EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Blue Social Token
Platform:	Ethereum
Website:	https://blue.social
Language:	Solidity
Date:	May 9th, 2023

Table of contents

Introduction

EtherAuthority was contracted by Blue Social to perform the Security audit of the Blue Social Token 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 May 9th, 2023.

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

- Blue Social is a protocol having functionalities like Presale, Tokensale, Airdrop, etc.
- The Blue Social contract inherits IERC20, MerkleProof, ERC20Upgradeable, ERC20BurnableUpgradeable, PausableUpgradeable, SafeERC20, OwnableUpgradeable, Initializable, UUPSUpgradeable, ERC20VotesUpgradeable, standard smart contracts from the OpenZeppelin library.
- These OpenZeppelin contracts are considered community audited and time tested, and hence are not part of the audit scope.

Audit scope

Name	**Code Review and Security Analysis Report for Blue Social Smart Contracts**
Platform	**Ethereum / Solidity**
File 1	BlueSocialToken.sol
File 1 MD5 Hash	C4C42B5223A1398E304CFFE32DDF268C
File 2	Presale.sol
File 2 MD5 Hash	67EA87A5EBA2AE47596DE0FFF4A29FB8
File 3	AirdropController.sol
File 3 MD5 Hash	354DE86CDA35ED07C423060357F0D246
File 4	BSTAirdrop.sol

File 4 MD5 Hash	2A8DC029D35459F6256698E3DCDEC224
File 5	EIP712Base.sol
File 5 MD5 Hash	F86C897939A9626FE8A428DF3199D05E
File 6	Forwarder.sol
File 6 MD5 Hash	1C8E9ED96F25DAB04BB90C194D96872C
File 7	NetworkAgnostic.sol
File 7 MD5 Hash	1BE6DE17F2F55FC4D4CA4F0B50F3F440
File 8	TokenSale.sol
File 8 MD5 Hash	B5B160D0C144B41B8D8269929B6AEBB4
File 9	MetaTransactionLib.sol
File 9 MD5 Hash	3DD60BFF324E1002D6411DC826DBF7CC
Audit Date	May 9th, 2023

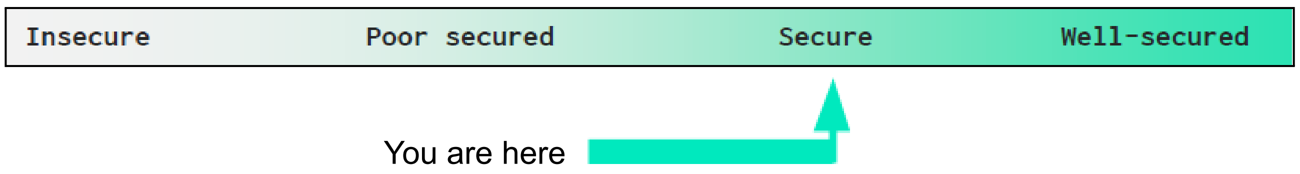
Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 BlueSocialToken.sol **<u>Owner Specifications:</u>** ● Triggers stopped state. ● Returns to normal state. **<u>Other Specifications:</u>** ● Name: BlueSocialToken ● Symbol: BST ● Total Supply: 1 Million	**YES, This is valid.**
File 2 Presale.sol **<u>Owner Specifications:</u>** ● Set a start - stop pre-sale. ● withdraw tokens. ● Set a rate value. ● Set Available tokens. ● Set a Wallet Receiver address. ● Set a HardCap and SoftCap value.	**YES, This is valid.**
File 3 AirdropController.sol ● This contract is used to control bulk airdrop different tokens.	**YES, This is valid.**
File 4 BSTAirdrop.sol ● BlueSocialAirdrop is used to claim data.	**YES, This is valid.**
File 5 EIP712Base.sol ● EIP712Base is used to recover signers from signature signed using EIP712 formatted data.	**YES, This is valid.**

File 6 Forwarder.sol ● Forwarded contract is used to pause and unpause the state. **<u>Owner Specifications:</u>** ● Triggers stopped state. ● Returns to normal state.	**YES, This is valid.**
File 7 NetworkAgnostic.sol ● NetworkAgnostic has functionality like: executeMetaTransaction, hashMetaTransaction, etc.	**YES, This is valid.**
File 8 TokenSale.sol **<u>Owner Specifications:</u>** ● The Owner can end the sale. ● Update Price by the Owner.	**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 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	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	Passed
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 9 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 Blue Social Token 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 Blue Social Token.

The Blue Social team has not provided 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 a BST Token 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 not well commented. But the logic is straightforward. 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://blue.social which provided rich information about the project architecture and tokenomics.

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

BlueSocialToken.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	__ERC20_init	internal	access only Initializing	No Issue
7	__ERC20_init_unchained	internal	access only Initializing	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	balanceOf	read	Passed	No Issue
13	transfer	write	Passed	No Issue
14	allowance	read	Passed	No Issue
15	approve	write	Passed	No Issue
16	transferFrom	write	Passed	No Issue
17	increaseAllowance	write	Passed	No Issue
18	decreaseAllowance	write	Passed	No Issue
19	_transfer	internal	Passed	No Issue
20	_mint	internal	Passed	No Issue
21	_burn	internal	Passed	No Issue
22	_approve	internal	Passed	No Issue
23	_spendAllowance	internal	Passed	No Issue
24	_beforeTokenTransfer	internal	Passed	No Issue
25	_afterTokenTransfer	internal	Passed	No Issue
26	__ERC20Burnable_init	internal	access only Initializing	No Issue
27	__ERC20Burnable_init_unchained	internal	access only Initializing	No Issue
28	burn	write	Passed	No Issue
29	burnFrom	write	Passed	No Issue
30	__Pausable_init	internal	access only Initializing	No Issue
31	__Pausable_init_unchained	internal	access only Initializing	No Issue
32	whenNotPaused	modifier	Passed	No Issue
33	whenPaused	modifier	Passed	No Issue
34	paused	read	Passed	No Issue

35	_requireNotPaused	internal	Passed	No Issue
36	_requirePaused	internal	Passed	No Issue
37	_pause	internal	Passed	No Issue
38	_unpause	internal	Passed	No Issue
39	__Ownable_init	internal	access only Initializing	No Issue
40	__Ownable_init_unchained	internal	access only Initializing	No Issue
41	onlyOwner	modifier	Passed	No Issue
42	owner	read	Passed	No Issue
43	_checkOwner	internal	Passed	No Issue
44	renounceOwnership	write	access onlyOwner	No Issue
45	transferOwnership	write	access onlyOwner	No Issue
46	_transferOwnership	internal	Passed	No Issue
47	__ERC20Votes_init	internal	access only Initializing	No Issue
48	__ERC20Votes_init_unchained	internal	access only Initializing	No Issue
49	checkpoints	read	Passed	No Issue
50	numCheckpoints	read	Passed	No Issue
51	delegates	read	Passed	No Issue
52	getVotes	read	Passed	No Issue
53	getPastVotes	read	Passed	No Issue
54	getPastTotalSupply	read	Passed	No Issue
55	_checkpointsLookup	read	Passed	No Issue
56	delegate	write	Passed	No Issue
57	delegateBySig	write	Passed	No Issue
58	_maxSupply	internal	Passed	No Issue
59	_mint	internal	Passed	No Issue
60	_burn	internal	Passed	No Issue
61	_afterTokenTransfer	internal	Passed	No Issue
62	_delegate	internal	Passed	No Issue
63	_moveVotingPower	write	Passed	No Issue
64	_writeCheckpoint	write	Passed	No Issue
65	_add	write	Passed	No Issue
66	_subtract	write	Passed	No Issue
67	_unsafeAccess	write	Passed	No Issue
68	__UUPSUpgradeable_init	internal	access only Initializing	No Issue
69	__UUPSUpgradeable_init_unchained	internal	access only Initializing	No Issue
70	onlyProxy	modifier	Passed	No Issue
71	notDelegated	modifier	Passed	No Issue
72	proxiableUUID	external	Passed	No Issue
73	upgradeTo	write	Passed	No Issue
74	upgradeToAndCall	write	Passed	No Issue
75	_authorizeUpgrade	internal	Passed	No Issue

76	initialize	write	access only Initializing	No Issue
77	pause	write	access onlyOwner	No Issue
78	unpause	write	access onlyOwner	No Issue
79	_mint	internal	Passed	No Issue
80	_burn	internal	Passed	No Issue
81	_beforeTokenTransfer	internal	Passed	No Issue
82	_authorizeUpgrade	internal	access onlyOwner	No Issue
83	_afterTokenTransfer	internal	Passed	No Issue
84	approveAndCall	write	Passed	No Issue

Presale.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	transferOwnership	write	access onlyOwner	No Issue
5	nonReentrant	modifier	Passed	No Issue
6	receive	external	Passed	No Issue
7	startICO	external	access onlyOwner	No Issue
8	stopICO	external	access onlyOwner	No Issue
9	buyTokens	write	Passed	No Issue
10	_preValidatePurchase	internal	Passed	No Issue
11	claimTokens	external	Passed	No Issue
12	_getTokenAmount	internal	Passed	No Issue
13	_forwardFunds	internal	Passed	No Issue
14	withdraw	external	access onlyOwner	No Issue
15	checkContribution	read	Passed	No Issue
16	setRate	external	access onlyOwner	No Issue
17	setAvailableTokens	write	access onlyOwner	No Issue
18	weiRaised	read	Passed	No Issue
19	setWalletReceiver	external	access onlyOwner	No Issue
20	setHardCap	external	access onlyOwner	No Issue
21	setSoftCap	external	access onlyOwner	No Issue
22	setMaxPurchase	external	access onlyOwner	No Issue
23	setMinPurchase	external	access onlyOwner	No Issue
24	takeTokens	write	access onlyOwner	No Issue
25	refundMe	write	Passed	No Issue
26	icoActive	modifier	Passed	No Issue
27	icoNotActive	modifier	Passed	No Issue

AirdropController.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	bulkAirdropERC20	write	Passed	No Issue
3	bulkAirdropERC721	write	Passed	No Issue
4	bulkAirdropERC1155	write	Passed	No Issue

BSTAirdrop.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	claim	external	Passed	No Issue
3	canClaim	read	Passed	No Issue

EIP712Base.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	__EIP712BaseInit	write	Passed	No Issue
7	getDomainSeperator	read	Passed	No Issue
8	toTypedMessageHash	internal	Passed	No Issue

Forwarder.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	transferOwnership	write	access onlyOwner	No Issue
5	whenNotPaused	modifier	Passed	No Issue
6	whenPaused	modifier	Passed	No Issue
7	paused	read	Passed	No Issue
8	_requireNotPaused	internal	Passed	No Issue
9	_requirePaused	internal	Passed	No Issue
10	_pause	internal	Passed	No Issue
11	_unpause	internal	Passed	No Issue

12	pause	write	access onlyOwner	No Issue
13	unpause	write	access onlyOwner	No Issue
14	DOMAIN_SEPARATOR	external	Passed	No Issue
15	getNonce	read	Passed	No Issue
16	verify	read	Passed	No Issue
17	execute	write	Passed	No Issue
18	addSenderToWhitelist	write	access onlyOwner	No Issue
19	removeSenderFromWhitelist	write	access onlyOwner	No Issue
20	isWhitelisted	read	Passed	No Issue
21	killForwarder	write	access onlyOwner	No Issue
22	_checkOwner	internal	Passed	No Issue
23	renounceOwnership	write	access onlyOwner	No Issue
24	_transferOwnership	internal	Passed	No Issue

NetworkAgnostic.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__EIP712BaseInit	write	Passed	No Issue
3	getDomainSeperator	read	Passed	No Issue
4	toTypedMessageHash	internal	Passed	No Issue
5	executeMetaTransaction	write	Passed	No Issue
6	hashMetaTransaction	internal	Passed	No Issue
7	getNonce	read	Passed	No Issue
8	verify	internal	Passed	No Issue
9	receive	external	Passed	No Issue

TokenSale.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	__BlueSocialTokenSaleInit	write	Passed	No Issue
7	multiply	internal	Passed	No Issue
8	buyTokens	write	Passed	No Issue
9	endSale	write	Passed	No Issue
10	updatePrice	write	Passed	No Issue

MetaTransactionLib.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_msgSender	internal	Passed	No Issue
3	__EIP712BaseInit	write	Passed	No Issue
4	getDomainSeperator	read	Passed	No Issue
5	toTypedMessageHash	internal	Passed	No Issue
6	executeMetaTransaction	write	Passed	No Issue
7	hashMetaTransaction	internal	Passed	No Issue
8	getNonce	read	Passed	No Issue
9	verify	internal	Passed	No Issue
10	receive	external	Passed	No Issue
11	__AccessControl_init	internal	Passed	No Issue
12	__AccessControl_init_unchained	internal	Passed	No Issue
13	onlyRole	modifier	Passed	No Issue
14	supportsInterface	read	Passed	No Issue
15	hasRole	read	Passed	No Issue
16	_checkRole	internal	Passed	No Issue
17	_checkRole	internal	Passed	No Issue
18	getRoleAdmin	read	Passed	No Issue
19	grantRole	write	Passed	No Issue
20	revokeRole	write	Passed	No Issue
21	renounceRole	write	Passed	No Issue
22	_setupRole	internal	Passed	No Issue
23	_setRoleAdmin	internal	Passed	No Issue
24	_grantRole	internal	Passed	No Issue
25	_revokeRole	internal	Passed	No Issue
26	__ERC20_init	internal	Passed	No Issue
27	__ERC20_init_unchained	internal	Passed	No Issue
28	name	read	Passed	No Issue
29	symbol	read	Passed	No Issue
30	decimals	read	Passed	No Issue
31	totalSupply	read	Passed	No Issue
32	balanceOf	read	Passed	No Issue
33	transfer	write	Passed	No Issue
34	allowance	read	Passed	No Issue
35	approve	write	Passed	No Issue
36	transferFrom	write	Passed	No Issue
37	increaseAllowance	write	Passed	No Issue
38	decreaseAllowance	write	Passed	No Issue
39	_transfer	internal	Passed	No Issue
40	_mint	internal	Passed	No Issue
41	_burn	internal	Passed	No Issue
42	_approve	internal	Passed	No Issue

43	_spendAllowance	internal	Passed	No Issue
44	_beforeTokenTransfer	internal	Passed	No Issue
45	_afterTokenTransfer	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 Severity

No critical severity vulnerabilities were found in the smart contract code.

High Severity

No high severity vulnerabilities were found in the contract code.

Medium

No medium severity vulnerabilities were found in the contract code.

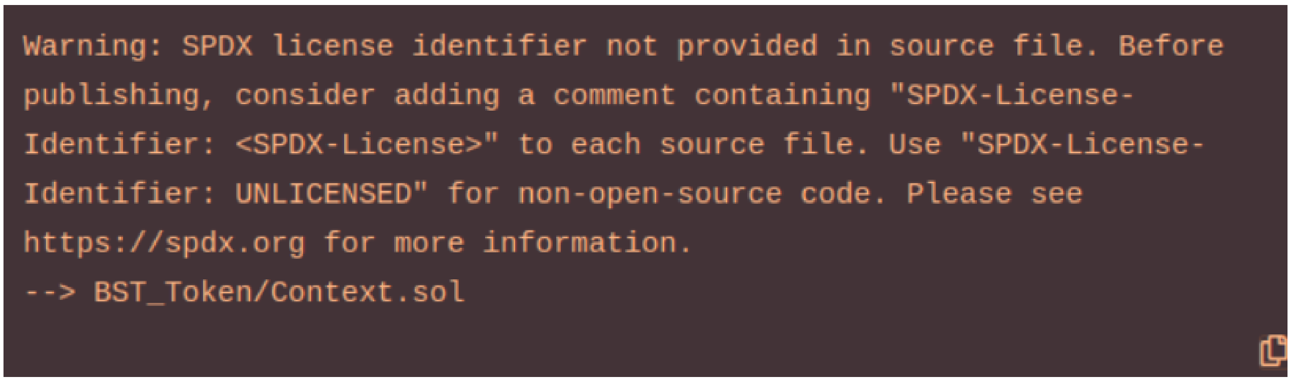
Low

No low severity vulnerabilities were found in the contract code.

Very Low / Informational / Best practices:

(1) Compile time warnings:

Presale.sol

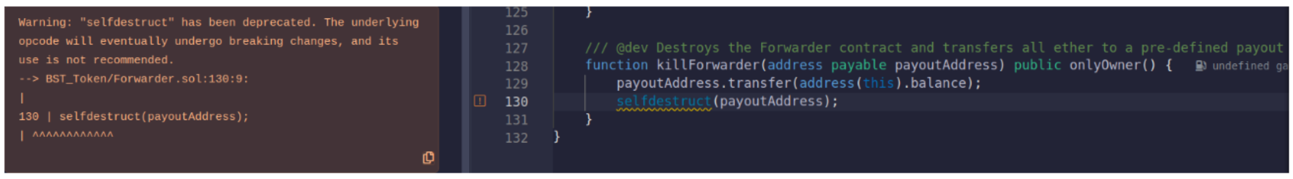


```
Warning: SPDX license identifier not provided in source file. Before
publishing, consider adding a comment containing "SPDX-License-
Identifier: <SPDX-License>" to each source file. Use "SPDX-License-
Identifier: UNLICENSED" for non-open-source code. Please see
https://spdx.org for more information.
--> BST_Token/Context.sol
```

Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code.

Resolution: Add SPDX-License-Identifier in Context.sol file.

Forwarder.sol



```
Warning: "selfdestruct" has been deprecated. The underlying
opcode will eventually undergo breaking changes, and its
use is not recommended.
--> BST_Token/Forwarder.sol:130:9:
|
130 | selfdestruct(payoutAddress);
| ^^^^^^^^^^^^
```

```
125         }
126 
127         /// @dev Destroys the Forwarder contract and transfers all ether to a pre-defined payout
128         function killForwarder(address payable payoutAddress) public onlyOwner() {
129             payoutAddress.transfer(address(this).balance);
130             selfdestruct(payoutAddress);
131         }
132     }
```

Warning: "selfdestruct" has been deprecated. The underlying opcode will eventually undergo breaking changes, and its use is not recommended.

(2) SafeMath Library: **Presale.sol**

SafeMath Library is used in this contract code, but the compiler version is greater than or equal to 0.8.0, Then it will be not required to use, solidity automatically handles overflow/underflow.

Resolution: Remove the SafeMath library and use normal math operators, It will improve code size, and less gas consumption.

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:

BlueSocialToken.sol

- pause: Triggers stopped by the owner.
- unpause: Returns to normal state by the owner.
- _authorizeUpgrade: Authorize upgrade by the owner.

Presale.sol

- startICO: Start Pre-sale can be set by the owner.
- stopICO: Stop Pe-sale can be set by the owner.
- withdraw: Owner can withdraw tokens.

- setRate: Rate value can be set by the owner.
- setAvailableTokens: Available tokens can be set by the owner.
- setWalletReceiver: Wallet Receiver address can be set by the owner.
- setHardCap: HardCap value can be set by the owner.
- setSoftCap:SoftCap value can be set by the owner.
- setMaxPurchase: Maximum purchase value can be set by the owner.
- setMinPurchase: Minimum purchase value can be set by the owner.

Ownable.sol

- transferOwnership: Current owner can transfer ownership of the contract to a new account.

OwnableUpgradeable.sol

- renounceOwnership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transferOwnership: Current owner can transfer ownership of the contract to a new account.
- _checkOwner: Throws if the sender is not the owner.

Forwarder.sol

- pause: Triggers stopped by the owner.
- unpause: Returns to normal state by the owner.
- addSenderToWhitelist: Only whitelisted addresses are allowed to broadcast meta-transactions by the owner.
- removeSenderFromWhitelist: Removes a whitelisted address by the owner.

TokenSale.sol

- endSale: End sale by the owner.
- updatePrice: Update price by the owner.

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 the given objects as files. We had observed some informational issues in the smart contracts, but those are not critical ones. **So, the 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.

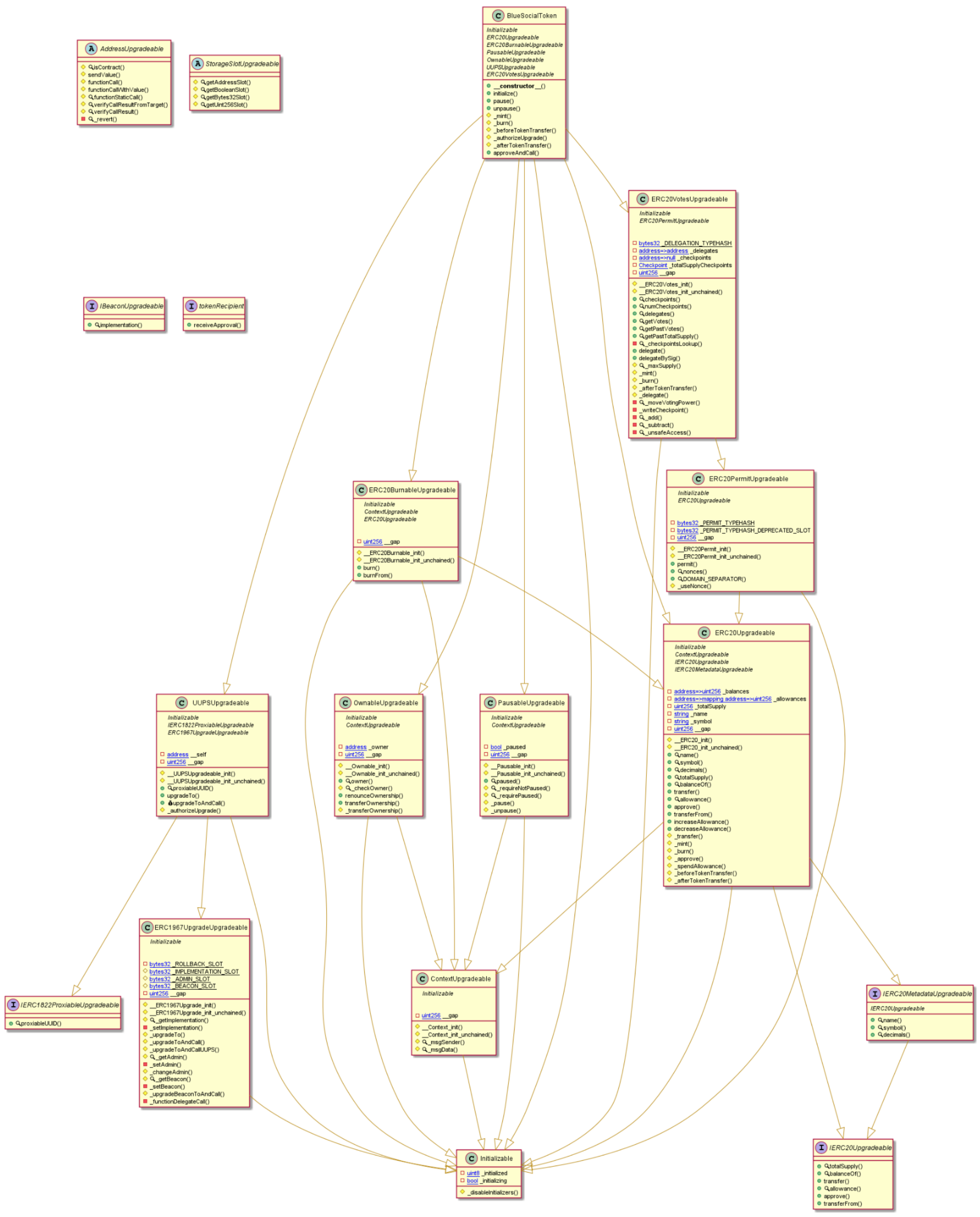
Technical Disclaimer

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.

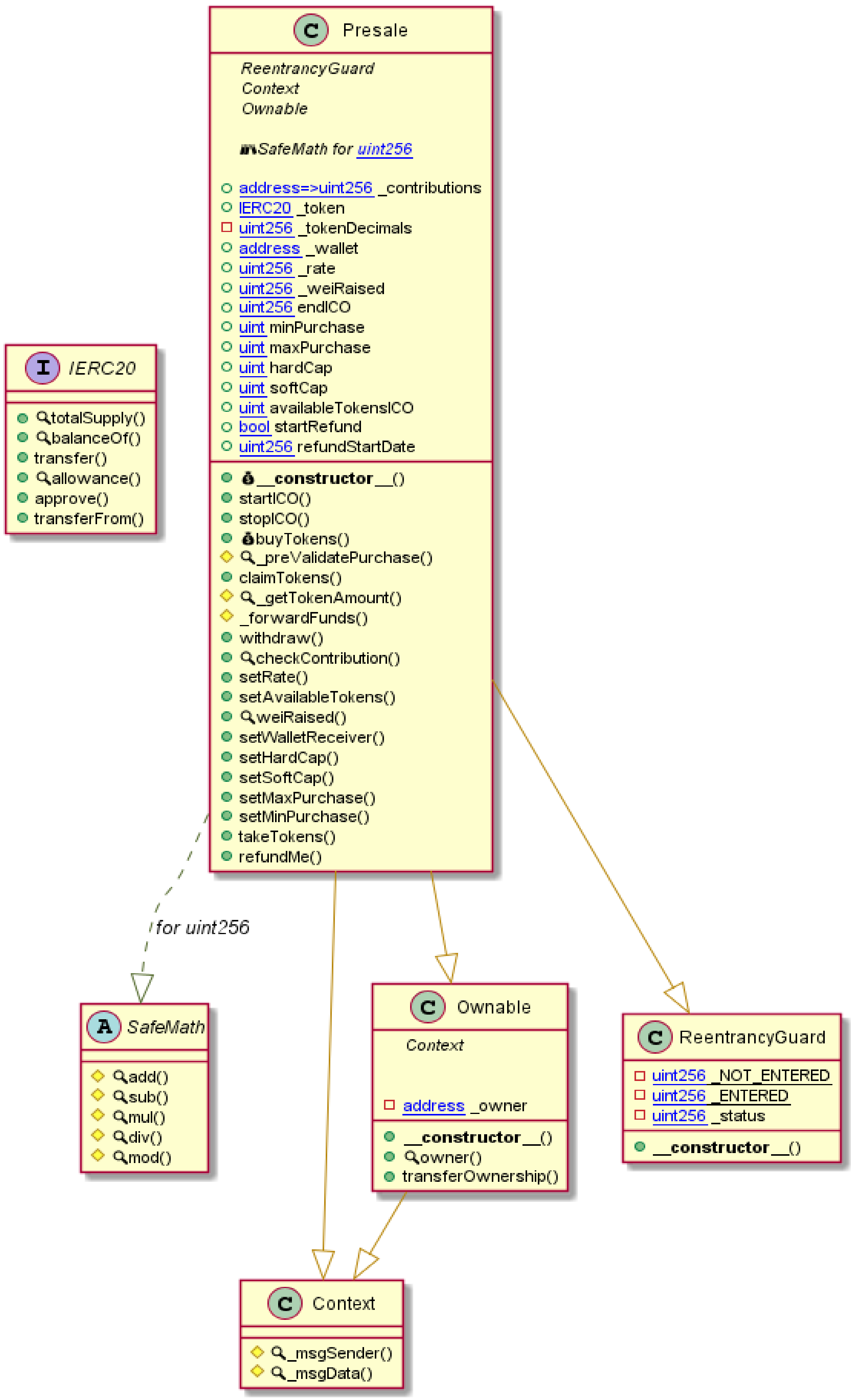
Appendix

Code Flow Diagram - Blue Social Token

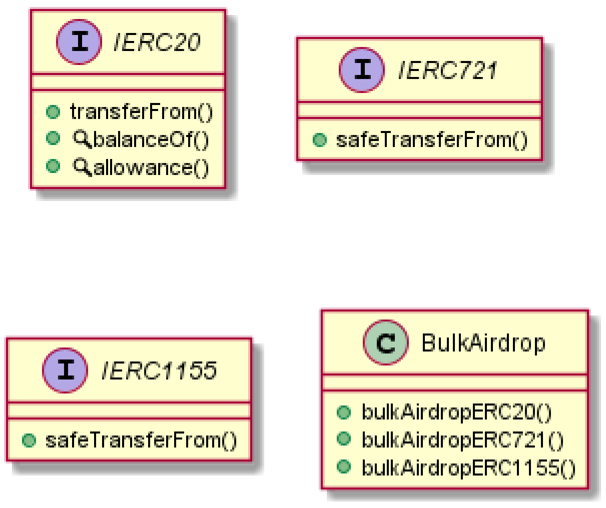
BlueSocialToken Diagram



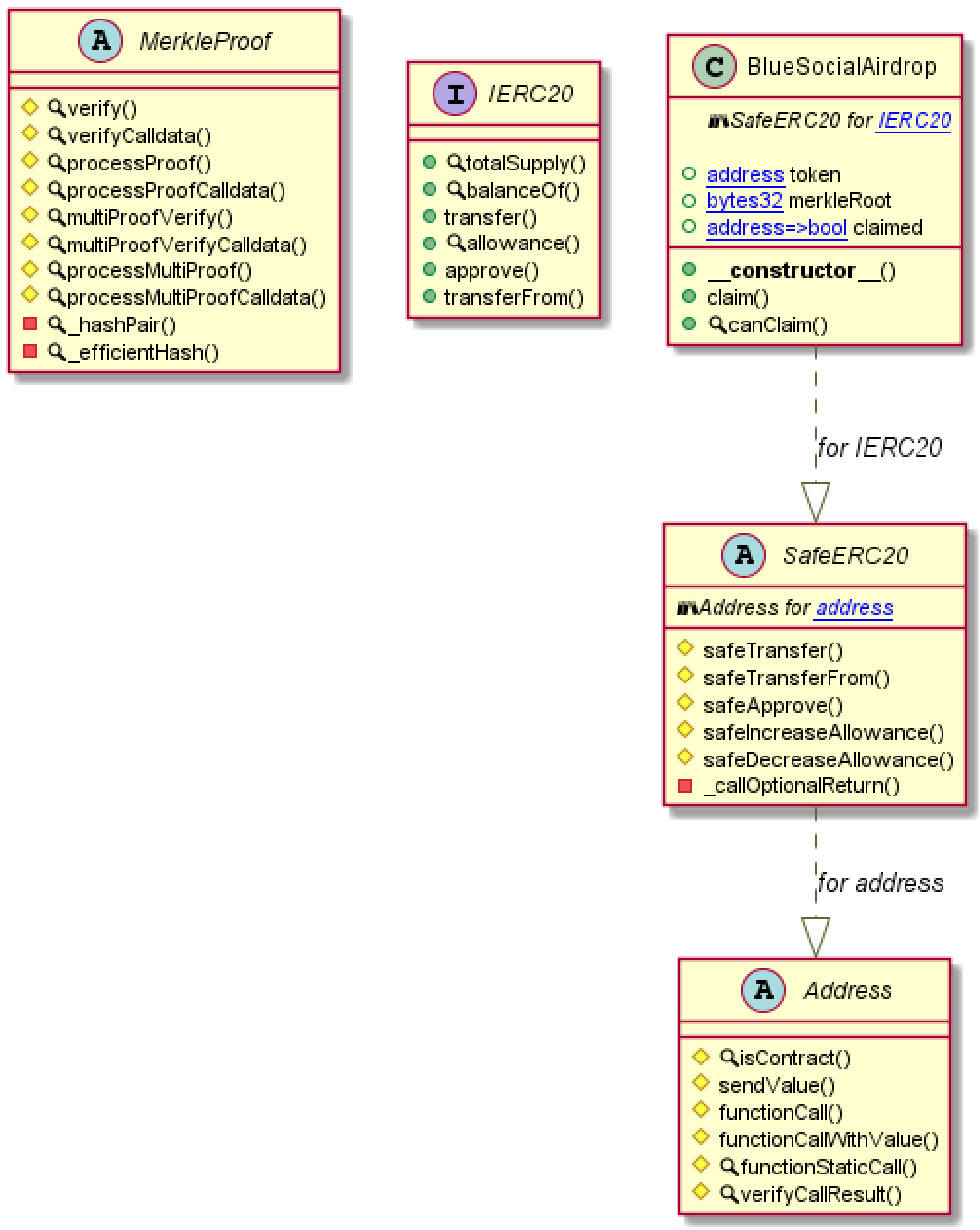
Presale Diagram



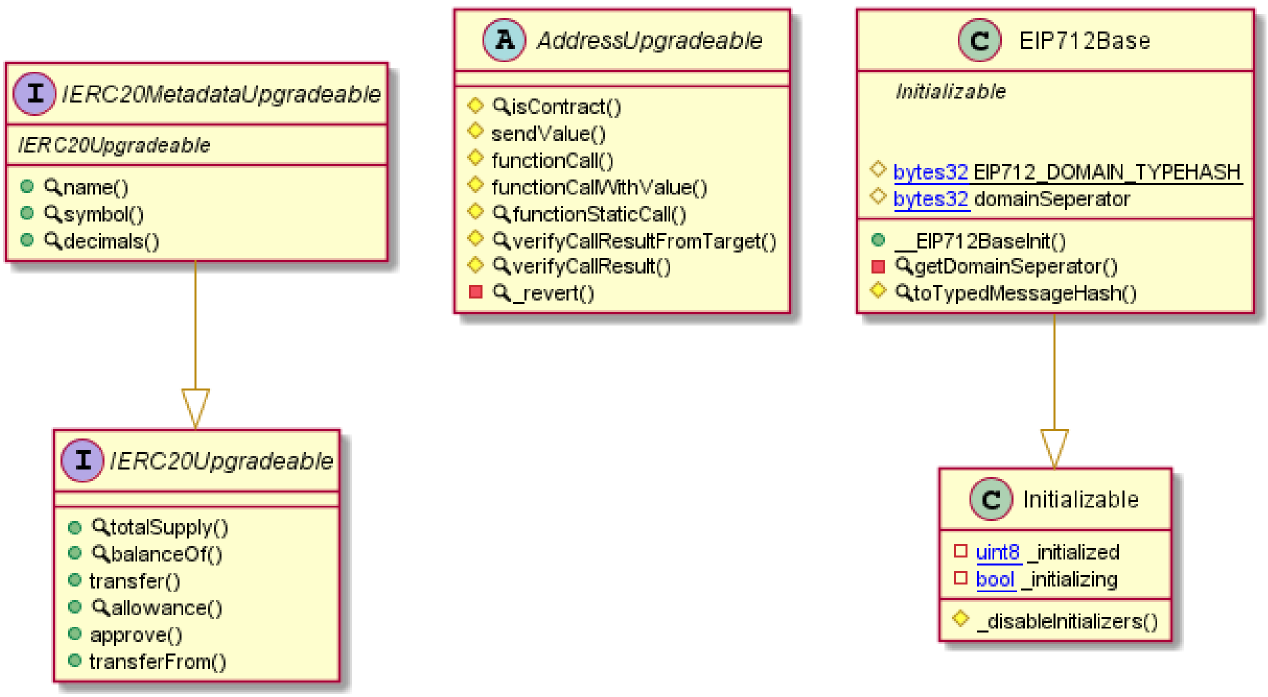
AirdropController Diagram



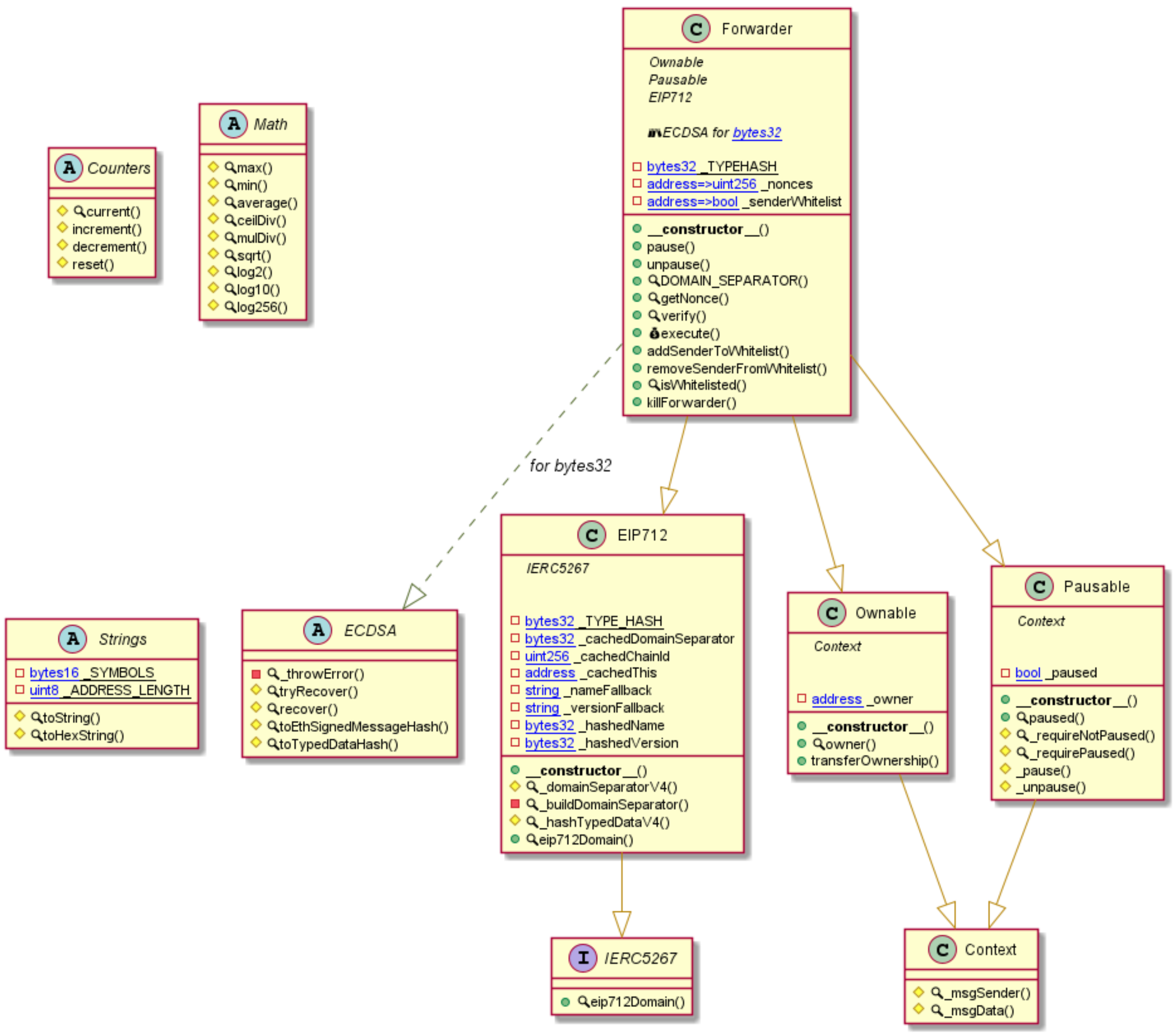
BSTAirdrop Diagram



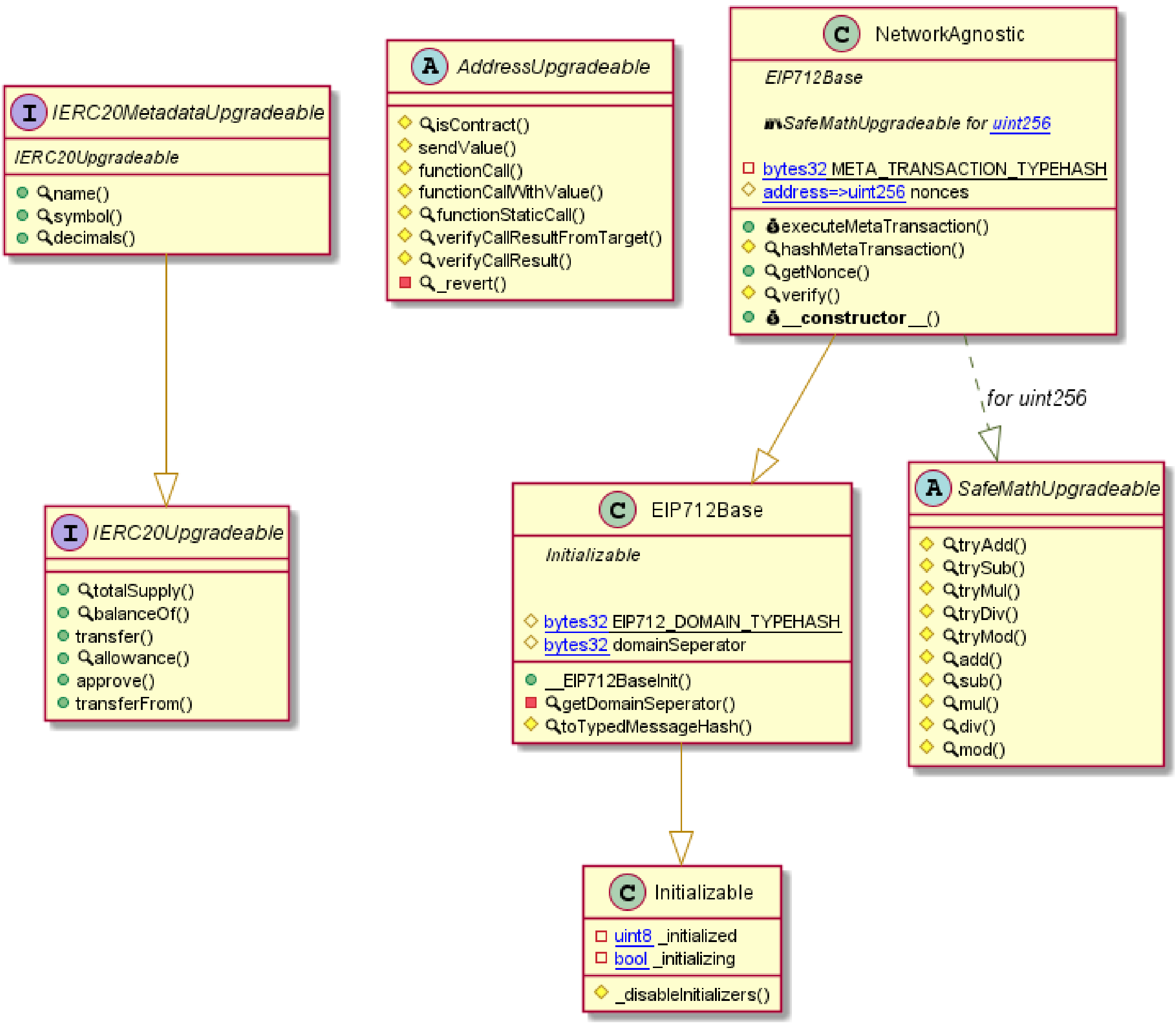
EIP712Base Diagram



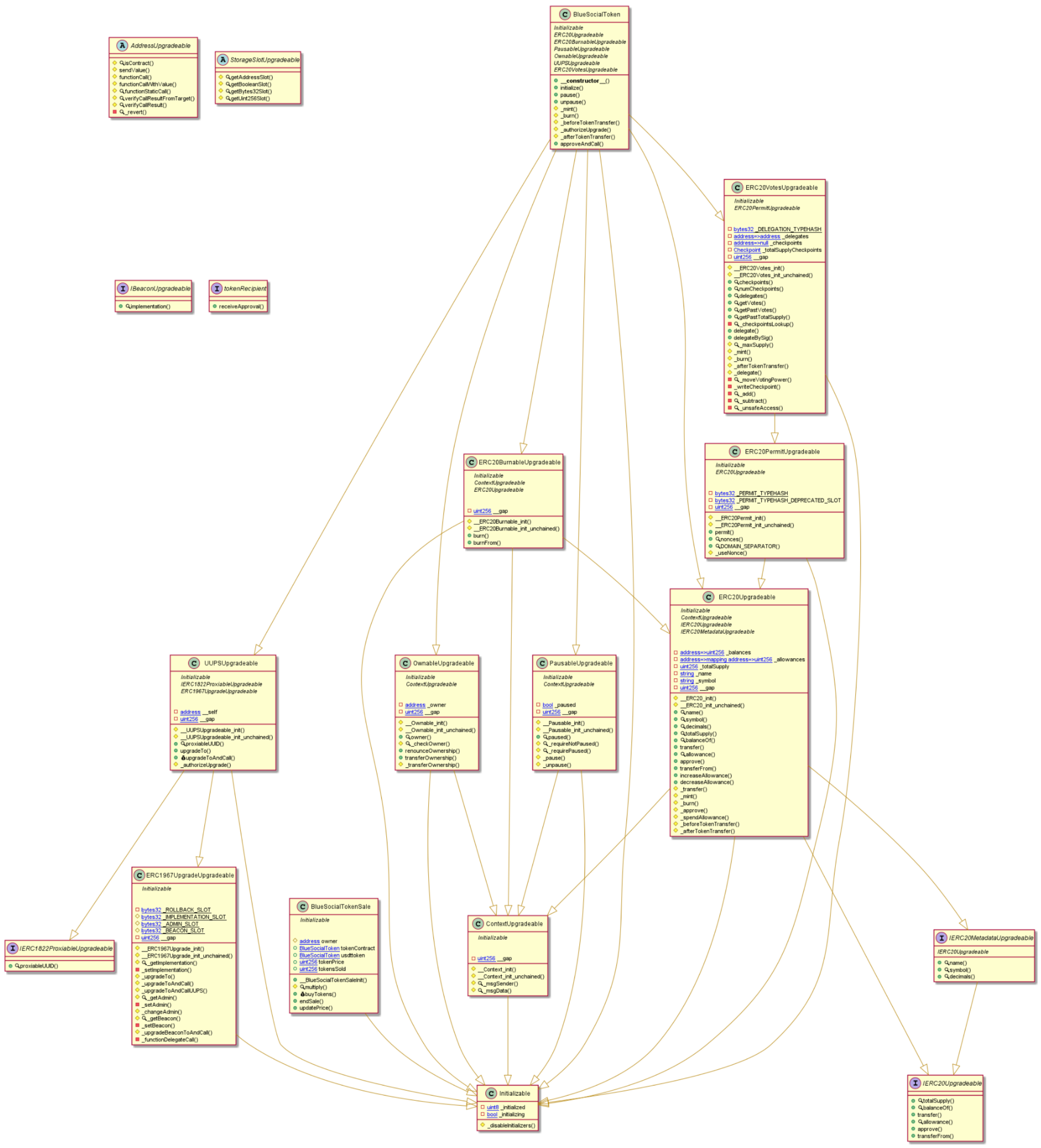
Forwarder Diagram



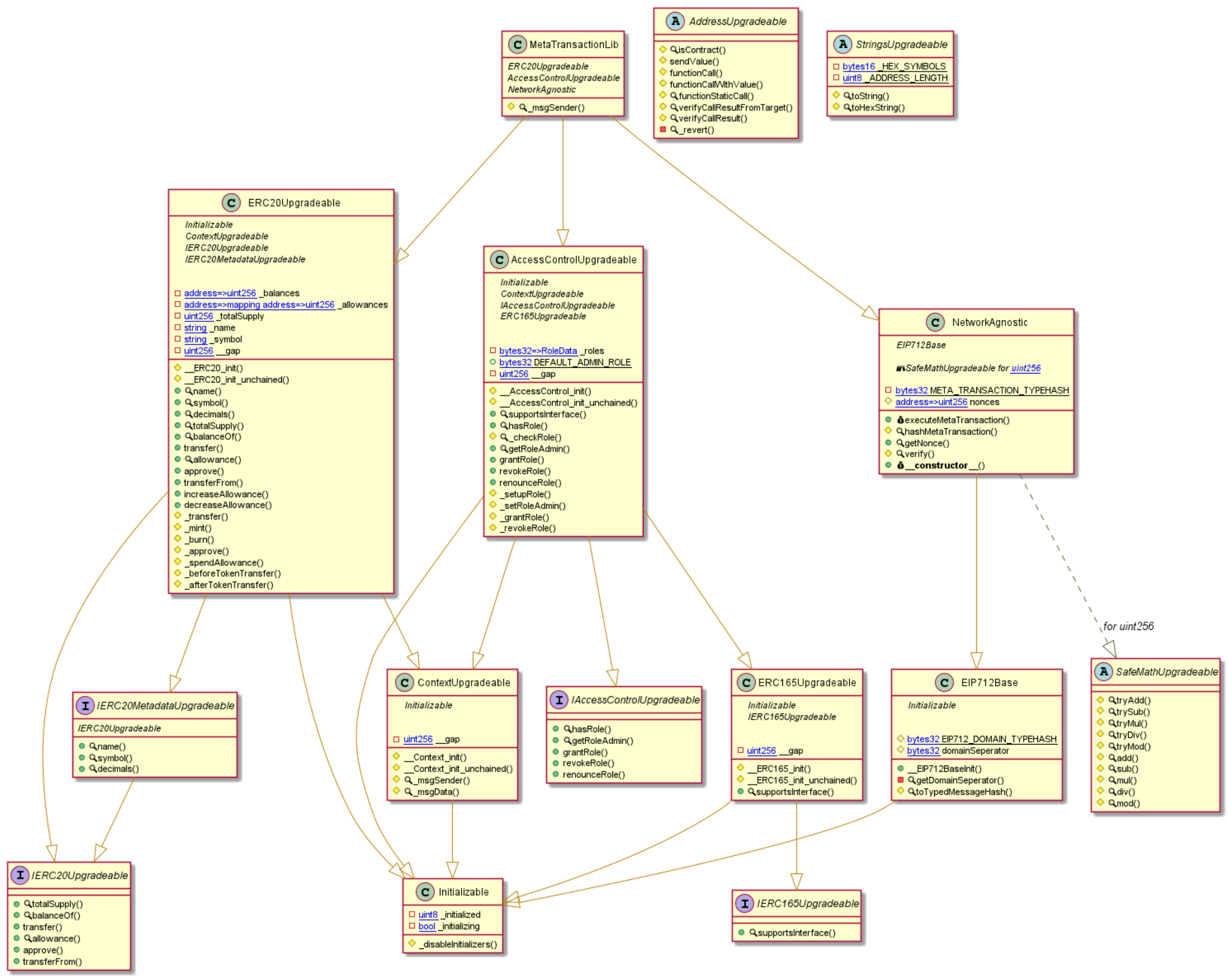
NetworkAgnostic Diagram



TokenSale Diagram

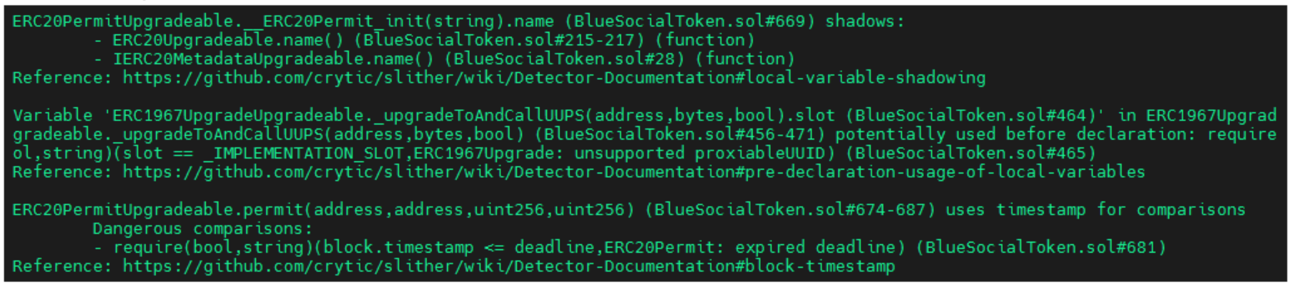


MetaTransactionLib Diagram

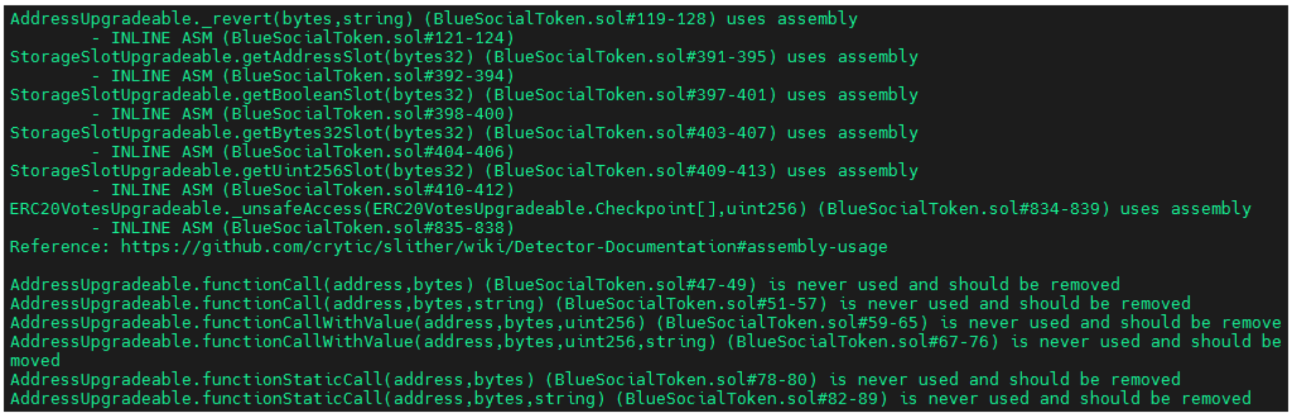


Slither Results Log

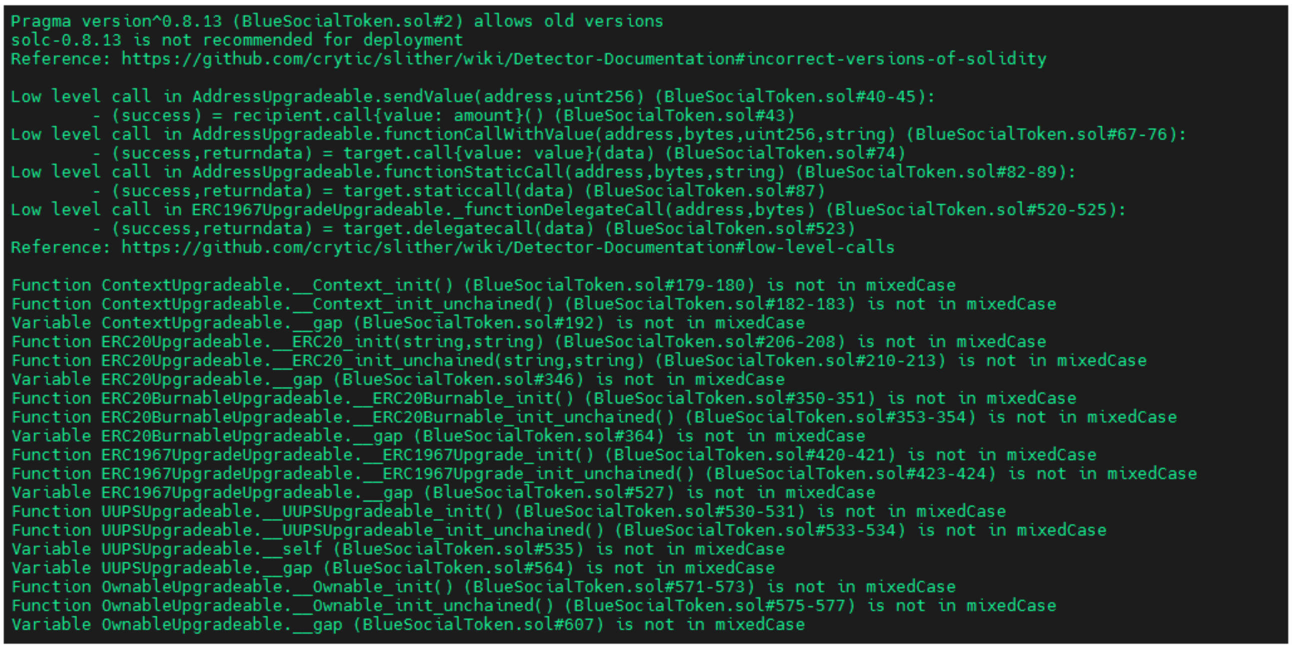
Slither log >> BlueSocialToken.sol



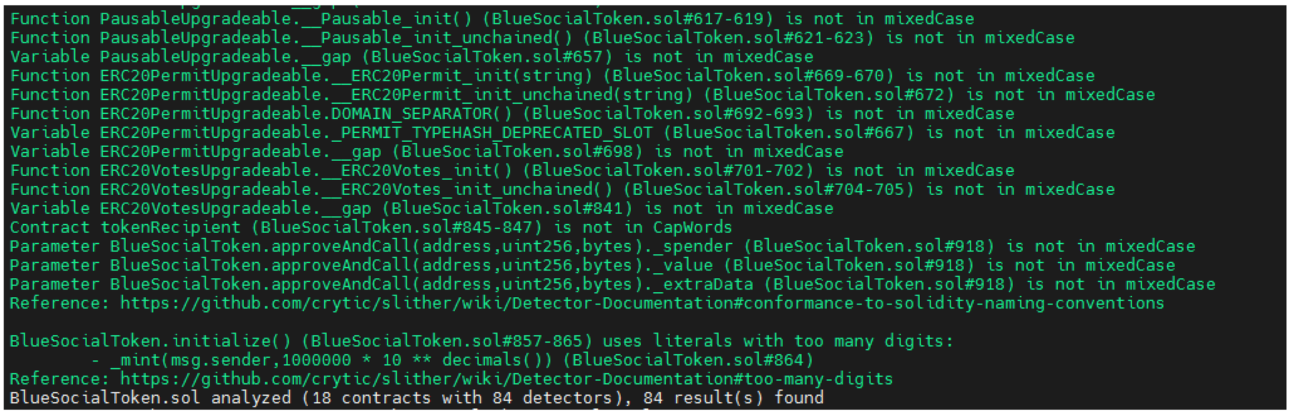
```
ERC20PermitUpgradeable.__ERC20Permit_init(string).name (BlueSocialToken.sol#669) shadows:
        - ERC20Upgradeable.name() (BlueSocialToken.sol#215-217) (function)
        - IERC20MetadataUpgradeable.name() (BlueSocialToken.sol#28) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Variable 'ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool).slot (BlueSocialToken.sol#464)' in ERC1967Upgrad
gradeable._upgradeToAndCallUUPS(address,bytes,bool) (BlueSocialToken.sol#456-471) potentially used before declaration: require
ol,string)(slot == _IMPLEMENTATION_SLOT,ERC1967Upgrade: unsupported proxiableUUID) (BlueSocialToken.sol#465)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

ERC20PermitUpgradeable.permit(address,address,uint256,uint256) (BlueSocialToken.sol#674-687) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,ERC20Permit: expired deadline) (BlueSocialToken.sol#681)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```



```
AddressUpgradeable._revert(bytes,string) (BlueSocialToken.sol#119-128) uses assembly
        - INLINE ASM (BlueSocialToken.sol#121-124)
StorageSlotUpgradeable.getAddressSlot(bytes32) (BlueSocialToken.sol#391-395) uses assembly
        - INLINE ASM (BlueSocialToken.sol#392-394)
StorageSlotUpgradeable.getBooleanSlot(bytes32) (BlueSocialToken.sol#397-401) uses assembly
        - INLINE ASM (BlueSocialToken.sol#398-400)
StorageSlotUpgradeable.getBytes32Slot(bytes32) (BlueSocialToken.sol#403-407) uses assembly
        - INLINE ASM (BlueSocialToken.sol#404-406)
StorageSlotUpgradeable.getUint256Slot(bytes32) (BlueSocialToken.sol#409-413) uses assembly
        - INLINE ASM (BlueSocialToken.sol#410-412)
ERC20VotesUpgradeable._unsafeAccess(ERC20VotesUpgradeable.Checkpoint[],uint256) (BlueSocialToken.sol#834-839) uses assembly
        - INLINE ASM (BlueSocialToken.sol#835-838)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

AddressUpgradeable.functionCall(address,bytes) (BlueSocialToken.sol#47-49) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (BlueSocialToken.sol#51-57) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (BlueSocialToken.sol#59-65) is never used and should be remove
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (BlueSocialToken.sol#67-76) is never used and should be
moved
AddressUpgradeable.functionStaticCall(address,bytes) (BlueSocialToken.sol#78-80) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (BlueSocialToken.sol#82-89) is never used and should be removed
```

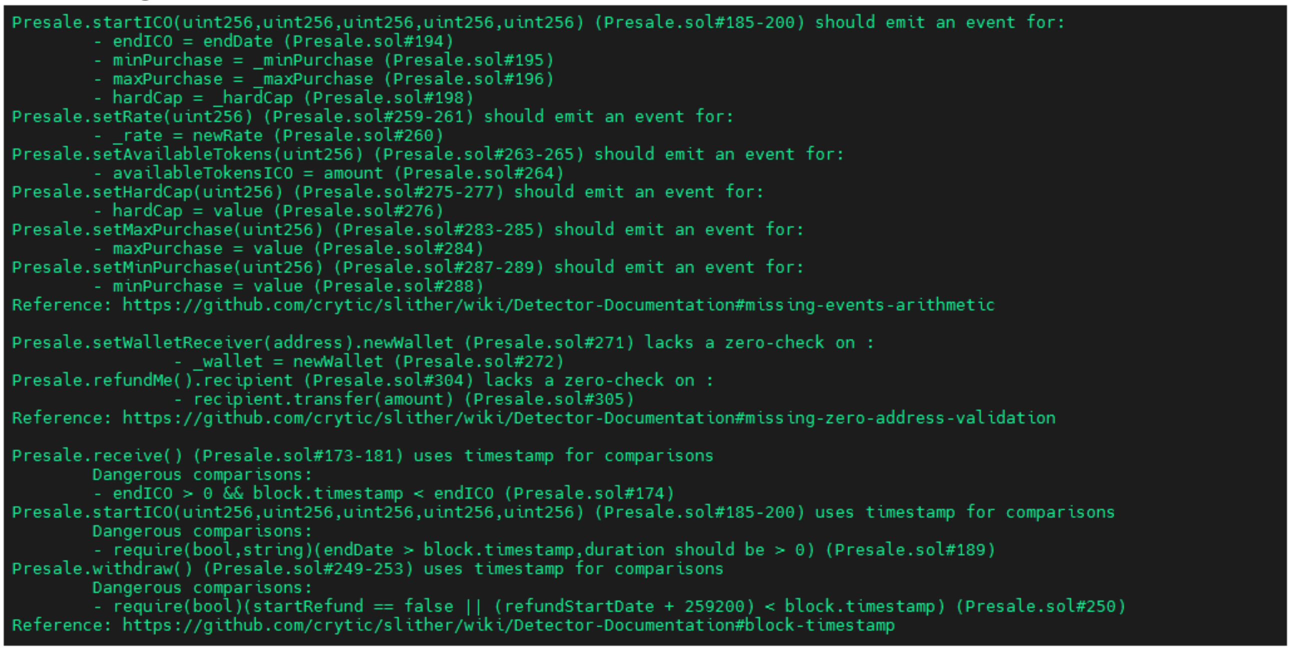


```
Pragma version^0.8.13 (BlueSocialToken.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (BlueSocialToken.sol#40-45):
        - (success) = recipient.call{value: amount}() (BlueSocialToken.sol#43)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (BlueSocialToken.sol#67-76):
        - (success,returndata) = target.call{value: value}(data) (BlueSocialToken.sol#74)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (BlueSocialToken.sol#82-89):
        - (success,returndata) = target.staticcall(data) (BlueSocialToken.sol#87)
Low level call in ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (BlueSocialToken.sol#520-525):
        - (success,returndata) = target.delegatecall(data) (BlueSocialToken.sol#523)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function ContextUpgradeable.__Context_init() (BlueSocialToken.sol#179-180) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (BlueSocialToken.sol#182-183) is not in mixedCase
Variable ContextUpgradeable.__gap (BlueSocialToken.sol#192) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (BlueSocialToken.sol#206-208) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (BlueSocialToken.sol#210-213) is not in mixedCase
Variable ERC20Upgradeable.__gap (BlueSocialToken.sol#346) is not in mixedCase
Function ERC20BurnableUpgradeable.__ERC20Burnable_init() (BlueSocialToken.sol#350-351) is not in mixedCase
Function ERC20BurnableUpgradeable.__ERC20Burnable_init_unchained() (BlueSocialToken.sol#353-354) is not in mixedCase
Variable ERC20BurnableUpgradeable.__gap (BlueSocialToken.sol#364) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init() (BlueSocialToken.sol#420-421) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init_unchained() (BlueSocialToken.sol#423-424) is not in mixedCase
Variable ERC1967UpgradeUpgradeable.__gap (BlueSocialToken.sol#527) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init() (BlueSocialToken.sol#530-531) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init_unchained() (BlueSocialToken.sol#533-534) is not in mixedCase
Variable UUPSUpgradeable.__self (BlueSocialToken.sol#535) is not in mixedCase
Variable UUPSUpgradeable.__gap (BlueSocialToken.sol#564) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (BlueSocialToken.sol#571-573) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (BlueSocialToken.sol#575-577) is not in mixedCase
Variable OwnableUpgradeable.__gap (BlueSocialToken.sol#607) is not in mixedCase
```

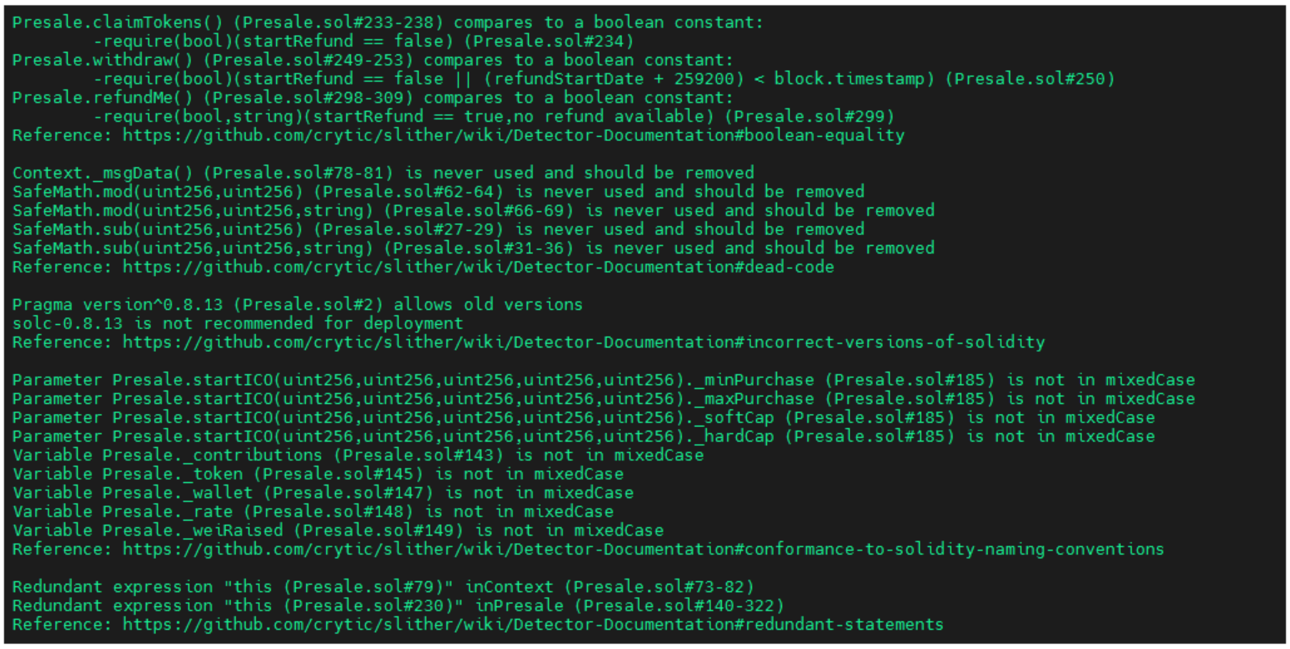


```
Function PausableUpgradeable.__Pausable_init() (BlueSocialToken.sol#617-619) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (BlueSocialToken.sol#621-623) is not in mixedCase
Variable PausableUpgradeable.__gap (BlueSocialToken.sol#657) is not in mixedCase
Function ERC20PermitUpgradeable.__ERC20Permit_init(string) (BlueSocialToken.sol#669-670) is not in mixedCase
Function ERC20PermitUpgradeable.__ERC20Permit_init_unchained(string) (BlueSocialToken.sol#672) is not in mixedCase
Function ERC20PermitUpgradeable.DOMAIN_SEPARATOR() (BlueSocialToken.sol#692-693) is not in mixedCase
Variable ERC20PermitUpgradeable._PERMIT_TYPEHASH_DEPRECATED_SLOT (BlueSocialToken.sol#667) is not in mixedCase
Variable ERC20PermitUpgradeable.__gap (BlueSocialToken.sol#698) is not in mixedCase
Function ERC20VotesUpgradeable.__ERC20Votes_init() (BlueSocialToken.sol#701-702) is not in mixedCase
Function ERC20VotesUpgradeable.__ERC20Votes_init_unchained() (BlueSocialToken.sol#704-705) is not in mixedCase
Variable ERC20VotesUpgradeable.__gap (BlueSocialToken.sol#841) is not in mixedCase
Contract tokenRecipient (BlueSocialToken.sol#845-847) is not in CapWords
Parameter BlueSocialToken.approveAndCall(address,uint256,bytes)._spender (BlueSocialToken.sol#918) is not in mixedCase
Parameter BlueSocialToken.approveAndCall(address,uint256,bytes)._value (BlueSocialToken.sol#918) is not in mixedCase
Parameter BlueSocialToken.approveAndCall(address,uint256,bytes)._extraData (BlueSocialToken.sol#918) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BlueSocialToken.initialize() (BlueSocialToken.sol#857-865) uses literals with too many digits:
        - _mint(msg.sender,1000000 * 10 ** decimals()) (BlueSocialToken.sol#864)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
BlueSocialToken.sol analyzed (18 contracts with 84 detectors), 84 result(s) found
```

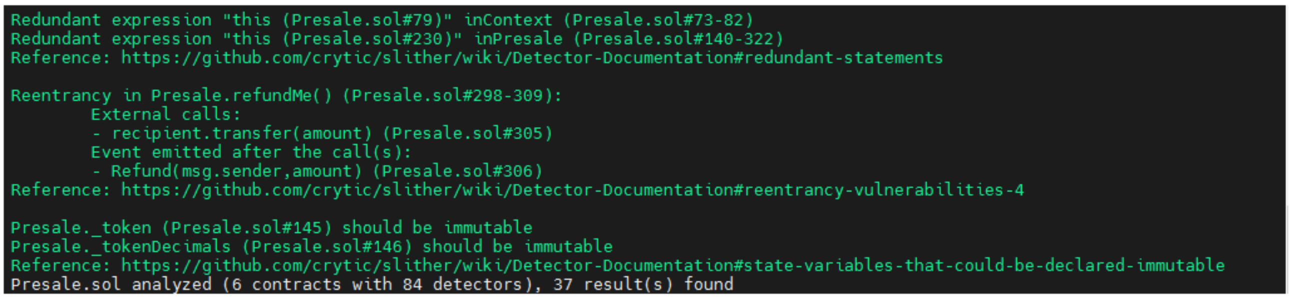
Slither log >> Presale.sol



```
Presale.startICO(uint256,uint256,uint256,uint256,uint256) (Presale.sol#185-200) should emit an event for:
        - endICO = endDate (Presale.sol#194)
        - minPurchase = _minPurchase (Presale.sol#195)
        - maxPurchase = _maxPurchase (Presale.sol#196)
        - hardCap = _hardCap (Presale.sol#198)
Presale.setRate(uint256) (Presale.sol#259-261) should emit an event for:
        - _rate = newRate (Presale.sol#260)
Presale.setAvailableTokens(uint256) (Presale.sol#263-265) should emit an event for:
        - availableTokensICO = amount (Presale.sol#264)
Presale.setHardCap(uint256) (Presale.sol#275-277) should emit an event for:
        - hardCap = value (Presale.sol#276)
Presale.setMaxPurchase(uint256) (Presale.sol#283-285) should emit an event for:
        - maxPurchase = value (Presale.sol#284)
Presale.setMinPurchase(uint256) (Presale.sol#287-289) should emit an event for:
        - minPurchase = value (Presale.sol#288)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Presale.setWalletReceiver(address).newWallet (Presale.sol#271) lacks a zero-check on :
                - _wallet = newWallet (Presale.sol#272)
Presale.refundMe().recipient (Presale.sol#304) lacks a zero-check on :
                - recipient.transfer(amount) (Presale.sol#305)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Presale.receive() (Presale.sol#173-181) uses timestamp for comparisons
        Dangerous comparisons:
        - endICO > 0 && block.timestamp < endICO (Presale.sol#174)
Presale.startICO(uint256,uint256,uint256,uint256,uint256) (Presale.sol#185-200) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(endDate > block.timestamp,duration should be > 0) (Presale.sol#189)
Presale.withdraw() (Presale.sol#249-253) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool)(startRefund == false || (refundStartDate + 259200) < block.timestamp) (Presale.sol#250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

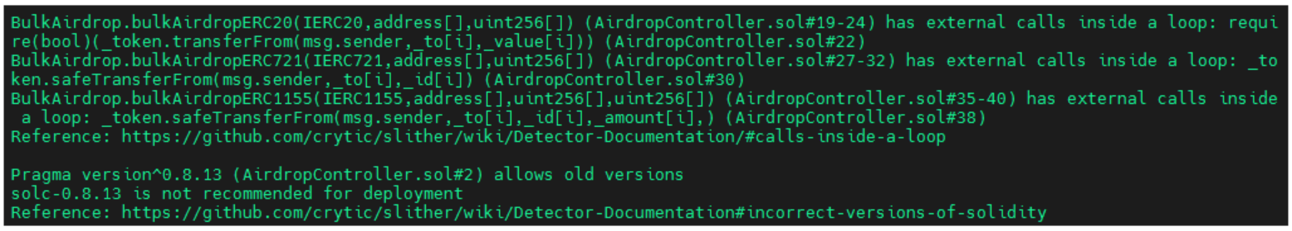


```
Presale.claimTokens() (Presale.sol#233-238) compares to a boolean constant:
        -require(bool)(startRefund == false) (Presale.sol#234)
Presale.withdraw() (Presale.sol#249-253) compares to a boolean constant:
        -require(bool)(startRefund == false || (refundStartDate + 259200) < block.timestamp) (Presale.sol#250)
Presale.refundMe() (Presale.sol#298-309) compares to a boolean constant:
        -require(bool,string)(startRefund == true,no refund available) (Presale.sol#299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Context._msgData() (Presale.sol#78-81) is never used and should be removed
SafeMath.mod(uint256,uint256) (Presale.sol#62-64) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Presale.sol#66-69) is never used and should be removed
SafeMath.sub(uint256,uint256) (Presale.sol#27-29) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (Presale.sol#31-36) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (Presale.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter Presale.startICO(uint256,uint256,uint256,uint256,uint256)._minPurchase (Presale.sol#185) is not in mixedCase
Parameter Presale.startICO(uint256,uint256,uint256,uint256,uint256)._maxPurchase (Presale.sol#185) is not in mixedCase
Parameter Presale.startICO(uint256,uint256,uint256,uint256,uint256)._softCap (Presale.sol#185) is not in mixedCase
Parameter Presale.startICO(uint256,uint256,uint256,uint256,uint256)._hardCap (Presale.sol#185) is not in mixedCase
Variable Presale._contributions (Presale.sol#143) is not in mixedCase
Variable Presale._token (Presale.sol#145) is not in mixedCase
Variable Presale._wallet (Presale.sol#147) is not in mixedCase
Variable Presale._rate (Presale.sol#148) is not in mixedCase
Variable Presale._weiRaised (Presale.sol#149) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (Presale.sol#79)" inContext (Presale.sol#73-82)
Redundant expression "this (Presale.sol#230)" inPresale (Presale.sol#140-322)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

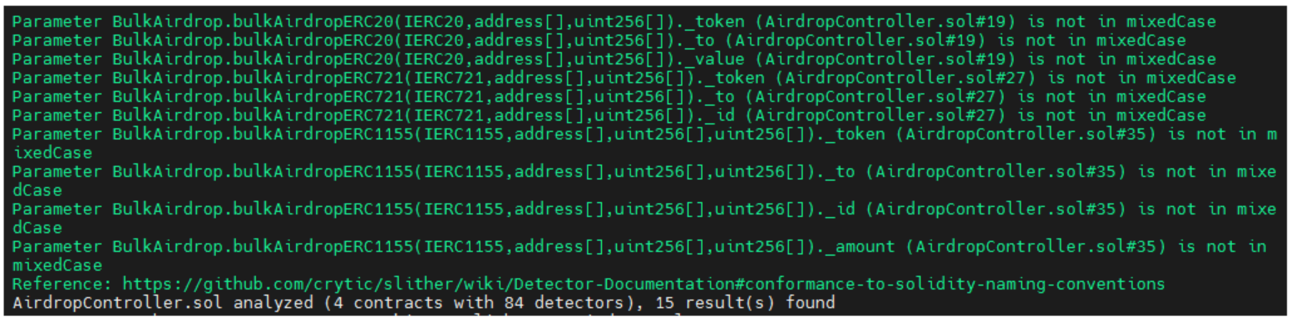


```
Redundant expression "this (Presale.sol#79)" inContext (Presale.sol#73-82)
Redundant expression "this (Presale.sol#230)" inPresale (Presale.sol#140-322)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Reentrancy in Presale.refundMe() (Presale.sol#298-309):
        External calls:
        - recipient.transfer(amount) (Presale.sol#305)
        Event emitted after the call(s):
        - Refund(msg.sender,amount) (Presale.sol#306)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Presale._token (Presale.sol#145) should be immutable
Presale._tokenDecimals (Presale.sol#146) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Presale.sol analyzed (6 contracts with 84 detectors), 37 result(s) found
```

Slither log >> AirdropController.sol

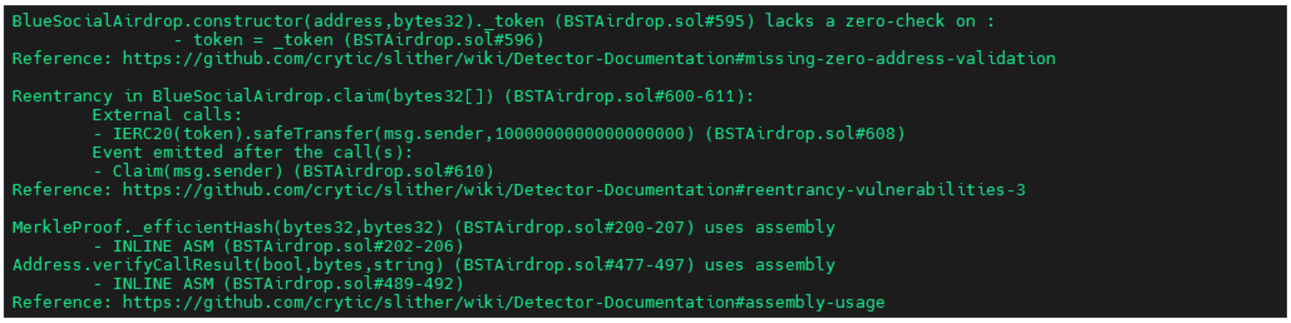


```
BulkAirdrop.bulkAirdropERC20(IERC20,address[],uint256[]) (AirdropController.sol#19-24) has external calls inside a loop: requi
re(bool)(_token.transferFrom(msg.sender,_to[i],_value[i])) (AirdropController.sol#22)
BulkAirdrop.bulkAirdropERC721(IERC721,address[],uint256[]) (AirdropController.sol#27-32) has external calls inside a loop: _to
ken.safeTransferFrom(msg.sender,_to[i],_id[i]) (AirdropController.sol#30)
BulkAirdrop.bulkAirdropERC1155(IERC1155,address[],uint256[],uint256[]) (AirdropController.sol#35-40) has external calls inside
 a loop: _token.safeTransferFrom(msg.sender,_to[i],_id[i],_amount[i],) (AirdropController.sol#38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Pragma version^0.8.13 (AirdropController.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

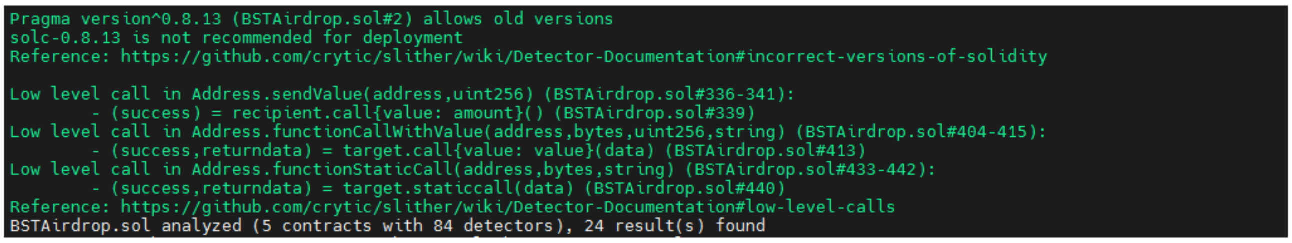


```
Parameter BulkAirdrop.bulkAirdropERC20(IERC20,address[],uint256[])._token (AirdropController.sol#19) is not in mixedCase
Parameter BulkAirdrop.bulkAirdropERC20(IERC20,address[],uint256[])._to (AirdropController.sol#19) is not in mixedCase
Parameter BulkAirdrop.bulkAirdropERC20(IERC20,address[],uint256[])._value (AirdropController.sol#19) is not in mixedCase
Parameter BulkAirdrop.bulkAirdropERC721(IERC721,address[],uint256[])._token (AirdropController.sol#27) is not in mixedCase
Parameter BulkAirdrop.bulkAirdropERC721(IERC721,address[],uint256[])._to (AirdropController.sol#27) is not in mixedCase
Parameter BulkAirdrop.bulkAirdropERC721(IERC721,address[],uint256[])._id (AirdropController.sol#27) is not in mixedCase
Parameter BulkAirdrop.bulkAirdropERC1155(IERC1155,address[],uint256[],uint256[])._token (AirdropController.sol#35) is not in m
ixedCase
Parameter BulkAirdrop.bulkAirdropERC1155(IERC1155,address[],uint256[],uint256[])._to (AirdropController.sol#35) is not in mixe
dCase
Parameter BulkAirdrop.bulkAirdropERC1155(IERC1155,address[],uint256[],uint256[])._id (AirdropController.sol#35) is not in mixe
dCase
Parameter BulkAirdrop.bulkAirdropERC1155(IERC1155,address[],uint256[],uint256[])._amount (AirdropController.sol#35) is not in
mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
AirdropController.sol analyzed (4 contracts with 84 detectors), 15 result(s) found
```

Slither log >> BSTAirdrop.sol

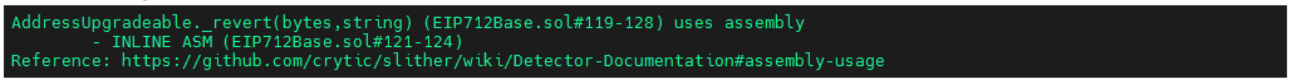


```
BlueSocialAirdrop.constructor(address,bytes32)._token (BSTAirdrop.sol#595) lacks a zero-check on :
                - token = _token (BSTAirdrop.sol#596)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in BlueSocialAirdrop.claim(bytes32[]) (BSTAirdrop.sol#600-611):
        External calls:
        - IERC20(token).safeTransfer(msg.sender,1000000000000000000) (BSTAirdrop.sol#608)
        Event emitted after the call(s):
        - Claim(msg.sender) (BSTAirdrop.sol#610)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

MerkleProof._efficientHash(bytes32,bytes32) (BSTAirdrop.sol#200-207) uses assembly
        - INLINE ASM (BSTAirdrop.sol#202-206)
Address.verifyCallResult(bool,bytes,string) (BSTAirdrop.sol#477-497) uses assembly
        - INLINE ASM (BSTAirdrop.sol#489-492)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

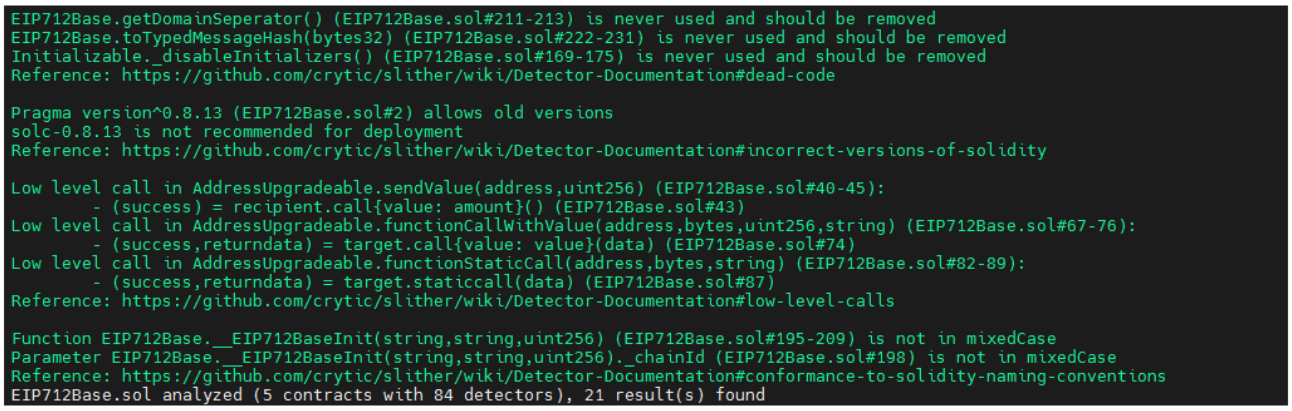


```
Pragma version^0.8.13 (BSTAirdrop.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (BSTAirdrop.sol#336-341):
        - (success) = recipient.call{value: amount}() (BSTAirdrop.sol#339)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (BSTAirdrop.sol#404-415):
        - (success,returndata) = target.call{value: value}(data) (BSTAirdrop.sol#413)
Low level call in Address.functionStaticCall(address,bytes,string) (BSTAirdrop.sol#433-442):
        - (success,returndata) = target.staticcall(data) (BSTAirdrop.sol#440)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
BSTAirdrop.sol analyzed (5 contracts with 84 detectors), 24 result(s) found
```

Slither log >> EIP712Base.sol

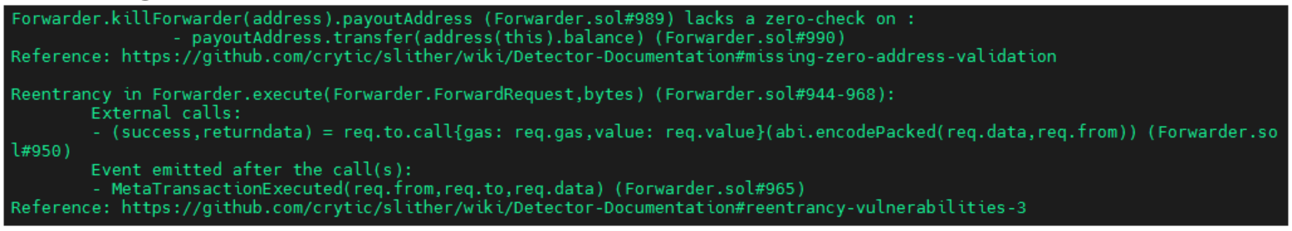


```
AddressUpgradeable._revert(bytes,string) (EIP712Base.sol#119-128) uses assembly
        - INLINE ASM (EIP712Base.sol#121-124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

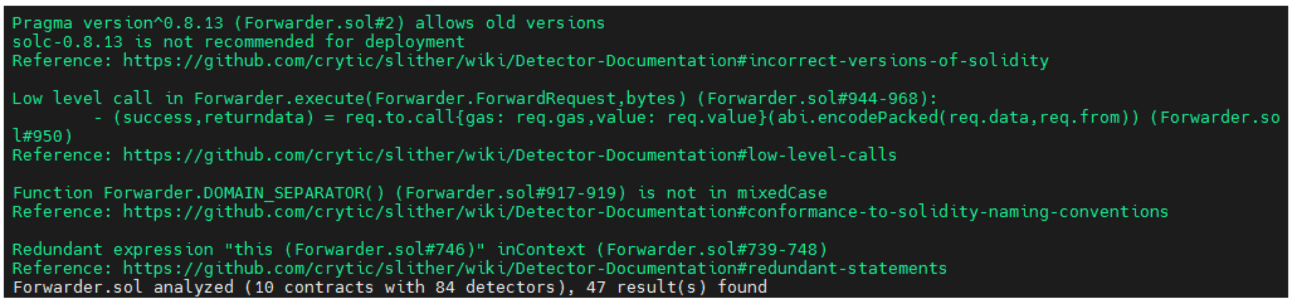


```
EIP712Base.getDomainSeperator() (EIP712Base.sol#211-213) is never used and should be removed
EIP712Base.toTypedMessageHash(bytes32) (EIP712Base.sol#222-231) is never used and should be removed
Initializable._disableInitializers() (EIP712Base.sol#169-175) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (EIP712Base.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (EIP712Base.sol#40-45):
        - (success) = recipient.call{value: amount}() (EIP712Base.sol#43)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (EIP712Base.sol#67-76):
        - (success,returndata) = target.call{value: value}(data) (EIP712Base.sol#74)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (EIP712Base.sol#82-89):
        - (success,returndata) = target.staticcall(data) (EIP712Base.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function EIP712Base.__EIP712BaseInit(string,string,uint256) (EIP712Base.sol#195-209) is not in mixedCase
Parameter EIP712Base.__EIP712BaseInit(string,string,uint256)._chainId (EIP712Base.sol#198) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
EIP712Base.sol analyzed (5 contracts with 84 detectors), 21 result(s) found
```

Slither log >> Forwarder.sol

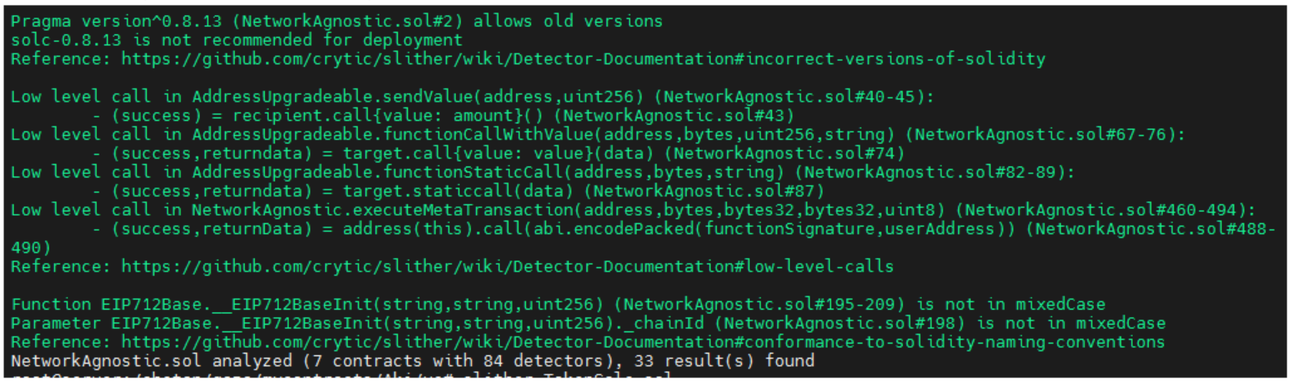


```
Forwarder.killForwarder(address).payoutAddress (Forwarder.sol#989) lacks a zero-check on :
                - payoutAddress.transfer(address(this).balance) (Forwarder.sol#990)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Forwarder.execute(Forwarder.ForwardRequest,bytes) (Forwarder.sol#944-968):
        External calls:
        - (success,returndata) = req.to.call{gas: req.gas,value: req.value}(abi.encodePacked(req.data,req.from)) (Forwarder.so
l#950)
        Event emitted after the call(s):
        - MetaTransactionExecuted(req.from,req.to,req.data) (Forwarder.sol#965)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```



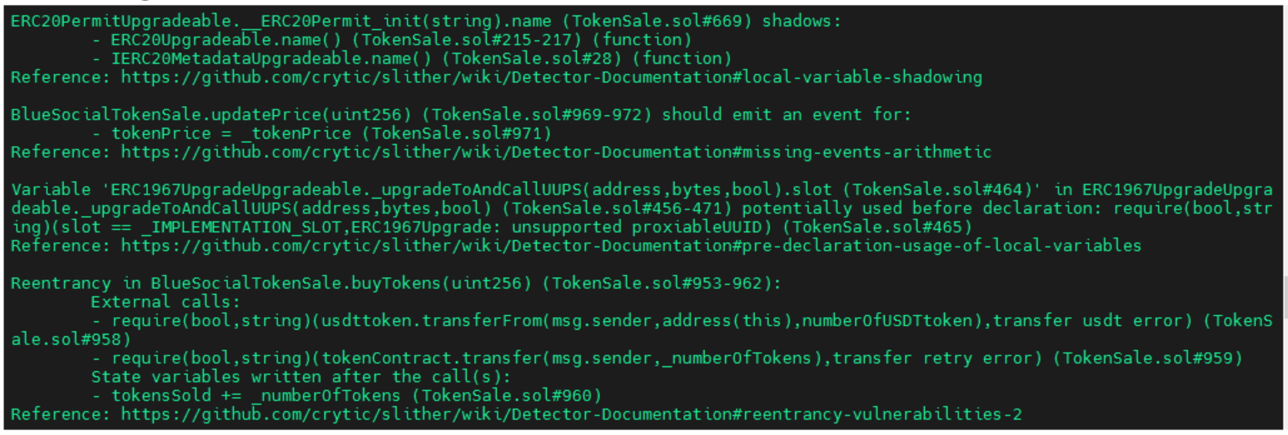
```
Pragma version^0.8.13 (Forwarder.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Forwarder.execute(Forwarder.ForwardRequest,bytes) (Forwarder.sol#944-968):
        - (success,returndata) = req.to.call{gas: req.gas,value: req.value}(abi.encodePacked(req.data,req.from)) (Forwarder.so
l#950)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function Forwarder.DOMAIN_SEPARATOR() (Forwarder.sol#917-919) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (Forwarder.sol#746)" inContext (Forwarder.sol#739-748)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
Forwarder.sol analyzed (10 contracts with 84 detectors), 47 result(s) found
```

Slither log >> NetworkAgnostic.sol

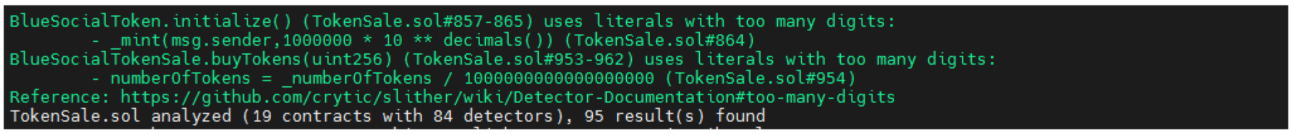


```
Pragma version^0.8.13 (NetworkAgnostic.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (NetworkAgnostic.sol#40-45):
        - (success) = recipient.call{value: amount}() (NetworkAgnostic.sol#43)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (NetworkAgnostic.sol#67-76):
        - (success,returndata) = target.call{value: value}(data) (NetworkAgnostic.sol#74)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (NetworkAgnostic.sol#82-89):
        - (success,returndata) = target.staticcall(data) (NetworkAgnostic.sol#87)
Low level call in NetworkAgnostic.executeMetaTransaction(address,bytes,bytes32,bytes32,uint8) (NetworkAgnostic.sol#460-494):
        - (success,returnData) = address(this).call(abi.encodePacked(functionSignature,userAddress)) (NetworkAgnostic.sol#488-
490)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function EIP712Base.__EIP712BaseInit(string,string,uint256) (NetworkAgnostic.sol#195-209) is not in mixedCase
Parameter EIP712Base.__EIP712BaseInit(string,string,uint256)._chainId (NetworkAgnostic.sol#198) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
NetworkAgnostic.sol analyzed (7 contracts with 84 detectors), 33 result(s) found
```

Slither log >> TokenSale.sol

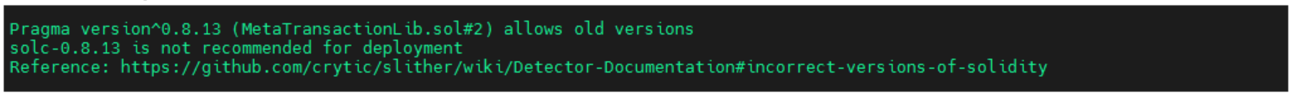


```
ERC20PermitUpgradeable.__ERC20Permit_init(string).name (TokenSale.sol#669) shadows:
        - ERC20Upgradeable.name() (TokenSale.sol#215-217) (function)
        - IERC20MetadataUpgradeable.name() (TokenSale.sol#28) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

BlueSocialTokenSale.updatePrice(uint256) (TokenSale.sol#969-972) should emit an event for:
        - tokenPrice = _tokenPrice (TokenSale.sol#971)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Variable 'ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool).slot (TokenSale.sol#464)' in ERC1967UpgradeUpgra
deable._upgradeToAndCallUUPS(address,bytes,bool) (TokenSale.sol#456-471) potentially used before declaration: require(bool,str
ing)(slot == _IMPLEMENTATION_SLOT,ERC1967Upgrade: unsupported proxiableUUID) (TokenSale.sol#465)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in BlueSocialTokenSale.buyTokens(uint256) (TokenSale.sol#953-962):
        External calls:
        - require(bool,string)(usdttoken.transferFrom(msg.sender,address(this),numberOfUSDTtoken),transfer usdt error) (TokenS
ale.sol#958)
        - require(bool,string)(tokenContract.transfer(msg.sender,_numberOfTokens),transfer retry error) (TokenSale.sol#959)
        State variables written after the call(s):
        - tokensSold += _numberOfTokens (TokenSale.sol#960)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

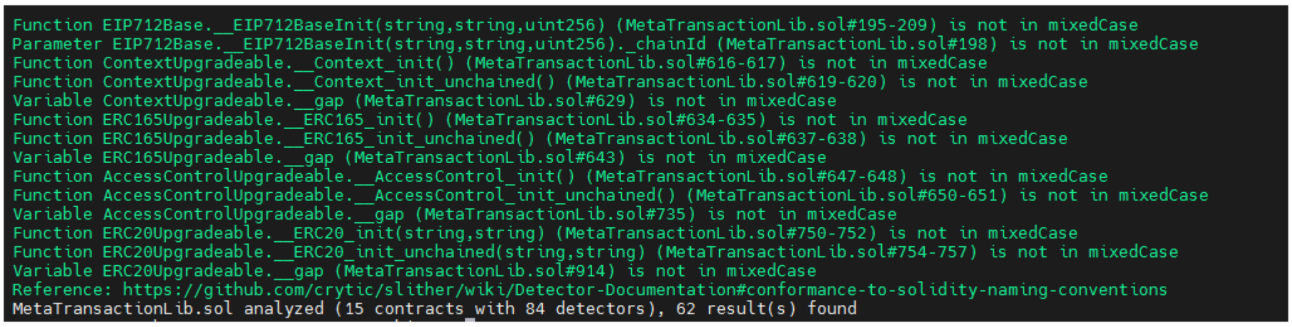


```
BlueSocialToken.initialize() (TokenSale.sol#857-865) uses literals with too many digits:
        - _mint(msg.sender,1000000 * 10 ** decimals()) (TokenSale.sol#864)
BlueSocialTokenSale.buyTokens(uint256) (TokenSale.sol#953-962) uses literals with too many digits:
        - numberOfTokens = _numberOfTokens / 1000000000000000000 (TokenSale.sol#954)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
TokenSale.sol analyzed (19 contracts with 84 detectors), 95 result(s) found
```

Slither log >> MetaTransactionLib.sol



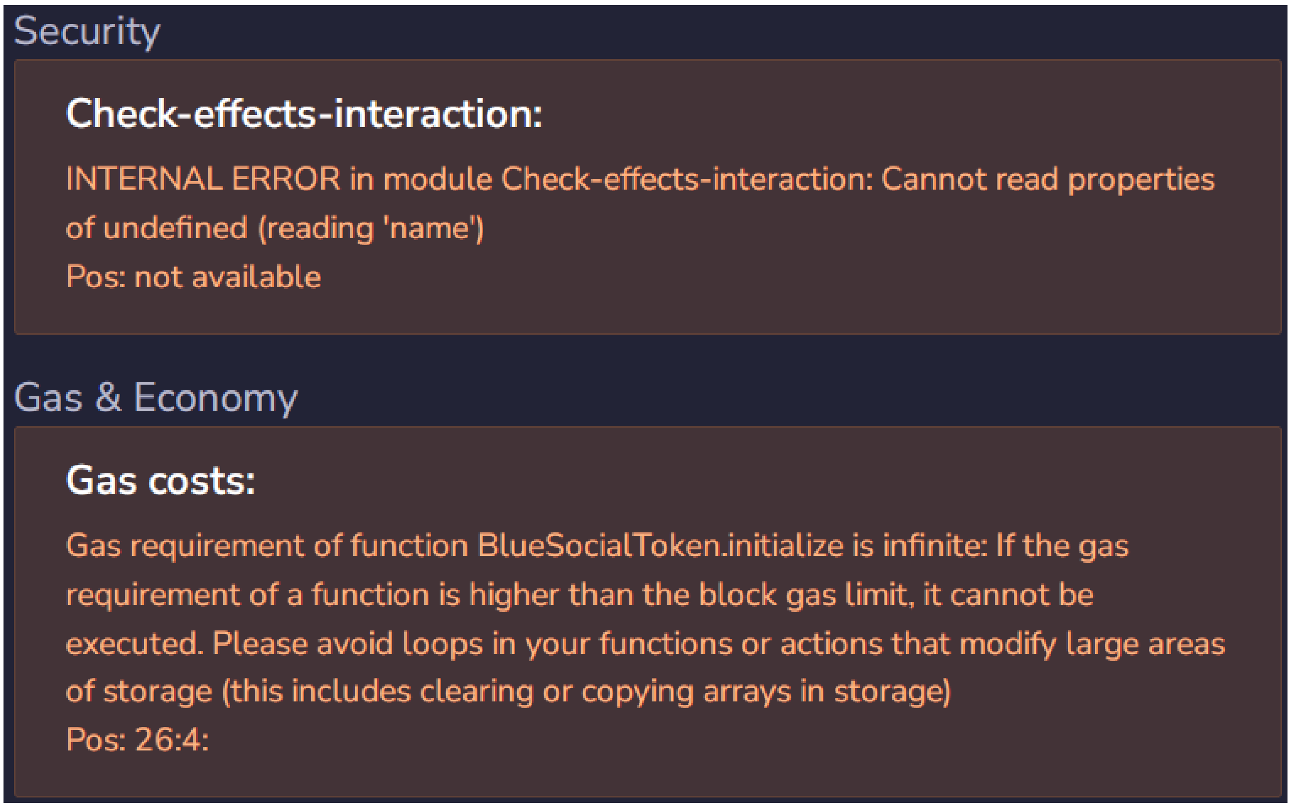
```
Pragma version^0.8.13 (MetaTransactionLib.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```



```
Function EIP712Base.__EIP712BaseInit(string,string,uint256) (MetaTransactionLib.sol#195-209) is not in mixedCase
Parameter EIP712Base.__EIP712BaseInit(string,string,uint256)._chainId (MetaTransactionLib.sol#198) is not in mixedCase
Function ContextUpgradeable.__Context_init() (MetaTransactionLib.sol#616-617) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (MetaTransactionLib.sol#619-620) is not in mixedCase
Variable ContextUpgradeable.__gap (MetaTransactionLib.sol#629) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (MetaTransactionLib.sol#634-635) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (MetaTransactionLib.sol#637-638) is not in mixedCase
Variable ERC165Upgradeable.__gap (MetaTransactionLib.sol#643) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init() (MetaTransactionLib.sol#647-648) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init_unchained() (MetaTransactionLib.sol#650-651) is not in mixedCase
Variable AccessControlUpgradeable.__gap (MetaTransactionLib.sol#735) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (MetaTransactionLib.sol#750-752) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (MetaTransactionLib.sol#754-757) is not in mixedCase
Variable ERC20Upgradeable.__gap (MetaTransactionLib.sol#914) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
MetaTransactionLib.sol analyzed (15 contracts with 84 detectors), 62 result(s) found
```

Solidity Static Analysis

BlueSocialToken.sol



Security

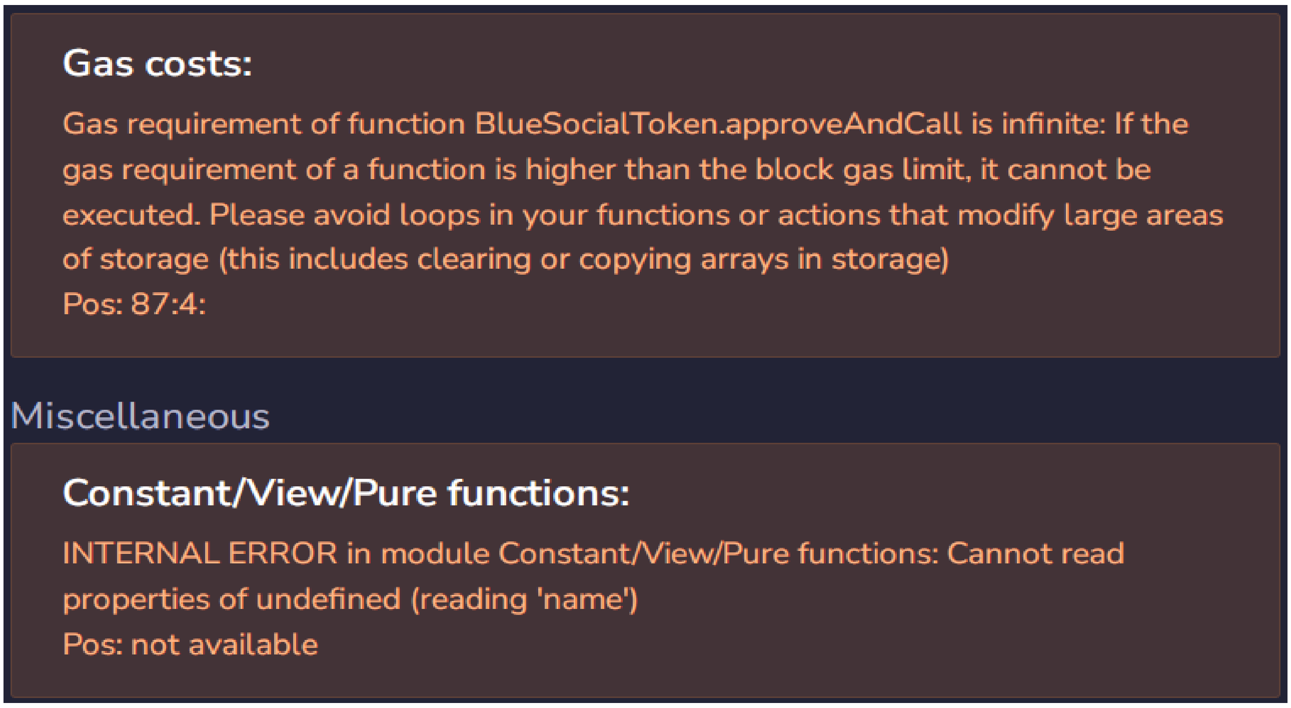
Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')
Pos: not available

Gas & Economy

Gas costs:

Gas requirement of function BlueSocialToken.initialize 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: 26:4:



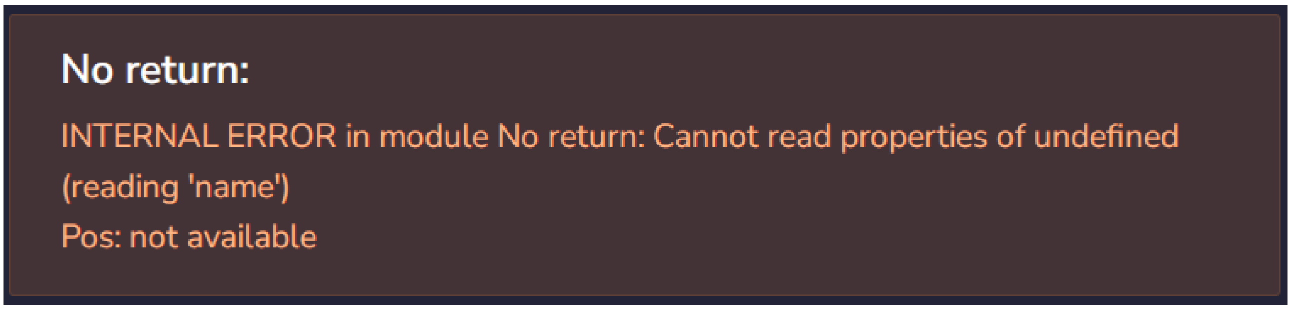
Gas costs:

Gas requirement of function BlueSocialToken.approveAndCall 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: 87:4:

Miscellaneous

Constant/View/Pure functions:

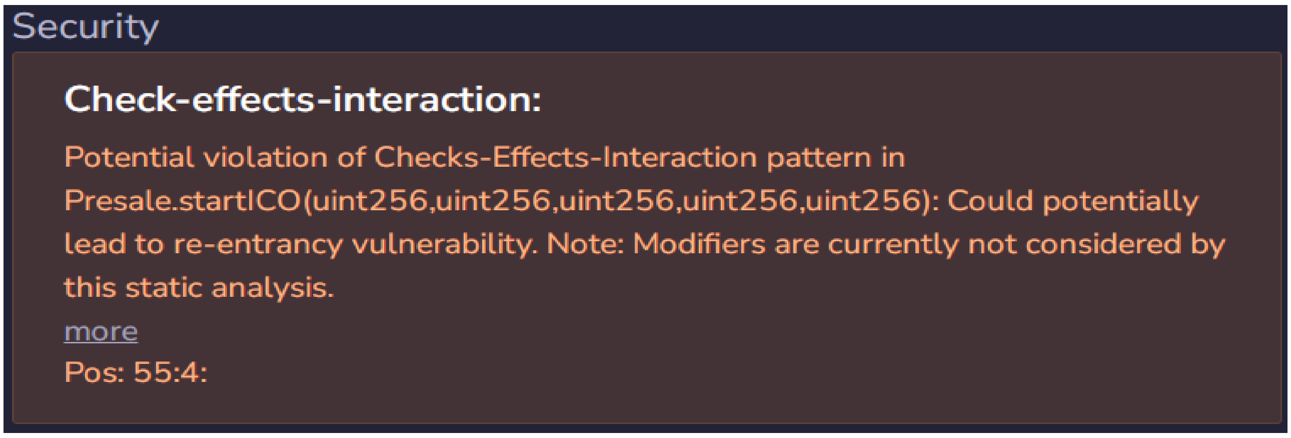
INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')
Pos: not available



No return:

INTERNAL ERROR in module No return: Cannot read properties of undefined (reading 'name')
Pos: not available

Presale.sol



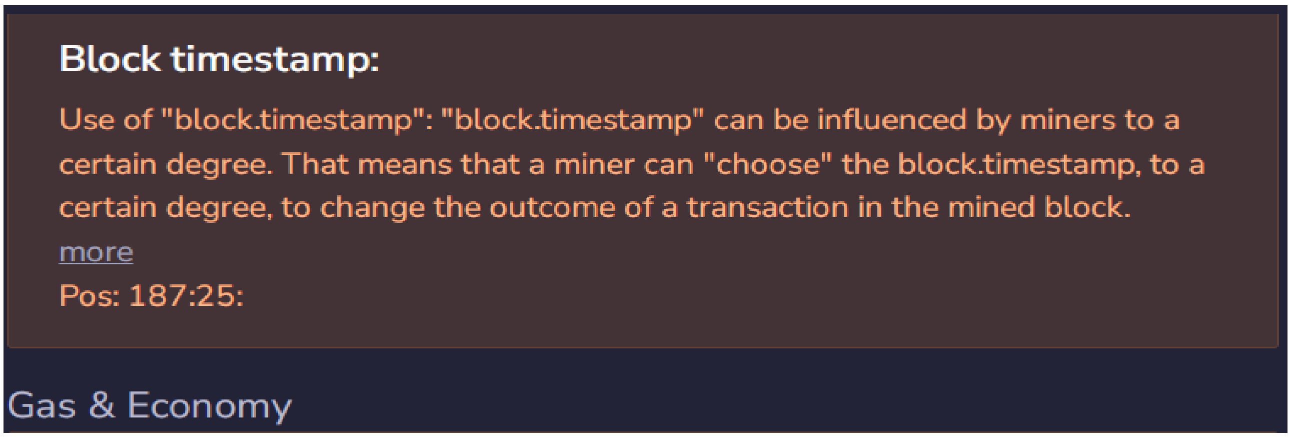
Security

Check-effects-interaction:

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

more

Pos: 55:4:



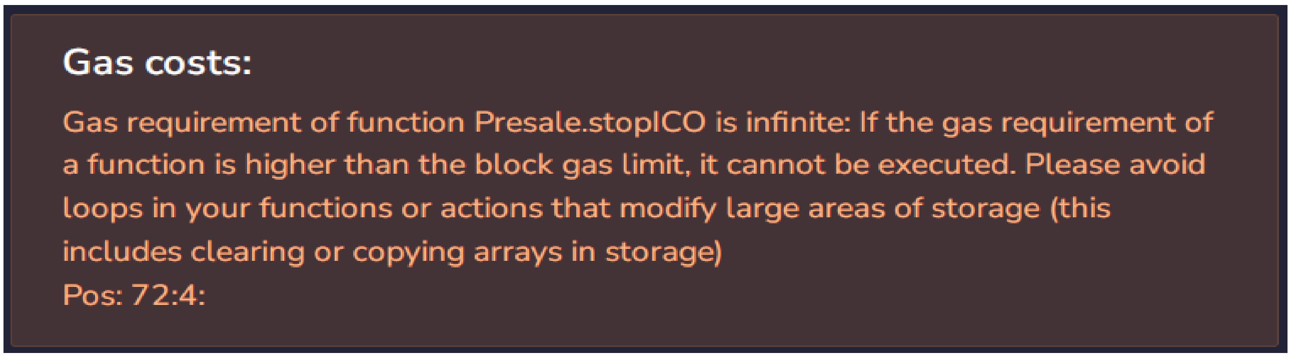
Block timestamp:

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: 187:25:

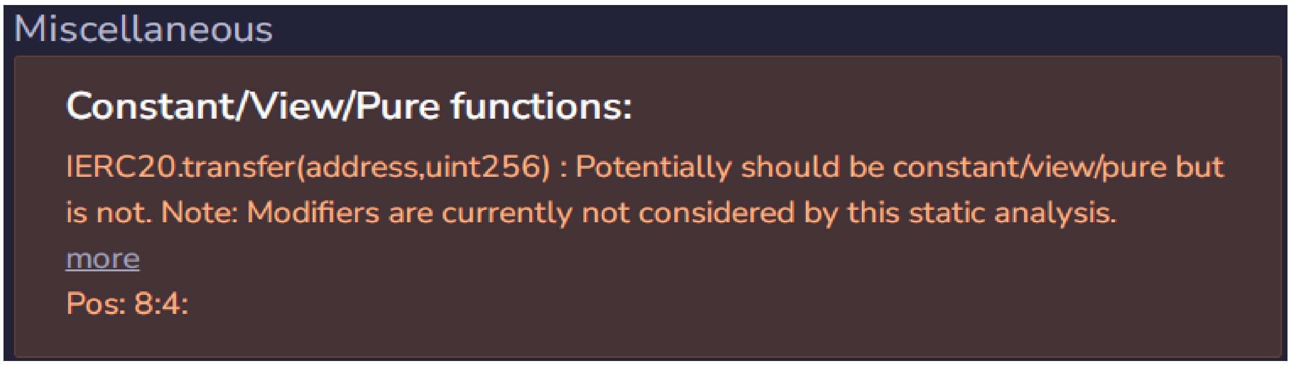
Gas & Economy



Gas costs:

Gas requirement of function Presale.stopICO 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: 72:4:



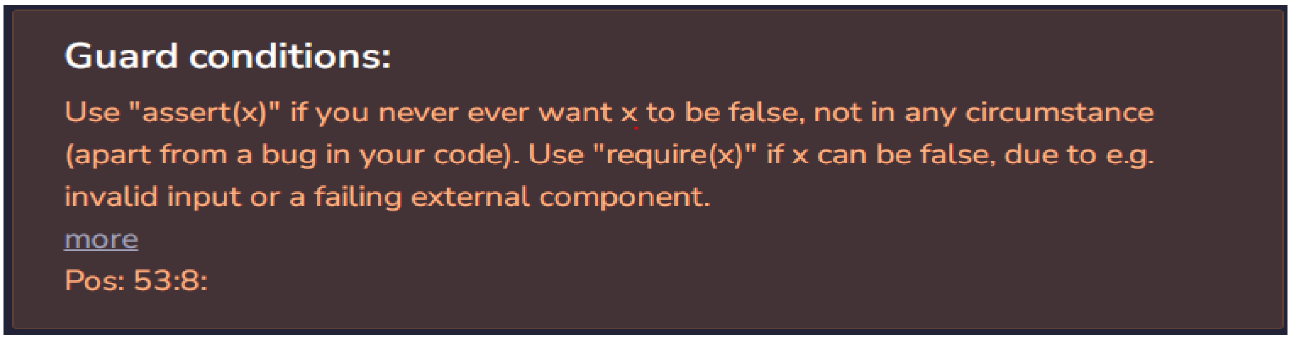
Miscellaneous

Constant/View/Pure functions:

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 8:4:



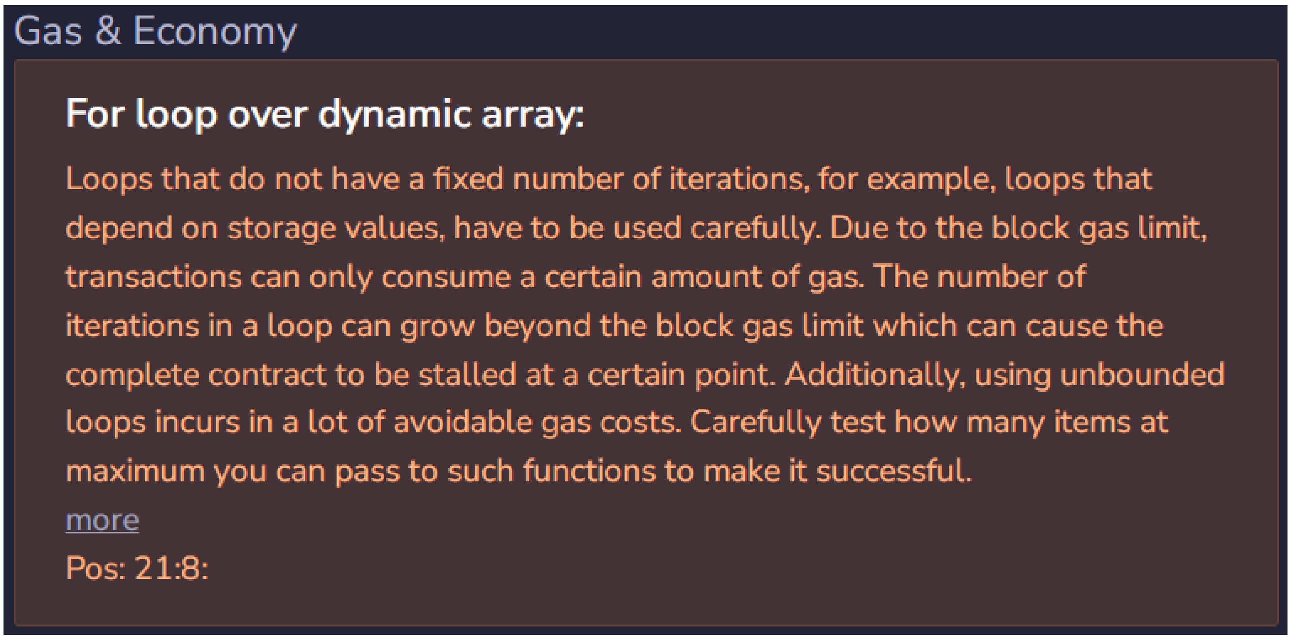
Guard conditions:

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: 53:8:

AirdropController.sol



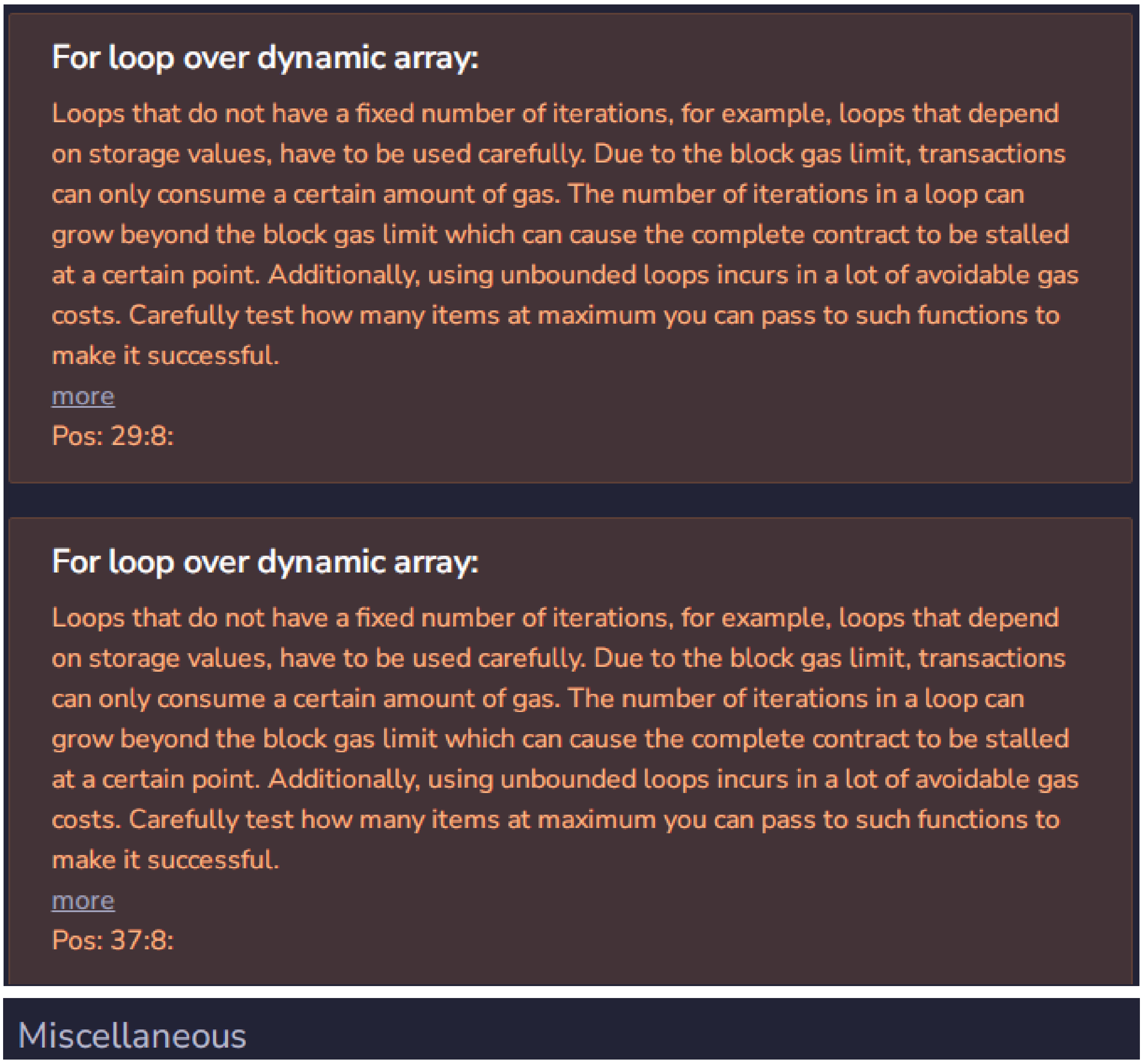
Gas & Economy

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: 21:8:



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: 29:8:

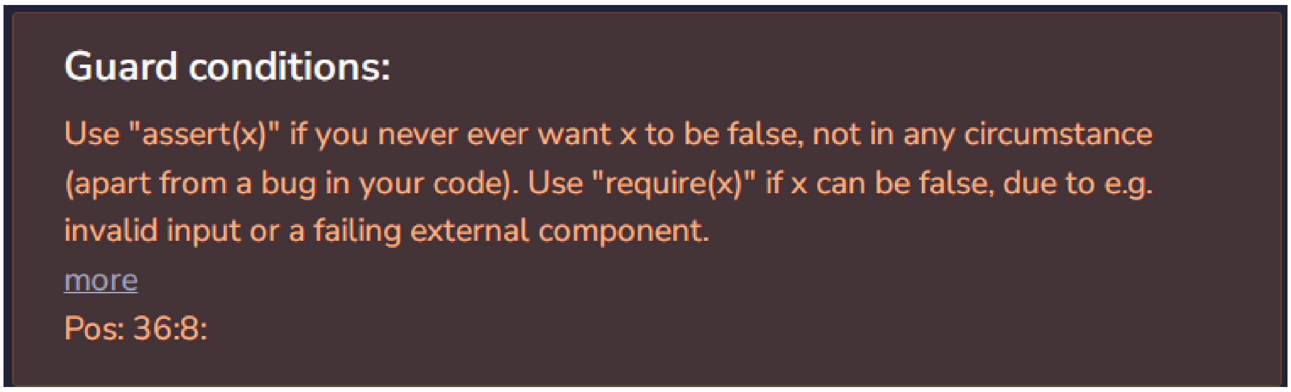
For loop over dynamic array:

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: 37:8:

Miscellaneous



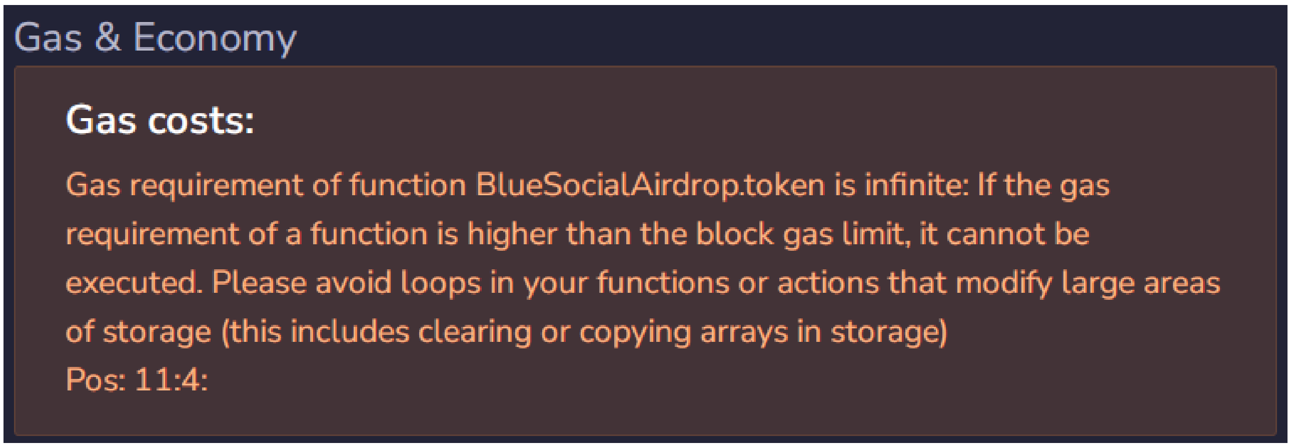
Guard conditions:

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: 36:8:

BSTAirdrop.sol

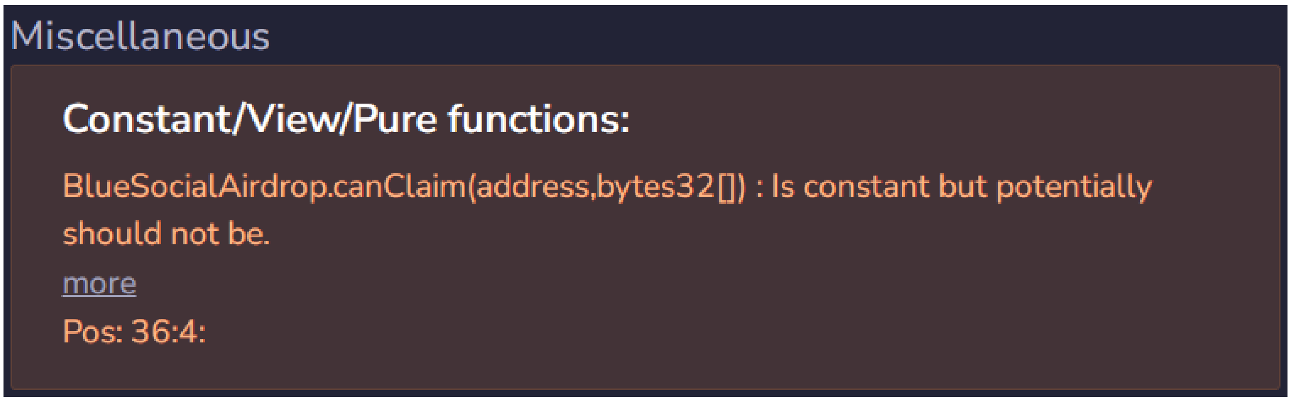


Gas & Economy

Gas costs:

Gas requirement of function BlueSocialAirdrop.token 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: 11:4:



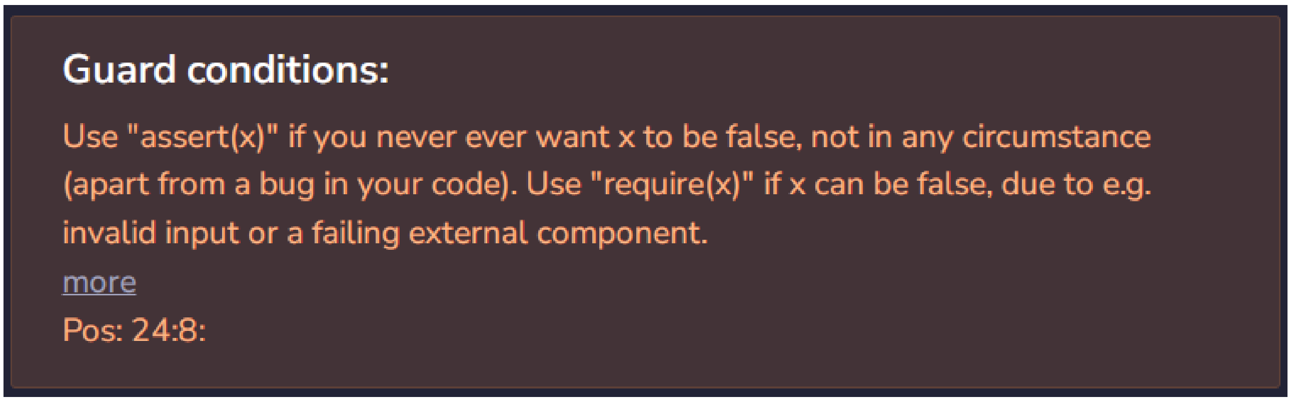
Miscellaneous

Constant/View/Pure functions:

BlueSocialAirdrop.canClaim(address,bytes32[]) : Is constant but potentially should not be.

more

Pos: 36:4:

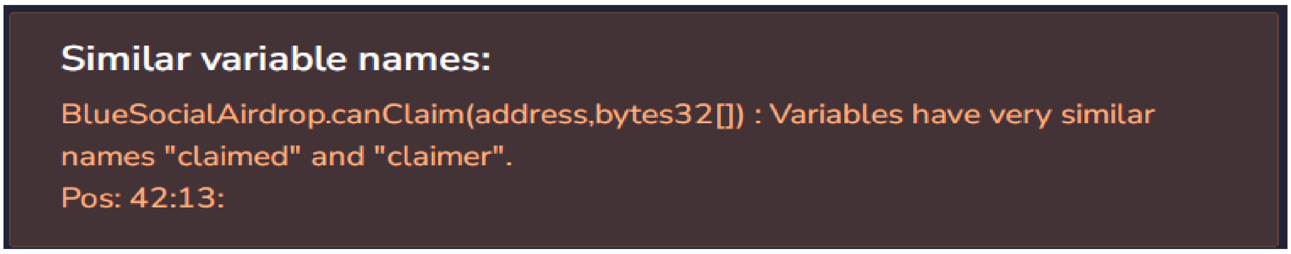


Guard conditions:

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: 24:8:

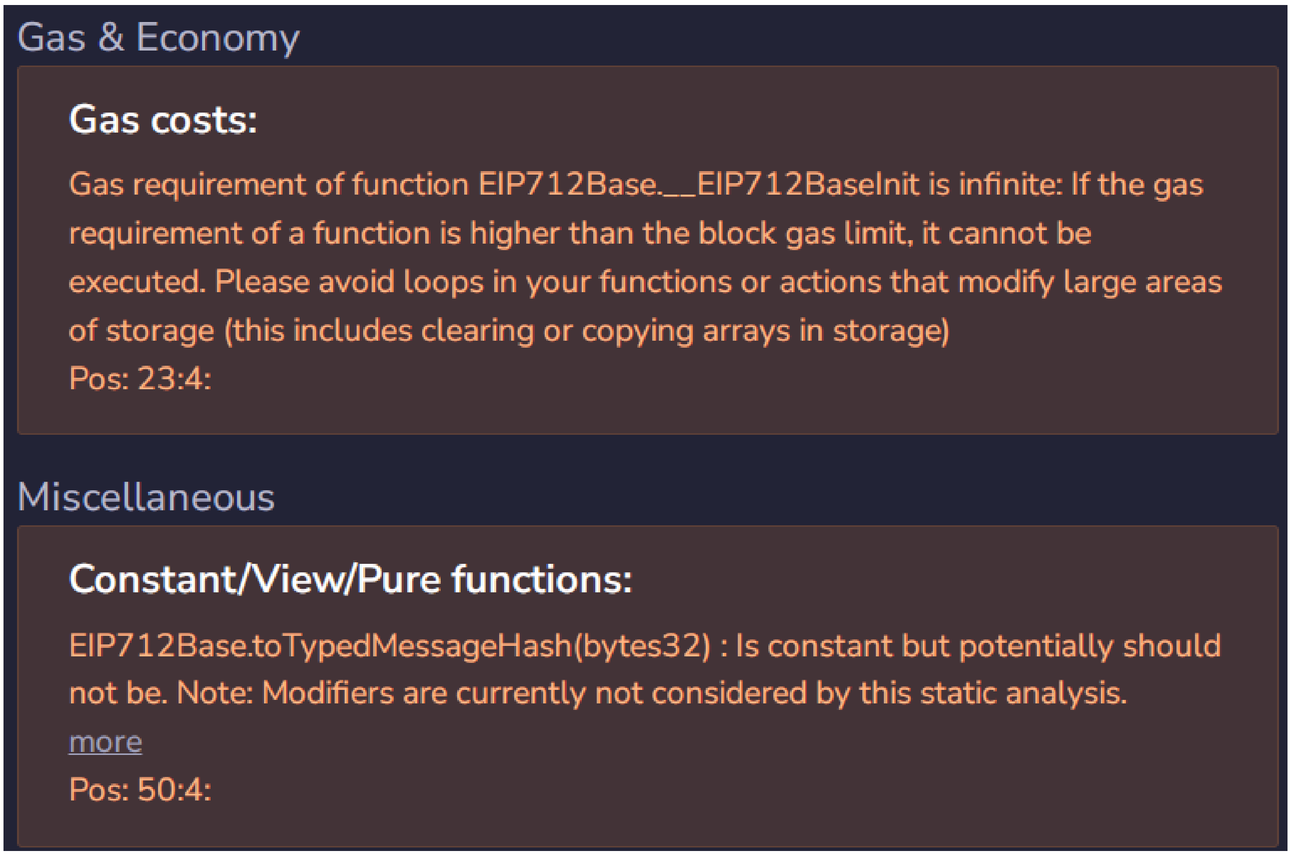


Similar variable names:

BlueSocialAirdrop.canClaim(address,bytes32[]) : Variables have very similar names "claimed" and "claimer".

Pos: 42:13:

EIP712Base.sol



Gas & Economy

Gas costs:

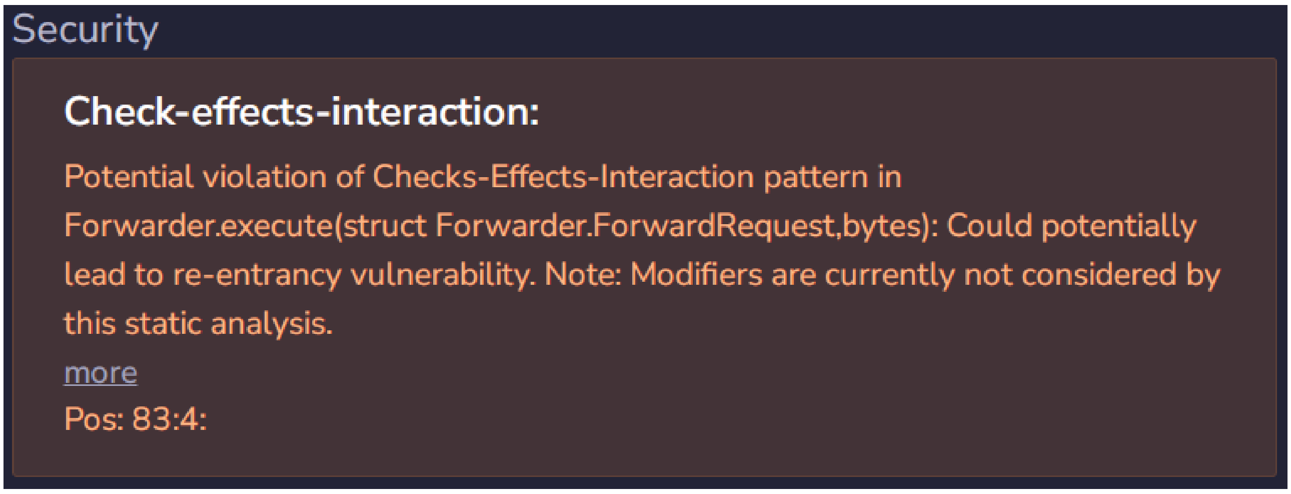
Gas requirement of function EIP712Base.__EIP712BaseInit is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 23:4:

Miscellaneous

Constant/View/Pure functions:

EIP712Base.toTypedMessageHash(bytes32) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 50:4:

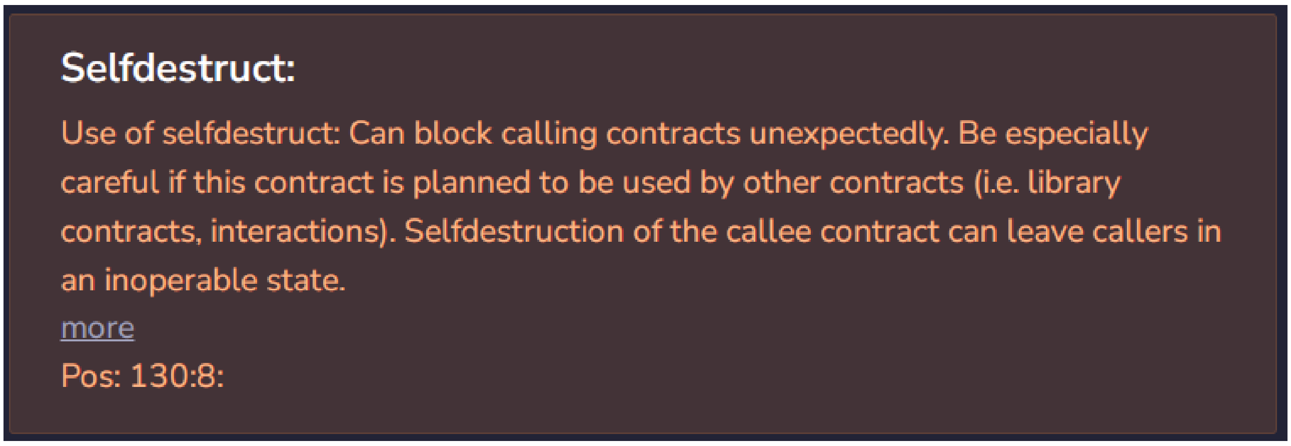
Forwarder.sol



Security

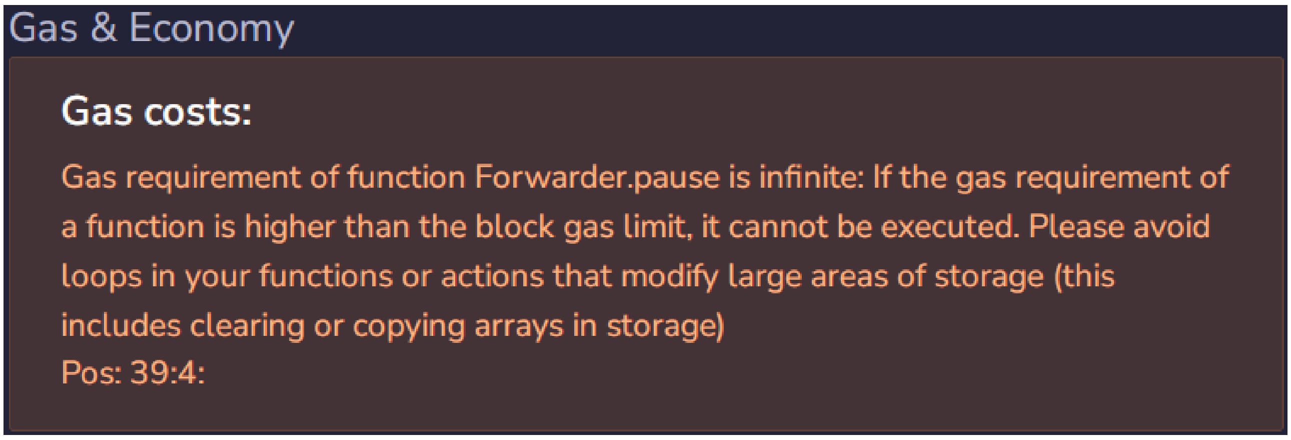
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Forwarder.execute(struct Forwarder.ForwardRequest,bytes): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 83:4:



Selfdestruct:

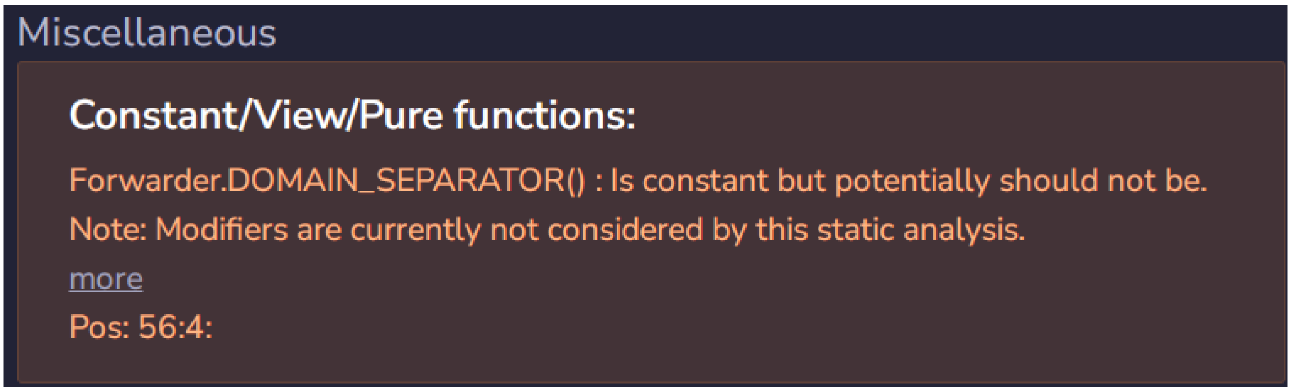
Use of selfdestruct: Can block calling contracts unexpectedly. Be especially careful if this contract is planned to be used by other contracts (i.e. library contracts, interactions). Selfdestruction of the callee contract can leave callers in an inoperable state.
more
Pos: 130:8:



Gas & Economy

Gas costs:

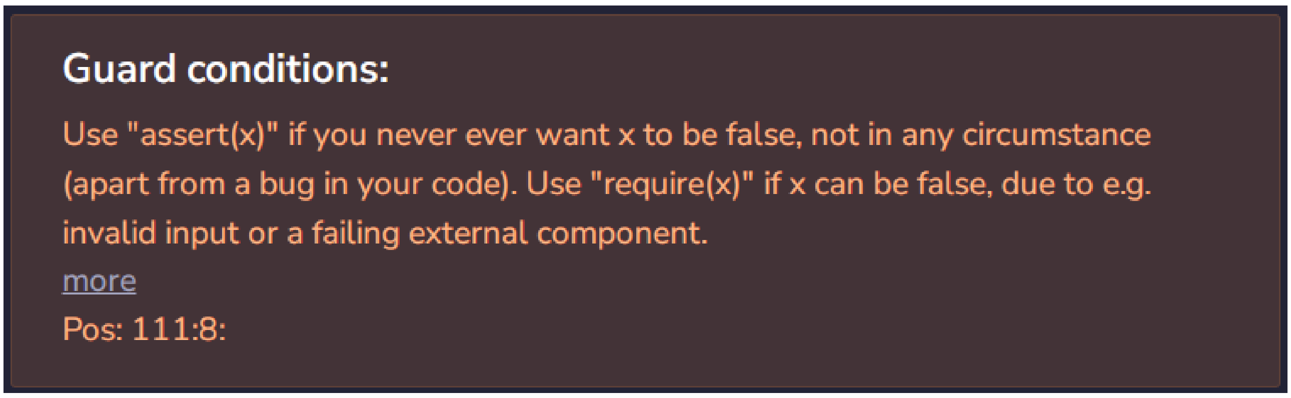
Gas requirement of function Forwarder.pause is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 39:4:



Miscellaneous

Constant/View/Pure functions:

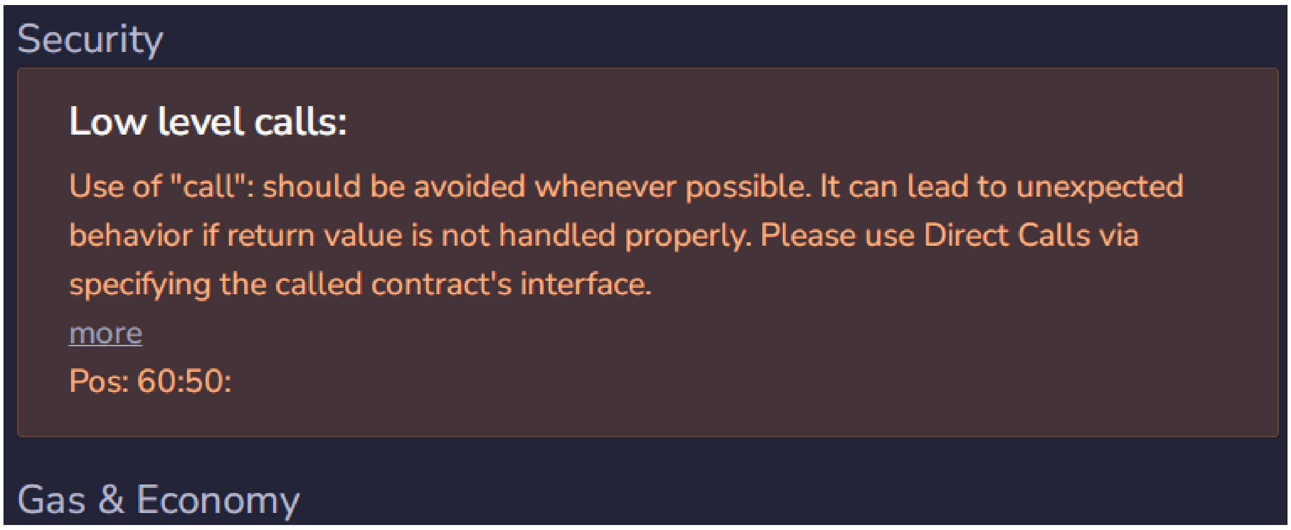
Forwarder.DOMAIN_SEPARATOR() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 56:4:



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: 111:8:

NetworkAgnostic.sol

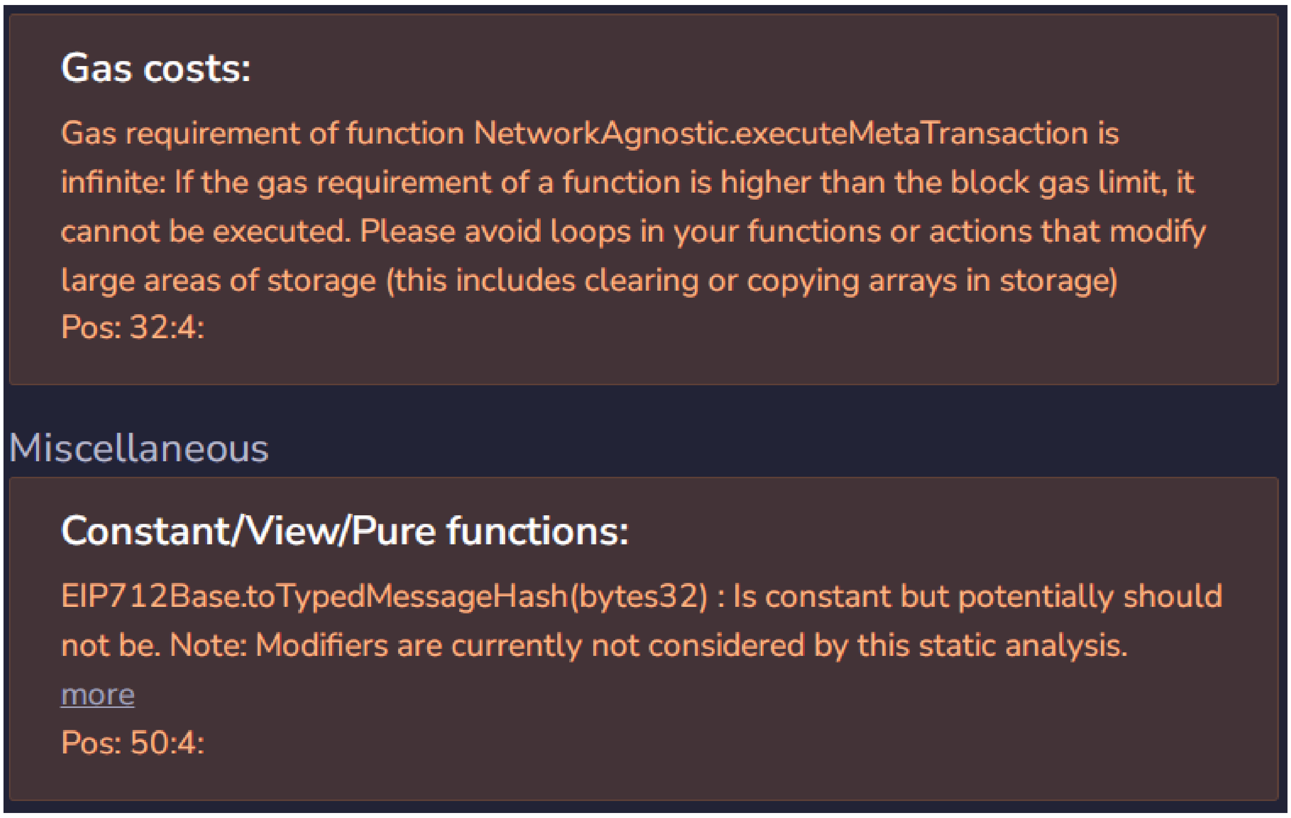


Security

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: 60:50:

Gas & Economy



Gas costs:

Gas requirement of function NetworkAgnostic.executeMetaTransaction 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: 32:4:

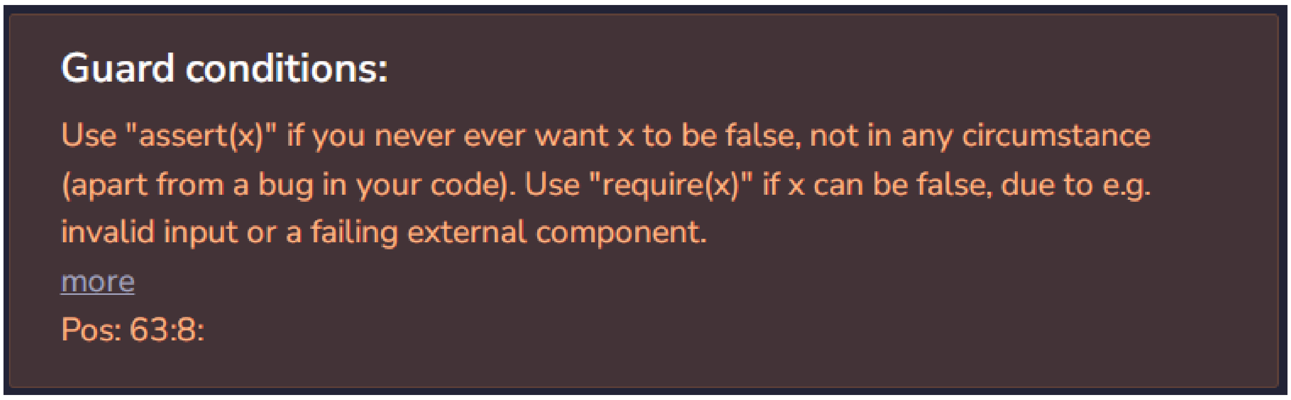
Miscellaneous

Constant/View/Pure functions:

EIP712Base.toTypedMessageHash(bytes32) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 50:4:



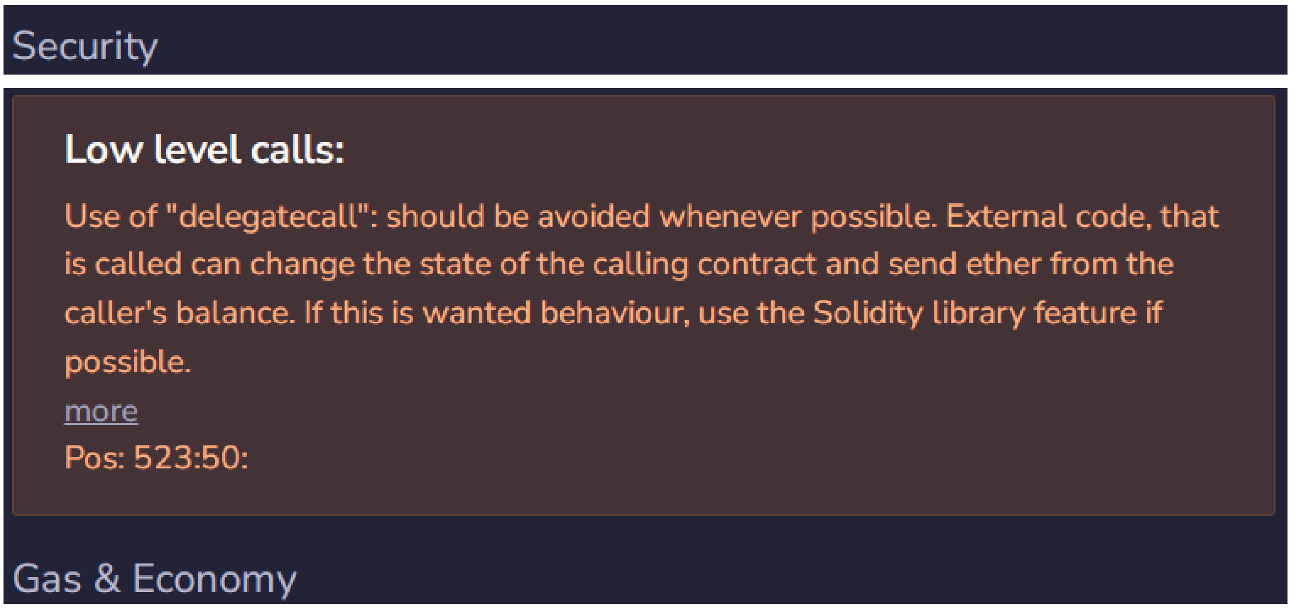
Guard conditions:

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: 63:8:

TokenSale.sol



Security

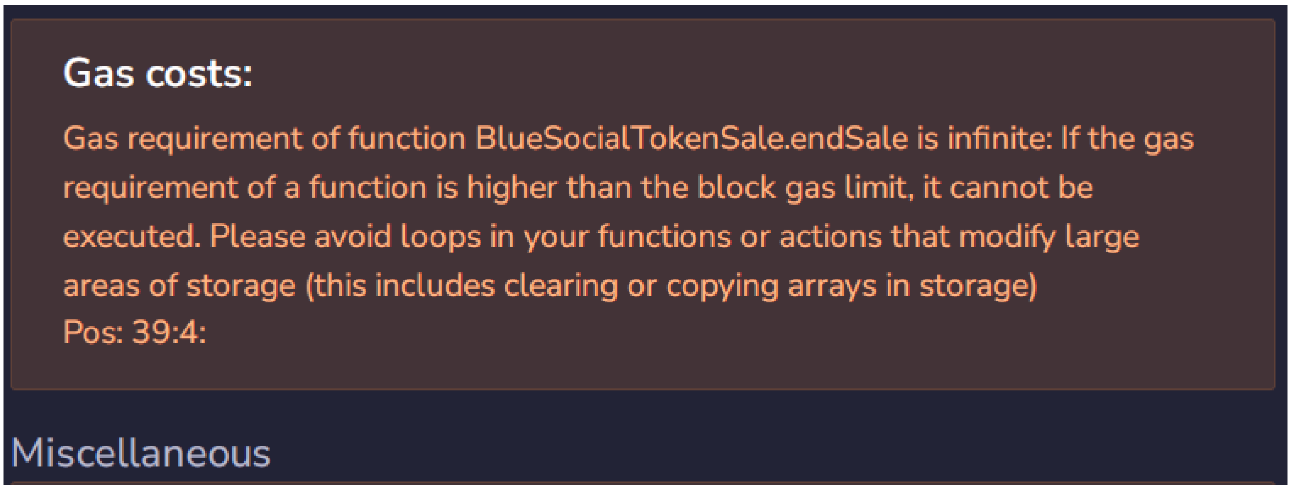
Low level calls:

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: 523:50:

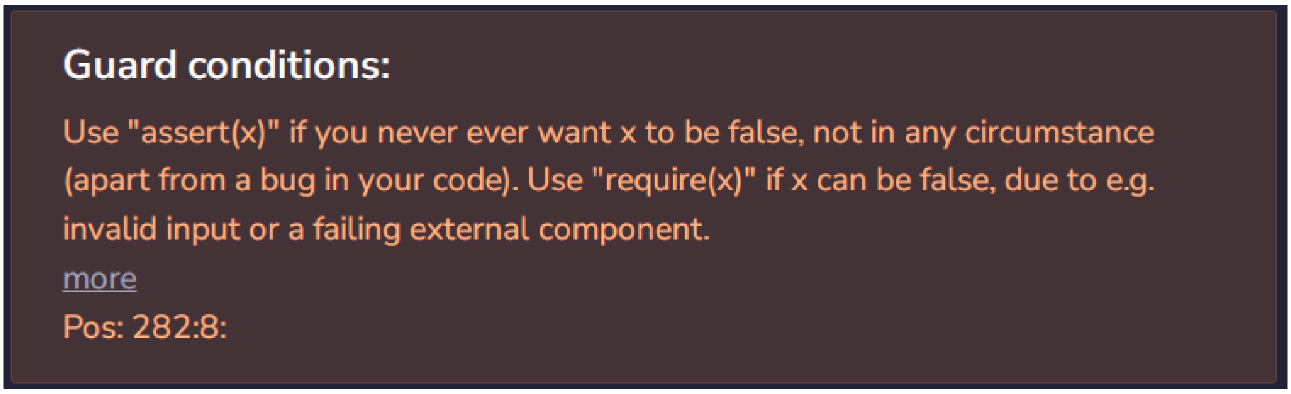
Gas & Economy



Gas costs:

Gas requirement of function BlueSocialTokenSale.endSale 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: 39:4:

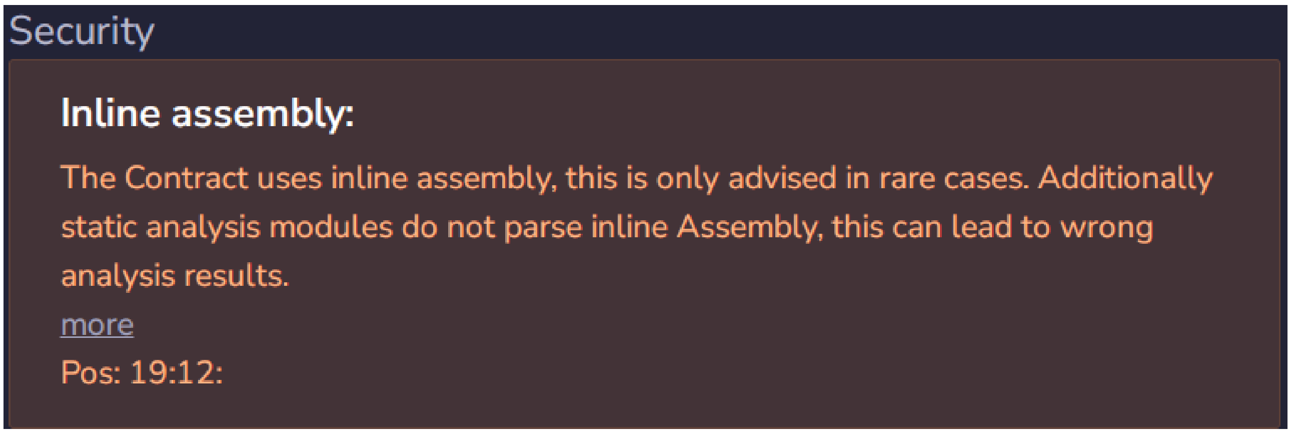
Miscellaneous



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: 282:8:

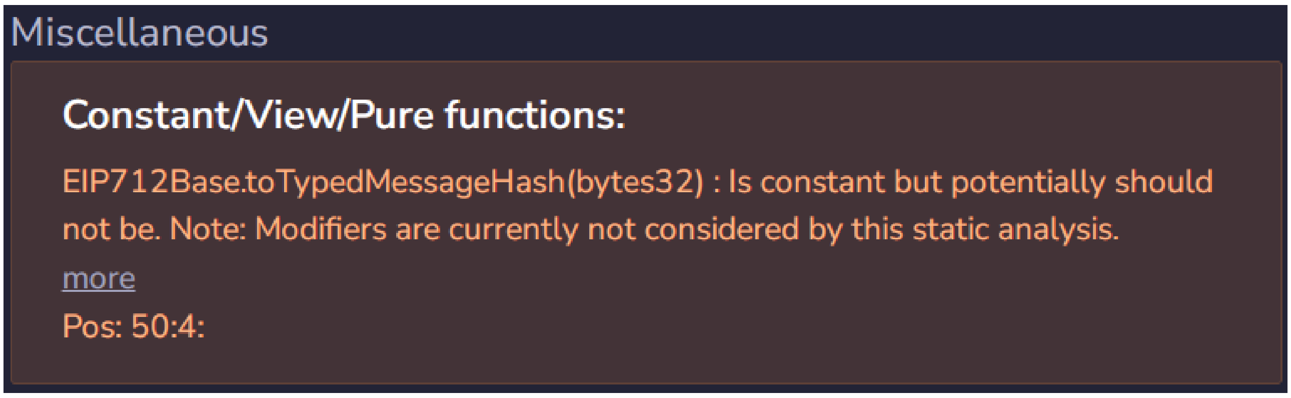
MetaTransactionLib.sol



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: 19:12:



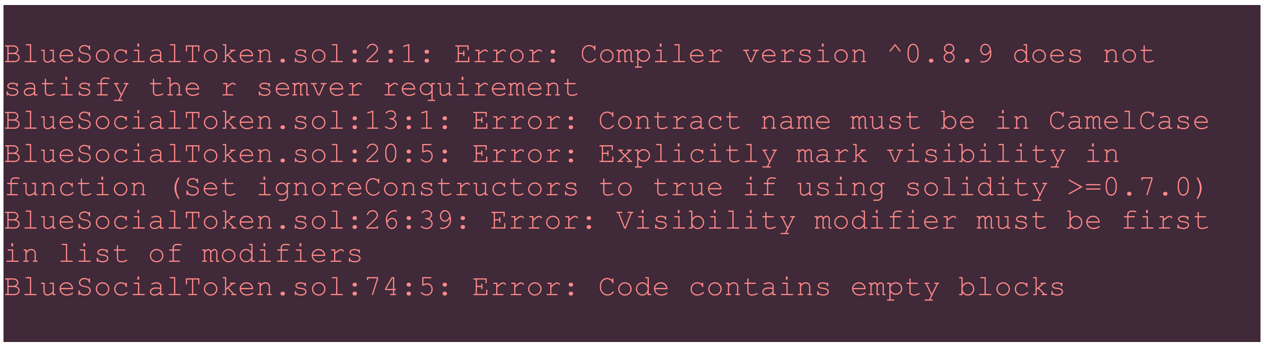
Miscellaneous

Constant/View/Pure functions:

EIP712Base.toTypedMessageHash(bytes32) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 50:4:

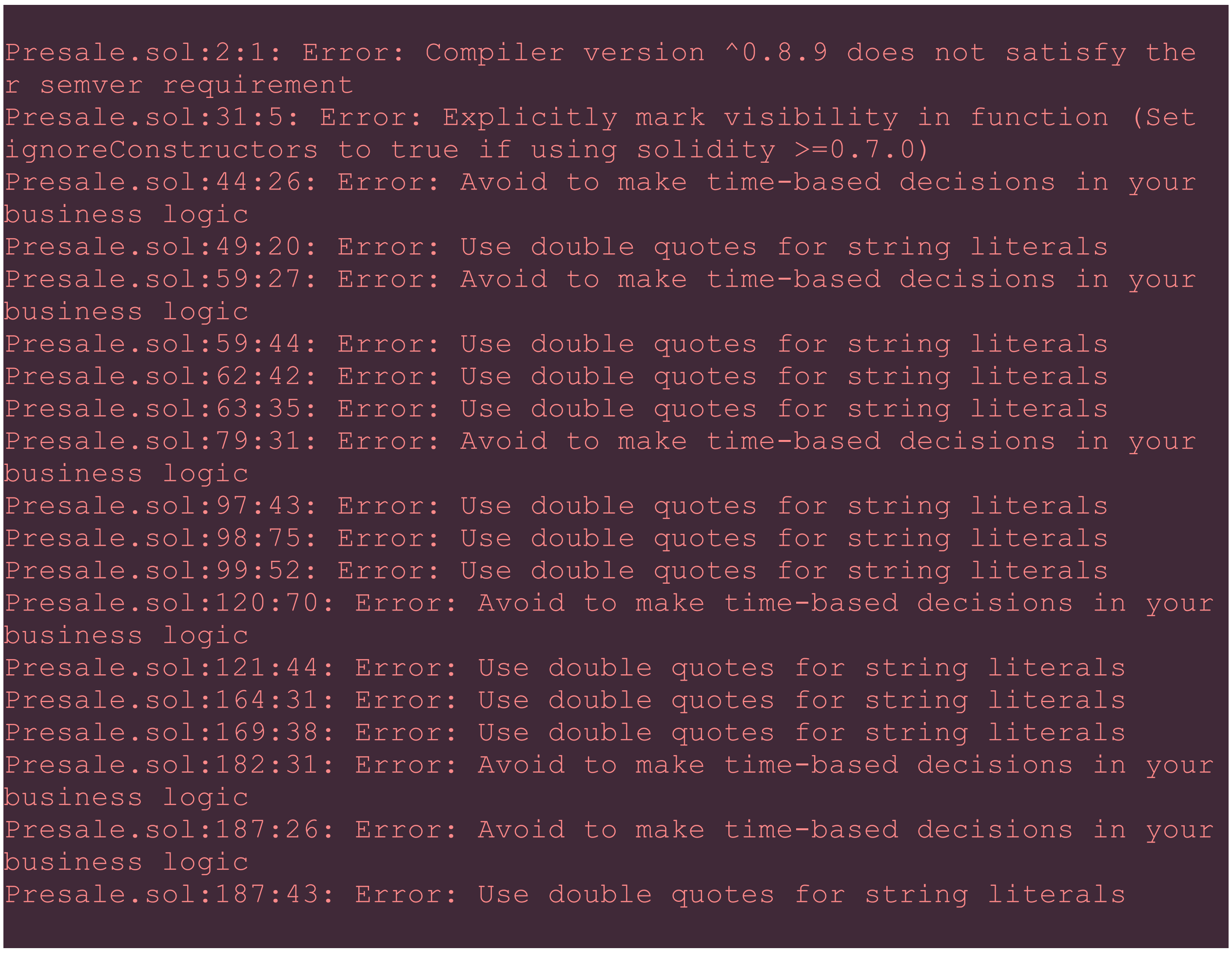
Solhint Linter

BlueSocialToken.sol



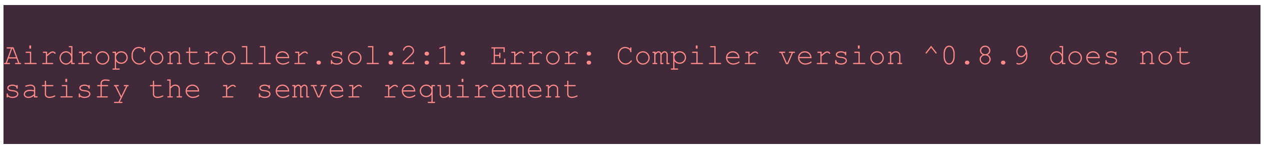
```
BlueSocialToken.sol:2:1: Error: Compiler version ^0.8.9 does not
satisfy the r semver requirement
BlueSocialToken.sol:13:1: Error: Contract name must be in CamelCase
BlueSocialToken.sol:20:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
BlueSocialToken.sol:26:39: Error: Visibility modifier must be first
in list of modifiers
BlueSocialToken.sol:74:5: Error: Code contains empty blocks
```

Presale.sol



```
Presale.sol:2:1: Error: Compiler version ^0.8.9 does not satisfy the
r semver requirement
Presale.sol:31:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
Presale.sol:44:26: Error: Avoid to make time-based decisions in your
business logic
Presale.sol:49:20: Error: Use double quotes for string literals
Presale.sol:59:27: Error: Avoid to make time-based decisions in your
business logic
Presale.sol:59:44: Error: Use double quotes for string literals
Presale.sol:62:42: Error: Use double quotes for string literals
Presale.sol:63:35: Error: Use double quotes for string literals
Presale.sol:79:31: Error: Avoid to make time-based decisions in your
business logic
Presale.sol:97:43: Error: Use double quotes for string literals
Presale.sol:98:75: Error: Use double quotes for string literals
Presale.sol:99:52: Error: Use double quotes for string literals
Presale.sol:120:70: Error: Avoid to make time-based decisions in your
business logic
Presale.sol:121:44: Error: Use double quotes for string literals
Presale.sol:164:31: Error: Use double quotes for string literals
Presale.sol:169:38: Error: Use double quotes for string literals
Presale.sol:182:31: Error: Avoid to make time-based decisions in your
business logic
Presale.sol:187:26: Error: Avoid to make time-based decisions in your
business logic
Presale.sol:187:43: Error: Use double quotes for string literals
```

AirdropController.sol



```
AirdropController.sol:2:1: Error: Compiler version ^0.8.9 does not
satisfy the r semver requirement
```

BSTAirdrop.sol

```
BSTAirdrop.sol:2:1: Error: Compiler version ^0.8.9 does not satisfy
the r semver requirement
BSTAirdrop.sol:18:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

EIP712Base.sol

```
EIP712Base.sol:2:1: Error: Compiler version ^0.8.9 does not satisfy
the r semver requirement
EIP712Base.sol:23:5: Error: Function name must be in mixedCase
```

Forwarder.sol

```
Forwarder.sol:2:1: Error: Compiler version ^0.8.9 does not satisfy
the r semver requirement
Forwarder.sol:30:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

NetworkAgnostic.sol

```
NetworkAgnostic.sol:2:1: Error: Compiler version ^0.8.9 does not
satisfy the r semver requirement
NetworkAgnostic.sol:19:5: Error: Explicitly mark visibility of state
NetworkAgnostic.sol:60:51: Error: Avoid using low level calls.
NetworkAgnostic.sol:106:32: Error: Code contains empty blocks
```

TokenSale.sol

```
TokenSale.sol:2:1: Error: Compiler version ^0.8.9 does not satisfy
the r semver requirement
TokenSale.sol:9:5: Error: Explicitly mark visibility of state
TokenSale.sol:17:5: Error: Function name must be in mixedCase
TokenSale.sol:35:9: Error: Possible reentrancy vulnerabilities. Avoid
state changes after transfer.
```

MetaTransactionLib.sol

```
MetaTransactionLib.sol:2:1: Error: Compiler version ^0.8.9 does not
```

```
satisfy the r semver requirement
MetaTransactionLib.sol:19:13: Error: Avoid using inline assembly. It
is acceptable only in rare cases
```

Software analysis result:

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

EtherAuthority