EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Estrella Tera
Platform:	Binance Smart Chain
Language:	Solidity
Date:	April 29th, 2023

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 Estrella Tera team to perform the Security audit of the Estrella Tera Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on April 29th, 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

- In the Estrella Tera (ETA) system, users can withdraw their ETA tokens (BEP20) from the dApp and receive 10 X USD of ETA tokens (outstanding capital). After withdrawal, ETA tokens will not participate in the token split.
- Estrella Tera Token is a dapp that has buy, register, WithdrawUSDToken functionalities.
- Estrella Tera (ETA) is a BEP20 standard token contract on the Binance Smart Chain blockchain.

Audit scope

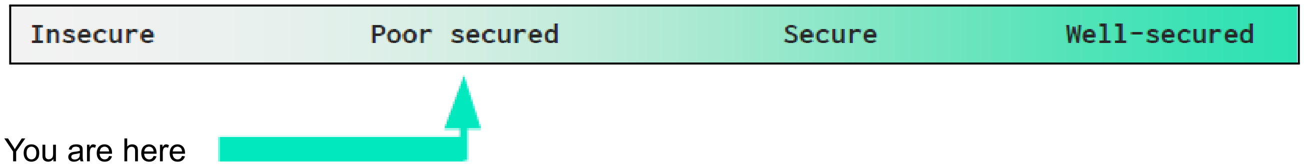
Name	**Code Review and Security Analysis Report for Estrella Tera Token Smart Contract**
Platform	**BSC / Solidity**
File	EstrellaTera.sol
File MD5 Hash	E47C5BB0660F0D18DFC1F026B3006825
Online code link	0x0211616971151acc438f9671e8abd8ad89022c31
Audit Date	April 29th, 2023
Revised Audit Date	May 5th, 2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: ● Decimals: 18 ● Mulplier: 4 ● End Cycles: 100 ● Refer Depth: 30 ● Fixed Price: 200 Quadrillion ● Token Price: 200 Quadrillion ● Owner Percentage: 81 ● Minimum Deposit: 1 ● Maximum Deposit: 10 Thousand ● cycleSupply: 10 Thousand ● roundSupply: 1 Million ● base Divider: 100 ● Token Price Increment: 2 Quadrillion ● Total seed funding 2,000,000 ETA	YES, This is valid.
Owner has control over following functions: ● Update withdrawal percentage by the owner. ● Withdraw USD token by the owner. ● Current owner can transfer ownership of the contract to a new account. ● Deleting ownership will leave the contract without an owner, removing any owner-only functionality.	YES, This is valid.

Audit Summary

According to the standard audit assessment, Customer`s solidity based smart contracts are **"Poor Secured"**. This token contract does contain owner control, which does not make it 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, 2 high, 1 medium and 0 low and some very low level issues.

We confirm that 1 high severity issue, 1 medium severity issue and 4 informational severity issues are solved in the revised smart contract.

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	Moderated
	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: Failed

Code Quality

This audit scope has 1 smart contract. Smart contract contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the Estrella Tera 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 Estrella Tera Token.

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

Code parts are not well commented on in the smart contracts. Ethereum's NatSpec commenting style is used, which is a good thing.

Documentation

We were given a Estrella Tera Token smart contract code in the form of a BSCScan web link 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.

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.

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

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	register	write	Passed	No Issue
9	_updateTeamNum	write	Passed	No Issue
10	updateReferral	write	Passed	No Issue
11	buy	write	Passed	Fixed
12	countBuyers	read	Passed	No Issue
13	trasnferAmount	write	Passed	No Issue
14	Commission	write	Passed	No Issue
15	buyerReferralCommission	write	Passed	Fixed
16	sellerReferralCommission	write	Passed	No Issue
17	checkUSDACE	read	Passed	No Issue
18	totalEarned	read	Passed	No Issue
19	TotalClaimed	read	Passed	No Issue
20	claimedCommission	write	Passed	No Issue
21	maxWithdrwa	read	Passed	Fixed
22	checkRemainingToken	read	Passed	No Issue
23	checktoken	read	Passed	No Issue
24	checkbalance	read	Passed	No Issue
25	checkPrice	read	Passed	No Issue
26	maxToken	read	Passed	No Issue
27	getPriceAfterTwoRunds	read	Passed	No Issue
28	getPrice1	read	Passed	No Issue
29	CheckCycle	read	Passed	No Issue
30	checkRound	read	Passed	No Issue
31	checkSellerOrder	read	Passed	No Issue
32	getPrice	read	Passed	No Issue
33	getETAWithdraw	read	Passed	No Issue
34	multiplerofETA	write	The getETAWithdraw function only calculating for USDTEarned	Refer Audit Findings
35	userWithdrawETAToken	write	Withdrawing ETA before updating the balance	Refer Audit Findings
36	updatebalance	write	Passed	No Issue

37	changeWithdrawPercentage	write	access only Owner	No Issue
38	WithdrawUSDToken	write	access only Owner	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

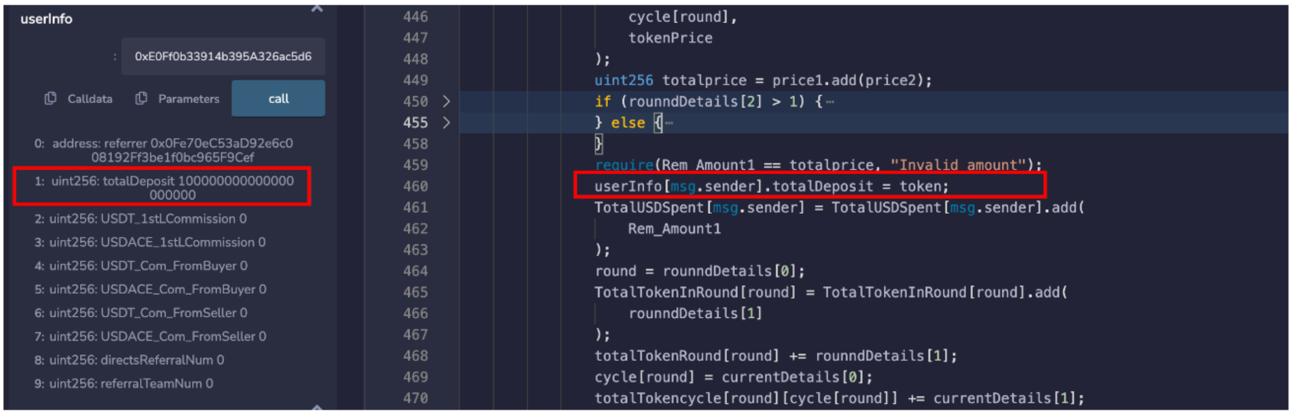
Critical Severity

No Critical severity vulnerabilities were found in the revised code.

High Severity

(1) Total deposit for the user is overriding the previous deposit:

Function: buy()



```
: 0xE0Ff0b33914b395A326ac5d6

0: address: referrer 0x0Fe70eC53aD92e6c0
      08192Ff3be1f0bc965F9Cef
1: uint256: totalDeposit 100000000000000
      000000
2: uint256: USDT_1stLCommission 0
3: uint256: USDACE_1stLCommission 0
4: uint256: USDT_Com_FromBuyer 0
5: uint256: USDACE_Com_FromBuyer 0
6: uint256: USDT_Com_FromSeller 0
7: uint256: USDACE_Com_FromSeller 0
8: uint256: directsReferralNum 0
9: uint256: referralTeamNum 0
```

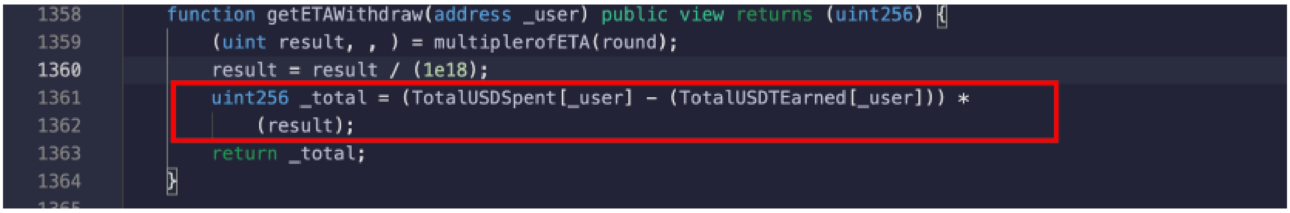
```
        cycle[round],
        tokenPrice
    );
    uint256 totalprice = price1.add(price2);
    if (rounndDetails[2] > 1) {⋯
    } else {⋯
    }
    require(Rem_Amount1 == totalprice, "Invalid amount");
    userInfo[msg.sender].totalDeposit = token;
    TotalUSDSpent[msg.sender] = TotalUSDSpent[msg.sender].add(
        Rem_Amount1
    );
    round = rounndDetails[0];
    TotalTokenInRound[round] = TotalTokenInRound[round].add(
        rounndDetails[1]
    );
    totalTokenRound[round] += rounndDetails[1];
    cycle[round] = currentDetails[0];
    totalTokencycle[round][cycle[round]] += currentDetails[1];
```

- Buy the ETA by depositing 100 REGETA tokens at this point the totaldeposited value is 100.
- Again buy the ETA by depositing 100 REGETA tokens at this point the totaldeposited value should be 200 REGETA tokens but it's overridden with the new deposited value which is 100.

Resolution: The solution is userInfo[msg.sender].totalDeposit += token;

Status: This is fixed in the revised code.

(2) The getETAWithdraw function only calculating for USDTEarned:



```
function getETAWithdraw(address _user) public view returns (uint256) {
    (uint result, , ) = multiplerofETA(round);
    result = result / (1e18);
    uint256 _total = (TotalUSDSpent[_user] - (TotalUSDTEarned[_user])) *
        (result);
    return _total;
}
```

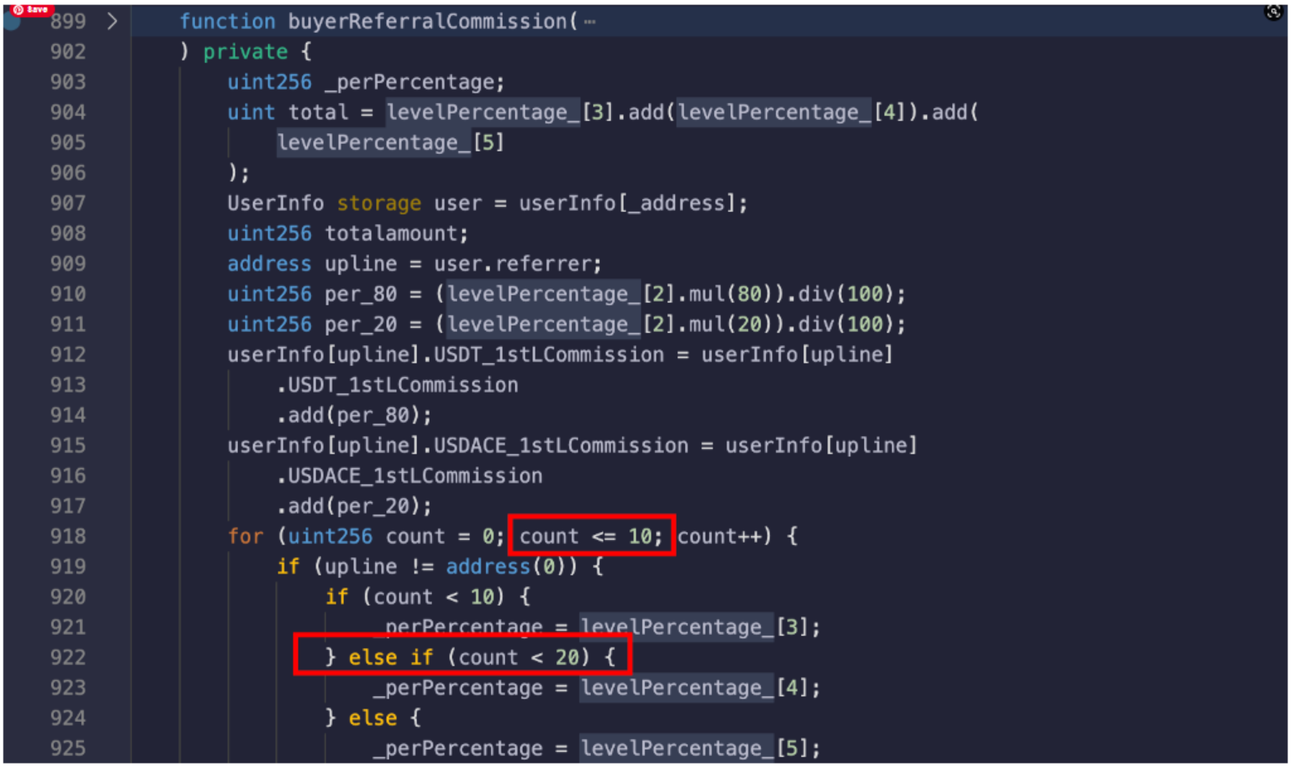
The getETAWithdraw function only calculates USDTEarned.

Resolution: We suggest checking the logic and calculate USDSspent - USSDT+USDACE or as per logic.

Status: Open

Medium

(1) Else If condition is redundant:



```
function buyerReferralCommission(⋯
) private {
    uint256 _perPercentage;
    uint total = levelPercentage_[3].add(levelPercentage_[4]).add(
        levelPercentage_[5]
    );
    UserInfo storage user = userInfo[_address];
    uint256 totalamount;
    address upline = user.referrer;
    uint256 per_80 = (levelPercentage_[2].mul(80)).div(100);
    uint256 per_20 = (levelPercentage_[2].mul(20)).div(100);
    userInfo[upline].USDT_1stLCommission = userInfo[upline]
        .USDT_1stLCommission
        .add(per_80);
    userInfo[upline].USDACE_1stLCommission = userInfo[upline]
        .USDACE_1stLCommission
        .add(per_20);
    for (uint256 count = 0; count <= 10; count++) {
        if (upline != address(0)) {
            if (count < 10) {
                _perPercentage = levelPercentage_[3];
            } else if (count < 20) {
                _perPercentage = levelPercentage_[4];
            } else {
                _perPercentage = levelPercentage_[5];
```

The for loop condition is limited to <= 10, but else if the condition is checking for < 20, which never met.

Resolution: We suggest checking the logic and correct it.

Status: This is fixed in the revised code.

Low

No Low severity vulnerabilities were found in the revised code.

Very Low / Informational / Best practices:

(1) SafeMath Library:

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.

Status: This is fixed in the revised code.

(2) Unused variables:

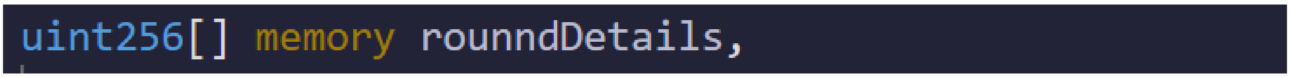
Below listed variables are defined but not used:

- uint256 private totalPrice;
- uint256 private totlalToken;
- mapping(uint256 => mapping(uint256 => uint256)) public totalTokencyclePrice;

Resolution: We suggest removing unused variables.

Status: This is fixed in the revised code.

(3) Local array Wrong spelled:



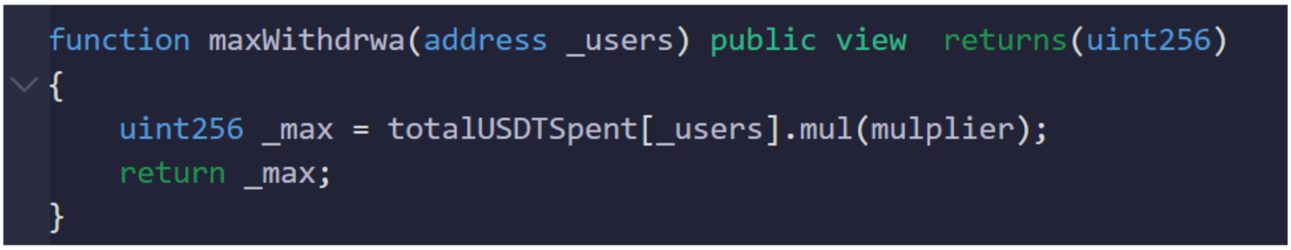
```
uint256[] memory rounndDetails,
```

Local array Wrong spelled as “**rounndDetails**.”

Resolution: For clean code, suggest to correct the spelling to “**roundDetails**”.

Status: This is fixed in the revised code.

(4) Method wrong spelled:



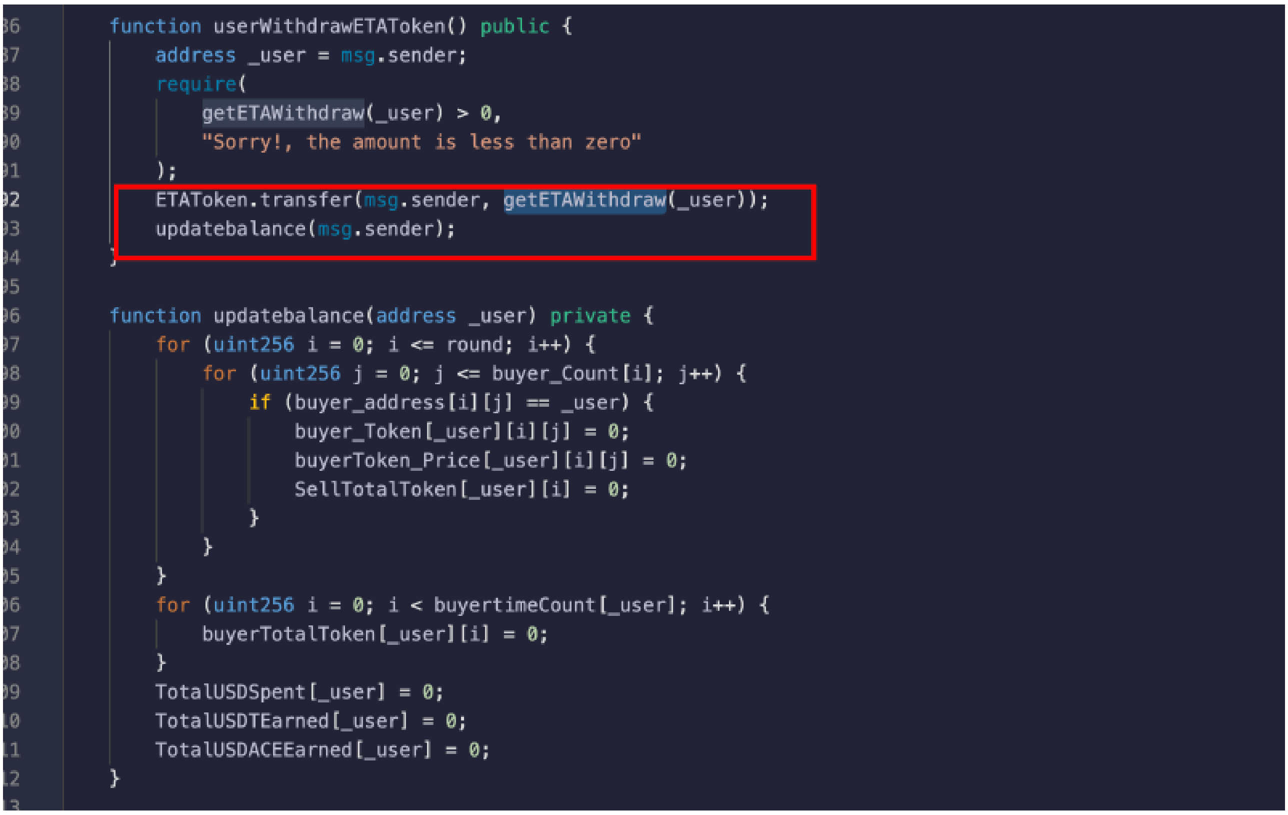
```
function maxWithdrwa(address _users) public view  returns(uint256)
{
    uint256 _max = totalUSDTSpent[_users].mul(mulplier);
    return _max;
}
```

Method wrong spelled.

Resolution: For clean code, suggest to correct the spelling to maxWithdrwa.

Status: This is fixed in the revised code.

(5) Withdrawing ETA before updating the balance:



```
    function userWithdrawETAToken() public {
        address _user = msg.sender;
        require(
            getETAWithdraw(_user) > 0,
            "Sorry!, the amount is less than zero"
        );
        ETAToken.transfer(msg.sender, getETAWithdraw(_user));
        updatebalance(msg.sender);
    }

    function updatebalance(address _user) private {
        for (uint256 i = 0; i <= round; i++) {
            for (uint256 j = 0; j <= buyer_Count[i]; j++) {
                if (buyer_address[i][j] == _user) {
                    buyer_Token[_user][i][j] = 0;
                    buyerToken_Price[_user][i][j] = 0;
                    SellTotalToken[_user][i] = 0;
                }
            }
        }
        for (uint256 i = 0; i < buyertimeCount[_user]; i++) {
            buyerTotalToken[_user][i] = 0;
        }
        TotalUSDSpent[_user] = 0;
        TotalUSDTEarned[_user] = 0;
        TotalUSDACEEarned[_user] = 0;
    }
```

Withdrawing ETA before updating the balance.

Resolution: Update the balance first then withdraw in order to avoid some attacks.

Status: Open

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:

EstrellaTera.sol

- buy: Toke bought by the owner.
- changeWithdrawPercentage: Update withdrawal percentage by the owner.
- WithdrawUSDToken: Withdraw USD token by the owner.

Ownable.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.

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 bscscan.com link, and we have used all possible tests based on the given objects as files. We confirm that 1 high-severity issue, 1 medium-severity issue, and 4 informational-severity issues are solved in the revised code. But a high issue is still there. So, **it's good to go for the mainnet deployment after correcting the high issue**.

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 smart contract, based on standard audit procedure scope, is **"Poor 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).

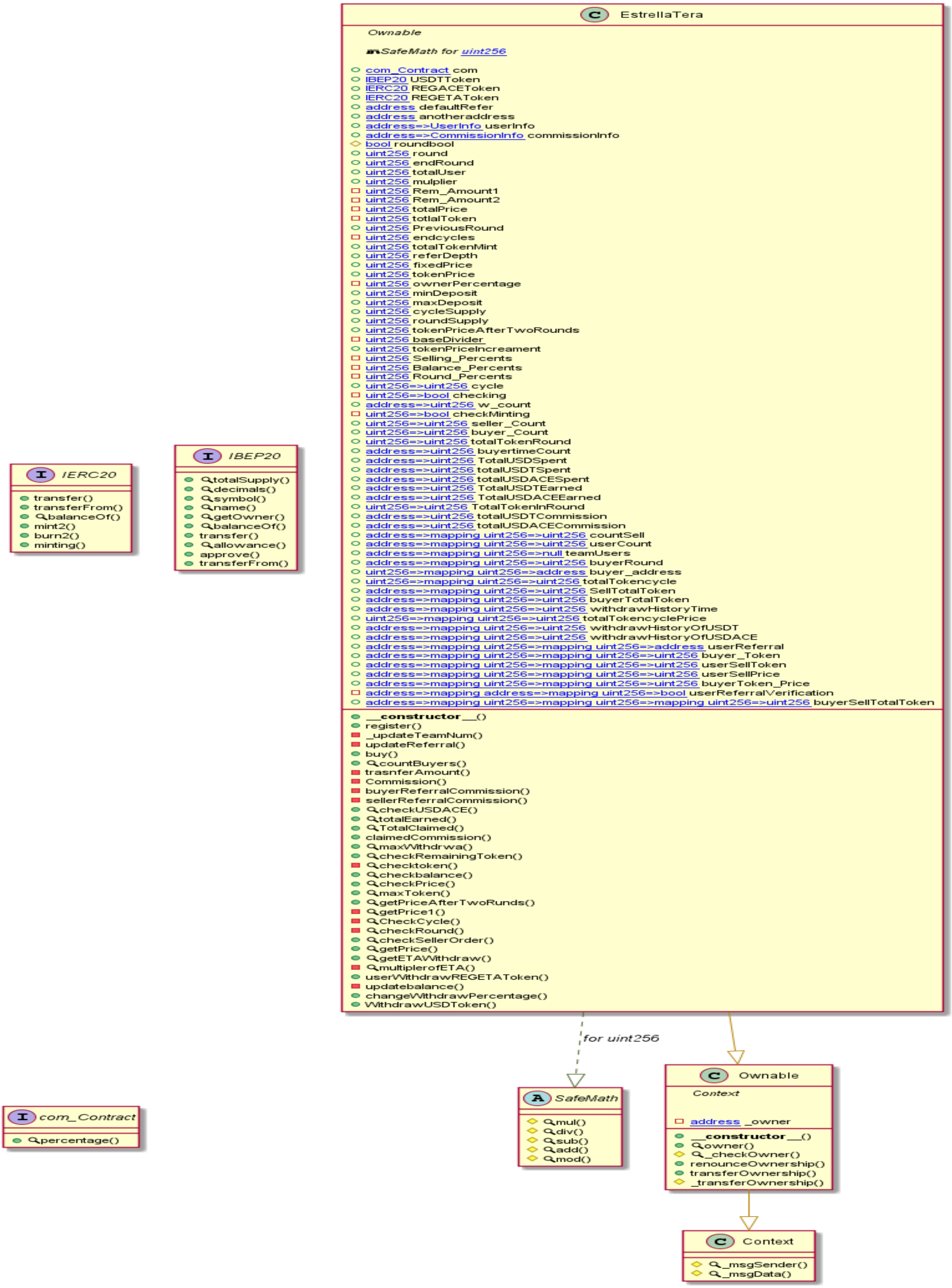
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we 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.

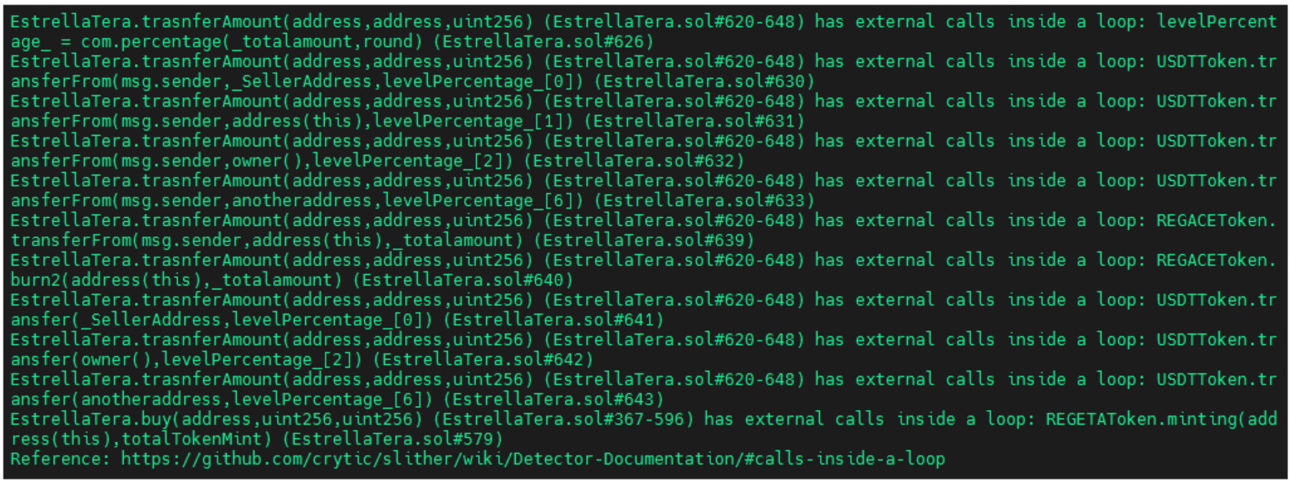
Appendix

Code Flow Diagram - Estrella Tera Token

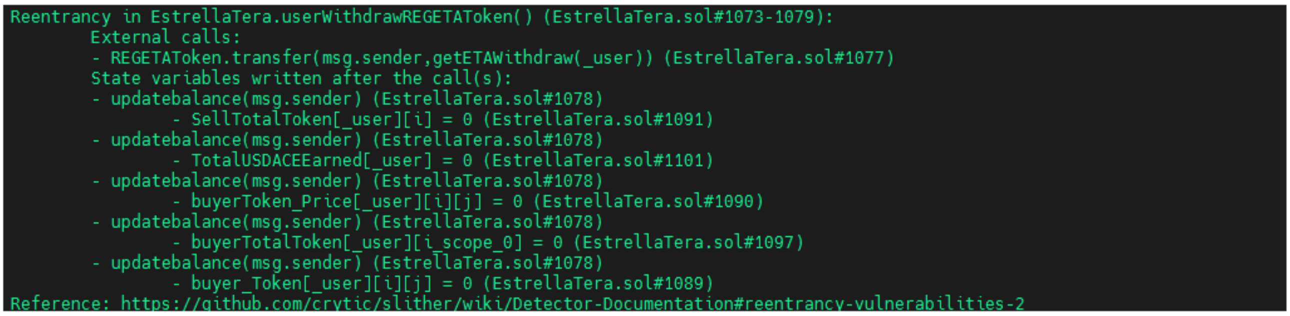

C EstrellaTera
Ownable
SafeMath for uint256
com_Contract com
IBEP20 USDTToken
IERC20 REGACEToken
IERC20 REGETAToken
address defaultRefer
address anotheraddress
address=>UserInfo userInfo
address=>CommissionInfo commissionInfo
bool roundbool
uint256 round
uint256 endRound
uint256 totalUser
uint256 mulplier
uint256 Rem_Amount1
uint256 Rem_Amount2
uint256 totalPrice
uint256 totlalToken
uint256 PreviousRound
uint256 endcycles
uint256 totalTokenMint
uint256 referDepth
uint256 fixedPrice
uint256 tokenPrice
uint256 ownerPercentage
uint256 minDeposit
uint256 maxDeposit
uint256 cycleSupply
uint256 roundSupply
uint256 tokenPriceAfterTwoRounds
uint256 baseDivider
uint256 tokenPriceIncreament
uint256 Selling_Percents
uint256 Balance_Percents
uint256 Round_Percents
uint256=>uint256 cycle
uint256=>bool checking
address=>uint256 w_count
uint256=>bool checkMinting
uint256=>uint256 seller_Count
uint256=>uint256 buyer_Count
uint256=>uint256 totalTokenRound
address=>uint256 buyertimeCount
address=>uint256 TotalUSDSpent
address=>uint256 totalUSDTSpent
address=>uint256 totalUSDACESpent
address=>uint256 TotalUSDTEarned
address=>uint256 TotalUSDACEEarned
uint256=>uint256 TotalTokenInRound
address=>uint256 totalUSDTCommission
address=>uint256 totalUSDACECommission
address=>mapping uint256=>uint256 countSell
address=>mapping uint256=>uint256 userCount
address=>mapping uint256=>null teamUsers
address=>mapping uint256=>uint256 buyerRound
uint256=>mapping uint256=>address buyer_address
uint256=>mapping uint256=>uint256 totalTokencycle
address=>mapping uint256=>uint256 SellTotalToken
address=>mapping uint256=>uint256 buyerTotalToken
address=>mapping uint256=>uint256 withdrawHistoryTime
uint256=>mapping uint256=>uint256 totalTokencyclePrice
address=>mapping uint256=>uint256 withdrawHistoryOfUSDT
address=>mapping uint256=>uint256 withdrawHistoryOfUSDACE
address=>mapping uint256=>mapping uint256=>address userReferral
address=>mapping uint256=>mapping uint256=>uint256 buyer_Token
address=>mapping uint256=>mapping uint256=>uint256 userSellToken
address=>mapping uint256=>mapping uint256=>uint256 userSellPrice
address=>mapping uint256=>mapping uint256=>uint256 buyerToken_Price
address=>mapping address=>mapping uint256=>bool userReferralVerification
address=>mapping uint256=>mapping uint256=>mapping uint256=>uint256 buyerSellTotalToken
__constructor__()
register()
_updateTeamNum()
updateReferral()
buy()
countBuyers()
trasnferAmount()
Commission()
buyerReferralCommission()
sellerReferralCommission()
checkUSDACE()
totalEarned()
TotalClaimed()
claimedCommission()
maxWithdrwa()
checkRemainingToken()
checktoken()
checkbalance()
checkPrice()
maxToken()
getPriceAfterTwoRunds()
getPrice1()
CheckCycle()
checkRound()
checkSellerOrder()
getPrice()
getETAWithdraw()
multiplerofETA()
userWithdrawREGETAToken()
updatebalance()
changeWithdrawPercentage()
WithdrawUSDToken()
I IERC20
transfer()
transferFrom()
balanceOf()
mint2()
burn2()
minting()
I IBEP20
totalSupply()
decimals()
symbol()
name()
getOwner()
balanceOf()
transfer()
allowance()
approve()
transferFrom()
for uint256
A SafeMath
mul()
div()
sub()
add()
mod()
C Ownable
Context
address _owner
__constructor__()
owner()
_checkOwner()
renounceOwnership()
transferOwnership()
_transferOwnership()
I com_Contract
percentage()
C Context
_msgSender()
_msgData()

Slither Results Log

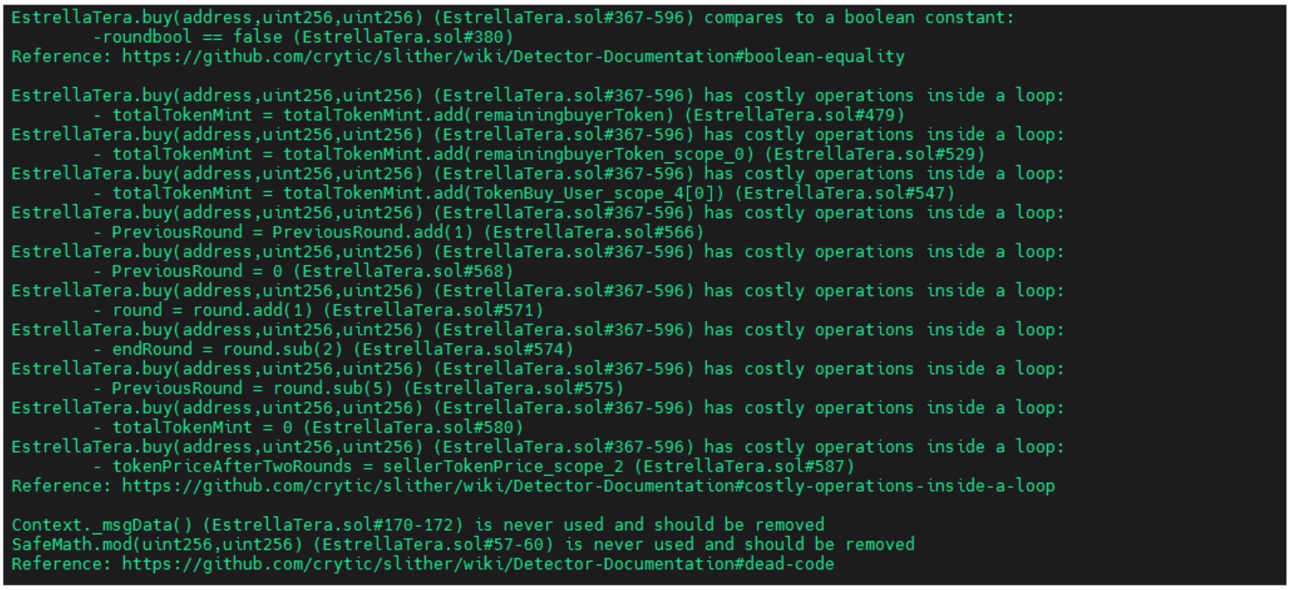
Slither Log >> EstrellaTera.sol



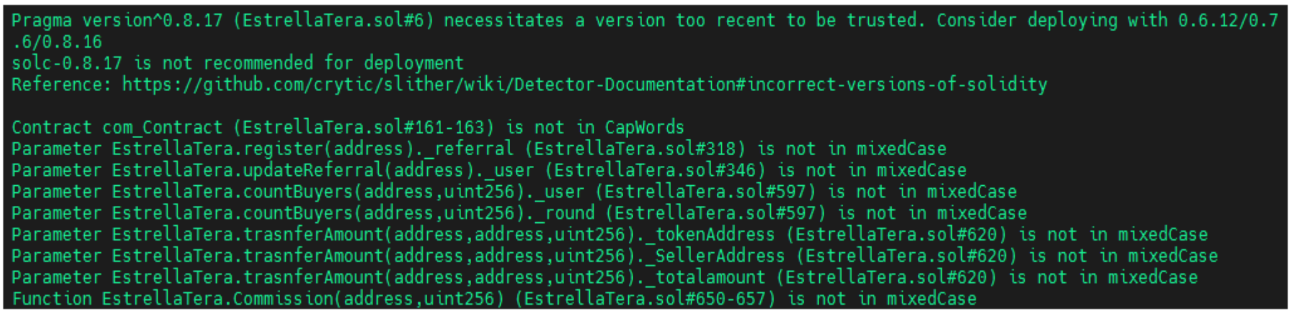
```
EstrellaTera.trasnferAmount(address,address,uint256) (EstrellaTera.sol#620-648) has external calls inside a loop: levelPercent
age_ = com.percentage(_totalamount,round) (EstrellaTera.sol#626)
EstrellaTera.trasnferAmount(address,address,uint256) (EstrellaTera.sol#620-648) has external calls inside a loop: USDTToken.tr
ansferFrom(msg.sender,_SellerAddress,levelPercentage_[0]) (EstrellaTera.sol#630)
EstrellaTera.trasnferAmount(address,address,uint256) (EstrellaTera.sol#620-648) has external calls inside a loop: USDTToken.tr
ansferFrom(msg.sender,address(this),levelPercentage_[1]) (EstrellaTera.sol#631)
EstrellaTera.trasnferAmount(address,address,uint256) (EstrellaTera.sol#620-648) has external calls inside a loop: USDTToken.tr
ansferFrom(msg.sender,owner(),levelPercentage_[2]) (EstrellaTera.sol#632)
EstrellaTera.trasnferAmount(address,address,uint256) (EstrellaTera.sol#620-648) has external calls inside a loop: USDTToken.tr
ansferFrom(msg.sender,anotheraddress,levelPercentage_[6]) (EstrellaTera.sol#633)
EstrellaTera.trasnferAmount(address,address,uint256) (EstrellaTera.sol#620-648) has external calls inside a loop: REGACEToken.
transferFrom(msg.sender,address(this),_totalamount) (EstrellaTera.sol#639)
EstrellaTera.trasnferAmount(address,address,uint256) (EstrellaTera.sol#620-648) has external calls inside a loop: REGACEToken.
burn2(address(this),_totalamount) (EstrellaTera.sol#640)
EstrellaTera.trasnferAmount(address,address,uint256) (EstrellaTera.sol#620-648) has external calls inside a loop: USDTToken.tr
ansfer(_SellerAddress,levelPercentage_[0]) (EstrellaTera.sol#641)
EstrellaTera.trasnferAmount(address,address,uint256) (EstrellaTera.sol#620-648) has external calls inside a loop: USDTToken.tr
ansfer(owner(),levelPercentage_[2]) (EstrellaTera.sol#642)
EstrellaTera.trasnferAmount(address,address,uint256) (EstrellaTera.sol#620-648) has external calls inside a loop: USDTToken.tr
ansfer(anotheraddress,levelPercentage_[6]) (EstrellaTera.sol#643)
EstrellaTera.buy(address,uint256,uint256) (EstrellaTera.sol#367-596) has external calls inside a loop: REGETAToken.minting(add
ress(this),totalTokenMint) (EstrellaTera.sol#579)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```



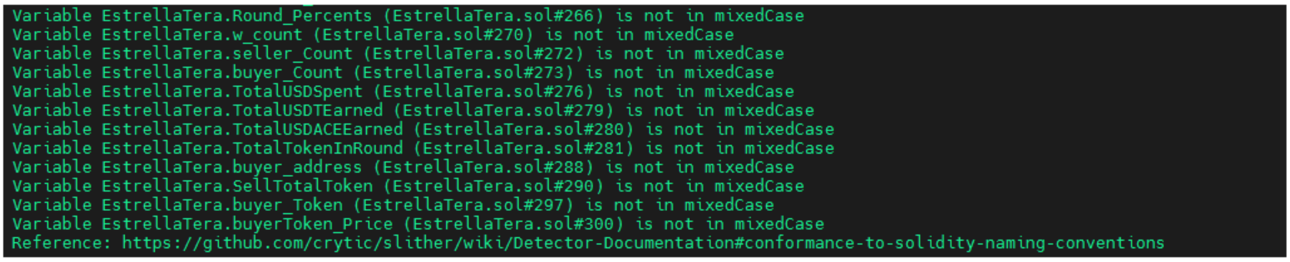
```
Reentrancy in EstrellaTera.userWithdrawREGETAToken() (EstrellaTera.sol#1073-1079):
        External calls:
        - REGETAToken.transfer(msg.sender,getETAWithdraw(_user)) (EstrellaTera.sol#1077)
        State variables written after the call(s):
        - updatebalance(msg.sender) (EstrellaTera.sol#1078)
                - SellTotalToken[_user][i] = 0 (EstrellaTera.sol#1091)
        - updatebalance(msg.sender) (EstrellaTera.sol#1078)
                - TotalUSDACEEarned[_user] = 0 (EstrellaTera.sol#1101)
        - updatebalance(msg.sender) (EstrellaTera.sol#1078)
                - buyerToken_Price[_user][i][j] = 0 (EstrellaTera.sol#1090)
        - updatebalance(msg.sender) (EstrellaTera.sol#1078)
                - buyerTotalToken[_user][i_scope_0] = 0 (EstrellaTera.sol#1097)
        - updatebalance(msg.sender) (EstrellaTera.sol#1078)
                - buyer_Token[_user][i][j] = 0 (EstrellaTera.sol#1089)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```



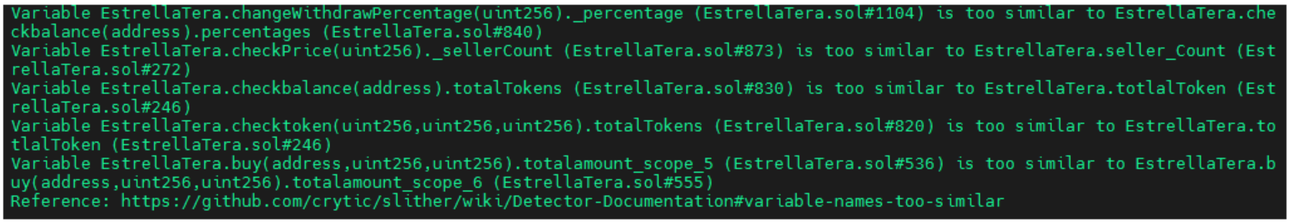
```
EstrellaTera.buy(address,uint256,uint256) (EstrellaTera.sol#367-596) compares to a boolean constant:
        -roundbool == false (EstrellaTera.sol#380)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

EstrellaTera.buy(address,uint256,uint256) (EstrellaTera.sol#367-596) has costly operations inside a loop:
        - totalTokenMint = totalTokenMint.add(remainingbuyerToken) (EstrellaTera.sol#479)
EstrellaTera.buy(address,uint256,uint256) (EstrellaTera.sol#367-596) has costly operations inside a loop:
        - totalTokenMint = totalTokenMint.add(remainingbuyerToken_scope_0) (EstrellaTera.sol#529)
EstrellaTera.buy(address,uint256,uint256) (EstrellaTera.sol#367-596) has costly operations inside a loop:
        - totalTokenMint = totalTokenMint.add(TokenBuy_User_scope_4[0]) (EstrellaTera.sol#547)
EstrellaTera.buy(address,uint256,uint256) (EstrellaTera.sol#367-596) has costly operations inside a loop:
        - PreviousRound = PreviousRound.add(1) (EstrellaTera.sol#566)
EstrellaTera.buy(address,uint256,uint256) (EstrellaTera.sol#367-596) has costly operations inside a loop:
        - PreviousRound = 0 (EstrellaTera.sol#568)
EstrellaTera.buy(address,uint256,uint256) (EstrellaTera.sol#367-596) has costly operations inside a loop:
        - round = round.add(1) (EstrellaTera.sol#571)
EstrellaTera.buy(address,uint256,uint256) (EstrellaTera.sol#367-596) has costly operations inside a loop:
        - endRound = round.sub(2) (EstrellaTera.sol#574)
EstrellaTera.buy(address,uint256,uint256) (EstrellaTera.sol#367-596) has costly operations inside a loop:
        - PreviousRound = round.sub(5) (EstrellaTera.sol#575)
EstrellaTera.buy(address,uint256,uint256) (EstrellaTera.sol#367-596) has costly operations inside a loop:
        - totalTokenMint = 0 (EstrellaTera.sol#580)
EstrellaTera.buy(address,uint256,uint256) (EstrellaTera.sol#367-596) has costly operations inside a loop:
        - tokenPriceAfterTwoRounds = sellerTokenPrice_scope_2 (EstrellaTera.sol#587)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Context._msgData() (EstrellaTera.sol#170-172) is never used and should be removed
SafeMath.mod(uint256,uint256) (EstrellaTera.sol#57-60) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```



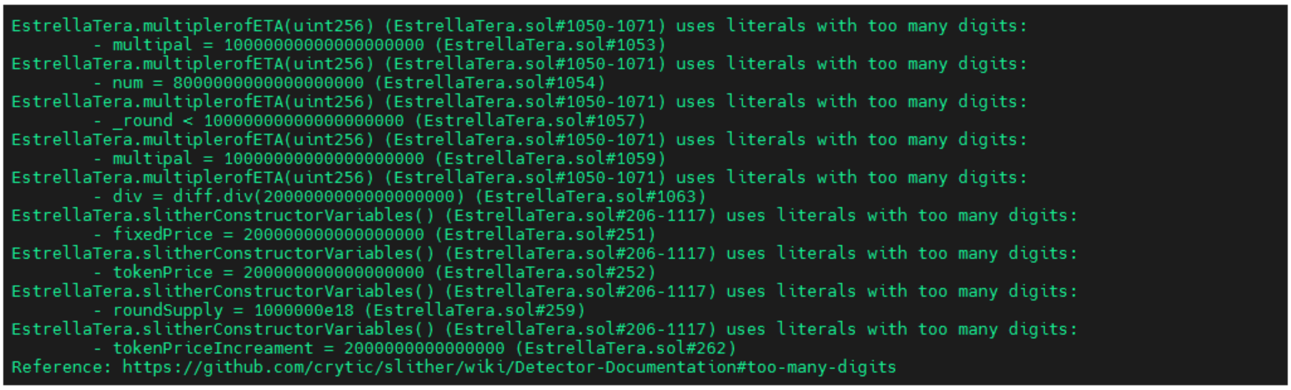
```
Pragma version^0.8.17 (EstrellaTera.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Contract com_Contract (EstrellaTera.sol#161-163) is not in CapWords
Parameter EstrellaTera.register(address)._referral (EstrellaTera.sol#318) is not in mixedCase
Parameter EstrellaTera.updateReferral(address)._user (EstrellaTera.sol#346) is not in mixedCase
Parameter EstrellaTera.countBuyers(address,uint256)._user (EstrellaTera.sol#597) is not in mixedCase
Parameter EstrellaTera.countBuyers(address,uint256)._round (EstrellaTera.sol#597) is not in mixedCase
Parameter EstrellaTera.trasnferAmount(address,address,uint256)._tokenAddress (EstrellaTera.sol#620) is not in mixedCase
Parameter EstrellaTera.trasnferAmount(address,address,uint256)._SellerAddress (EstrellaTera.sol#620) is not in mixedCase
Parameter EstrellaTera.trasnferAmount(address,address,uint256)._totalamount (EstrellaTera.sol#620) is not in mixedCase
Function EstrellaTera.Commission(address,uint256) (EstrellaTera.sol#650-657) is not in mixedCase
```



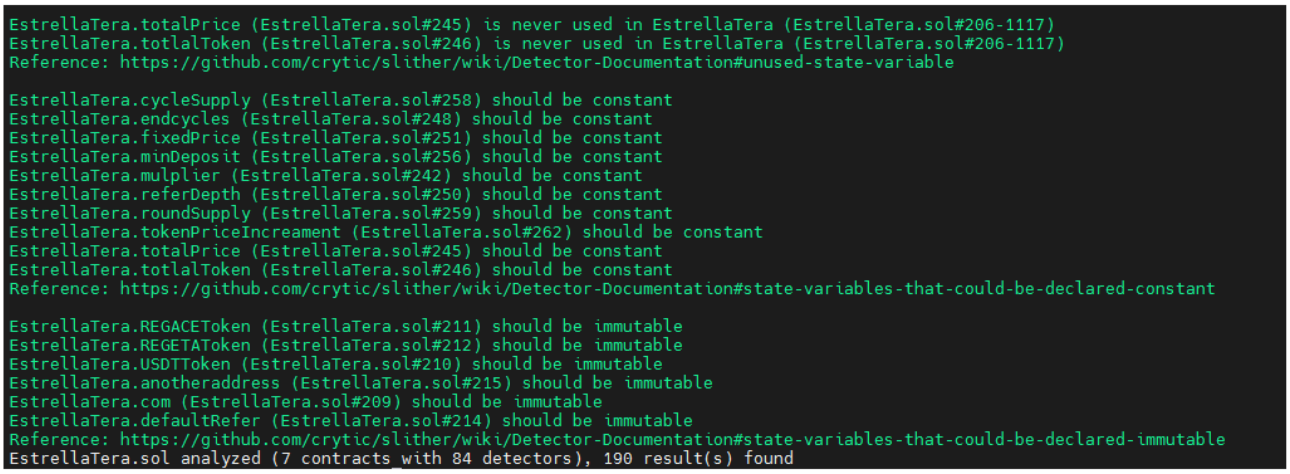
```
Variable EstrellaTera.Round_Percents (EstrellaTera.sol#266) is not in mixedCase
Variable EstrellaTera.w_count (EstrellaTera.sol#270) is not in mixedCase
Variable EstrellaTera.seller_Count (EstrellaTera.sol#272) is not in mixedCase
Variable EstrellaTera.buyer_Count (EstrellaTera.sol#273) is not in mixedCase
Variable EstrellaTera.TotalUSDSpent (EstrellaTera.sol#276) is not in mixedCase
Variable EstrellaTera.TotalUSDTEarned (EstrellaTera.sol#279) is not in mixedCase
Variable EstrellaTera.TotalUSDACEEarned (EstrellaTera.sol#280) is not in mixedCase
Variable EstrellaTera.TotalTokenInRound (EstrellaTera.sol#281) is not in mixedCase
Variable EstrellaTera.buyer_address (EstrellaTera.sol#288) is not in mixedCase
Variable EstrellaTera.SellTotalToken (EstrellaTera.sol#290) is not in mixedCase
Variable EstrellaTera.buyer_Token (EstrellaTera.sol#297) is not in mixedCase
Variable EstrellaTera.buyerToken_Price (EstrellaTera.sol#300) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```



```
Variable EstrellaTera.changeWithdrawPercentage(uint256)._percentage (EstrellaTera.sol#1104) is too similar to EstrellaTera.che
ckbalance(address).percentages (EstrellaTera.sol#840)
Variable EstrellaTera.checkPrice(uint256)._sellerCount (EstrellaTera.sol#873) is too similar to EstrellaTera.seller_Count (Est
rellaTera.sol#272)
Variable EstrellaTera.checkbalance(address).totalTokens (EstrellaTera.sol#830) is too similar to EstrellaTera.totlalToken (Est
rellaTera.sol#246)
Variable EstrellaTera.checktoken(uint256,uint256,uint256).totalTokens (EstrellaTera.sol#820) is too similar to EstrellaTera.to
tlalToken (EstrellaTera.sol#246)
Variable EstrellaTera.buy(address,uint256,uint256).totalamount_scope_5 (EstrellaTera.sol#536) is too similar to EstrellaTera.b
uy(address,uint256,uint256).totalamount_scope_6 (EstrellaTera.sol#555)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
```



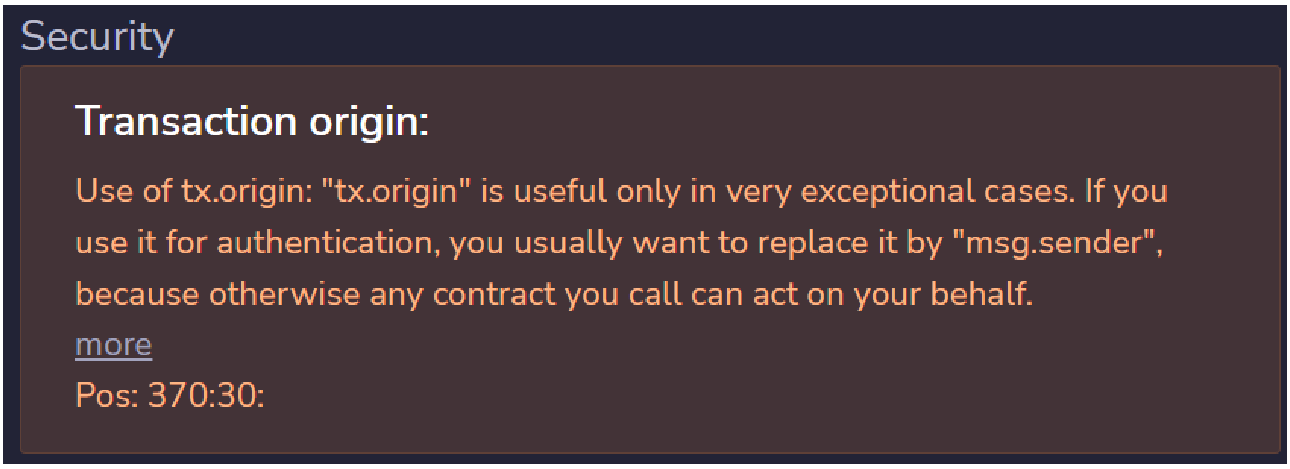
```
EstrellaTera.multiplerofETA(uint256) (EstrellaTera.sol#1050-1071) uses literals with too many digits:
        - multipal = 10000000000000000000 (EstrellaTera.sol#1053)
EstrellaTera.multiplerofETA(uint256) (EstrellaTera.sol#1050-1071) uses literals with too many digits:
        - num = 8000000000000000000 (EstrellaTera.sol#1054)
EstrellaTera.multiplerofETA(uint256) (EstrellaTera.sol#1050-1071) uses literals with too many digits:
        - _round < 10000000000000000000 (EstrellaTera.sol#1057)
EstrellaTera.multiplerofETA(uint256) (EstrellaTera.sol#1050-1071) uses literals with too many digits:
        - multipal = 10000000000000000000 (EstrellaTera.sol#1059)
EstrellaTera.multiplerofETA(uint256) (EstrellaTera.sol#1050-1071) uses literals with too many digits:
        - div = diff.div(20000000000000000000) (EstrellaTera.sol#1063)
EstrellaTera.slitherConstructorVariables() (EstrellaTera.sol#206-1117) uses literals with too many digits:
        - fixedPrice = 200000000000000000 (EstrellaTera.sol#251)
EstrellaTera.slitherConstructorVariables() (EstrellaTera.sol#206-1117) uses literals with too many digits:
        - tokenPrice = 200000000000000000 (EstrellaTera.sol#252)
EstrellaTera.slitherConstructorVariables() (EstrellaTera.sol#206-1117) uses literals with too many digits:
        - roundSupply = 1000000e18 (EstrellaTera.sol#259)
EstrellaTera.slitherConstructorVariables() (EstrellaTera.sol#206-1117) uses literals with too many digits:
        - tokenPriceIncreament = 2000000000000000 (EstrellaTera.sol#262)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```



```
EstrellaTera.totalPrice (EstrellaTera.sol#245) is never used in EstrellaTera (EstrellaTera.sol#206-1117)
EstrellaTera.totlalToken (EstrellaTera.sol#246) is never used in EstrellaTera (EstrellaTera.sol#206-1117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

EstrellaTera.cycleSupply (EstrellaTera.sol#258) should be constant
EstrellaTera.endcycles (EstrellaTera.sol#248) should be constant
EstrellaTera.fixedPrice (EstrellaTera.sol#251) should be constant
EstrellaTera.minDeposit (EstrellaTera.sol#256) should be constant
EstrellaTera.mulplier (EstrellaTera.sol#242) should be constant
EstrellaTera.referDepth (EstrellaTera.sol#250) should be constant
EstrellaTera.roundSupply (EstrellaTera.sol#259) should be constant
EstrellaTera.tokenPriceIncreament (EstrellaTera.sol#262) should be constant
EstrellaTera.totalPrice (EstrellaTera.sol#245) should be constant
EstrellaTera.totlalToken (EstrellaTera.sol#246) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

EstrellaTera.REGACEToken (EstrellaTera.sol#211) should be immutable
EstrellaTera.REGETAToken (EstrellaTera.sol#212) should be immutable
EstrellaTera.USDTToken (EstrellaTera.sol#210) should be immutable
EstrellaTera.anotheraddress (EstrellaTera.sol#215) should be immutable
EstrellaTera.com (EstrellaTera.sol#209) should be immutable
EstrellaTera.defaultRefer (EstrellaTera.sol#214) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
EstrellaTera.sol analyzed (7 contracts with 84 detectors), 190 result(s) found
```

Solidity Static Analysis

EstrellaTera.sol



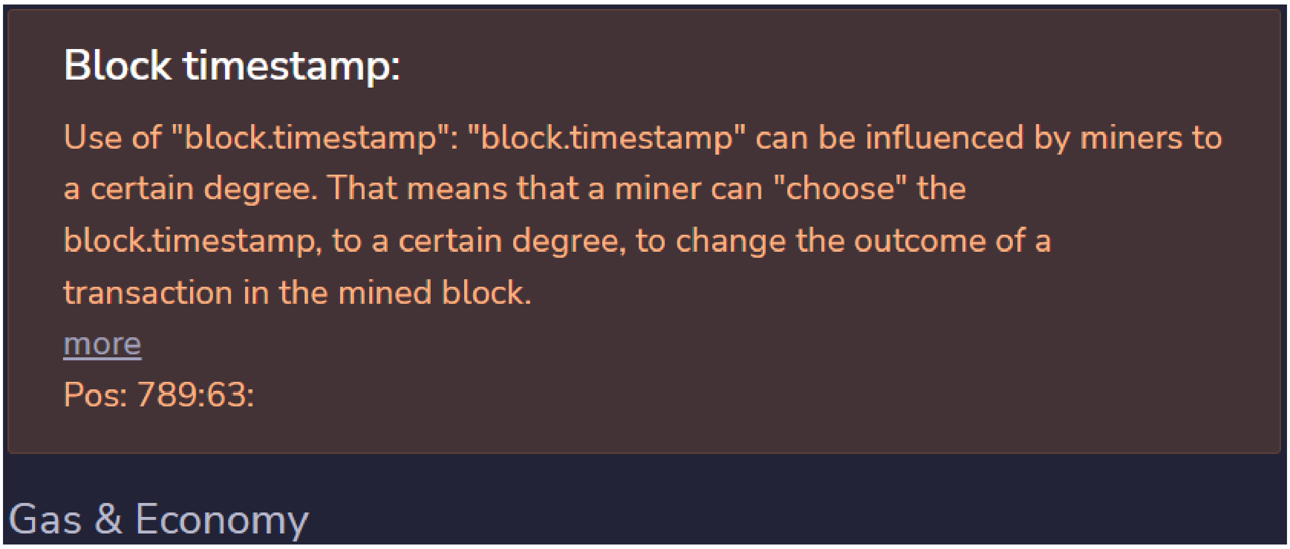
Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.

more

Pos: 370:30:



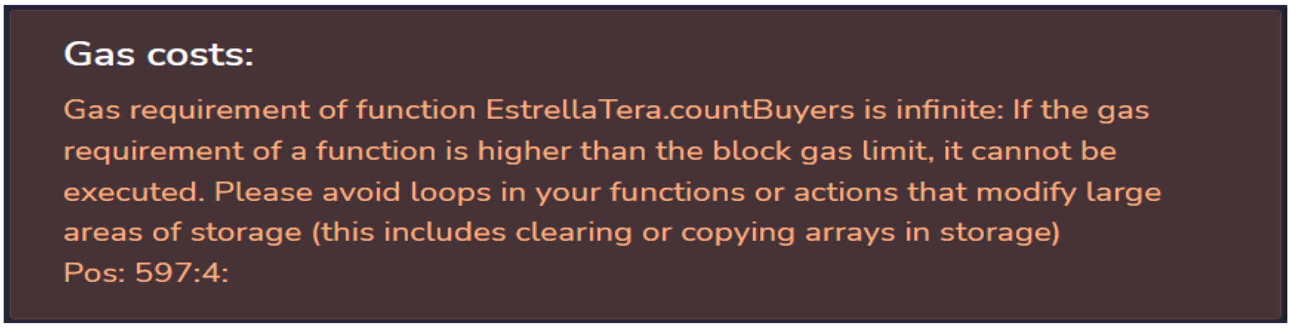
Block timestamp:

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: 789:63:

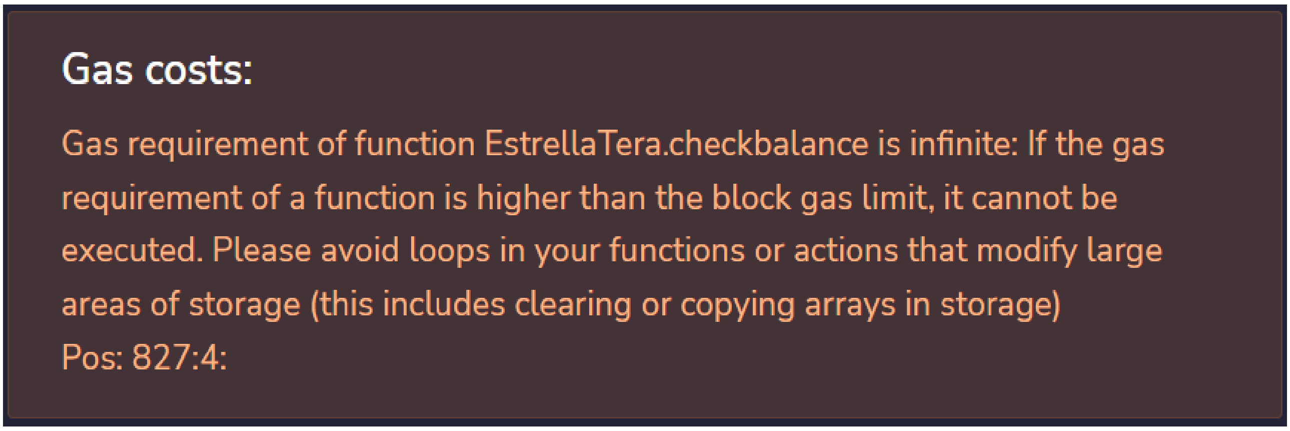
Gas & Economy



Gas costs:

Gas requirement of function EstrellaTera.countBuyers 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)

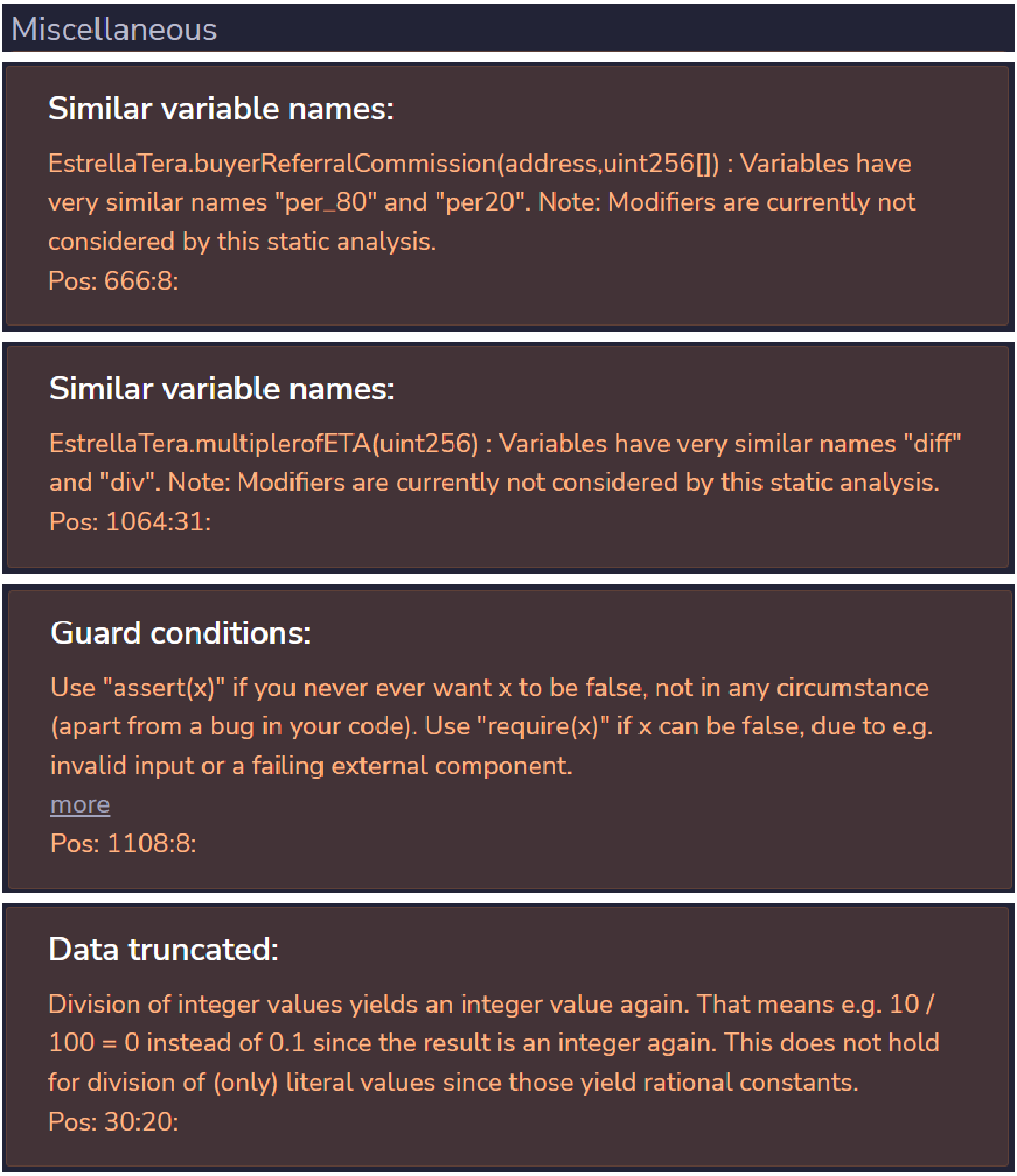
Pos: 597:4:



Gas costs:

Gas requirement of function EstrellaTera.checkbalance 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: 827:4:



Miscellaneous

Similar variable names:

EstrellaTera.buyerReferralCommission(address,uint256[]) : Variables have very similar names "per_80" and "per20". Note: Modifiers are currently not considered by this static analysis.
Pos: 666:8:

Similar variable names:

EstrellaTera.multiplerofETA(uint256) : Variables have very similar names "diff" and "div". Note: Modifiers are currently not considered by this static analysis.
Pos: 1064:31:

Guard conditions:

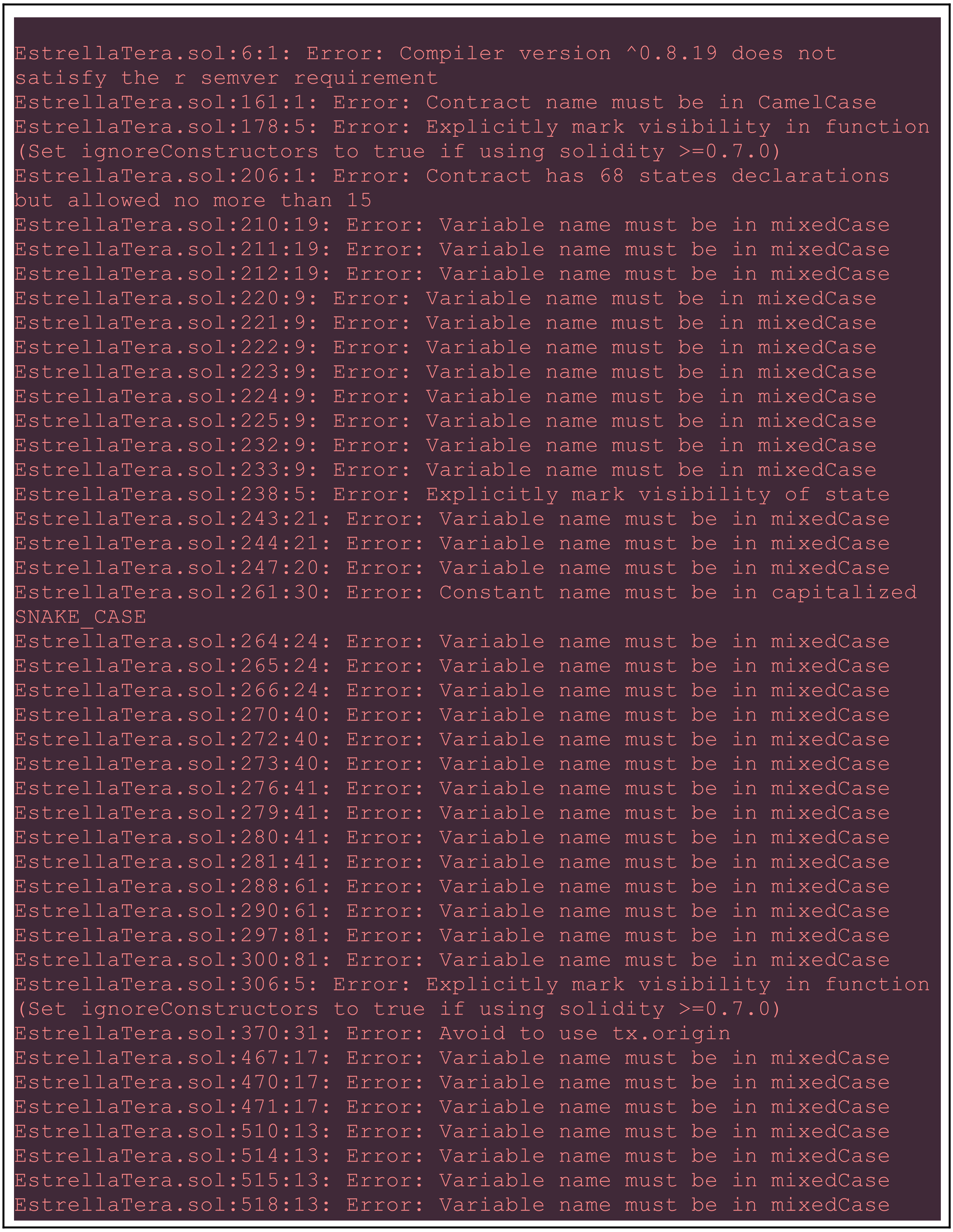
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1108:8:

Data truncated:

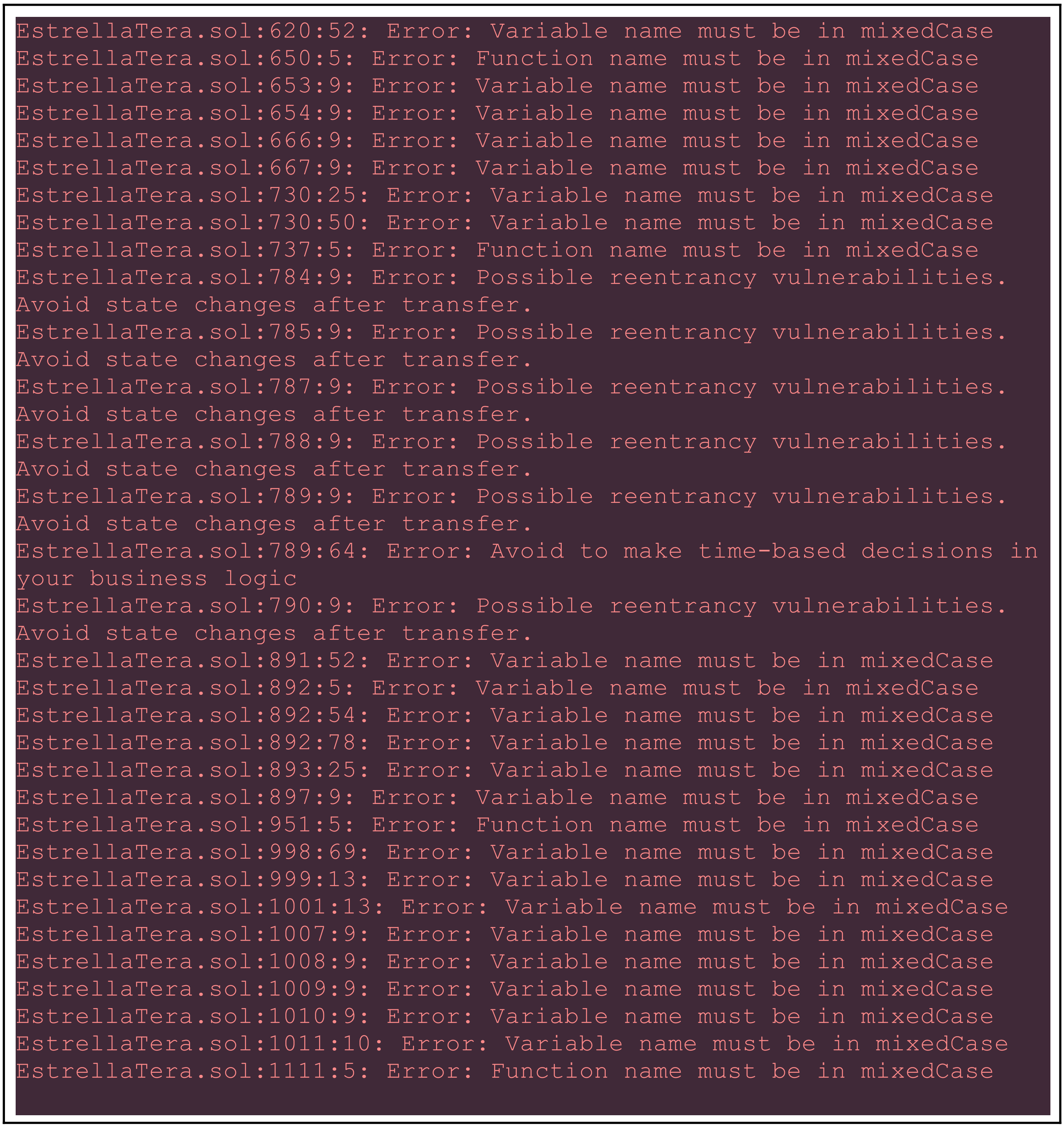
Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 30:20:

Solhint Linter

EstrellaTera.sol



```
EstrellaTera.sol:6:1: Error: Compiler version ^0.8.19 does not
satisfy the r semver requirement
EstrellaTera.sol:161:1: Error: Contract name must be in CamelCase
EstrellaTera.sol:178:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
EstrellaTera.sol:206:1: Error: Contract has 68 states declarations
but allowed no more than 15
EstrellaTera.sol:210:19: Error: Variable name must be in mixedCase
EstrellaTera.sol:211:19: Error: Variable name must be in mixedCase
EstrellaTera.sol:212:19: Error: Variable name must be in mixedCase
EstrellaTera.sol:220:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:221:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:222:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:223:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:224:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:225:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:232:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:233:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:238:5: Error: Explicitly mark visibility of state
EstrellaTera.sol:243:21: Error: Variable name must be in mixedCase
EstrellaTera.sol:244:21: Error: Variable name must be in mixedCase
EstrellaTera.sol:247:20: Error: Variable name must be in mixedCase
EstrellaTera.sol:261:30: Error: Constant name must be in capitalized
SNAKE_CASE
EstrellaTera.sol:264:24: Error: Variable name must be in mixedCase
EstrellaTera.sol:265:24: Error: Variable name must be in mixedCase
EstrellaTera.sol:266:24: Error: Variable name must be in mixedCase
EstrellaTera.sol:270:40: Error: Variable name must be in mixedCase
EstrellaTera.sol:272:40: Error: Variable name must be in mixedCase
EstrellaTera.sol:273:40: Error: Variable name must be in mixedCase
EstrellaTera.sol:276:41: Error: Variable name must be in mixedCase
EstrellaTera.sol:279:41: Error: Variable name must be in mixedCase
EstrellaTera.sol:280:41: Error: Variable name must be in mixedCase
EstrellaTera.sol:281:41: Error: Variable name must be in mixedCase
EstrellaTera.sol:288:61: Error: Variable name must be in mixedCase
EstrellaTera.sol:290:61: Error: Variable name must be in mixedCase
EstrellaTera.sol:297:81: Error: Variable name must be in mixedCase
EstrellaTera.sol:300:81: Error: Variable name must be in mixedCase
EstrellaTera.sol:306:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
EstrellaTera.sol:370:31: Error: Avoid to use tx.origin
EstrellaTera.sol:467:17: Error: Variable name must be in mixedCase
EstrellaTera.sol:470:17: Error: Variable name must be in mixedCase
EstrellaTera.sol:471:17: Error: Variable name must be in mixedCase
EstrellaTera.sol:510:13: Error: Variable name must be in mixedCase
EstrellaTera.sol:514:13: Error: Variable name must be in mixedCase
EstrellaTera.sol:515:13: Error: Variable name must be in mixedCase
EstrellaTera.sol:518:13: Error: Variable name must be in mixedCase
```



```
EstrellaTera.sol:620:52: Error: Variable name must be in mixedCase
EstrellaTera.sol:650:5: Error: Function name must be in mixedCase
EstrellaTera.sol:653:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:654:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:666:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:667:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:730:25: Error: Variable name must be in mixedCase
EstrellaTera.sol:730:50: Error: Variable name must be in mixedCase
EstrellaTera.sol:737:5: Error: Function name must be in mixedCase
EstrellaTera.sol:784:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
EstrellaTera.sol:785:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
EstrellaTera.sol:787:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
EstrellaTera.sol:788:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
EstrellaTera.sol:789:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
EstrellaTera.sol:789:64: Error: Avoid to make time-based decisions in
your business logic
EstrellaTera.sol:790:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
EstrellaTera.sol:891:52: Error: Variable name must be in mixedCase
EstrellaTera.sol:892:5: Error: Variable name must be in mixedCase
EstrellaTera.sol:892:54: Error: Variable name must be in mixedCase
EstrellaTera.sol:892:78: Error: Variable name must be in mixedCase
EstrellaTera.sol:893:25: Error: Variable name must be in mixedCase
EstrellaTera.sol:897:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:951:5: Error: Function name must be in mixedCase
EstrellaTera.sol:998:69: Error: Variable name must be in mixedCase
EstrellaTera.sol:999:13: Error: Variable name must be in mixedCase
EstrellaTera.sol:1001:13: Error: Variable name must be in mixedCase
EstrellaTera.sol:1007:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:1008:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:1009:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:1010:9: Error: Variable name must be in mixedCase
EstrellaTera.sol:1011:10: Error: Variable name must be in mixedCase
EstrellaTera.sol:1111:5: Error: Function name must be in mixedCase
```

Software analysis result:

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

EtherAuthority

Email: audit@EtherAuthority.io