AntiHACK.me

SMART CONTRACT

Security Audit Report

Project:	Cash Click
Platform:	Binance Smart Chain
Language:	Solidity
Date:	January 26th, 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.

Introduction

AntiHACK.me was contracted by the Cash Click team to perform the Security audit of the Cash Click smart contract 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 January 26th, 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

Cash Click is a standard BEP20 token smart contract. This audit only considers Cash Click and ACT which are BEP20 tokens , Proxy admin and Transparent upgradable proxies.

Audit scope

Name	Code Review and Security Analysis Report for Cash Click Smart Contracts
Platform	**BSC / Solidity**
File 1	Acts.sol
File 1 MD5 Hash	24B64272B2D5319D8B172CD935090
File 2	CASHCLICK.sol
File 2 MD5 Hash	24B64272B2D5319D8B172CD935090
File 3	ProxyAdmin.sol
File 3 MD5 Hash	AA29F22D4DB8537A99F1265A4C642
File 4	TransparentUpgradeableProxy.sol
File 4 MD5 Hash	24B64272B2D5319D8B172CD935090
Audit Date	January 26th,2022

Claimed Smart Contracts Features

Claimed Feature Detail	Our Observation
File 1 Acts.sol • Name: Acts • Decimals: 18 • Symbol: ACTS • The owner can access functions like: mint, burn.	**YES, This is valid.**
File 2 CASHCLICK.sol • Name: Cash Click • Decimals: 18 • Symbol: CASH CLICK • The owner can access functions like: mint, burn.	**YES, This is valid.**
File 3 ProxyAdmin.sol • The ProxyAdmin can return the current implementation of `proxy`, the current admin of `proxy` etc.	**YES, This is valid.**
File 4 TransparentUpgradeableProxy.sol • The TransparentUpgradeableProxy can access functions like: admin, implementation, changeAdmin,etc.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. These contracts also have owner functions (described in the centralization section below), which does not make everything 100% decentralized. Thus, the owner must execute those smart contract functions as per the business plan.

We used various tools like MythX, Slither 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 the AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 2 medium and 0 low and some very low-level issues.

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	Moderated
	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
	Random number generation/use vulnerability	Passed
	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
	Unnecessary 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	Moderated
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 4 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 Cash Click 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 Cash Click.

The Cash Click team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **not** well commented on smart contracts.

Documentation

We were given an Cash Click smart contract code in the form of a BSCScan web link. The details of that code are mentioned above in the table.

As mentioned above, code parts are **Not well** commented. So, it is not 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.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well-known industry standard open-source projects. And their core code blocks are written well.

Apart from libraries, its functions are not used in external smart contract calls.

AS-IS overview

Acts.sol

(1) Interface

(a) IBEP20

(2) Inherited contracts

(a) TokenStorage

(b) Initializable

(c) Context

(d) Ownable

(e) BEP20

(3) Events

(a) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

(4) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	init	write	Passed	No Issue
3	mint	write	Unlimited token minting	Refer Audit Findings
4	burn	write	Owner can burn anyone's token	Refer Audit Findings
5	getOwner	external	Passed	No Issue
6	name	read	Passed	No Issue
7	decimals	read	Passed	No Issue
8	symbol	read	Passed	No Issue
9	totalSupply	read	Passed	No Issue
10	balanceOf	read	Passed	No Issue

11	transfer	write	Passed	No Issue
12	allowance	read	Passed	No Issue
13	approve	write	Passed	No Issue
14	transferFrom	write	Passed	No Issue
15	increaseAllowance	write	Passed	No Issue
16	decreaseAllowance	write	Passed	No Issue
17	_transfer	internal	Passed	No Issue
18	_mint	internal	Passed	No Issue
19	_burn	internal	Passed	No Issue
20	_approve	internal	Passed	No Issue
21	_burnFrom	internal	Passed	No Issue
22	initializer	modifier	Passed	No Issue
23	owner	read	Passed	No Issue
24	onlyOwner	modifier	Passed	No Issue
25	isOwner	read	Passed	No Issue
26	renounceOwnership	write	access only Owner	No Issue
27	transferOwnership	write	access only Owner	No Issue
28	_transferOwnership	internal	Passed	No Issue

CASHCLICK.sol

(1) Interface

(a) IBEP20

(2) Inherited contracts

(a) TokenStorage

(b) Initializable

(c) Context

(d) Ownable

(e) BEP20

(3) Events

(a) event DelegateChanged(address indexed delegator, address indexed fromDelegate, address indexed toDelegate);

(b) event DelegateVotesChanged(address indexed delegate, uint previousBalance, uint newBalance);

(c) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

(4) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	owner	read	Passed	No Issue
5	isOwner	read	Passed	No Issue
6	renounceOwnership	write	access only Owner	No Issue
7	transferOwnership	write	access only Owner	No Issue
8	_transferOwnership	internal	Passed	No Issue
9	getOwner	external	Passed	No Issue
10	name	read	Passed	No Issue
11	decimals	read	Passed	No Issue
12	symbol	read	Passed	No Issue
13	totalSupply	read	Passed	No Issue
14	supplyHardCap	read	Passed	No Issue
15	totalMinted	read	Passed	No Issue
16	balanceOf	read	Passed	No Issue
17	transfer	write	Passed	No Issue
18	allowance	read	Passed	No Issue
19	approve	write	Passed	No Issue

20	transferFrom	write	Passed	No Issue
21	increaseAllowance	write	Passed	No Issue
22	decreaseAllowance	write	Passed	No Issue
23	_transfer	internal	Passed	No Issue
24	_mint	internal	Passed	No Issue
25	_burn	internal	Passed	No Issue
26	_approve	internal	Passed	No Issue
27	_burnFrom	internal	Passed	No Issue
28	init	write	Passed	No Issue
29	mint	write	access only Owner	No Issue
30	burn	write	Owner can burn anyone's token	Refer Audit Findings
31	delegates	external	Passed	No Issue
32	delegate	external	Passed	No Issue
33	delegateBySig	external	Passed	No Issue
34	getCurrentVotes	external	Passed	No Issue
35	getPriorVotes	external	Passed	No Issue
36	_delegate	internal	Passed	No Issue
37	_moveDelegates	internal	Passed	No Issue
38	_writeCheckpoint	internal	Passed	No Issue
39	safe32	internal	Passed	No Issue
40	getChainId	internal	Passed	No Issue

ProxyAdmin.sol

(1) Inherited contracts

(a) Context

(b) Ownable

(c) Proxy

(d) ProxyAdmin

(e) UpgradeableProxy

(f) TransparentUpgradeableProxy

(2) Events

(a) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

(b) event Upgraded(address indexed implementation);

(c) event AdminChanged(address previousAdmin, address newAdmin);

(3) 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	isOwner	read	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	_delegate	internal	Passed	No Issue
9	_implementation	internal	Passed	No Issue
10	_fallback	internal	Passed	No Issue
11	_beforeFallback	internal	Passed	No Issue
12	getProxyImplementation	read	Passed	No Issue
13	getProxyAdmin	read	Passed	No Issue
14	changeProxyAdmin	write	access only Owner	No Issue
15	upgrade	write	access only Owner	No Issue
16	upgradeAndCall	write	access only Owner	No Issue
17	_implementation	internal	Passed	No Issue
18	_upgradeTo	internal	Passed	No Issue

19	_setImplementation	write	Passed	No Issue
20	ifAdmin	modifier	Passed	No Issue
21	admin	external	access by ifAdmin	No Issue
22	implementation	external	access by ifAdmin	No Issue
23	changeAdmin	external	access by ifAdmin	No Issue
24	upgradeTo	external	access by ifAdmin	No Issue
25	upgradeToAndCall	external	access by ifAdmin	No Issue
26	_admin	internal	Passed	No Issue
27	_setAdmin	write	Passed	No Issue
28	_beforeFallback	internal	Passed	No Issue

TransparentUpgradeableProxy.sol

(1) Inherited contracts

(a) Proxy

(b) UpgradeableProxy

(2) Events

(a) event Upgraded(address indexed implementation);

(b) event AdminChanged(address previousAdmin, address newAdmin);

(3) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_delegate	internal	Passed	No Issue
3	_implementation	internal	Passed	No Issue
4	_fallback	internal	Passed	No Issue
5	_beforeFallback	internal	Passed	No Issue
6	_implementation	internal	Passed	No Issue
7	_upgradeTo	internal	Passed	No Issue

8	_setImplementation	write	Passed	No Issue
9	ifAdmin	modifier	Passed	No Issue
10	admin	external	access by ifAdmin	No Issue
11	implementation	external	access by ifAdmin	No Issue
12	changeAdmin	external	access by ifAdmin	No Issue
13	upgradeTo	external	access by ifAdmin	No Issue
14	upgradeToAndCall	external	access by ifAdmin	No Issue
15	_admin	internal	Passed	No Issue
16	_setAdmin	write	Passed	No Issue
17	_beforeFallback	internal	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.

Audit Findings

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

(1) Unlimited token minting: **Acts.sol**

```
/// @notice Creates `_amount` token to `_to`. Must only be called by the owner.
function mint(address _to, uint255 _amount) public onlyOwner {
    _mint(_to, _amount);
}
```

Token minting without any maximum limit is considered inappropriate for tokenomics.

Resolution: We recommend placing some limit on token minting to mitigate this issue. If this is a part of the plan then disregard this issue.

Status: Acknowledged

(2) Owner can burn anyone's token: **Acts.sol, CASHCLICK.sol**

```
function burn(address _account, uint255 _amount) public onlyOwner {
    _burn(_account, _amount);
}
```

```
function burn(address _account, uint255 _amount) public onlyOwner {
    _burn(_account, _amount);
    _moveDelegates(_delegates[_account], address(0), _amount);
}
```

Owner can burn any users' tokens.

Resolution: We suggest changing the code so only token holders can burn their own tokens and not anyone else. Not even a contract creator.

Low

No Low severity vulnerabilities were found.

Very Low / Discussion / Best practices:

(1) Use latest solidity version: **TransparentUpgradeableProxy.sol, Acts.sol, CASHCLICK.sol , ProxyAdmin.sol**

Using the latest solidity will prevent any compiler level bugs.

Resolution: We suggest using version > 0.8.0.

Centralization

These smart contracts have 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:

- mint: The Acts owner can create a `_amount` token to `_to`.
- burn: The Acts owner can burn an amount from the account.
- changeProxyAdmin: The ProxyAdmin owner can change the admin of `proxy` to `New Admin`.
- upgrade: The ProxyAdmin owner can upgrade `proxy` to `implementation`.
- upgradeAndCall: The ProxyAdmin owner can upgrade `proxy` to `implementation` and call a function for the new implementation.
- admin: The TransparentUpgradeableProxy admin can return the current admin.
- implementation: The TransparentUpgradeableProxy admin can return the current implementation.
- changeAdmin: The TransparentUpgradeableProxy admin can change the admin of the proxy.
- upgradeTo: The TransparentUpgradeableProxy admin can upgrade the implementation of the proxy.
- upgradeToAndCall:The TransparentUpgradeableProxy admin can upgrade the implementation of the proxy, and then call a function from the new implementation as specified by `data`, which should be an encoded function call. This is useful to initialize new storage variables in the proxied contract.
- mint: The CASH CLICK owner can create a `_amount` token to `_to`.
- burn: The CASH CLICK owner can burn an amount from the address.

Conclusion

We were given all contract codes in the form of BSCscan links. And we have used all possible tests based on given objects as files. We observed some issues, but they are not critical. So, **it's good to go to production**.

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.

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

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

AntiHACK.me Disclaimer

AntiHACK.me 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).

Because 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.

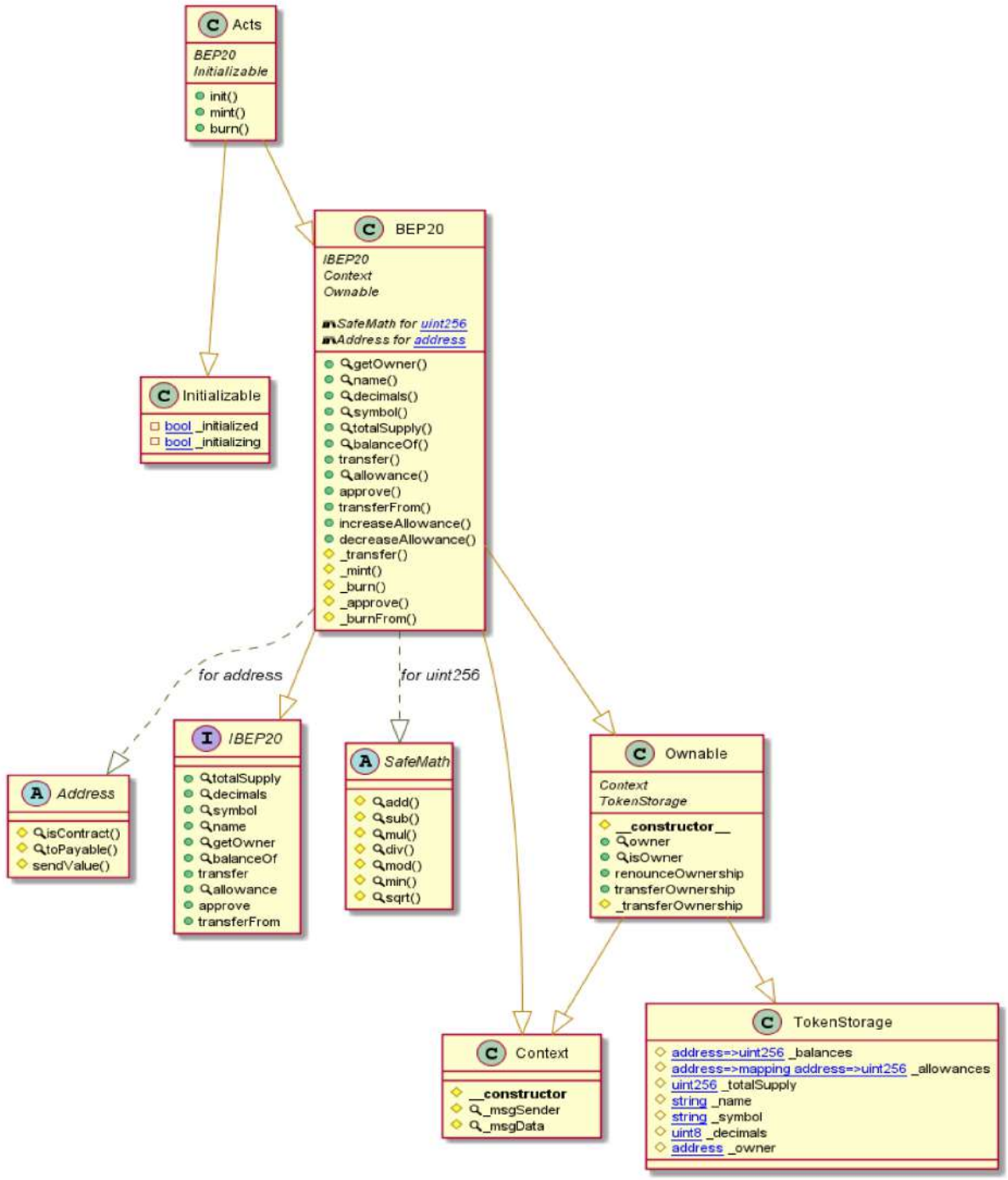
Technical Disclaimer

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.

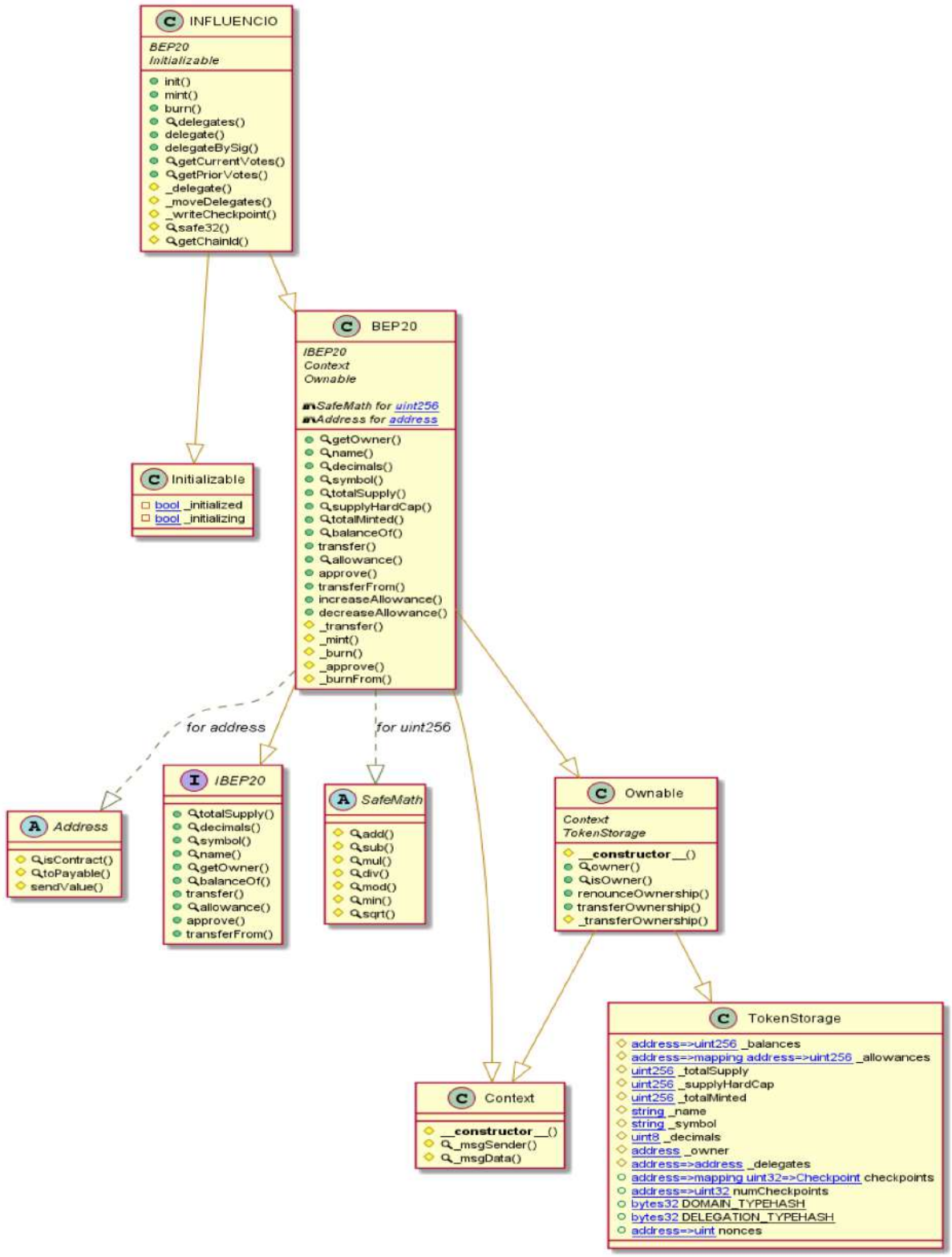
Appendix

Code Flow Diagram - Cash Click

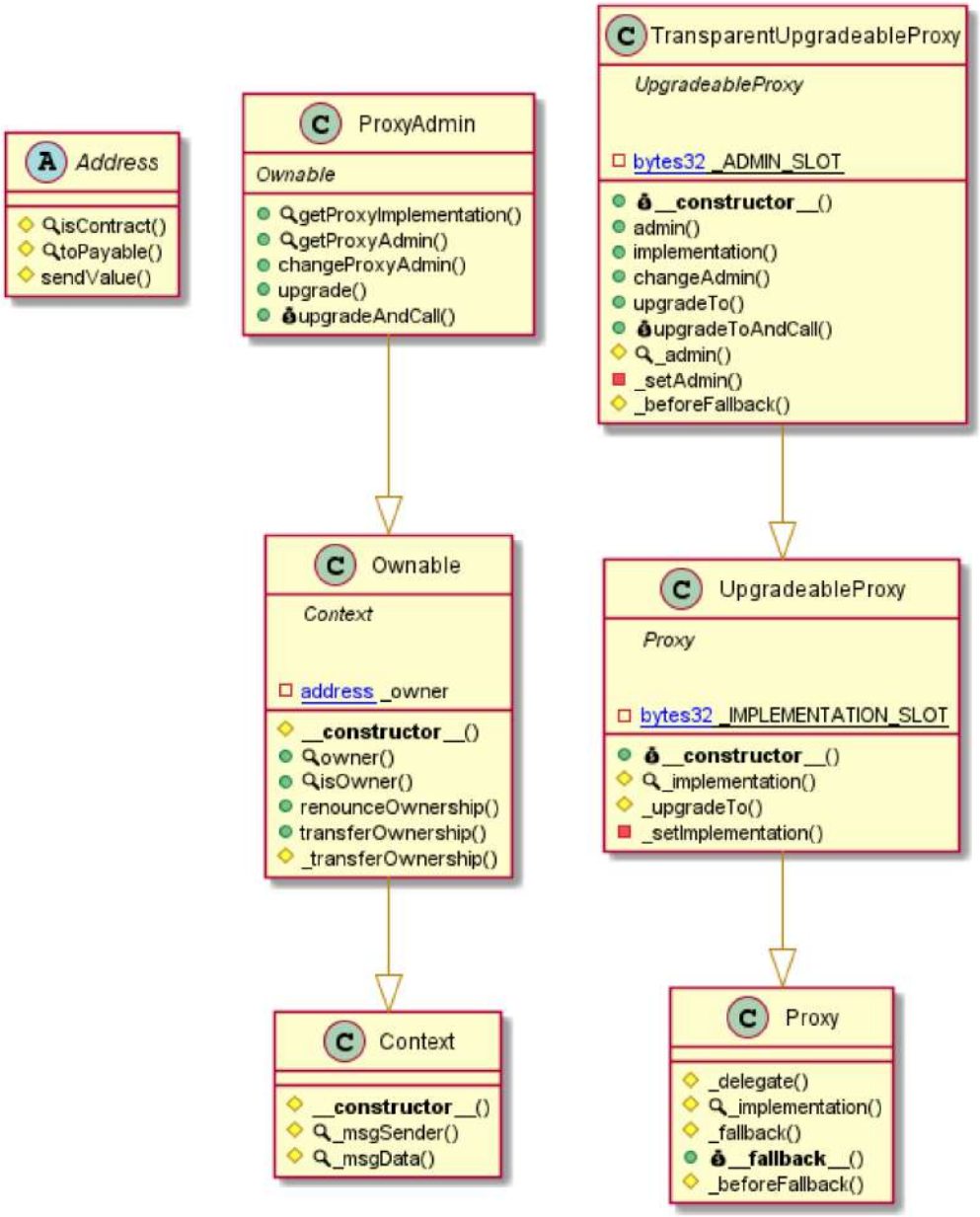
Acts Diagram



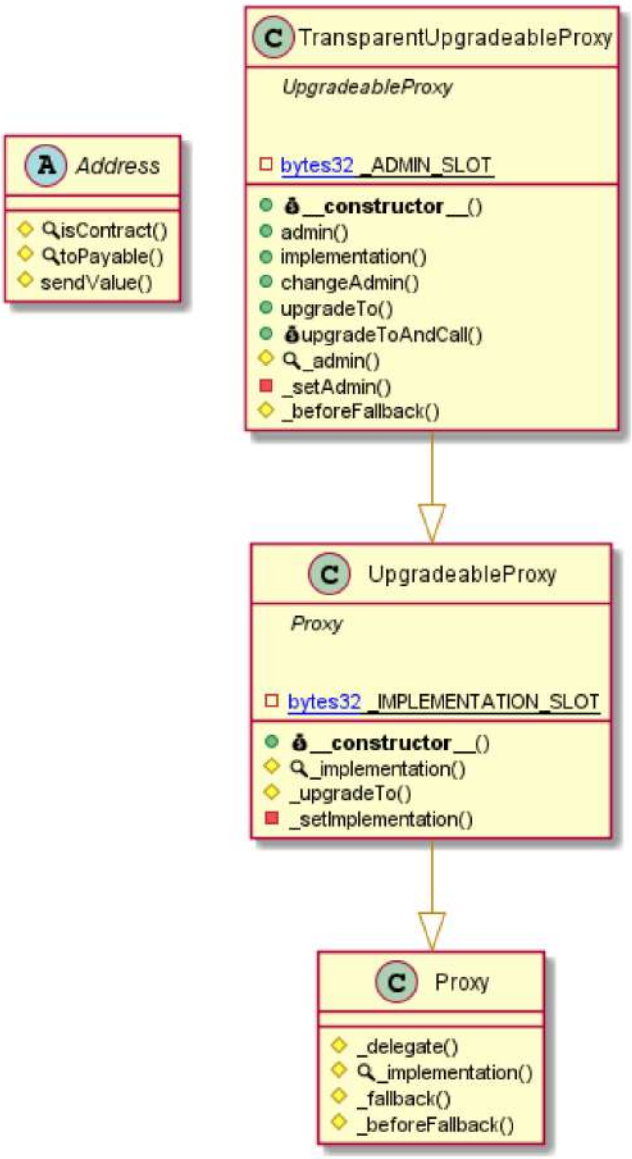
CASH CLICK DIAGRAM



ProxyAdmin Diagram



Transparent Upgradeable Proxy Diagram



Slither Results Log

Slither log >> Acts.sol

```
INFO:Detectors:
BEP20.allowance(address,address).owner (Acts.sol#536) shadows:
        - Ownable.owner() (Acts.sol#413-415) (function)
BEP20._approve(address,address,uint256).owner (Acts.sol#695) shadows:
        - Ownable.owner() (Acts.sol#413-415) (function)
Acts.init(string,string,uint8).name (Acts.sol#726) shadows:
        - BEP20.name() (Acts.sol#487-490) (function)
        - IBEP20.name() (Acts.sol#88) (function)
Acts.init(string,string,uint8).symbol (Acts.sol#727) shadows:
        - BEP20.symbol() (Acts.sol#502-504) (function)
        - IBEP20.symbol() (Acts.sol#83) (function)
Acts.init(string,string,uint8).decimals (Acts.sol#728) shadows:
        - BEP20.decimals() (Acts.sol#495-497) (function)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Address.isContract(address) (Acts.sol#22-31) uses assembly
        - INLINE ASM (Acts.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.isContract(address) (Acts.sol#22-31) is never used and should be removed
Address.sendValue(address,uint256) (Acts.sol#61-67) is never used and should be removed
Address.toPayable(address) (Acts.sol#39-41) is never used and should be removed
BEP20._burnFrom(address,uint256) (Acts.sol#712-719) is never used and should be removed
Context._msgData() (Acts.sol#392-395) is never used and should be removed
SafeMath.div(uint256,uint256) (Acts.sol#252-254) is never used and should be removed
SafeMath.div(uint256,uint256,string) (Acts.sol#268-278) is never used and should be removed
SafeMath.min(uint256,uint256) (Acts.sol#317-319) is never used and should be removed
SafeMath.mod(uint256,uint256) (Acts.sol#292-294) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Acts.sol#308-315) is never used and should be removed
SafeMath.mul(uint256,uint256) (Acts.sol#226-238) is never used and should be removed
SafeMath.sqrt(uint256) (Acts.sol#322-333) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
INFO:Detectors:
Variable TokenStorage._balances (Acts.sol#338) is not in mixedCase
Variable TokenStorage._allowances (Acts.sol#339) is not in mixedCase
Variable TokenStorage._totalSupply (Acts.sol#340) is not in mixedCase
Variable TokenStorage._name (Acts.sol#344) is not in mixedCase
Variable TokenStorage._symbol (Acts.sol#345) is not in mixedCase
Variable TokenStorage._decimals (Acts.sol#346) is not in mixedCase
Variable TokenStorage._owner (Acts.sol#349) is not in mixedCase
Parameter Acts.mint(address,uint256)._to (Acts.sol#738) is not in mixedCase
Parameter Acts.mint(address,uint256)._amount (Acts.sol#738) is not in mixedCase
Parameter Acts.burn(address,uint256)._account (Acts.sol#742) is not in mixedCase
Parameter Acts.burn(address,uint256)._amount (Acts.sol#742) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Acts.sol#393)" inContext (Acts.sol#383-396)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (Acts.sol#413-415)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Acts.sol#439-442)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Acts.sol#448-450)
name() should be declared external:
        - BEP20.name() (Acts.sol#487-490)
decimals() should be declared external:
        - BEP20.decimals() (Acts.sol#495-497)
symbol() should be declared external:
        - BEP20.symbol() (Acts.sol#502-504)
totalSupply() should be declared external:
        - BEP20.totalSupply() (Acts.sol#509-511)
balanceOf(address) should be declared external:
        - BEP20.balanceOf(address) (Acts.sol#516-518)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (Acts.sol#528-531)
allowance(address,address) should be declared external:
```

```
        - BEP20.approve(address,uint256) (Acts.sol#547-550)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (Acts.sol#564-576)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (Acts.sol#590-593)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (Acts.sol#609-616)
init(string,string,uint8) should be declared external:
        - Acts.init(string,string,uint8) (Acts.sol#725-734)
mint(address,uint256) should be declared external:
        - Acts.mint(address,uint256) (Acts.sol#738-740)
burn(address,uint256) should be declared external:
        - Acts.burn(address,uint256) (Acts.sol#742-744)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Acts.sol analyzed (9 contracts with 75 detectors), 48 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> CASHCLICK.sol

```
INFO:Detectors:
   Cash click._writeCheckpoint(address,uint32,uint256,uint256) (   Cash click.sol#960-978) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (   Cash click.sol#970)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
   Cash click.getChainId().chainId (   Cash click.sol#986) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
BEP20.allowance(address,address).owner (   Cash click.sol#581) shadows:
        - Ownable.owner() (   Cash click.sol#451-453) (function)
BEP20._approve(address,address,uint256).owner (   Cash click.sol#740) shadows:
        - Ownable.owner() (   Cash click.sol#451-453) (function)
INFLUENCIO.init(string,string,uint8,uint256).name (   Cash click.sol#770) shadows:
        - BEP20.name() (   Cash click.sol#525-527) (function)
        - IBEP20.name() (   Cash click.sol#90) (function)
INFLUENCIO.init(string,string,uint8,uint256).symbol (   Cash click.sol#771) shadows:
        - BEP20.symbol() (   Cash click.sol#539-541) (function)
        - IBEP20.symbol() (   Cash click.sol#85) (function)
INFLUENCIO.init(string,string,uint8,uint256).decimals (   Cash click.sol#772) shadows:
        - BEP20.decimals() (   Cash click.sol#532-534) (function)
        - IBEP20.decimals() (   Cash click.sol#80) (function)
INFLUENCIO.init(string,string,uint8,uint256).supplyHardCap (   Cash click.sol#773) shadows:
        - BEP20.supplyHardCap() (   Cash click.sol#550-552) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
          .delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (   Cash click.sol#826-867) uses timestamp for comparison
s
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,   Cash click::delegateBySig: signature expired) (   Cash click.sol#865)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (   Cash click.sol#22-31) uses assembly
        - INLINE ASM (          .sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.isContract(address) (   Cash click.sol#22-31) is never used and should be removed
```

```
Address.sendValue(address,uint256) (   Cash click.sol#61-67) is never used and should be removed
Address.toPayable(address) (   Cash click.sol#39-41) is never used and should be removed
BEP20._burnFrom(address,uint256) (   Cash click.sol#757-764) is never used and should be removed
Context._msgData() (          .sol#430-433) is never used and should be removed
SafeMath.div(uint256,uint256) (   Cash click.sol#254-256) is never used and should be removed
SafeMath.div(uint256,uint256,string) (   Cash click.sol#270-280) is never used and should be removed
SafeMath.min(uint256,uint256) (   Cash click.sol#319-321) is never used and should be removed
SafeMath.mod(uint256,uint256) (   Cash click.sol#294-296) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (   Cash click.sol#310-317) is never used and should be removed
SafeMath.mul(uint256,uint256) (   Cash click.sol#228-240) is never used and should be removed
SafeMath.sqrt(uint256) (   Cash click.sol#324-335) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (   Cash click.sol#61-67):
        - (success) = recipient.call.value(amount)() (   Cash click.sol#65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable TokenStorage._balances (   Cash click.sol#342) is not in mixedCase
Variable TokenStorage._allowances (   Cash click.sol#343) is not in mixedCase
Variable TokenStorage._totalSupply (   Cash click.sol#344) is not in mixedCase
Variable TokenStorage._supplyHardCap (   Cash click.sol#346) is not in mixedCase
Variable TokenStorage._totalMinted (   Cash click.sol#348) is not in mixedCase
Variable TokenStorage._name (   Cash click.sol#351) is not in mixedCase
Variable TokenStorage._symbol (   Cash click.sol#352) is not in mixedCase
Variable TokenStorage._decimals (   Cash click.sol#353) is not in mixedCase
Variable TokenStorage._owner (   Cash click.sol#356) is not in mixedCase
Variable TokenStorage._delegates (   Cash click.sol#359) is not in mixedCase
Parameter INFLUENCIO.mint(address,uint256)._to (   Cash click.sol#784) is not in mixedCase
Parameter INFLUENCIO.mint(address,uint256)._amount (   Cash click.sol#784) is not in mixedCase
Parameter INFLUENCIO.burn(address,uint256)._account (   Cash click.sol#791) is not in mixedCase
Parameter INFLUENCIO.burn(address,uint256)._amount (   Cash click.sol#791) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (   Cash click.sol#431)" inContext (   Cash click.sol#421-434)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
owner() should be declared external:
```

```
        - Ownable.owner() (   Cash click.sol#451-453)
renounceOwnership() should be declared external:

transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (   Cash click.sol#486-488)
decimals() should be declared external:

symbol() should be declared external:
        - BEP20.symbol() (   Cash click.sol#539-541)
totalSupply() should be declared external:

supplyHardCap() should be declared external:
        - BEP20.supplyHardCap() (   Cash click.sol#550-552)
totalMinted() should be declared external:
        - BEP20.totalMinted() (   Cash click.sol#554-556)
transfer(address,uint256) should be declared external:

allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (   Cash click.sol#581-583)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (   Cash click.sol#592-595)
transferFrom(address,address,uint256) should be declared external:

increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (   Cash click.sol#635-638)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (   Cash click.sol#654-661)
init(string,string,uint8,uint256) should be declared external:

mint(address,uint256) should be declared external:
        - INFLUENCIO.mint(address,uint256) (   Cash click.sol#784-789)
burn(address,uint256) should be declared external:
```

Slither log >> ProxyAdmin.sol

```
INFO:Detectors:
UpgradeableProxy.constructor(address,bytes)._logic (ProxyAdmin.sol#289) lacks a zero-check on :
                - (success) = _logic.delegatecall(_data) (ProxyAdmin.sol#294)
TransparentUpgradeableProxy.upgradeToAndCall(address,bytes).newImplementation (ProxyAdmin.sol#434) lacks a zero-check on :
                - (success) = newImplementation.delegatecall(data) (ProxyAdmin.sol#437)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Modifier TransparentUpgradeableProxy.ifAdmin() (ProxyAdmin.sol#371-377) does not always execute _; or revertReference: https:/
/github.com/crytic/slither/wiki/Detector-Documentation#incorrect-modifier
INFO:Detectors:
Address.isContract(address) (ProxyAdmin.sol#21-30) uses assembly
        - INLINE ASM (ProxyAdmin.sol#28)
Proxy._delegate(address) (ProxyAdmin.sol#156-179) uses assembly
        - INLINE ASM (ProxyAdmin.sol#157-178)
UpgradeableProxy._implementation() (ProxyAdmin.sol#314-320) uses assembly
        - INLINE ASM (ProxyAdmin.sol#317-319)
UpgradeableProxy._setImplementation(address) (ProxyAdmin.sol#335-344) uses assembly
        - INLINE ASM (ProxyAdmin.sol#341-343)
TransparentUpgradeableProxy._admin() (ProxyAdmin.sol#444-450) uses assembly
        - INLINE ASM (ProxyAdmin.sol#447-449)
TransparentUpgradeableProxy._setAdmin(address) (ProxyAdmin.sol#455-462) uses assembly


INFO:Detectors:
Address.sendValue(address,uint256) (ProxyAdmin.sol#60-66) is never used and should be removed
Address.toPayable(address) (ProxyAdmin.sol#38-40) is never used and should be removed
Context._msgData() (ProxyAdmin.sol#79-82) is never used and should be removed
Proxy._implementation() (ProxyAdmin.sol#185) is never used and should be removed

INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (ProxyAdmin.sol#60-66):
        - (success) = recipient.call.value(amount)() (ProxyAdmin.sol#64)
Low level call in ProxyAdmin.getProxyImplementation(TransparentUpgradeableProxy) (ProxyAdmin.sol#224-230):
        - (success,returndata) = address(proxy).staticcall(0x5c60da1b) (ProxyAdmin.sol#227)
Low level call in ProxyAdmin.getProxyAdmin(TransparentUpgradeableProxy) (ProxyAdmin.sol#239-245):
        - (success,returndata) = address(proxy).staticcall(0xf851a440) (ProxyAdmin.sol#242)
Low level call in UpgradeableProxy.constructor(address,bytes) (ProxyAdmin.sol#289-297):
```

```
Low level call in TransparentUpgradeableProxy.upgradeToAndCall(address,bytes) (ProxyAdmin.sol#434-439):
        - (success) = newImplementation.delegatecall(data) (ProxyAdmin.sol#437)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Redundant expression "this (ProxyAdmin.sol#80)" inContext (ProxyAdmin.sol#69-83)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (ProxyAdmin.sol#102-104)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (ProxyAdmin.sol#128-131)
transferOwnership(address) should be declared external:

getProxyImplementation(TransparentUpgradeableProxy) should be declared external:
        - ProxyAdmin.getProxyImplementation(TransparentUpgradeableProxy) (ProxyAdmin.sol#224-230)
getProxyAdmin(TransparentUpgradeableProxy) should be declared external:
        - ProxyAdmin.getProxyAdmin(TransparentUpgradeableProxy) (ProxyAdmin.sol#239-245)
changeProxyAdmin(TransparentUpgradeableProxy,address) should be declared external:

upgrade(TransparentUpgradeableProxy,address) should be declared external:
        - ProxyAdmin.upgrade(TransparentUpgradeableProxy,address) (ProxyAdmin.sol#265-267)
upgradeAndCall(TransparentUpgradeableProxy,address,bytes) should be declared external:
        - ProxyAdmin.upgradeAndCall(TransparentUpgradeableProxy,address,bytes) (ProxyAdmin.sol#277-279)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Slither log >> TransparentUpgradeableProxy.sol

```
INFO:Detectors:
UpgradeableProxy.constructor(address,bytes)._logic (TransparentUpgradeableProxy.sol#141) lacks a zero-check on :
                - (success) = _logic.delegatecall(_data) (TransparentUpgradeableProxy.sol#146)
TransparentUpgradeableProxy.upgradeToAndCall(address,bytes).newImplementation (TransparentUpgradeableProxy.sol#286) lacks a ze
ro-check on :
                - (success) = newImplementation.delegatecall(data) (TransparentUpgradeableProxy.sol#289)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Modifier TransparentUpgradeableProxy.ifAdmin() (TransparentUpgradeableProxy.sol#223-229) does not always execute _; or revertR
eference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-modifier
INFO:Detectors:
Address.isContract(address) (TransparentUpgradeableProxy.sol#23-32) uses assembly
        - INLINE ASM (TransparentUpgradeableProxy.sol#30)
Proxy._delegate(address) (TransparentUpgradeableProxy.sol#76-99) uses assembly
        - INLINE ASM (TransparentUpgradeableProxy.sol#77-98)
UpgradeableProxy._implementation() (TransparentUpgradeableProxy.sol#166-172) uses assembly
        - INLINE ASM (TransparentUpgradeableProxy.sol#169-171)


TransparentUpgradeableProxy._admin() (TransparentUpgradeableProxy.sol#296-302) uses assembly
        - INLINE ASM (TransparentUpgradeableProxy.sol#299-301)
TransparentUpgradeableProxy._setAdmin(address) (TransparentUpgradeableProxy.sol#307-314) uses assembly
        - INLINE ASM (TransparentUpgradeableProxy.sol#311-313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.sendValue(address,uint256) (TransparentUpgradeableProxy.sol#62-68) is never used and should be removed
Address.toPayable(address) (TransparentUpgradeableProxy.sol#40-42) is never used and should be removed
Proxy._implementation() (TransparentUpgradeableProxy.sol#105) is never used and should be removed

INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (TransparentUpgradeableProxy.sol#62-68):
        - (success) = recipient.call.value(amount)() (TransparentUpgradeableProxy.sol#66)
Low level call in UpgradeableProxy.constructor(address,bytes) (TransparentUpgradeableProxy.sol#141-149):
        - (success) = _logic.delegatecall(_data) (TransparentUpgradeableProxy.sol#146)
Low level call in TransparentUpgradeableProxy.upgradeToAndCall(address,bytes) (TransparentUpgradeableProxy.sol#286-291):
        - (success) = newImplementation.delegatecall(data) (TransparentUpgradeableProxy.sol#289)
```

```
INFO:Slither:TransparentUpgradeableProxy.sol analyzed (4 contracts with 75 detectors), 15 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

Acts.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 28:8:

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: 64:27:

Gas & Economy

Gas costs:

Gas requirement of function Acts.burn 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: 740:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 77:4:

Miscellaneous

Similar variable names:

Acts.burn(address,uint256) : Variables have very similar names "_account" and "_amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 741:24:

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: 698:8:

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: 327:20:

CASHCLICK.sol

Security

Inline assembly:
The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 28:8:

Block timestamp:
Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp".
"block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 870:16:

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: 64:27:

Gas & Economy

Gas costs:
Gas requirement of function CASHCLICK mint 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: 789:4:

Gas costs:
Gas requirement of function CASHCLICK getPriorVotes 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: 895:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 79:4:

Miscellaneous

Similar variable names:

CASH CLICK .burn(address,uint256) : Variables have very similar names "_account" and "_amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 797:24:

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: 790:8:

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: 920:36:

ProxyAdmin.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 28:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 459:8:

Low level calls:

Use of "delegatecall": should be avoided whenever possible.
External code, that is called can change the state of the calling contract and send ether from the caller's balance.
If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 437:26:

Gas & Economy

Gas costs:

Fallback function of contract TransparentUpgradeableProxy requires too much gas (infinite).
If the fallback function requires more than 2300 gas, the contract cannot receive Ether.
Pos: 201:4:

Gas costs:

Gas requirement of function TransparentUpgradeableProxy.upgradeToAndCall 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: 434:4:

Miscellaneous

Constant/View/Pure functions:

Address.isContract(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 21:4:

Constant/View/Pure functions:

TransparentUpgradeableProxy._admin() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 444:4:

No return:

TransparentUpgradeableProxy._admin(): Defines a return type but never explicitly returns a value.
Pos: 444:4:

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: 413:8:

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: 438:8:

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: 468:8:

TransparentUpgradeableProxy.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 28:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 309:8:

Low level calls:

Use of "delegatecall": should be avoided whenever possible.
External code, that is called can change the state of the calling contract and send ether from the caller's balance.
If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 287:26:

Gas & Economy

Gas costs:

Fallback function of contract TransparentUpgradeableProxy requires too much gas (infinite).
If the fallback function requires more than 2300 gas, the contract cannot receive Ether.
Pos: 119:4:

Gas costs:

Gas requirement of function TransparentUpgradeableProxy.upgradeToAndCall 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: 284:4:

Miscellaneous

Constant/View/Pure functions:

Address.isContract(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 21:4:

Constant/View/Pure functions:

TransparentUpgradeableProxy._admin() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 294:4:

No return:

TransparentUpgradeableProxy._admin(): Defines a return type but never explicitly returns a value.

Pos: 294:4:

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: 288:8:

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: 318:8:

Solhint Linter

Acts.sol

```
Acts.sol:1:1: Error: Compiler version ^0.5.5 does not satisfy the r
semver requirement
Acts.sol:64:28: Error: Avoid using low level calls.
Acts.sol:385:28: Error: Code contains empty blocks
```

CASHCLICK.sol

```
CASH CLICK.sol:1:1: Error: Compiler version ^0.5.5 does not satisfy
the r semver requirement
CASH CLICK.sol:64:28: Error: Avoid using low level calls.
CASH CLICK.sol:429:28: Error: Code contains empty blocks
CASH CLICK.sol:870:17: Error: Avoid to make time-based decisions in
your business logic
```

ProxyAdmin.sol

```
ProxyAdmin.sol:2:1: Error: Compiler version ^0.5.0 does not satisfy
the r semver requirement
ProxyAdmin.sol:64:28: Error: Avoid using low level calls.
ProxyAdmin.sol:157:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ProxyAdmin.sol:211:41: Error: Code contains empty blocks
```

TransparentUpgradeableProxy.sol

```
TransparentUpgradeableProxy.sol:1:1: Error: Compiler version ^0.5.5
does not satisfy the r semver requirement
TransparentUpgradeableProxy.sol:64:28: Error: Avoid to use low level
calls.
TransparentUpgradeableProxy.sol:75:9: Error: Avoid to use inline
assembly. It is acceptable only in rare cases
TransparentUpgradeableProxy.sol:129:41: Error: Code contains empty
blocks
```

Software analysis result:

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


AntiHACK
.me