Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer:	GBPST
Website	https://eurst.io
Platform:	Ethereum
Language:	Solidity
Date:	September 28th, 2021

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

EtherAuthority was contracted by the GBPST team to perform the Security audit of GBPST Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on September 28th, 2021.

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

GBPstablecoin (symbol: GBPST) is a fiat-backed stablecoin issued by Wallex Trust, built on the Ethereum network according to the ERC20 standard for tokens.

Audit scope

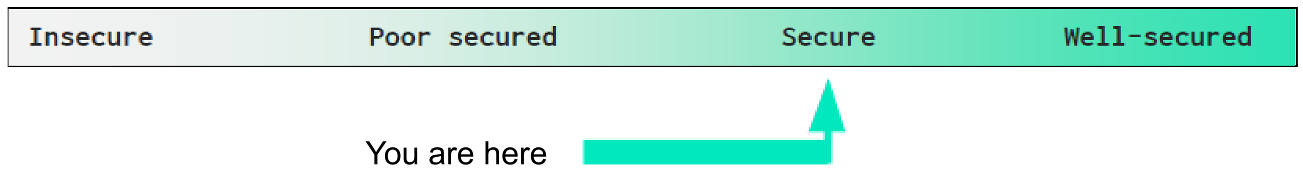
Name	**Code Review and Security Analysis Report for GBPST Token Smart Contracts**
Platform	**Ethereum / Solidity**
File 1	GBPST.sol
File 1 MD5 Hash	D1901958E3E97AA6AC5D2C1DE25717A4
File 2	UserRegistry.sol
File 2 MD5 Hash	81CBEA003C8C46E8C5B4020D5B161F7F
File 3	Climer.sol
File 3 MD5 Hash	B69BE11D9F34B43BBD62A93EC9028377
File 4	Registry.sol
File 4 MD5 Hash	81CBEA003C8C46E8C5B4020D5B161F7F
Audit Date	September 28th, 2021

Claimed Smart Contracts Features

Claimed Feature Detail	Our Observation
File 1: GBPST.sol ● Name: GBP Stable Token ● Symbol: GBPST ● Decimals: 18 ● Minting/Burning possible	**YES, This is valid.**
File 2: UserRegistry.sol ● The UserRegistry owner can access user KYC verifications, redeem status address, burn, etc.	**YES, This is valid.**
File 3: Climer.sol ● Allows the owner to claim ERC20 tokens or ETH sent to this contract.	**YES, This is valid.**
File 4: Registry.sol ● add/set/get Attribute.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Technically Secured"**. These contracts contain owner control, which does not make it fully decentralized. Owner (or authorized wallet) can mint/burn tokens, can control the user wallets, and possess many control processes.



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 1 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
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	Passed
	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	Not Set
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 4 smart contracts files. Smart contracts also contain Libraries, Smart contracts, inherits and Interfaces. These are compact and well written contracts.

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

The GBPST team has provided scenario and unit test scripts, which have helped to determine the integrity of the code in an automated way.

Code parts are **well** commented on smart contracts.

Documentation

We were given an GBPST Protocol smart contracts code in the form of the files. The hashes of those files are mentioned above in the table.

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

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 used in external smart contract calls.

AS-IS overview

GBPST.sol

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	transfer	write	Passed	No Issue
3	transferFrom	write	Passed	No Issue
4	_redeem	internal	Passed	No Issue
5	mint	write	access only Minter	No max minting set
6	wipeBlocklistedAccount	write	access only Wiper	No Issue
7	setUserRegistry	write	access only RegistryManager	No Issue
8	onlyMinter	modifier	Passed	No Issue
9	onlyWiper	modifier	Passed	No Issue
10	onlyRegistryManager	modifier	Passed	No Issue

UserRegistry.sol

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	setToken	write	access only Owner	No Issue
3	setMinBurnBound	write	access only Owner	No Issue
4	setMaxBurnBound	write	access only Owner	No Issue
5	registerNewUser	write	access only Owner	No Issue
6	getUser	read	access only Owner	No Issue
7	getUserById	read	access only Owner	No Issue
8	setUserId	write	access only Owner	No Issue
9	userKycVerified	write	access only Owner	No Issue
10	userKycUnverified	write	access only Owner	No Issue
11	enableRedeemAddress	write	access only Owner	No Issue
12	disableRedeemAddress	write	access only Owner	No Issue
13	verifyKycEnableRedeem	write	access only Owner	No Issue

14	unverifyKycDisableRedeem	write	access only Owner	No Issue
15	blockAccount	write	access only Owner	No Issue
16	unblockAccount	write	access only Owner	No Issue
17	getUserByRedeemAddress	read	Passed	No Issue
18	getRedeemAddress	read	Passed	No Issue
19	_isBlocked	internal	Passed	No Issue
20	_isUser	internal	Passed	No Issue
21	_isKyced	internal	Passed	No Issue
22	_isRedemptionAddress	internal	Passed	No Issue
23	isRedeem	external	access only Token	No Issue
24	isRedeemFrom	external	access only Token	No Issue
25	canTransfer	external	access only Token	No Issue
26	canTransferFrom	external	access only Token	No Issue
27	canMint	external	access only Token	No Issue
28	canBurn	external	access only Token	No Issue
29	canWipe	external	access only Token	No Issue
30	onlyToken	modifier	Passed	No Issue

Claimer.sol

Sl.	Functions	Type	Observation	Conclusion
1	claimToken	external	access only Owner	No Issue
2	claimEther	external	access only Owner	No Issue

Registry.sol

1	constructor	read	Passed	No Issue
2	setAttribute	write	access only Owner	No Issue
3	hasAttribute	read	Passed	No Issue
4	getAttribute	read	Passed	No Issue
5	getAttributeValue	read	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

No Medium severity vulnerabilities were found.

Low

(1) Unlimited token minting

```
function mint(address _to, uint256 _amount) public onlyMinter {
    userRegistry.canMint(_to);

    _mint(_to, _amount);

    emit Mint(_to, _amount);
}
```

Tokens minting without any maximum limit is considered inappropriate for the tokenomics. We recommend to place some limit of token minting to mitigate this issue.

Status: We got confirmation from GBPST team that this is a desired feature. Because GBPST is a stablecoin and thus minting of tokens is necessary as part of the business plan. So, this issue can be safely ignored.

Very Low / Informational / Best practices:

(1) Consider adding the latest solidity version:

```
pragma solidity ^0.8.0;
```

The current version is an upgraded one, but consider using 0.8.7 which is the latest at the time of this audit.

(2) Visibility external instead of public:

It is recommended to use the function visibility as external instead of public, if it is not called from inside the contract. This will save some gas.

https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

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 GBPST Minter can mint any amount of tokens.
- wipeBlocklistedAccount: The GBPST Wiper can destroy the tokens owned by a blocklisted account.
- setUserRegistry: The GBPST RegistryManager can set the user registry address.
- setToken: The UserRegistry owner can set a token.
- setMinBurnBound: The UserRegistry owner can set a minimum burn bound.
- setMaxBurnBound: The UserRegistry owner can set a maximum burn bound.
- registerNewUser: The UserRegistry owner can add a new register user.
- getUser: The UserRegistry owner can get the user address.
- getUserById: The UserRegistry owner can get a user by Id.
- setUserId: The UserRegistry owner can set user Id.
- userKycVerified: The UserRegistry owner can set users as KYC verified.
- enableRedeemAddress: The UserRegistry owner can set the user as KYC un-verified.

- disableRedeemAddress: The UserRegistry owner can enable `_account` redeem address to burn.
- verifyKycEnableRedeem: The UserRegistry owner can disable the `_account` redeem address to burn.
- unverifyKycDisableRedeem: The UserRegistry owner can set the user as KYC un-verified. Disables `_account` redeem address to burn.
- blockAccount: The UserRegistry owner can register `_account` as blocked.
- unblockAccount: The UserRegistry owner can register `_account` as unblocked.
- isRedeem: The UserRegistry token owner can determine if it is redeeming.
- isRedeemFrom: The UserRegistry token owner can determine if it is redeeming from.
- canTransfer: The UserRegistry token owner can throw if any of `_from` or `_to` is blocklisted.
- canTransferFrom: The UserRegistry token owner can throw if any of `_spender`, `_from` or `_to` is blocklisted.
- canMint: The UserRegistry token owner can throw if any of `_to` is not KYC verified or blocklisted.
- canBurn: The UserRegistry token owner can throw if any of `_from` is not enabled to burn or `_amount` lower than minBurnBound.
- canWipe: The UserRegistry token owner can throw if any of `_account` is not blocked.
- claimToken: The Claimer owner can send all token balance of an arbitrary erc20 token in the contract to another address.
- claimEther: The Claimer owner can send all eth balance in the contract to another address send eth balance to.
- setAttribute: The Registry owner can set attributes.

Conclusion

We were given contract codes. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and once they are resolved/acknowledged, **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 **"Technically 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.

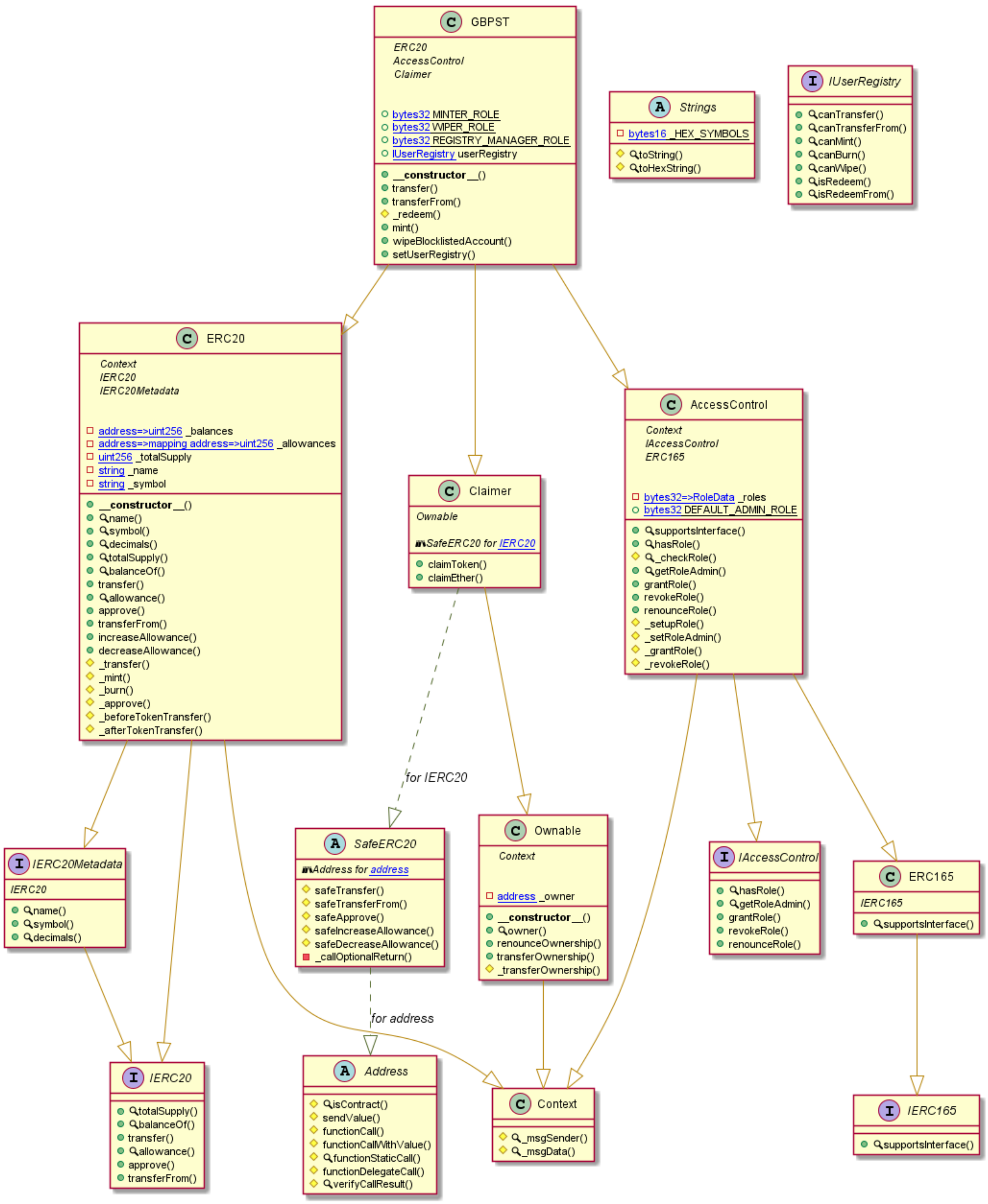
Technical Disclaimer

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.

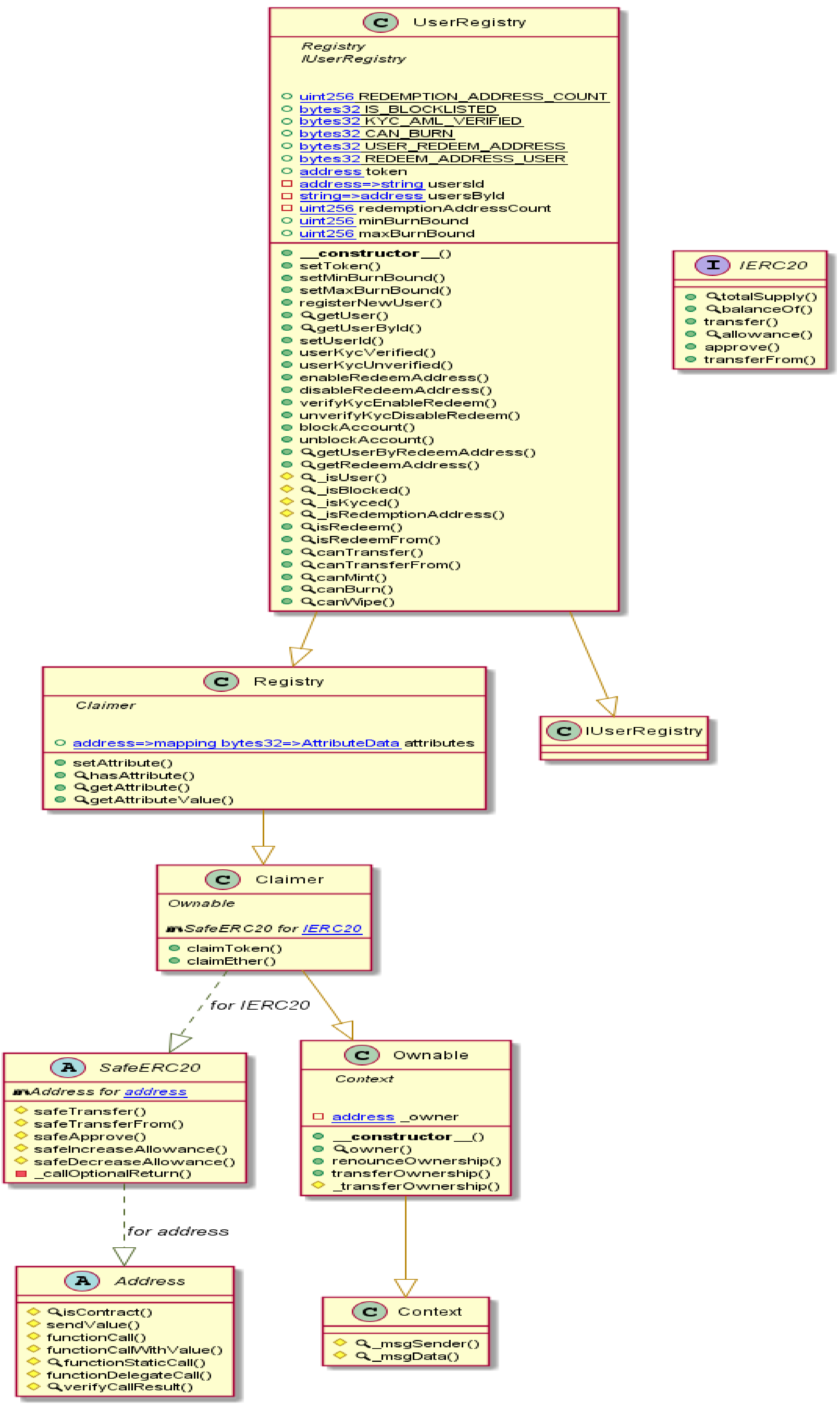
Appendix

Code Flow Diagram - GBPST Protocol

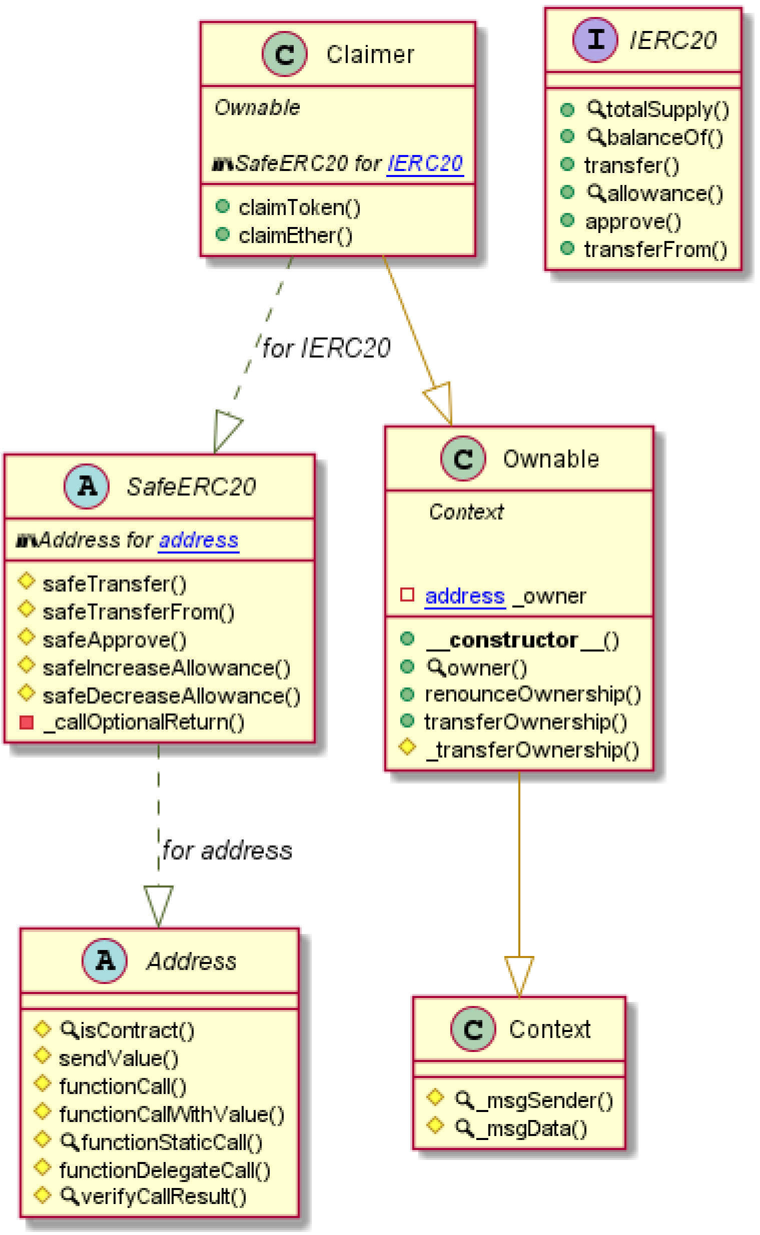
GBPST Token



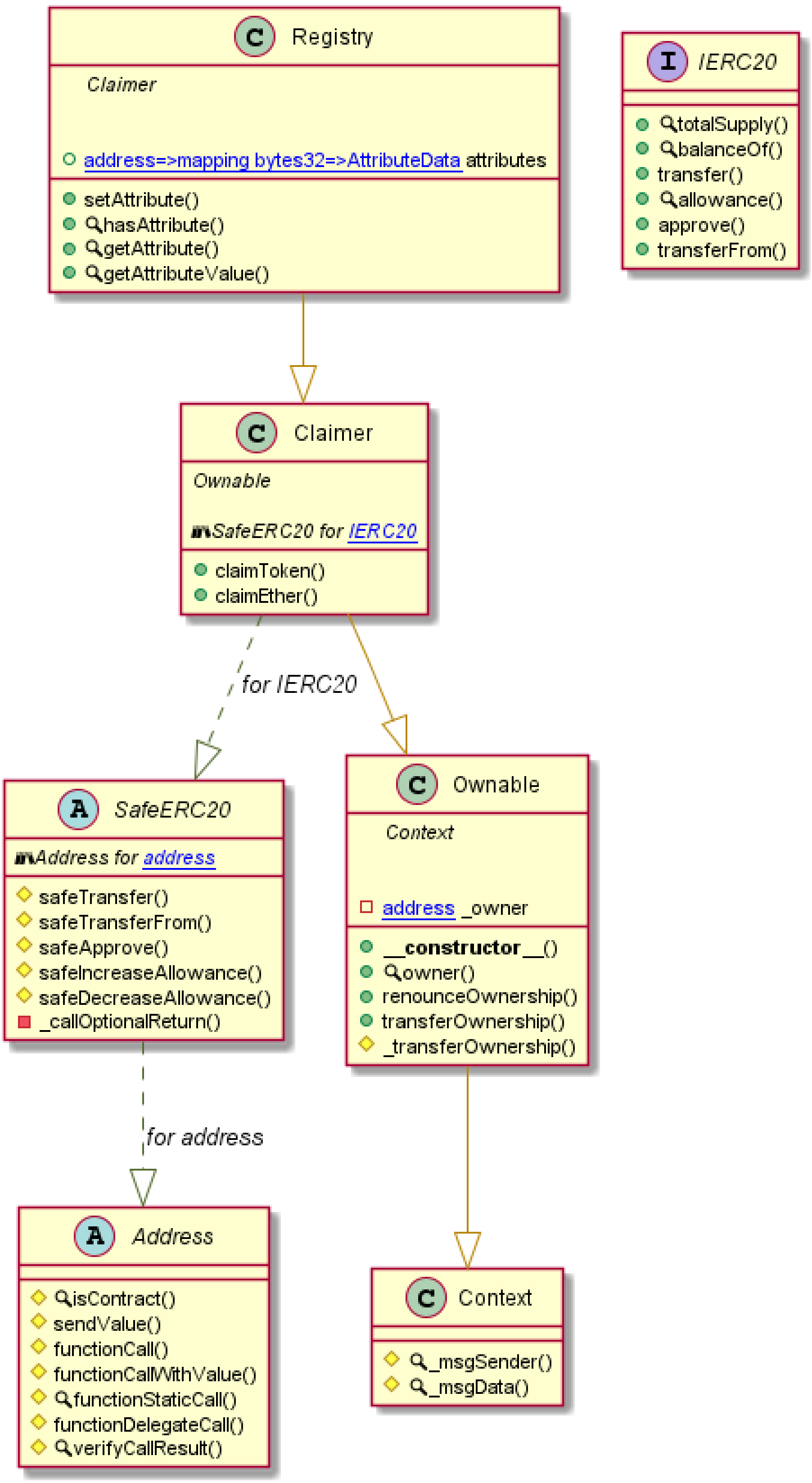
UserRegistry Token



Climer Token

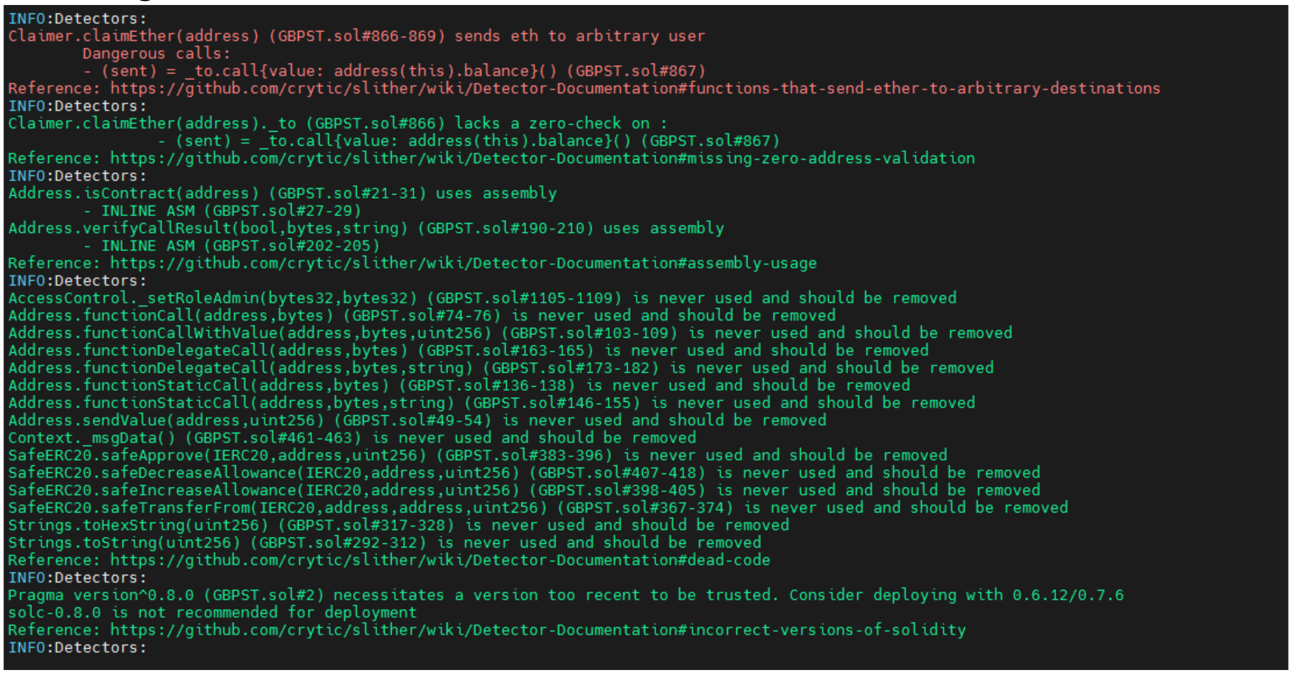


Registry Token

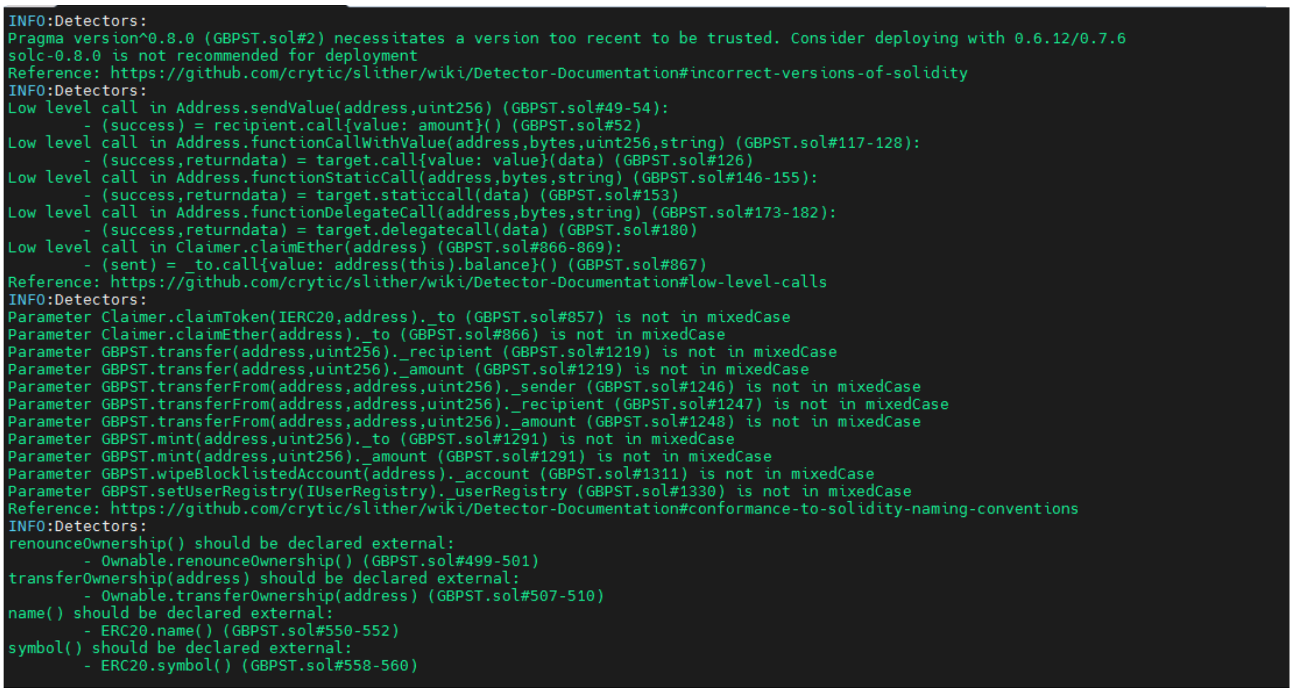


Slither Results Log

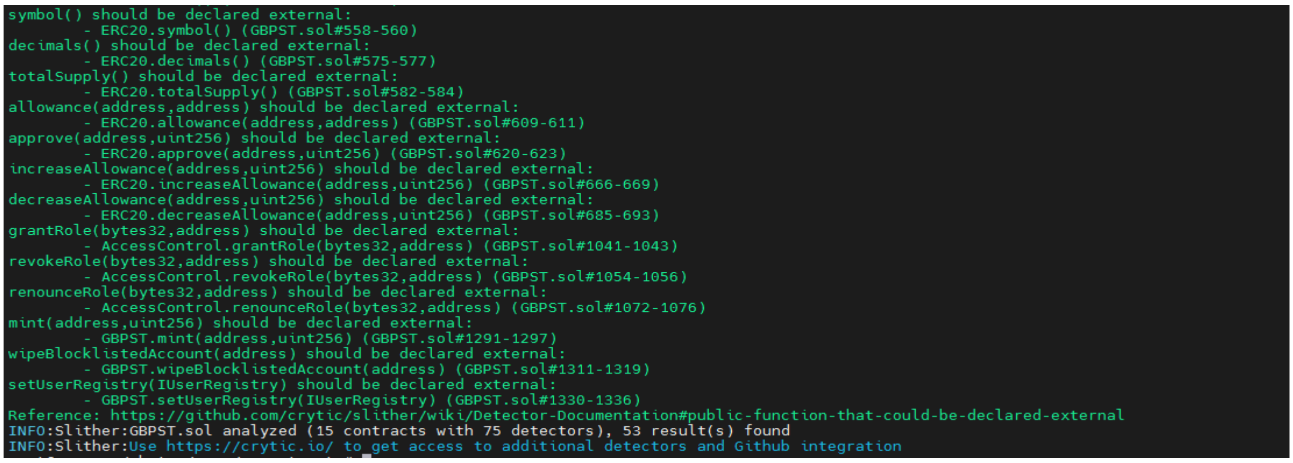
Slither log >> GBPST.sol



```
INFO:Detectors:
Claimer.claimEther(address) (GBPST.sol#866-869) sends eth to arbitrary user
        Dangerous calls:
        - (sent) = _to.call{value: address(this).balance}() (GBPST.sol#867)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Claimer.claimEther(address)._to (GBPST.sol#866) lacks a zero-check on :
                - (sent) = _to.call{value: address(this).balance}() (GBPST.sol#867)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Address.isContract(address) (GBPST.sol#21-31) uses assembly
        - INLINE ASM (GBPST.sol#27-29)
Address.verifyCallResult(bool,bytes,string) (GBPST.sol#190-210) uses assembly
        - INLINE ASM (GBPST.sol#202-205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AccessControl._setRoleAdmin(bytes32,bytes32) (GBPST.sol#1105-1109) is never used and should be removed
Address.functionCall(address,bytes) (GBPST.sol#74-76) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (GBPST.sol#103-109) is never used and should be removed
Address.functionDelegateCall(address,bytes) (GBPST.sol#163-165) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (GBPST.sol#173-182) is never used and should be removed
Address.functionStaticCall(address,bytes) (GBPST.sol#136-138) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (GBPST.sol#146-155) is never used and should be removed
Address.sendValue(address,uint256) (GBPST.sol#49-54) is never used and should be removed
Context._msgData() (GBPST.sol#461-463) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (GBPST.sol#383-396) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (GBPST.sol#407-418) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (GBPST.sol#398-405) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (GBPST.sol#367-374) is never used and should be removed
Strings.toHexString(uint256) (GBPST.sol#317-328) is never used and should be removed
Strings.toString(uint256) (GBPST.sol#292-312) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (GBPST.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
```

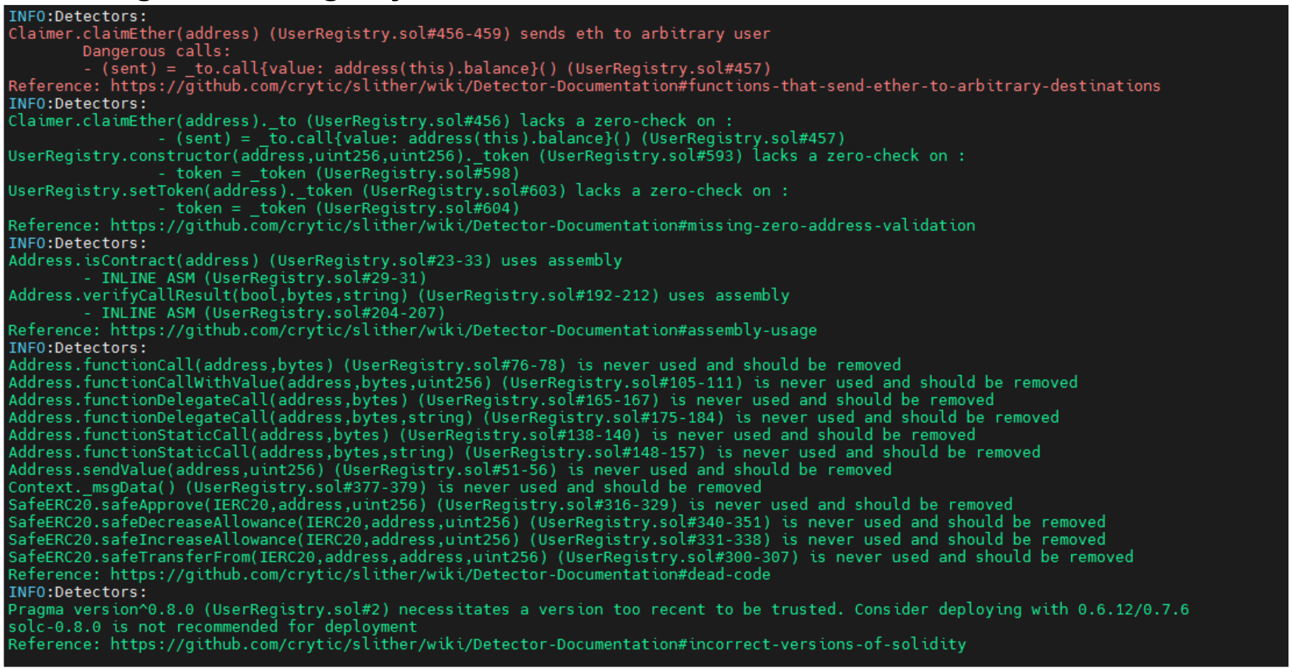


```
INFO:Detectors:
Pragma version^0.8.0 (GBPST.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (GBPST.sol#49-54):
        - (success) = recipient.call{value: amount}() (GBPST.sol#52)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (GBPST.sol#117-128):
        - (success,returndata) = target.call{value: value}(data) (GBPST.sol#126)
Low level call in Address.functionStaticCall(address,bytes,string) (GBPST.sol#146-155):
        - (success,returndata) = target.staticcall(data) (GBPST.sol#153)
Low level call in Address.functionDelegateCall(address,bytes,string) (GBPST.sol#173-182):
        - (success,returndata) = target.delegatecall(data) (GBPST.sol#180)
Low level call in Claimer.claimEther(address) (GBPST.sol#866-869):
        - (sent) = _to.call{value: address(this).balance}() (GBPST.sol#867)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Claimer.claimToken(IERC20,address)._to (GBPST.sol#857) is not in mixedCase
Parameter Claimer.claimEther(address)._to (GBPST.sol#866) is not in mixedCase
Parameter GBPST.transfer(address,uint256)._recipient (GBPST.sol#1219) is not in mixedCase
Parameter GBPST.transfer(address,uint256)._amount (GBPST.sol#1219) is not in mixedCase
Parameter GBPST.transferFrom(address,address,uint256)._sender (GBPST.sol#1246) is not in mixedCase
Parameter GBPST.transferFrom(address,address,uint256)._recipient (GBPST.sol#1247) is not in mixedCase
Parameter GBPST.transferFrom(address,address,uint256)._amount (GBPST.sol#1248) is not in mixedCase
Parameter GBPST.mint(address,uint256)._to (GBPST.sol#1291) is not in mixedCase
Parameter GBPST.mint(address,uint256)._amount (GBPST.sol#1291) is not in mixedCase
Parameter GBPST.wipeBlocklistedAccount(address)._account (GBPST.sol#1311) is not in mixedCase
Parameter GBPST.setUserRegistry(IUserRegistry)._userRegistry (GBPST.sol#1330) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (GBPST.sol#499-501)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (GBPST.sol#507-510)
name() should be declared external:
        - ERC20.name() (GBPST.sol#550-552)
symbol() should be declared external:
        - ERC20.symbol() (GBPST.sol#558-560)
```

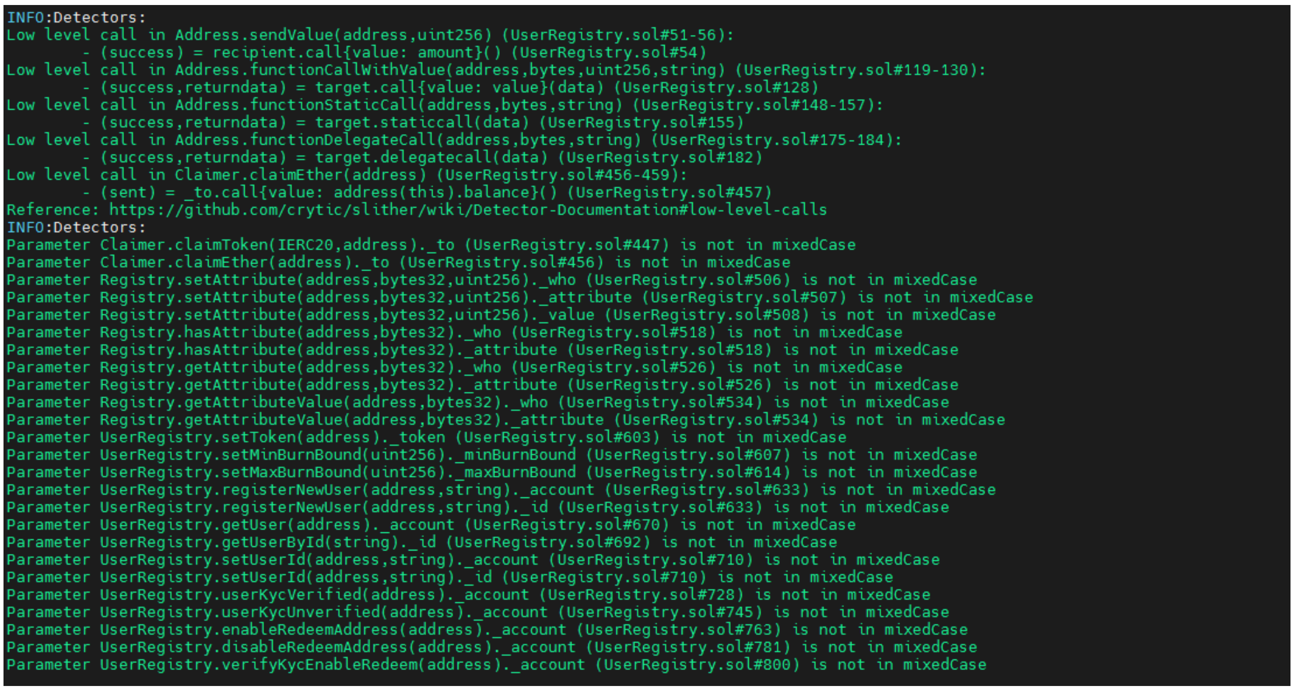


```
symbol() should be declared external:
        - ERC20.symbol() (GBPST.sol#558-560)
decimals() should be declared external:
        - ERC20.decimals() (GBPST.sol#575-577)
totalSupply() should be declared external:
        - ERC20.totalSupply() (GBPST.sol#582-584)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (GBPST.sol#609-611)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (GBPST.sol#620-623)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (GBPST.sol#666-669)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (GBPST.sol#685-693)
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (GBPST.sol#1041-1043)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (GBPST.sol#1054-1056)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (GBPST.sol#1072-1076)
mint(address,uint256) should be declared external:
        - GBPST.mint(address,uint256) (GBPST.sol#1291-1297)
wipeBlocklistedAccount(address) should be declared external:
        - GBPST.wipeBlocklistedAccount(address) (GBPST.sol#1311-1319)
setUserRegistry(IUserRegistry) should be declared external:
        - GBPST.setUserRegistry(IUserRegistry) (GBPST.sol#1330-1336)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:GBPST.sol analyzed (15 contracts with 75 detectors), 53 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

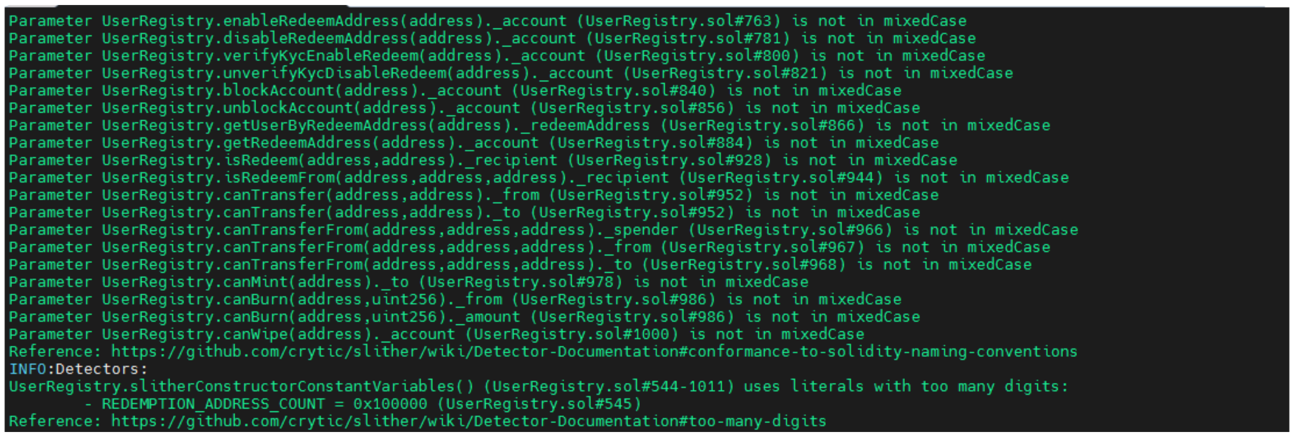
Slither log >> UserRegistry.sol



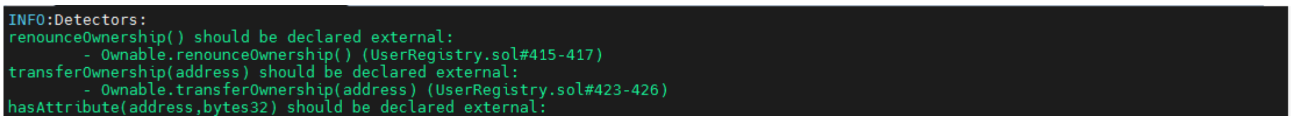
```
INFO:Detectors:
Claimer.claimEther(address) (UserRegistry.sol#456-459) sends eth to arbitrary user
        Dangerous calls:
        - (sent) = _to.call{value: address(this).balance}() (UserRegistry.sol#457)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Claimer.claimEther(address)._to (UserRegistry.sol#456) lacks a zero-check on :
                - (sent) = _to.call{value: address(this).balance}() (UserRegistry.sol#457)
UserRegistry.constructor(address,uint256,uint256)._token (UserRegistry.sol#593) lacks a zero-check on :
                - token = _token (UserRegistry.sol#598)
UserRegistry.setToken(address)._token (UserRegistry.sol#603) lacks a zero-check on :
                - token = _token (UserRegistry.sol#604)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Address.isContract(address) (UserRegistry.sol#23-33) uses assembly
        - INLINE ASM (UserRegistry.sol#29-31)
Address.verifyCallResult(bool,bytes,string) (UserRegistry.sol#192-212) uses assembly
        - INLINE ASM (UserRegistry.sol#204-207)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (UserRegistry.sol#76-78) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (UserRegistry.sol#105-111) is never used and should be removed
Address.functionDelegateCall(address,bytes) (UserRegistry.sol#165-167) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (UserRegistry.sol#175-184) is never used and should be removed
Address.functionStaticCall(address,bytes) (UserRegistry.sol#138-140) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (UserRegistry.sol#148-157) is never used and should be removed
Address.sendValue(address,uint256) (UserRegistry.sol#51-56) is never used and should be removed
Context._msgData() (UserRegistry.sol#377-379) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (UserRegistry.sol#316-329) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (UserRegistry.sol#340-351) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (UserRegistry.sol#331-338) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (UserRegistry.sol#300-307) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (UserRegistry.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```



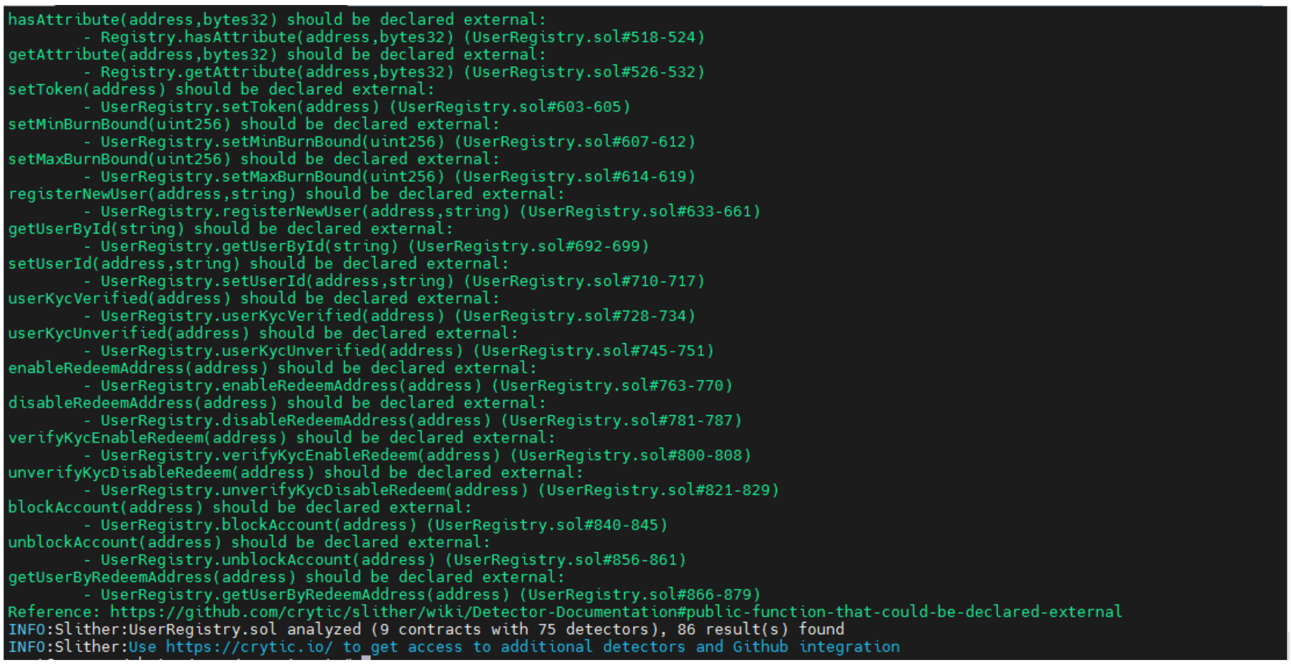
```
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (UserRegistry.sol#51-56):
        - (success) = recipient.call{value: amount}() (UserRegistry.sol#54)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (UserRegistry.sol#119-130):
        - (success,returndata) = target.call{value: value}(data) (UserRegistry.sol#128)
Low level call in Address.functionStaticCall(address,bytes,string) (UserRegistry.sol#148-157):
        - (success,returndata) = target.staticcall(data) (UserRegistry.sol#155)
Low level call in Address.functionDelegateCall(address,bytes,string) (UserRegistry.sol#175-184):
        - (success,returndata) = target.delegatecall(data) (UserRegistry.sol#182)
Low level call in Claimer.claimEther(address) (UserRegistry.sol#456-459):
        - (sent) = _to.call{value: address(this).balance}() (UserRegistry.sol#457)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Claimer.claimToken(IERC20,address)._to (UserRegistry.sol#447) is not in mixedCase
Parameter Claimer.claimEther(address)._to (UserRegistry.sol#456) is not in mixedCase
Parameter Registry.setAttribute(address,bytes32,uint256)._who (UserRegistry.sol#506) is not in mixedCase
Parameter Registry.setAttribute(address,bytes32,uint256)._attribute (UserRegistry.sol#507) is not in mixedCase
Parameter Registry.setAttribute(address,bytes32,uint256)._value (UserRegistry.sol#508) is not in mixedCase
Parameter Registry.hasAttribute(address,bytes32)._who (UserRegistry.sol#518) is not in mixedCase
Parameter Registry.hasAttribute(address,bytes32)._attribute (UserRegistry.sol#518) is not in mixedCase
Parameter Registry.getAttribute(address,bytes32)._who (UserRegistry.sol#526) is not in mixedCase
Parameter Registry.getAttribute(address,bytes32)._attribute (UserRegistry.sol#526) is not in mixedCase
Parameter Registry.getAttributeValue(address,bytes32)._who (UserRegistry.sol#534) is not in mixedCase
Parameter Registry.getAttributeValue(address,bytes32)._attribute (UserRegistry.sol#534) is not in mixedCase
Parameter UserRegistry.setToken(address)._token (UserRegistry.sol#603) is not in mixedCase
Parameter UserRegistry.setMinBurnBound(uint256)._minBurnBound (UserRegistry.sol#607) is not in mixedCase
Parameter UserRegistry.setMaxBurnBound(uint256)._maxBurnBound (UserRegistry.sol#614) is not in mixedCase
Parameter UserRegistry.registerNewUser(address,string)._account (UserRegistry.sol#633) is not in mixedCase
Parameter UserRegistry.registerNewUser(address,string)._id (UserRegistry.sol#633) is not in mixedCase
Parameter UserRegistry.getUser(address)._account (UserRegistry.sol#670) is not in mixedCase
Parameter UserRegistry.getUserById(string)._id (UserRegistry.sol#692) is not in mixedCase
Parameter UserRegistry.setUserId(address,string)._account (UserRegistry.sol#710) is not in mixedCase
Parameter UserRegistry.setUserId(address,string)._id (UserRegistry.sol#710) is not in mixedCase
Parameter UserRegistry.userKycVerified(address)._account (UserRegistry.sol#728) is not in mixedCase
Parameter UserRegistry.userKycUnverified(address)._account (UserRegistry.sol#745) is not in mixedCase
Parameter UserRegistry.enableRedeemAddress(address)._account (UserRegistry.sol#763) is not in mixedCase
Parameter UserRegistry.disableRedeemAddress(address)._account (UserRegistry.sol#781) is not in mixedCase
Parameter UserRegistry.verifyKycEnableRedeem(address)._account (UserRegistry.sol#800) is not in mixedCase
```



```
Parameter UserRegistry.enableRedeemAddress(address)._account (UserRegistry.sol#763) is not in mixedCase
Parameter UserRegistry.disableRedeemAddress(address)._account (UserRegistry.sol#781) is not in mixedCase
Parameter UserRegistry.verifyKycEnableRedeem(address)._account (UserRegistry.sol#800) is not in mixedCase
Parameter UserRegistry.unverifyKycDisableRedeem(address)._account (UserRegistry.sol#821) is not in mixedCase
Parameter UserRegistry.blockAccount(address)._account (UserRegistry.sol#840) is not in mixedCase
Parameter UserRegistry.unblockAccount(address)._account (UserRegistry.sol#856) is not in mixedCase
Parameter UserRegistry.getUserByRedeemAddress(address)._redeemAddress (UserRegistry.sol#866) is not in mixedCase
Parameter UserRegistry.getRedeemAddress(address)._account (UserRegistry.sol#884) is not in mixedCase
Parameter UserRegistry.isRedeem(address,address)._recipient (UserRegistry.sol#928) is not in mixedCase
Parameter UserRegistry.isRedeemFrom(address,address,address)._recipient (UserRegistry.sol#944) is not in mixedCase
Parameter UserRegistry.canTransfer(address,address)._from (UserRegistry.sol#952) is not in mixedCase
Parameter UserRegistry.canTransfer(address,address)._to (UserRegistry.sol#952) is not in mixedCase
Parameter UserRegistry.canTransferFrom(address,address,address)._spender (UserRegistry.sol#966) is not in mixedCase
Parameter UserRegistry.canTransferFrom(address,address,address)._from (UserRegistry.sol#967) is not in mixedCase
Parameter UserRegistry.canTransferFrom(address,address,address)._to (UserRegistry.sol#968) is not in mixedCase
Parameter UserRegistry.canMint(address)._to (UserRegistry.sol#978) is not in mixedCase
Parameter UserRegistry.canBurn(address,uint256)._from (UserRegistry.sol#986) is not in mixedCase
Parameter UserRegistry.canBurn(address,uint256)._amount (UserRegistry.sol#986) is not in mixedCase
Parameter UserRegistry.canWipe(address)._account (UserRegistry.sol#1000) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
UserRegistry.slitherConstructorConstantVariables() (UserRegistry.sol#544-1011) uses literals with too many digits:
        - REDEMPTION_ADDRESS_COUNT = 0x100000 (UserRegistry.sol#545)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

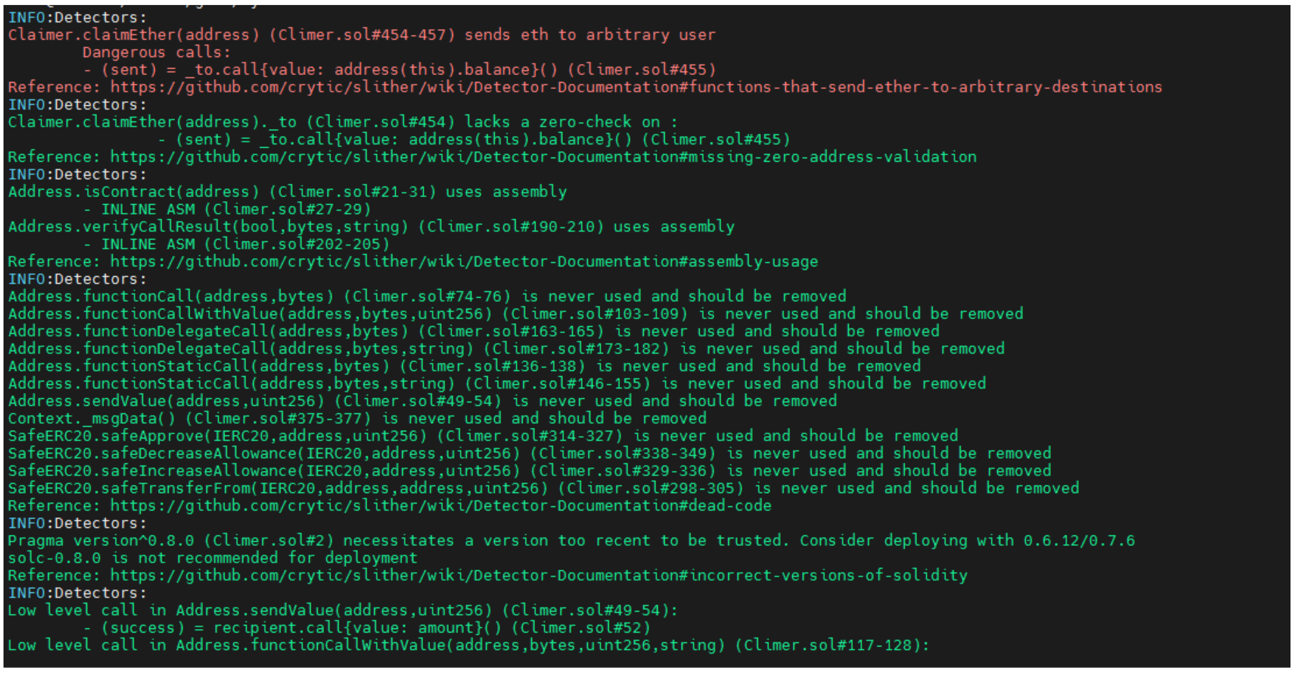


```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (UserRegistry.sol#415-417)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (UserRegistry.sol#423-426)
hasAttribute(address,bytes32) should be declared external:
```

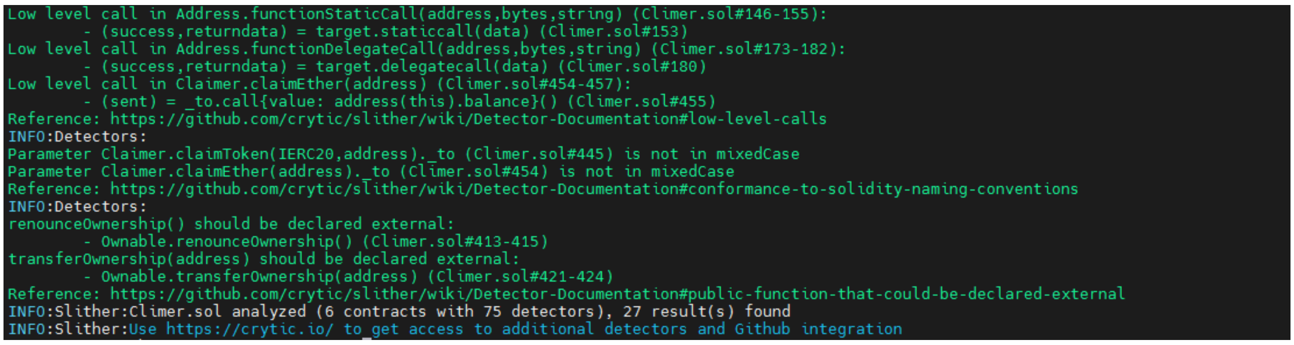


```
hasAttribute(address,bytes32) should be declared external:
        - Registry.hasAttribute(address,bytes32) (UserRegistry.sol#518-524)
getAttribute(address,bytes32) should be declared external:
        - Registry.getAttribute(address,bytes32) (UserRegistry.sol#526-532)
setToken(address) should be declared external:
        - UserRegistry.setToken(address) (UserRegistry.sol#603-605)
setMinBurnBound(uint256) should be declared external:
        - UserRegistry.setMinBurnBound(uint256) (UserRegistry.sol#607-612)
setMaxBurnBound(uint256) should be declared external:
        - UserRegistry.setMaxBurnBound(uint256) (UserRegistry.sol#614-619)
registerNewUser(address,string) should be declared external:
        - UserRegistry.registerNewUser(address,string) (UserRegistry.sol#633-661)
getUserById(string) should be declared external:
        - UserRegistry.getUserById(string) (UserRegistry.sol#692-699)
setUserId(address,string) should be declared external:
        - UserRegistry.setUserId(address,string) (UserRegistry.sol#710-717)
userKycVerified(address) should be declared external:
        - UserRegistry.userKycVerified(address) (UserRegistry.sol#728-734)
userKycUnverified(address) should be declared external:
        - UserRegistry.userKycUnverified(address) (UserRegistry.sol#745-751)
enableRedeemAddress(address) should be declared external:
        - UserRegistry.enableRedeemAddress(address) (UserRegistry.sol#763-770)
disableRedeemAddress(address) should be declared external:
        - UserRegistry.disableRedeemAddress(address) (UserRegistry.sol#781-787)
verifyKycEnableRedeem(address) should be declared external:
        - UserRegistry.verifyKycEnableRedeem(address) (UserRegistry.sol#800-808)
unverifyKycDisableRedeem(address) should be declared external:
        - UserRegistry.unverifyKycDisableRedeem(address) (UserRegistry.sol#821-829)
blockAccount(address) should be declared external:
        - UserRegistry.blockAccount(address) (UserRegistry.sol#840-845)
unblockAccount(address) should be declared external:
        - UserRegistry.unblockAccount(address) (UserRegistry.sol#856-861)
getUserByRedeemAddress(address) should be declared external:
        - UserRegistry.getUserByRedeemAddress(address) (UserRegistry.sol#866-879)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:UserRegistry.sol analyzed (9 contracts with 75 detectors), 86 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Climer.sol

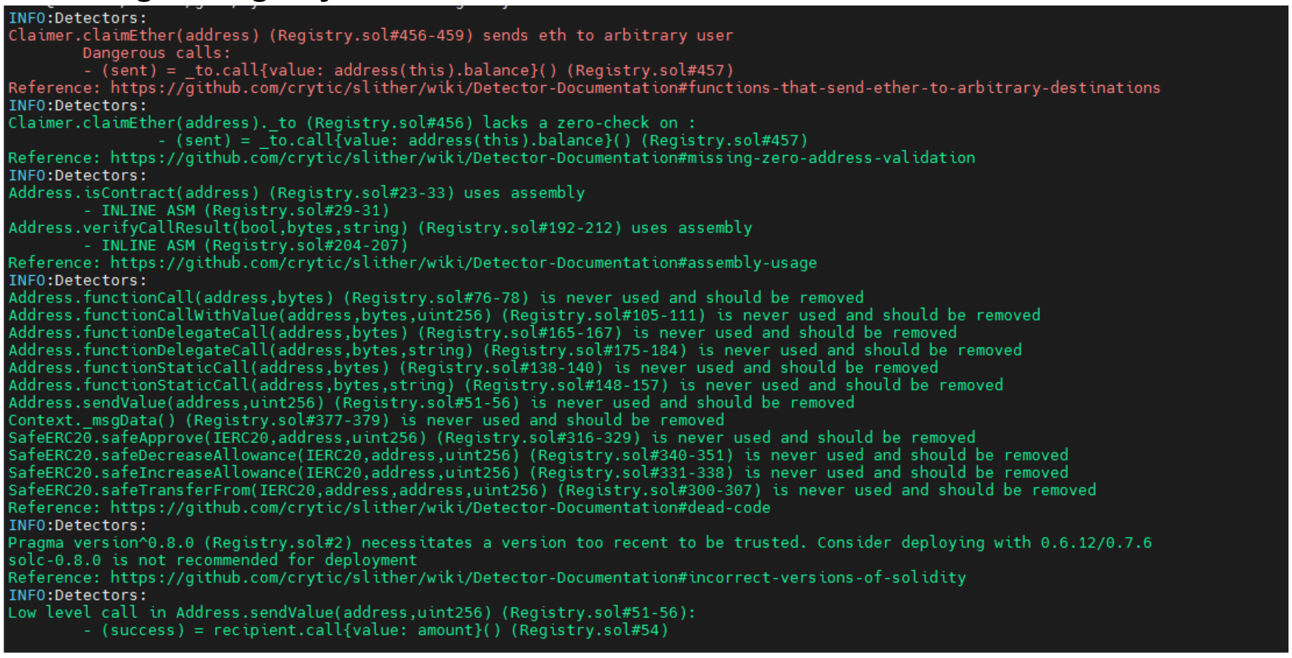


```
INFO:Detectors:
Claimer.claimEther(address) (Climer.sol#454-457) sends eth to arbitrary user
        Dangerous calls:
        - (sent) = _to.call{value: address(this).balance}() (Climer.sol#455)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Claimer.claimEther(address)._to (Climer.sol#454) lacks a zero-check on :
                - (sent) = _to.call{value: address(this).balance}() (Climer.sol#455)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Address.isContract(address) (Climer.sol#21-31) uses assembly
        - INLINE ASM (Climer.sol#27-29)
Address.verifyCallResult(bool,bytes,string) (Climer.sol#190-210) uses assembly
        - INLINE ASM (Climer.sol#202-205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (Climer.sol#74-76) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Climer.sol#103-109) is never used and should be removed
Address.functionDelegateCall(address,bytes) (Climer.sol#163-165) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (Climer.sol#173-182) is never used and should be removed
Address.functionStaticCall(address,bytes) (Climer.sol#136-138) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (Climer.sol#146-155) is never used and should be removed
Address.sendValue(address,uint256) (Climer.sol#49-54) is never used and should be removed
Context._msgData() (Climer.sol#375-377) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (Climer.sol#314-327) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (Climer.sol#338-349) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (Climer.sol#329-336) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (Climer.sol#298-305) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (Climer.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (Climer.sol#49-54):
        - (success) = recipient.call{value: amount}() (Climer.sol#52)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Climer.sol#117-128):
```

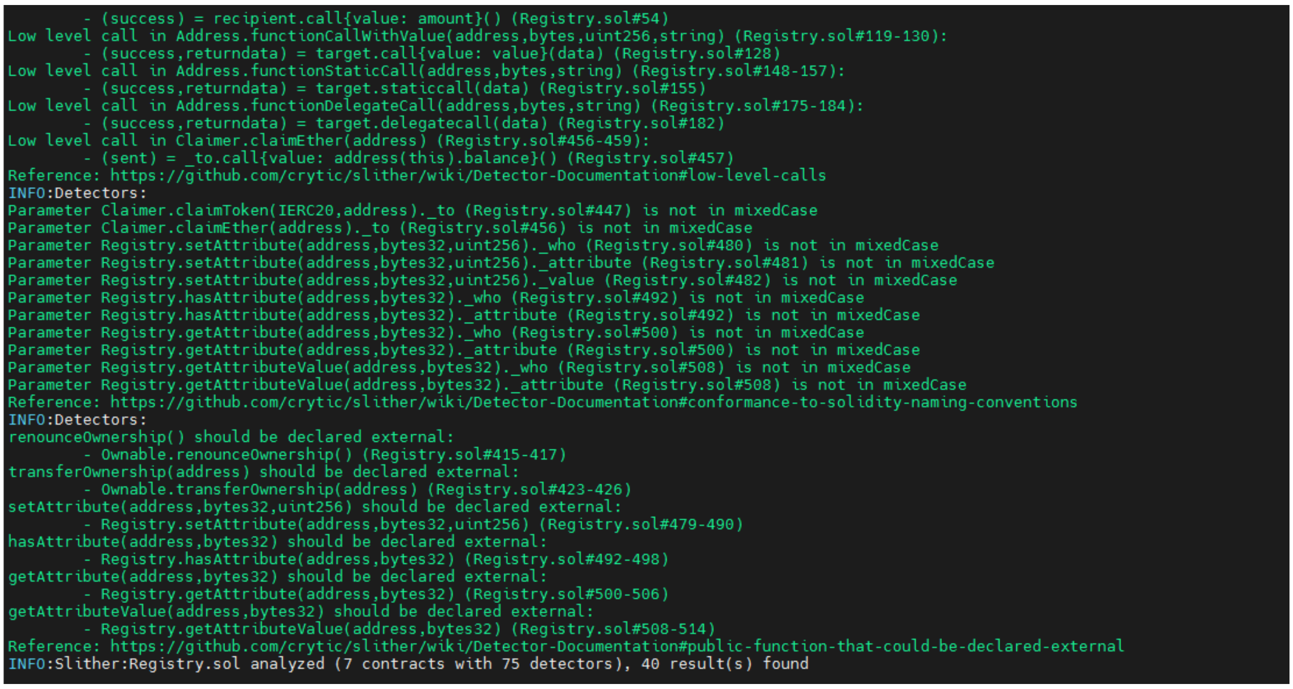


```
Low level call in Address.functionStaticCall(address,bytes,string) (Climer.sol#146-155):
        - (success,returndata) = target.staticcall(data) (Climer.sol#153)
Low level call in Address.functionDelegateCall(address,bytes,string) (Climer.sol#173-182):
        - (success,returndata) = target.delegatecall(data) (Climer.sol#180)
Low level call in Claimer.claimEther(address) (Climer.sol#454-457):
        - (sent) = _to.call{value: address(this).balance}() (Climer.sol#455)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Claimer.claimToken(IERC20,address)._to (Climer.sol#445) is not in mixedCase
Parameter Claimer.claimEther(address)._to (Climer.sol#454) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Climer.sol#413-415)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Climer.sol#421-424)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Climer.sol analyzed (6 contracts with 75 detectors), 27 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

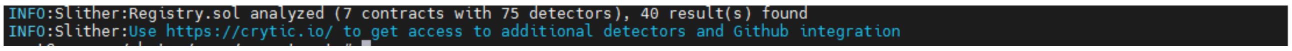
Slither log >> Registry.sol



```
INFO:Detectors:
Claimer.claimEther(address) (Registry.sol#456-459) sends eth to arbitrary user
        Dangerous calls:
        - (sent) = _to.call{value: address(this).balance}() (Registry.sol#457)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Claimer.claimEther(address)._to (Registry.sol#456) lacks a zero-check on :
                - (sent) = _to.call{value: address(this).balance}() (Registry.sol#457)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Address.isContract(address) (Registry.sol#23-33) uses assembly
        - INLINE ASM (Registry.sol#29-31)
Address.verifyCallResult(bool,bytes,string) (Registry.sol#192-212) uses assembly
        - INLINE ASM (Registry.sol#204-207)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (Registry.sol#76-78) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Registry.sol#105-111) is never used and should be removed
Address.functionDelegateCall(address,bytes) (Registry.sol#165-167) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (Registry.sol#175-184) is never used and should be removed
Address.functionStaticCall(address,bytes) (Registry.sol#138-140) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (Registry.sol#148-157) is never used and should be removed
Address.sendValue(address,uint256) (Registry.sol#51-56) is never used and should be removed
Context._msgData() (Registry.sol#377-379) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (Registry.sol#316-329) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (Registry.sol#340-351) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (Registry.sol#331-338) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (Registry.sol#300-307) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (Registry.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (Registry.sol#51-56):
        - (success) = recipient.call{value: amount}() (Registry.sol#54)
```



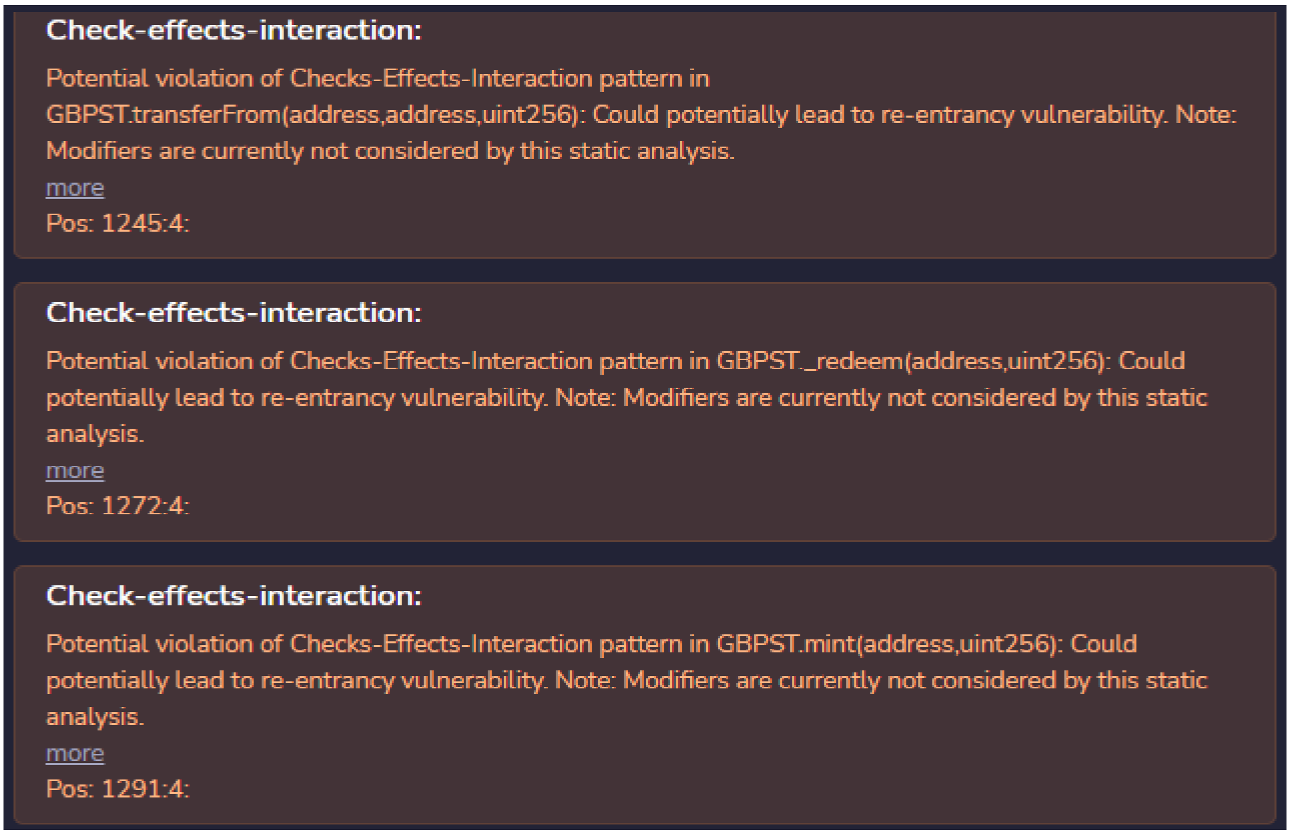
```
        - (success) = recipient.call{value: amount}() (Registry.sol#54)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Registry.sol#119-130):
        - (success,returndata) = target.call{value: value}(data) (Registry.sol#128)
Low level call in Address.functionStaticCall(address,bytes,string) (Registry.sol#148-157):
        - (success,returndata) = target.staticcall(data) (Registry.sol#155)
Low level call in Address.functionDelegateCall(address,bytes,string) (Registry.sol#175-184):
        - (success,returndata) = target.delegatecall(data) (Registry.sol#182)
Low level call in Claimer.claimEther(address) (Registry.sol#456-459):
        - (sent) = _to.call{value: address(this).balance}() (Registry.sol#457)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Claimer.claimToken(IERC20,address)._to (Registry.sol#447) is not in mixedCase
Parameter Claimer.claimEther(address)._to (Registry.sol#456) is not in mixedCase
Parameter Registry.setAttribute(address,bytes32,uint256)._who (Registry.sol#480) is not in mixedCase
Parameter Registry.setAttribute(address,bytes32,uint256)._attribute (Registry.sol#481) is not in mixedCase
Parameter Registry.setAttribute(address,bytes32,uint256)._value (Registry.sol#482) is not in mixedCase
Parameter Registry.hasAttribute(address,bytes32)._who (Registry.sol#492) is not in mixedCase
Parameter Registry.hasAttribute(address,bytes32)._attribute (Registry.sol#492) is not in mixedCase
Parameter Registry.getAttribute(address,bytes32)._who (Registry.sol#500) is not in mixedCase
Parameter Registry.getAttribute(address,bytes32)._attribute (Registry.sol#500) is not in mixedCase
Parameter Registry.getAttributeValue(address,bytes32)._who (Registry.sol#508) is not in mixedCase
Parameter Registry.getAttributeValue(address,bytes32)._attribute (Registry.sol#508) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Registry.sol#415-417)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Registry.sol#423-426)
setAttribute(address,bytes32,uint256) should be declared external:
        - Registry.setAttribute(address,bytes32,uint256) (Registry.sol#479-490)
hasAttribute(address,bytes32) should be declared external:
        - Registry.hasAttribute(address,bytes32) (Registry.sol#492-498)
getAttribute(address,bytes32) should be declared external:
        - Registry.getAttribute(address,bytes32) (Registry.sol#500-506)
getAttributeValue(address,bytes32) should be declared external:
        - Registry.getAttributeValue(address,bytes32) (Registry.sol#508-514)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Registry.sol analyzed (7 contracts with 75 detectors), 40 result(s) found
```



```
INFO:Slither:Registry.sol analyzed (7 contracts with 75 detectors), 40 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

GBPST.sol



Check-effects-interaction:

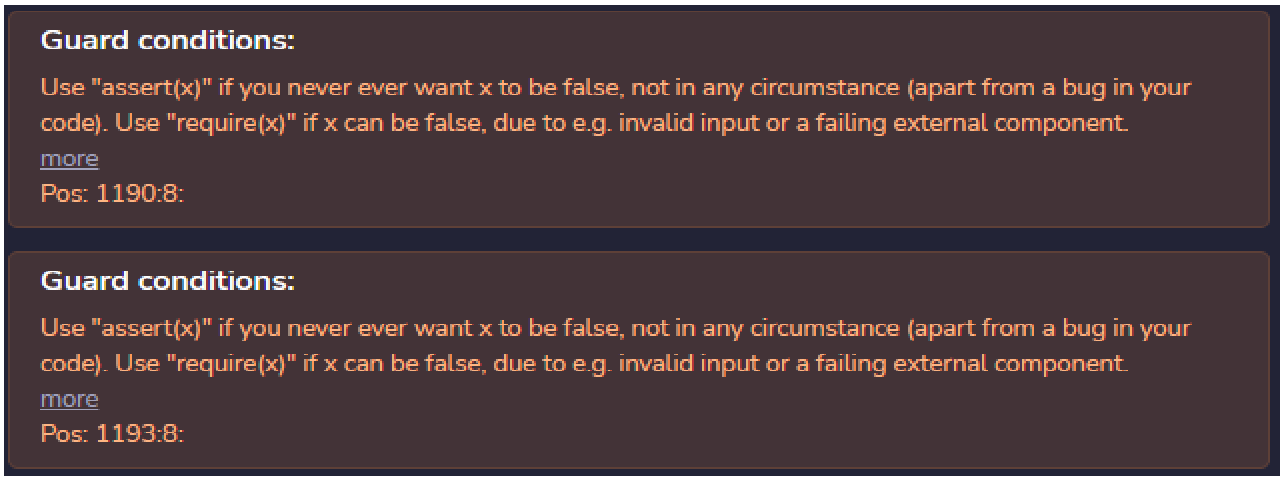
Potential violation of Checks-Effects-Interaction pattern in GBPST.transferFrom(address,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1245:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in GBPST._redeem(address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1272:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in GBPST.mint(address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1291:4:

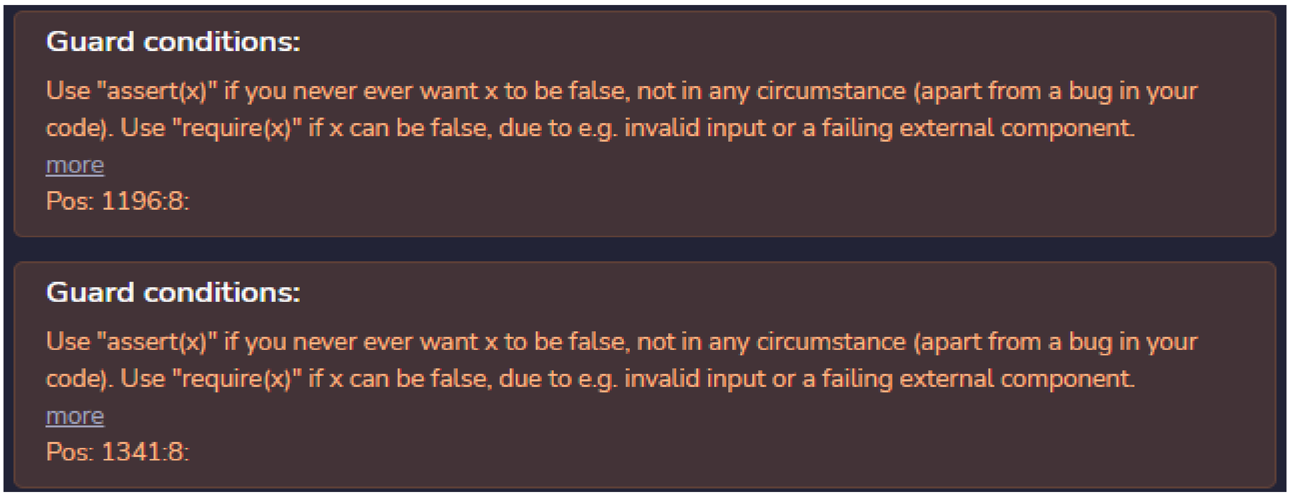


Guard conditions:

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: 1190:8:

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: 1193:8:



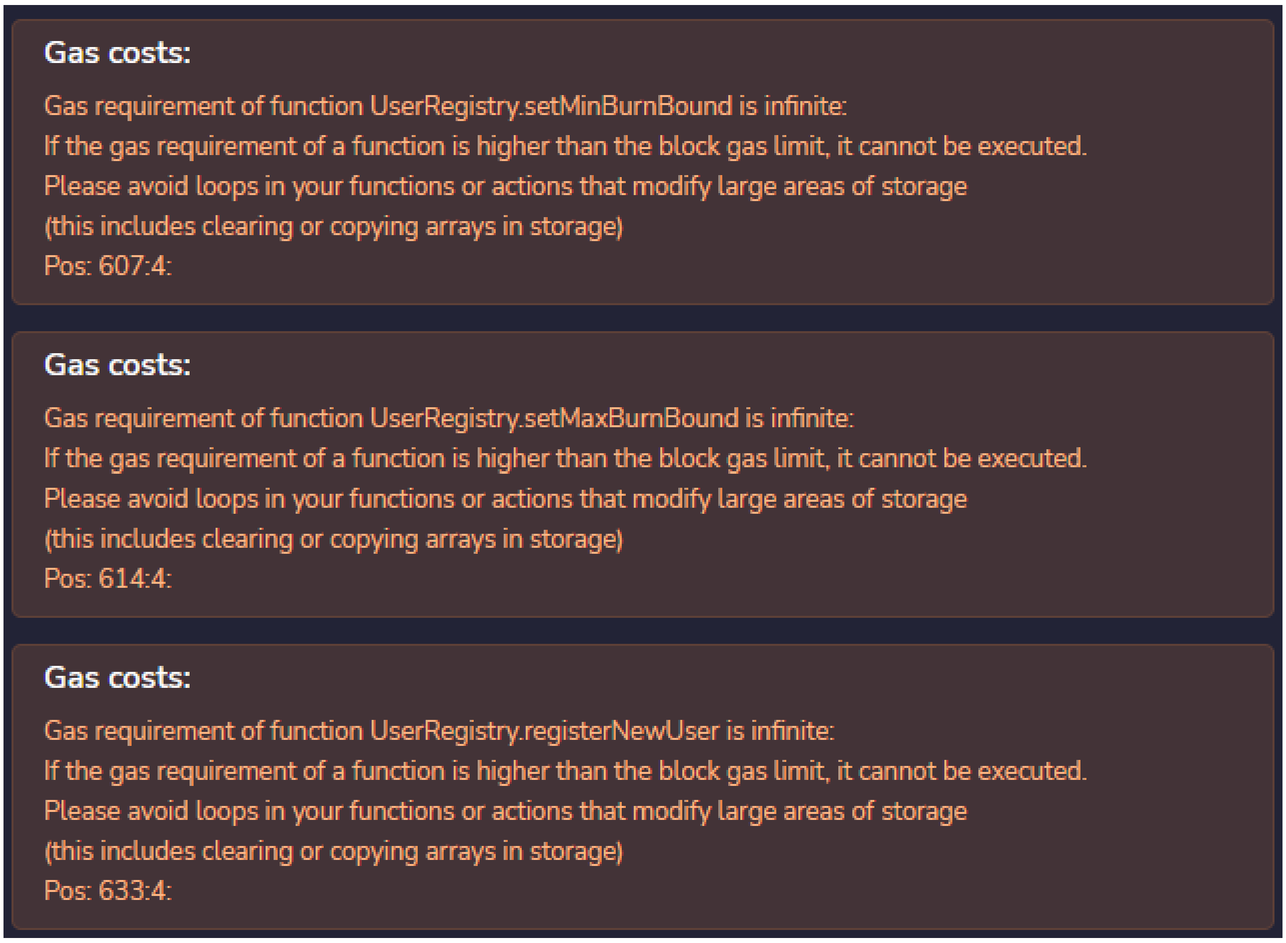
Guard conditions:

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: 1196:8:

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: 1341:8:

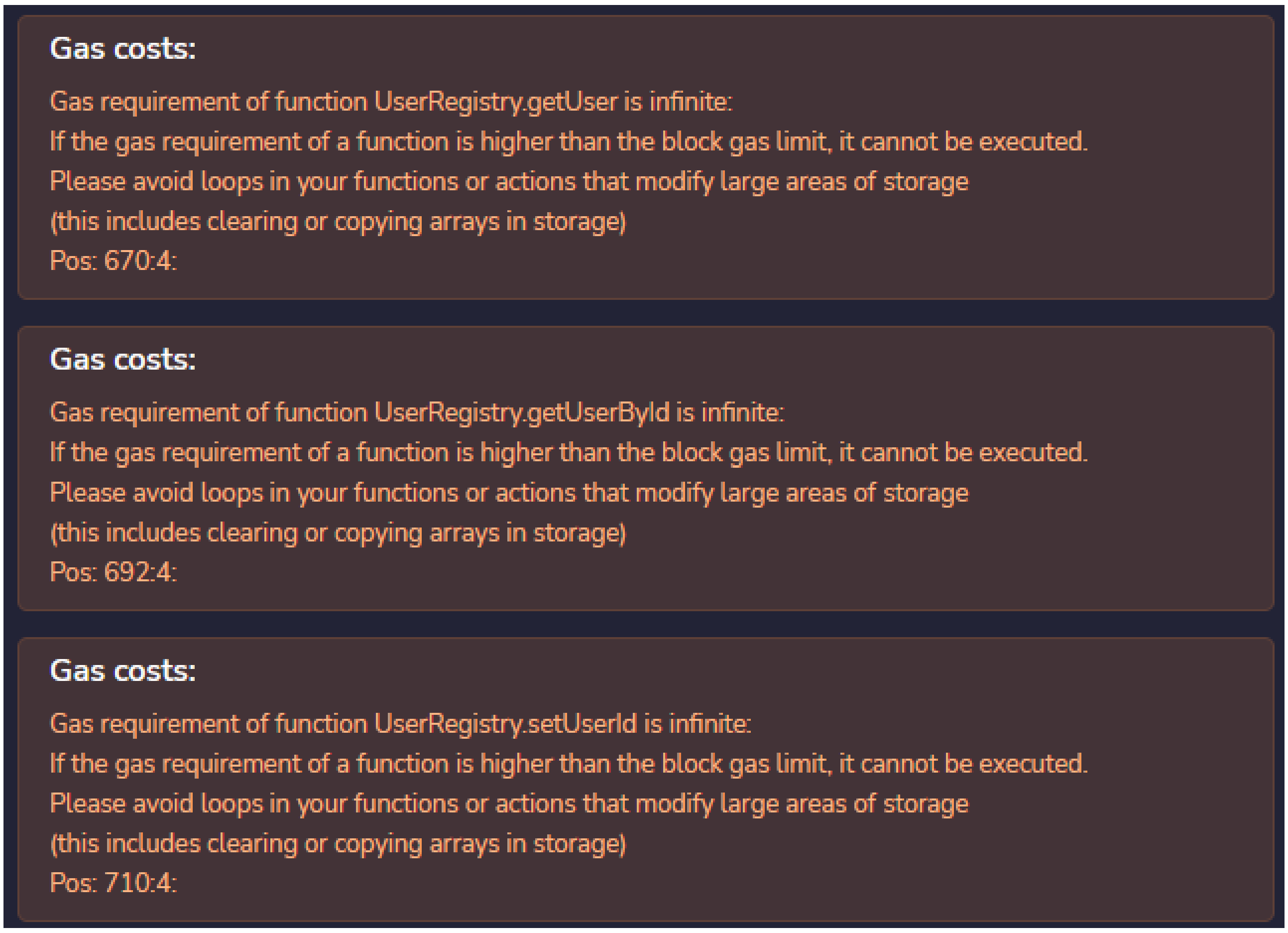
UserRegistry.sol



```
Gas costs:
Gas requirement of function UserRegistry.setMinBurnBound 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: 607:4:

Gas costs:
Gas requirement of function UserRegistry.setMaxBurnBound 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: 614:4:

Gas costs:
Gas requirement of function UserRegistry.registerNewUser 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: 633:4:
```

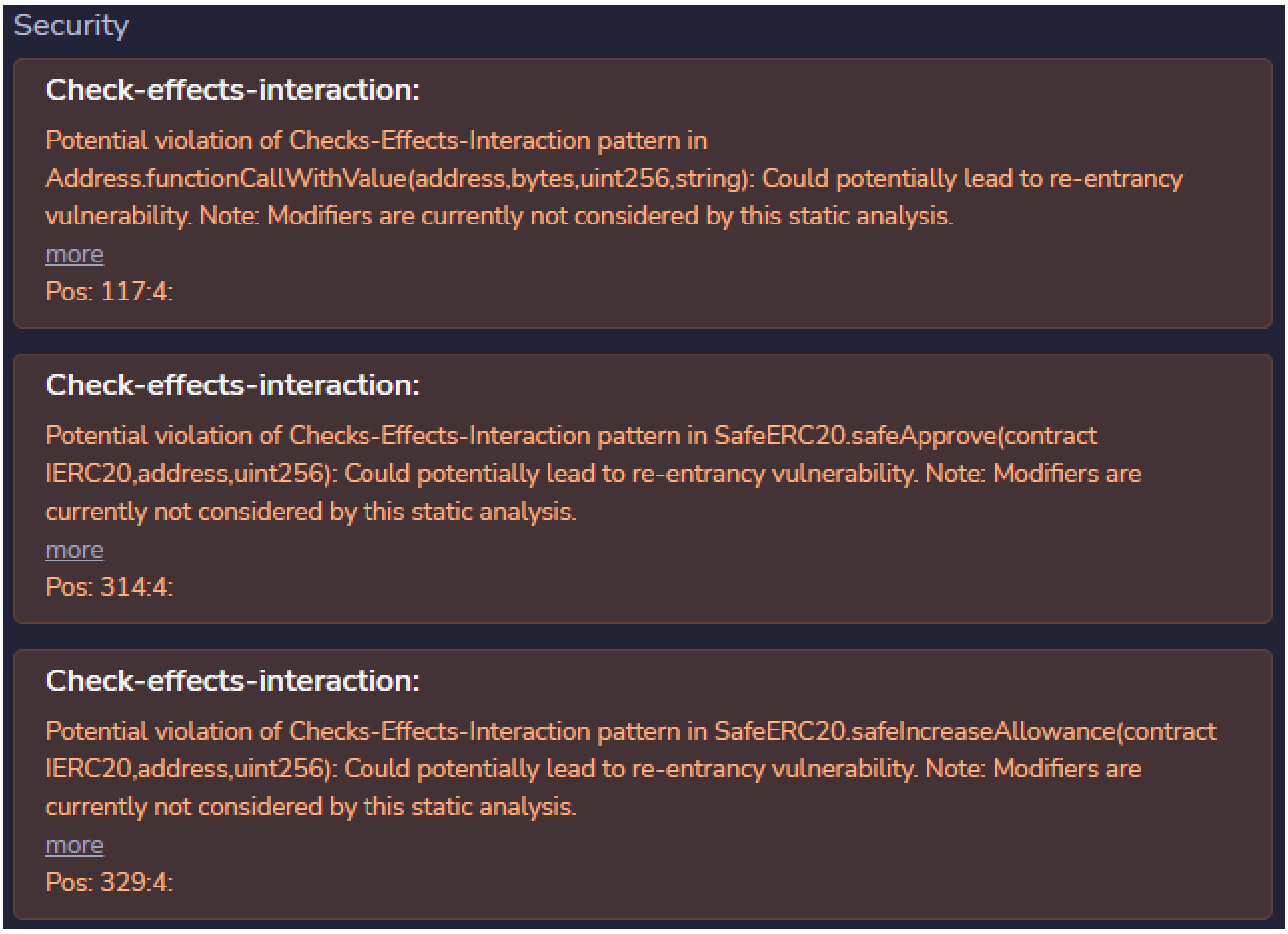


```
Gas costs:
Gas requirement of function UserRegistry.getUser is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 670:4:

Gas costs:
Gas requirement of function UserRegistry.getUserById 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: 692:4:

Gas costs:
Gas requirement of function UserRegistry.setUserId 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: 710:4:
```

Climer.sol



Security

Check-effects-interaction:

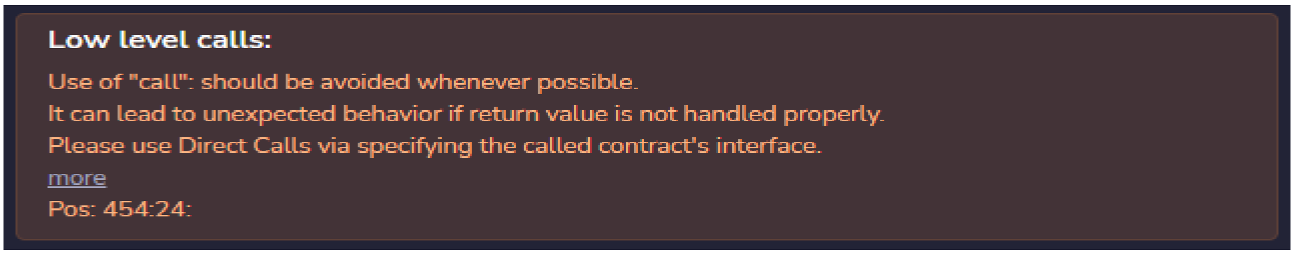
Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 117:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeApprove(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 314:4:

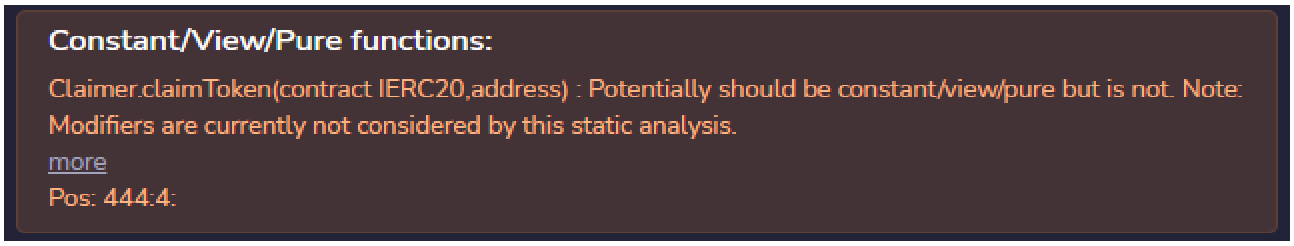
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeIncreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 329:4:



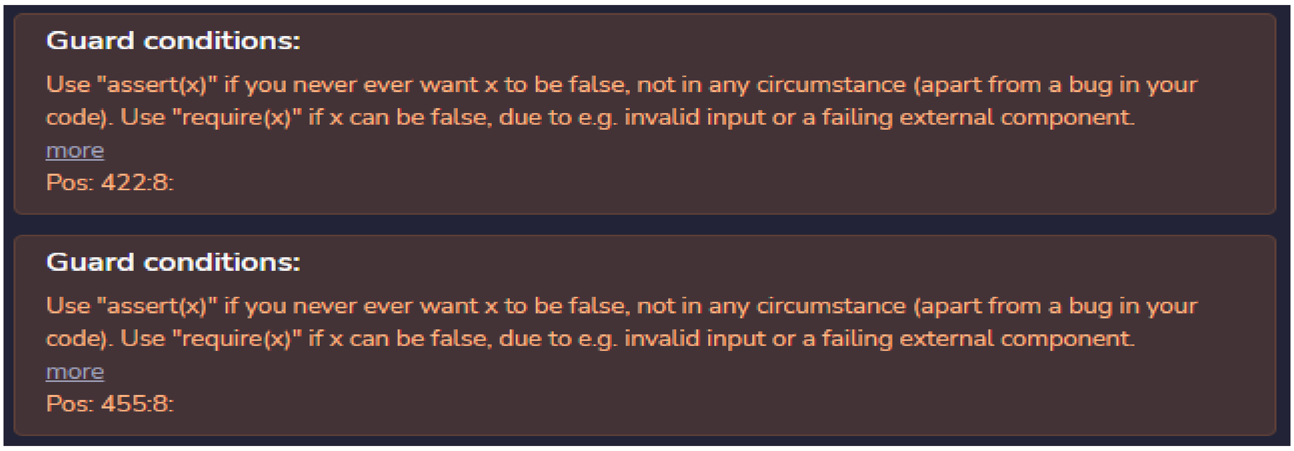
Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 454:24:



Constant/View/Pure functions:

Claimer.claimToken(contract IERC20,address) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 444:4:



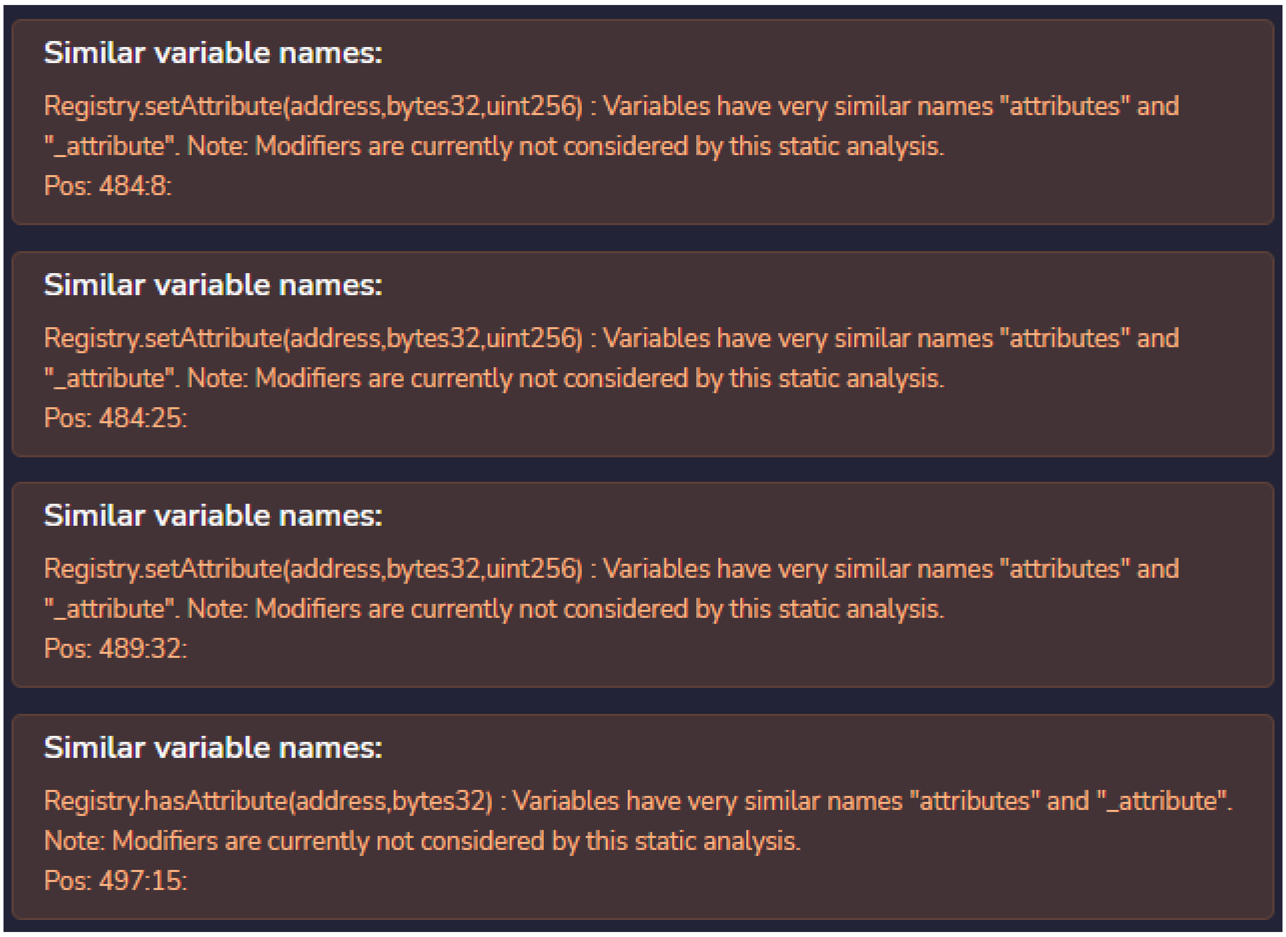
Guard conditions:

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: 422:8:

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: 455:8:

Registry.sol



Similar variable names:

Registry.setAttribute(address,bytes32,uint256) : Variables have very similar names "attributes" and "_attribute". Note: Modifiers are currently not considered by this static analysis.
Pos: 484:8:

Similar variable names:

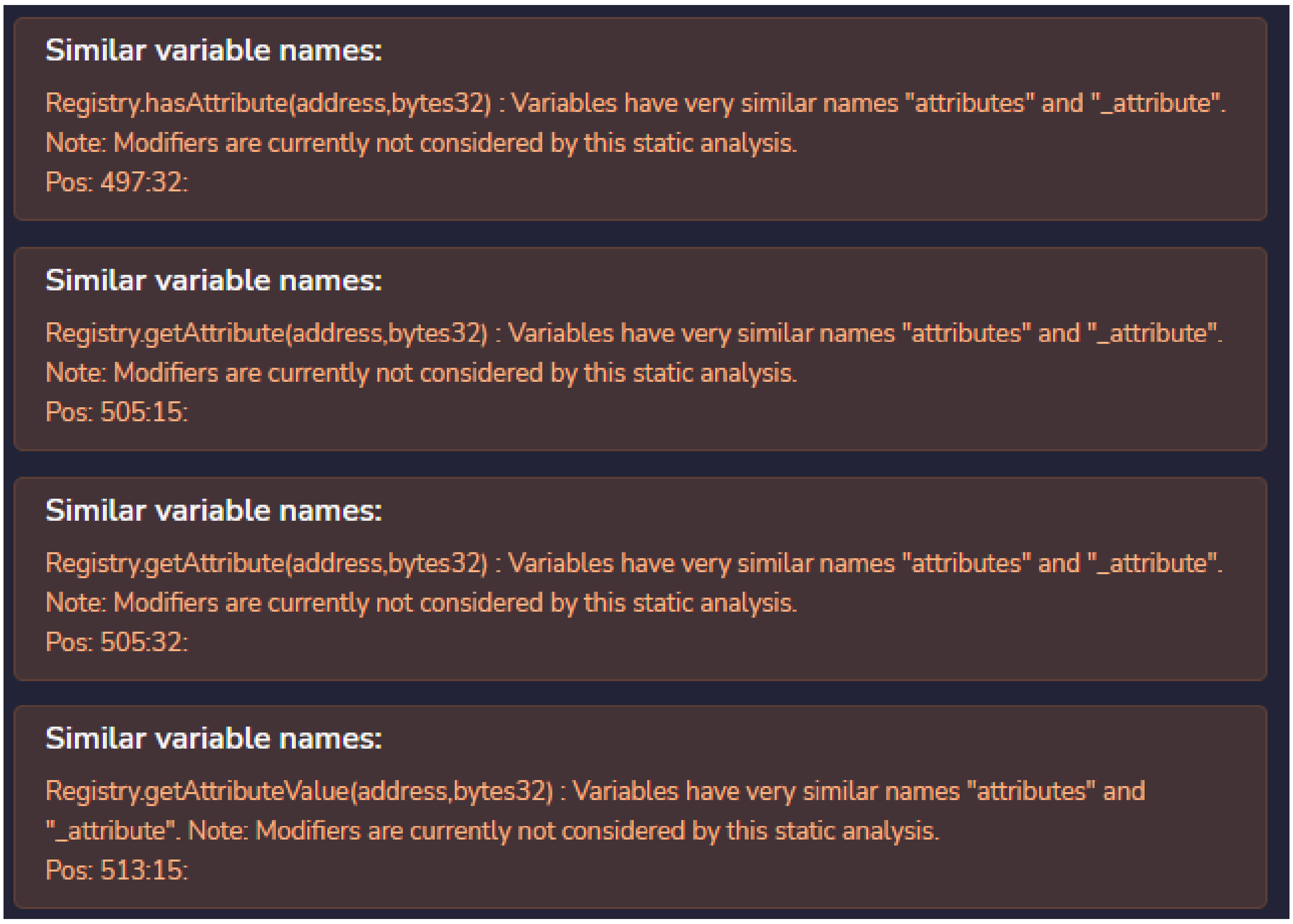
Registry.setAttribute(address,bytes32,uint256) : Variables have very similar names "attributes" and "_attribute". Note: Modifiers are currently not considered by this static analysis.
Pos: 484:25:

Similar variable names:

Registry.setAttribute(address,bytes32,uint256) : Variables have very similar names "attributes" and "_attribute". Note: Modifiers are currently not considered by this static analysis.
Pos: 489:32:

Similar variable names:

Registry.hasAttribute(address,bytes32) : Variables have very similar names "attributes" and "_attribute". Note: Modifiers are currently not considered by this static analysis.
Pos: 497:15:



Similar variable names:

Registry.hasAttribute(address,bytes32) : Variables have very similar names "attributes" and "_attribute". Note: Modifiers are currently not considered by this static analysis.
Pos: 497:32:

Similar variable names:

Registry.getAttribute(address,bytes32) : Variables have very similar names "attributes" and "_attribute". Note: Modifiers are currently not considered by this static analysis.
Pos: 505:15:

Similar variable names:

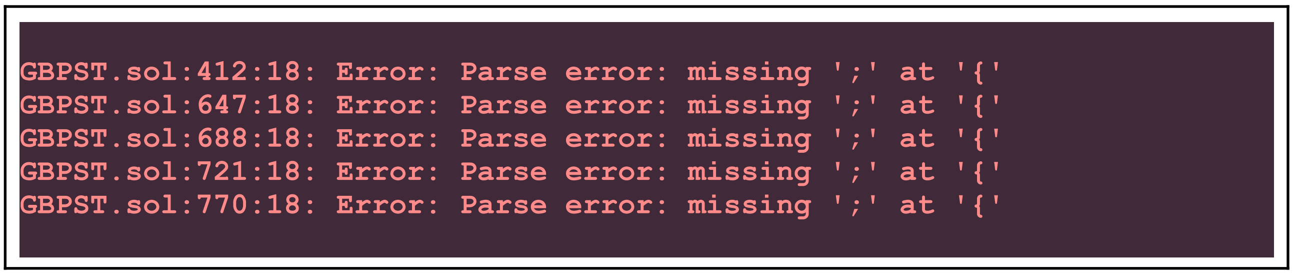
Registry.getAttribute(address,bytes32) : Variables have very similar names "attributes" and "_attribute". Note: Modifiers are currently not considered by this static analysis.
Pos: 505:32:

Similar variable names:

Registry.getAttributeValue(address,bytes32) : Variables have very similar names "attributes" and "_attribute". Note: Modifiers are currently not considered by this static analysis.
Pos: 513:15:

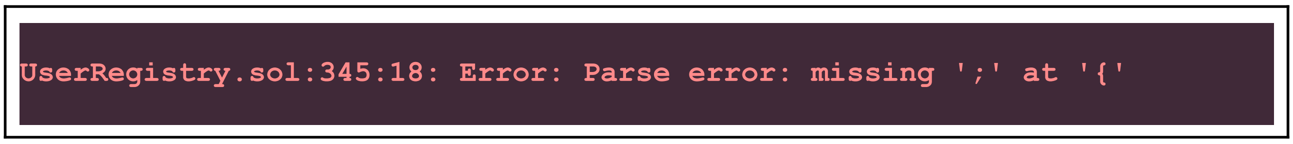
Solhint Linter

GBPST.sol



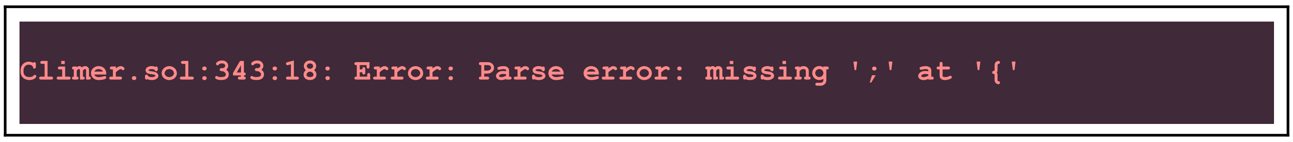
```
GBPST.sol:412:18: Error: Parse error: missing ';' at '{'
GBPST.sol:647:18: Error: Parse error: missing ';' at '{'
GBPST.sol:688:18: Error: Parse error: missing ';' at '{'
GBPST.sol:721:18: Error: Parse error: missing ';' at '{'
GBPST.sol:770:18: Error: Parse error: missing ';' at '{'
```

UserRegistry.sol



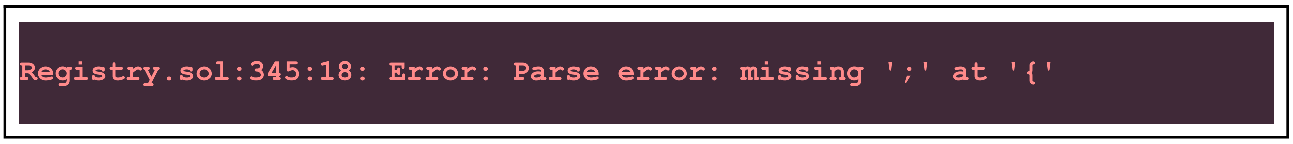
```
UserRegistry.sol:345:18: Error: Parse error: missing ';' at '{'
```

Climer.sol



```
Climer.sol:343:18: Error: Parse error: missing ';' at '{'
```

Registry.sol



```
Registry.sol:345:18: Error: Parse error: missing ';' at '{'
```

Software analysis result:

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

EtherAuthority