.sol#1113)
        Event emitted after the call(s):
        - Approval(owner_,spender,amount) (CharityToken.sol#1046)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allo
wance)) (CharityToken.sol#939-943)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
CharityToken._rTotal (CharityToken.sol#860) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
CharityToken._previousTaxFee (CharityToken.sol#867) is set pre-construction with a non-constant function or state variable:
        - _taxFee
CharityToken._previousCharityFee (CharityToken.sol#868) is set pre-construction with a non-constant function or state variable:
        - _charityFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version^0.8.4 (CharityToken.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (CharityToken.sol#398-403):
        - (success) = recipient.call{value: amount}() (CharityToken.sol#401)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (CharityToken.sol#466-477):
        - (success,returndata) = target.call{value: value}(data) (CharityToken.sol#475)
Low level call in Address.functionStaticCall(address,bytes,string) (CharityToken.sol#495-504):
        - (success,returndata) = target.staticcall(data) (CharityToken.sol#502)
Low level call in Address.functionDelegateCall(address,bytes,string) (CharityToken.sol#522-531):
        - (success,returndata) = target.delegatecall(data) (CharityToken.sol#529)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Router01.WETH() (CharityToken.sol#55) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (CharityToken.sol#278) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (CharityToken.sol#280) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (CharityToken.sol#306) is not in mixedCase
Function CharityToken._getETHBalance() (CharityToken.sol#1334-1336) is not in mixedCase
Function CharityToken._setTaxFee(uint256) (CharityToken.sol#1338-1341) is not in mixedCase
Function CharityToken._setCharityFee(uint256) (CharityToken.sol#1343-1346) is not in mixedCase
Function CharityToken._setCharityFactory(address) (CharityToken.sol#1348-1350) is not in mixedCase
Function CharityToken._setMaxTxAmount(uint256) (CharityToken.sol#1352-1355) is not in mixedCase
```

```
Variable CharityToken._charityFee (CharityToken.sol#866) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Reentrancy in CharityToken._transfer(address,address,uint256) (CharityToken.sol#1049-1092):
        External calls:
        - sendETHToCharity(address(this).balance) (CharityToken.sol#1078)
                - charityFactory.transfer(amount) (CharityToken.sol#1113)
        State variables written after the call(s):
        - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
                - _charityFee = _previousCharityFee (CharityToken.sol#1030)
                - _charityFee = 0 (CharityToken.sol#1025)
        - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
                - _previousCharityFee = _charityFee (CharityToken.sol#1022)
        - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
                - _previousTaxFee = _taxFee (CharityToken.sol#1021)
        - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rCharity) (CharityToken.sol#1241)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (CharityToken.sol#1187)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (CharityToken.sol#1167)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (CharityToken.sol#1209)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (CharityToken.sol#1230)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (CharityToken.sol#1168)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (CharityToken.sol#1210)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (CharityToken.sol#1189)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (CharityToken.sol#1232)
        - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
                - _rTotal = _rTotal.sub(rFee) (CharityToken.sol#1246)
        - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
                - _tFeeTotal = _tFeeTotal.add(tFee) (CharityToken.sol#1247)
        - _tokenTransfer(sender,recipient,amount,takeFee) (CharityToken.sol#1091)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tCharity) (CharityToken.sol#1242)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (CharityToken.sol#1229)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (CharityToken.sol#1208)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (CharityToken.sol#1188)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (CharityToken.sol#1231)
```

```
                - charityFactory.transfer(amount) (CharityToken.sol#1113)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance))
(CharityToken.sol#939-943)
                - _allowances[owner_][spender] = amount (CharityToken.sol#1045)
        Event emitted after the call(s):
        - Approval(owner_,spender,amount) (CharityToken.sol#1046)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allo
wance)) (CharityToken.sol#939-943)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Chari
tyToken.sol#60) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint2
56).amountBDesired (CharityToken.sol#61)
Variable CharityToken._transferFromExcluded(address,address,uint256).rTransferAmount (CharityToken.sol#1202) is too similar to
CharityToken._transferToExcluded(address,address,uint256).tTransferAmount (CharityToken.sol#1183)
Variable CharityToken._getRValues(uint256,uint256,uint256).rTransferAmount (CharityToken.sol#1305) is too similar to CharityTok
en._transferFromExcluded(address,address,uint256).tTransferAmount (CharityToken.sol#1204)
Variable CharityToken._transferFromExcluded(address,address,uint256).rTransferAmount (CharityToken.sol#1202) is too similar to
CharityToken._getValues(uint256).tTransferAmount (CharityToken.sol#1265)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
CharityToken._decimals (CharityToken.sol#864) should be constant
CharityToken._name (CharityToken.sol#862) should be constant
CharityToken._numOfTokensToExchangeForCharity (CharityToken.sol#874) should be constant
CharityToken._symbol (CharityToken.sol#863) should be constant
CharityToken._tTotal (CharityToken.sol#859) should be constant
CharityToken.swapEnabled (CharityToken.sol#871) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (CharityToken.sol#819-821)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (CharityToken.sol#827-830)
name() should be declared external:
        - CharityToken.name() (CharityToken.sol#894-896)
symbol() should be declared external:
        - CharityToken.symbol() (CharityToken.sol#898-900)
decimals() should be declared external:
        - CharityToken.decimals() (CharityToken.sol#902-904)
totalSupply() should be declared external:
        - CharityToken.totalSupply() (CharityToken.sol#906-908)
rate() should be declared external:
        - CharityToken.rate() (CharityToken.sol#910-912)
transfer(address,uint256) should be declared external:
        - CharityToken.transfer(address,uint256) (CharityToken.sol#919-922)
allowance(address,address) should be declared external:
        - CharityToken.allowance(address,address) (CharityToken.sol#924-926)
approve(address,uint256) should be declared external:
        - CharityToken.approve(address,uint256) (CharityToken.sol#928-931)
transferFrom(address,address,uint256) should be declared external:
        - CharityToken.transferFrom(address,address,uint256) (CharityToken.sol#933-945)
increaseAllowance(address,uint256) should be declared external:
        - CharityToken.increaseAllowance(address,uint256) (CharityToken.sol#947-950)
decreaseAllowance(address,uint256) should be declared external:
        - CharityToken.decreaseAllowance(address,uint256) (CharityToken.sol#952-959)
```

```
decreaseAllowance(address,uint256) should be declared external:
        - CharityToken.decreaseAllowance(address,uint256) (CharityToken.sol#952-959)
isExcluded(address) should be declared external:
        - CharityToken.isExcluded(address) (CharityToken.sol#961-963)
totalFees() should be declared external:
        - CharityToken.totalFees() (CharityToken.sol#965-967)
deliver(uint256) should be declared external:
        - CharityToken.deliver(uint256) (CharityToken.sol#969-976)
reflectionFromToken(uint256,bool) should be declared external:
        - CharityToken.reflectionFromToken(uint256,bool) (CharityToken.sol#978-987)
isExcludedFromFee(address) should be declared external:
        - CharityToken.isExcludedFromFee(address) (CharityToken.sol#1033-1035)
_getETHBalance() should be declared external:
        - CharityToken._getETHBalance() (CharityToken.sol#1334-1336)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CharityToken.sol analyzed (10 contracts with 75 detectors), 125 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

CharityFactory.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in CharityFactory.withdrawTokens(string,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 755:4:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

more

Pos: 401:50:

Gas & Economy

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 724:8:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 13:4:

Miscellaneous

Constant/View/Pure functions:

CharityFactory.createOrganization() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 683:4:

No return:

ChartityFactory.createOrganization(): Defines a return type but never explicitly returns a value.
Pos: 683:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 765:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 242:19:

CharityToken.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in CharityToken.swapTokensForEth(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1094:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1108:12:

Gas & Economy

Gas costs:

Gas requirement of function CharityToken.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 894:4:

Gas costs:

Gas requirement of function CharityToken.includeAccount is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1005:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1317:8:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 13:4:

Miscellaneous

Constant/View/Pure functions:

CharityToken.burnTokens(uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1129:4:

Similar variable names:

CharityToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1181:12:

Similar variable names:

CharityToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1183:12:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1344:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1353:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 744:19:

Solhint Linter

CharityFactory.sol

```
CharityFactory.sol:67:18: Error: Parse error: missing ';' at '{'
CharityFactory.sol:80:18: Error: Parse error: missing ';' at '{'
CharityFactory.sol:92:18: Error: Parse error: missing ';' at '{'
CharityFactory.sol:109:18: Error: Parse error: missing ';' at '{'
CharityFactory.sol:121:18: Error: Parse error: missing ';' at '{'
CharityFactory.sol:217:18: Error: Parse error: missing ';' at '{'
CharityFactory.sol:240:18: Error: Parse error: missing ';' at '{'
CharityFactory.sol:266:18: Error: Parse error: missing ';' at '{'
```

CharityToken.sol

```
CharityToken.sol:569:18: Error: Parse error: missing ';' at '{'
CharityToken.sol:582:18: Error: Parse error: missing ';' at '{'
CharityToken.sol:594:18: Error: Parse error: missing ';' at '{'
CharityToken.sol:611:18: Error: Parse error: missing ';' at '{'
CharityToken.sol:623:18: Error: Parse error: missing ';' at '{'
CharityToken.sol:719:18: Error: Parse error: missing ';' at '{'
CharityToken.sol:742:18: Error: Parse error: missing ';' at '{'
CharityToken.sol:768:18: Error: Parse error: missing ';' at '{'
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io