Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	SLATE-Citizen
Platform:	Ethereum
Language:	Solidity
Date:	August 12th, 2023

Table of contents

Email: audit@EtherAuthority.io

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

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the SLATE-Citizen team to perform the Security audit of the SLATE-Citizen 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 August 12th, 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

- The SLATE-Citizen protocol covers multiple ERC1155 contracts, and all contracts have different functions:
    - **BodyToken:** Contract for minting, burning, and creating new tokens.
    - **CitizenToken:** Contract permits adding/updating new children for Citizen composition.
    - **GlyphToken:** Contract updates citizen token address, royalty percentage, minting, burning, and creating new tokens.
    - **ParticleToken:** Contract for reserve mining, bidding and burning.
- There are 4 smart contracts, 3 internal contracts, 3 libraries, 5 interface files which were included in the audit scope. And there were some standard library code such as OpenZepelin, which were excluded. Because those standard library code is considered as time tested and community audited, so we can safely ignore them.
- The smart contracts have functions like burn, mintBatch, burnBatch, receive, mint, etc.

Audit scope

Name	Code Review and Security Analysis Report for SLATE-Citizen Smart Contracts
Platform	**Ethereum / Solidity**
File 1	BodyToken.sol
File 1 MD5 hash	FA998FCA64CB7DBBEDAAB12D36DA6BD1
File 2	CitizenToken.sol
File 2 MD5 hash	47B1A134CEC979A8BD77DCF407179A25
File 3	GlyphToken.sol
File 3 MD5 hash	3A28D35212CC173B0206A7747F671AD2
File 4	ParticleToken.sol
File 4 MD5 hash	F58D20C577A4FB6294F707DF40023096
Audit Date	August 12th, 2023

Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 BodyToken.sol **The Citizen/Owner has control over the following functions:** • Throws if the sender is the citizen contract. • Set a citizen token address by the owner. • Royalty percentage value can be set by the owner. • Owner can create new tokens. • Update token URI by the owner.	**YES, This is valid.**
File 2 CitizenToken.sol **The owner has control over the following functions:** • Owner can increase the maximum supply. • Royalty percentage value can be set by the owner. • Allows addition of new children/ Updation of existing children for Citizen composition by the owner. • Owner can mint citizens. • Owner can burn any existing citizen with its SKU id.	**YES, This is valid.**
File 3 GlyphToken.sol **The Citizen/Owner has control over the following functions:** • Throws if the sender is the citizen contract. • Set a citizen token address by the owner. • Royalty percentage value can be set by the owner. • Owner can create new tokens. • Update token URI by the owner.	**YES, This is valid.**
File 4 ParticleToken.sol **The owner has control over the following functions:** • Owner can reserve mint. • Increase maximum supply by the owner.	**YES, This is valid.**

• Set a citizen token address by the owner. • Citizen Token Contract owners can burn tokens. • Safe transfer tokens by the owner. • Safely transfer toke from multiple addresses by the owner.	

Audit Summary

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

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

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

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

Technical Quick Stats

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

Overall Audit Result: PASSED

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 4 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in SLATE-Citizen 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 SLATE-Citizen Protocol.

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

Code parts are not well commented on smart contracts.

Documentation

We were given a SLATE-Citizen smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are not well commented. but the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

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

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

AS-IS overview

BodyToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyCitizenOrOwner	modifier	Passed	No Issue
3	_checkCitizenOrOwner	internal	Passed	No Issue
4	receive	external	Passed	No Issue
5	supportsInterface	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	uri	write	Passed	No Issue
8	getTokenIds	external	Passed	No Issue
9	setCitizenTokenAddress	external	Require error message	Refer Audit Findings
10	setRoyaltyPercent	external	access only Owner	No Issue
11	createNewToken	write	access only Owner	No Issue
12	_createNewToken	internal	Passed	No Issue
13	updateTokenUri	external	access only Owner	No Issue
14	mint	external	access only Citizen Or Owner	No Issue
15	mintBatch	external	access only Citizen Or Owner	No Issue
16	burn	external	access only Citizen Or Owner	No Issue
17	burnBatch	external	access only Citizen Or Owner	No Issue
18	_beforeTokenTransfer	internal	Passed	No Issue

CitizenToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	receive	external	Passed	No Issue
3	supportsInterface	read	Passed	No Issue
4	totalSupply	read	Passed	No Issue
5	totalSupplyByid	read	Passed	No Issue
6	skusByWallet	external	Passed	No Issue
7	tokensByWallet	external	Passed	No Issue
8	citizenBalance	external	Passed	No Issue
9	getSkuOwner	read	Passed	No Issue
10	getCitizenId	read	Passed	No Issue
11	increaseMaxSupply	external	access only Owner	No Issue
12	setRoyaltyPercent	external	access only Owner	No Issue

13	addChildrenForCitizenComposition	external	access only Owner	No Issue
14	updateChildrenForCitizenComposition	external	access only Owner	No Issue
15	mint	external	Require error message	Refer Audit Findings
16	generateUniqueId	internal	Passed	No Issue
17	mintChildren	internal	Passed	No Issue
18	getChildrenMetadata	internal	Passed	No Issue
19	burn	external	access only Owner	No Issue
20	getChildBalances	internal	Passed	No Issue
21	modifySku	write	access only Owner	No Issue
22	modifySkusInBatch	external	access only Owner	No Issue
23	safeTransferFromBySkuID	write	access only Owner	No Issue
24	supportsInterface	write	Passed	No Issue
25	uri	read	Passed	No Issue
26	childBalance	external	Passed	No Issue
27	childContractsFor	external	Passed	No Issue
28	childIdsForOn	external	Passed	No Issue
29	updateBaseCID	external	access only Owner	No Issue
30	_afterTokenTransfer	internal	Passed	No Issue
31	identifyUIdFromCitizenId	internal	Passed	No Issue
32	createNewUniqueIds	internal	Passed	No Issue
33	generateUniqueId	internal	Passed	No Issue
34	mintChildren	internal	Passed	No Issue
35	removeTokenId	internal	Passed	No Issue
36	safeTransferChildFrom	write	access only Owner	No Issue
37	safeBatchTransferChildFrom	write	access only Owner	No Issue
38	onERC1155Received	write	Passed	No Issue
39	onERC1155BatchReceived	write	Passed	No Issue
40	_receiveChild	internal	Passed	No Issue
41	_removeChild	internal	Passed	No Issue
42	_beforeChildTransfer	internal	Passed	No Issue
43	supportsInterface	read	Passed	No Issue
44	royaltyInfo	read	Passed	No Issue
45	_feeDenominator	internal	Passed	No Issue
46	_setDefaultRoyalty	internal	Passed	No Issue
47	_deleteDefaultRoyalty	internal	Passed	No Issue
48	_setTokenRoyalty	internal	Passed	No Issue
49	_resetTokenRoyalty	internal	Passed	No Issue

GlyphToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyCitizenOrOwner	modifier	Passed	No Issue
3	_checkCitizenOrOwner	internal	Passed	No Issue

4	receive	external	Passed	No Issue
5	supportsInterface	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	uri	write	Passed	No Issue
8	getTokenIds	external	Passed	No Issue
9	setCitizenTokenAddress	external	Require error message	Refer Audit Findings
10	setRoyaltyPercent	external	access only Owner	No Issue
11	createNewToken	write	access only Owner	No Issue
12	_createNewToken	internal	Passed	No Issue
13	updateTokenUri	external	access only Owner	No Issue
14	mint	external	access only Citizen Or Owner	No Issue
15	mintBatch	external	access only Citizen Or Owner	No Issue
16	burn	external	access only Citizen Or Owner	No Issue
17	burnBatch	external	access only Citizen Or Owner	No Issue
18	_beforeTokenTransfer	internal	Passed	No Issue

ParticleToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	totalSupply	read	Passed	No Issue
3	reserveMint	external	access only Owner	No Issue
4	bidParticle	external	Passed	No Issue
5	increaseMaxSupply	external	access only Owner	No Issue
6	setCitizenTokenAddress	external	access only Owner	No Issue
7	burn	write	Passed	No Issue
8	safeTransferFrom	write	access only Owner	No Issue
9	safeBatchTransferFrom	write	access only Owner	No Issue
10	supportsInterface	read	Passed	No Issue
11	uri	read	Passed	No Issue
12	balanceOf	read	Passed	No Issue
13	balanceOfBatch	read	Passed	No Issue
14	setApprovalForAll	write	Passed	No Issue
15	isApprovedForAll	read	Passed	No Issue
16	safeTransferFrom	write	Passed	No Issue
17	safeBatchTransferFrom	write	Passed	No Issue
18	_setURI	internal	Passed	No Issue
19	_setApprovalForAll	internal	Passed	No Issue
20	_mint	internal	Passed	No Issue
21	_mintBatch	internal	Passed	No Issue
22	_burn	internal	Passed	No Issue
23	_burnBatch	internal	Passed	No Issue

24	_beforeTokenTransfer	internal	Passed	No Issue
25	_afterTokenTransfer	internal	Passed	No Issue
26	_doSafeTransferAcceptanceCheck	write	Passed	No Issue
27	_doSafeBatchTransferAcceptanceCheck	write	Passed	No Issue
28	_asSingletonArray	internal	Passed	No Issue
29	onlyOwner	modifier	Passed	No Issue
30	owner	read	Passed	No Issue
31	_checkOwner	internal	Passed	No Issue
32	renounceOwnership	write	access only Owner	No Issue
33	transferOwnership	write	access only Owner	No Issue
34	_transferOwnership	internal	Passed	No Issue

Severity Definitions

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

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No critical severity vulnerabilities were found in the contract code.

High Severity

No high severity vulnerabilities were found in the contract code.

Medium

No medium severity vulnerabilities were found in the contract code.

Low

No low severity vulnerabilities were found in the contract code.

Very Low / Informational / Best practices:

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

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

(2) Require error message:
CitizenToken.sol
In the mint function, it needs to check Address "_to" !=address(0) .

Resolution: We suggest validating with an error message.
require(_to!=address(0),'Invalid Address')

CitizenChildToken.sol

In the setCitizenTokenAddress function, it needs to check Address "_citizenAddress" !=address(0) .

Resolution: We suggest validating with an error message.

require(_citizenAddress!=address(0),'Invalid Address')

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:

CitizenToken.sol

- increaseMaxSupply: Owner can increase maximum supply.
- setRoyaltyPercent: Royalty percentage value can be set by the owner.
- addChildrenForCitizenComposition: Allows addition of new children for Citizen composition by the owner.
- updateChildrenForCitizenComposition: Updation of existing children for Citizen composition by the owner.
- mint: Owner can mint citizens.
- burn: Owner can burn any existing citizen with its SKU id.
- modifySku: Modify citizen type or citizen id for a given SKU by the owner.
- modifySkusInBatch: Modify SKUs In Batch by the owner.
- safeTransferFromBySkuID: Safe transfer tokens by the owner.

ParticleToken.sol

- reserveMint: Owner can reserve mint.
- increaseMaxSupply: Increase maximum supply by the owner.
- setCitizenTokenAddress: Set citizen token address by the owner.
- burn: Citizen Token Contract owners can burn tokens.
- safeTransferFrom: Safe transfer tokens by the owner.
- safeBatchTransferFrom: Safely transfer toke from multiple addresses by the owner.

CitizenChildToken.sol

- _checkCitizenOrOwner: Throws if the sender is the citizen contract.
- setCitizenTokenAddress: Set citizen token address by the owner.
- setRoyaltyPercent: Royalty percentage value can be set by the owner.
- createNewToken: Owner can create new tokens.
- updateTokenUri: Update token URI by the owner.

- mint: Citizen owners can mint tokens.
- mintBatch: Citizen owners can mint tokens Batch vise.
- burn: Citizen owners can burn tokens.
- burnBatch: Citizen owners can burn tokens Batch vise.

ERC998ERC1155TopDown.sol

- updateBaseCID: Base CID can be updated by the owner.
- safeTransferChildFrom: Transfers child token from a token ID.
- safeBatchTransferChildFrom: Transfers batch of child tokens from a token ID.

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.
- _checkOwner: Throws if the sender is not the owner.

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

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We had observed 2 Informational severity issues in the smart contracts. but those are not critical. **So, the smart contracts are ready for the mainnet deployment.**

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

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

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

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

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

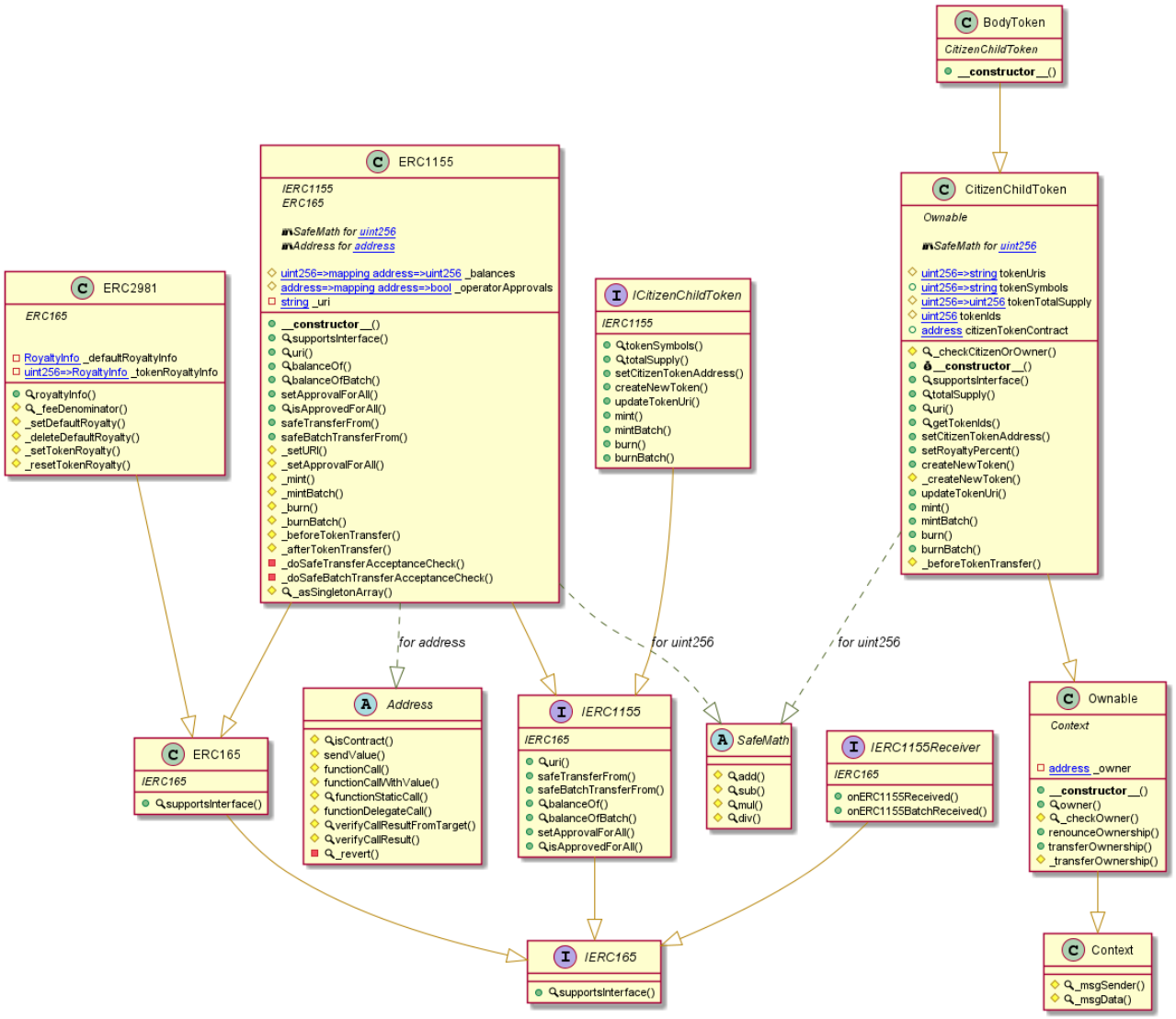
Technical Disclaimer

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

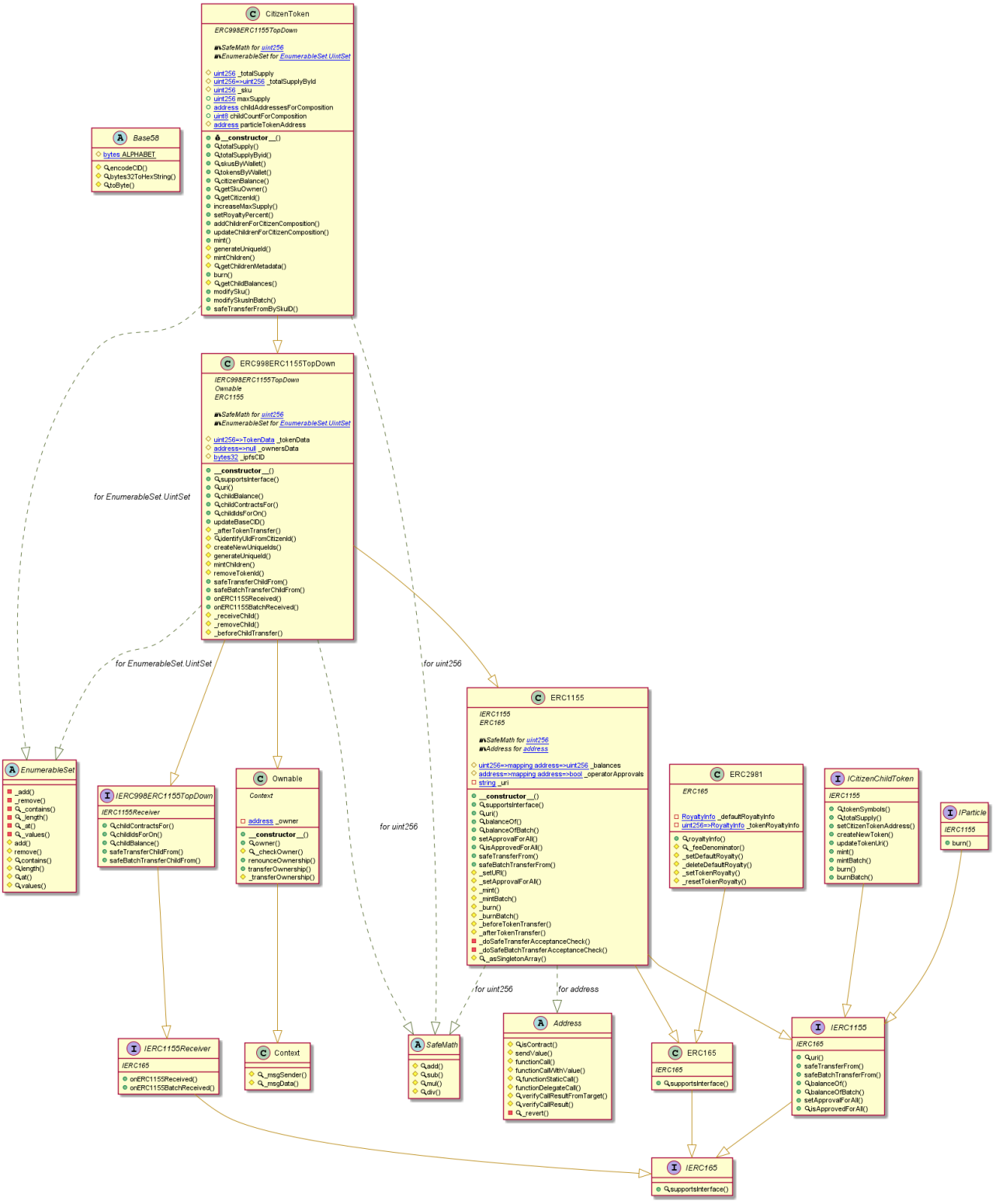
Appendix

Code Flow Diagram - SLATE-Citizen Protocol

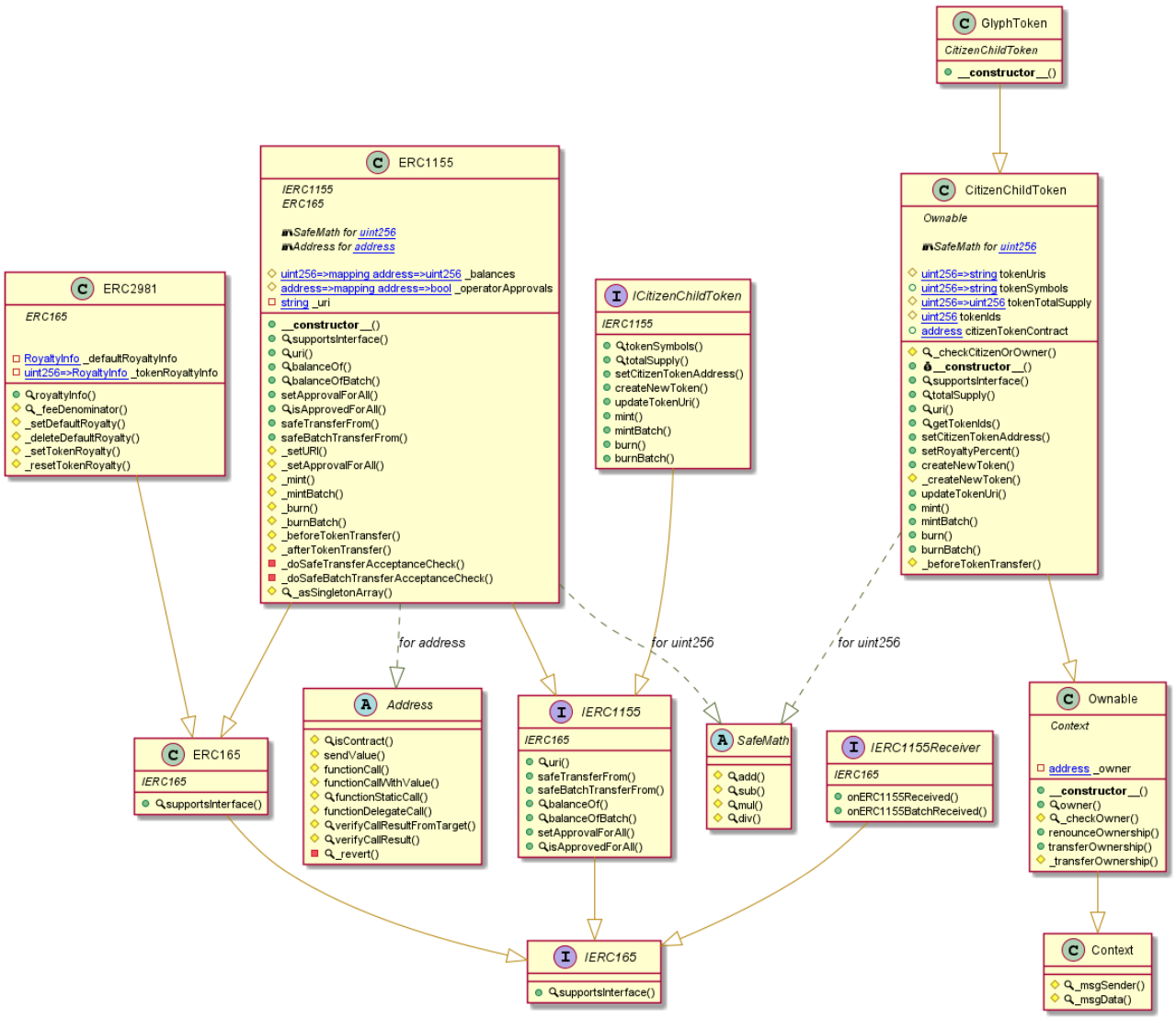
BodyToken Diagram



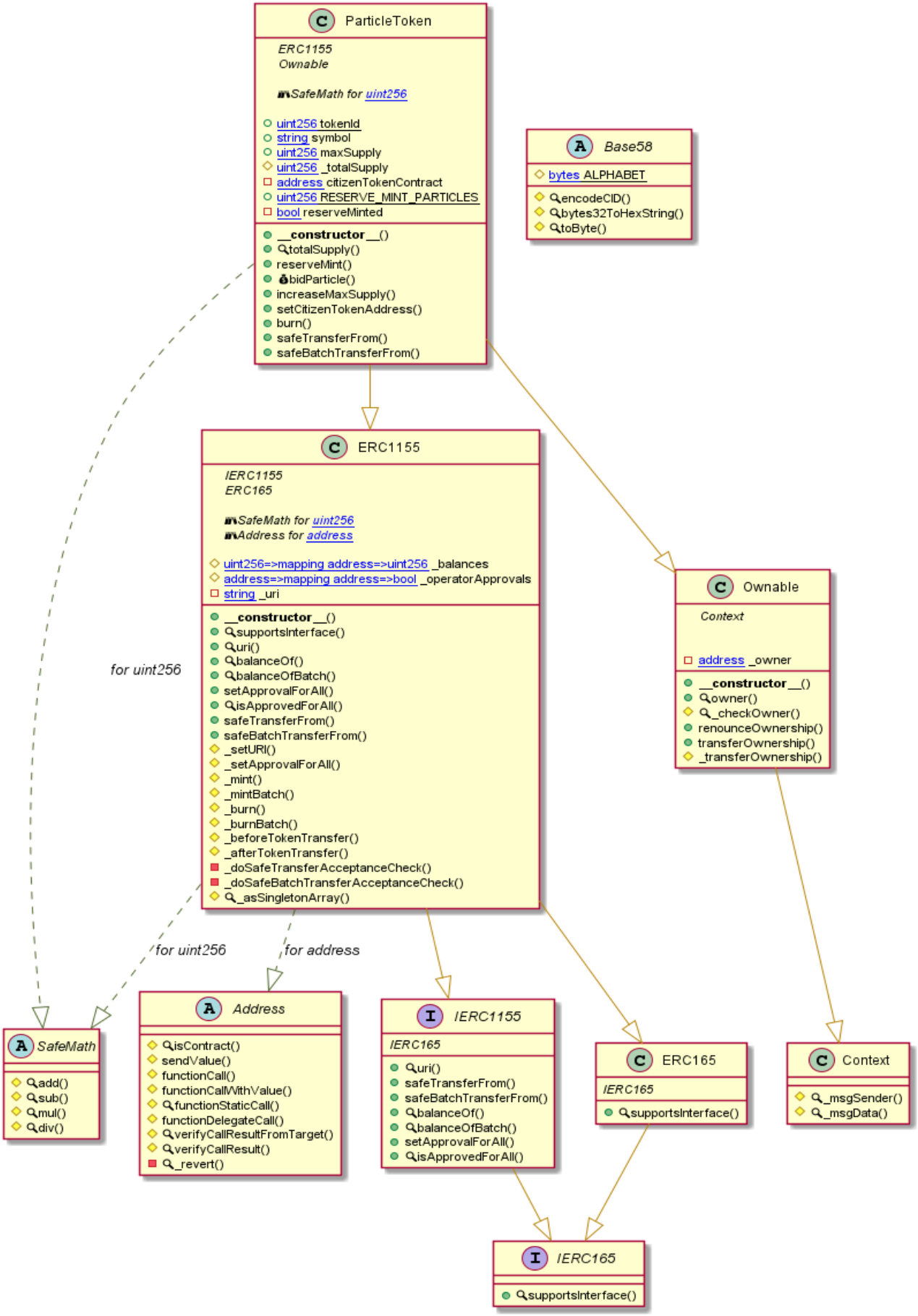
CitizenToken Diagram



GlyphToken Diagram



ParticleToken Diagram



Slither Results Log

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

Slither log >> BodyToken.sol

```
CitizenChildToken.setCitizenTokenAddress(address)._citizenAddress (BodyToken.sol#1191) lacks a zero-check on :
                - citizenTokenContract = _citizenAddress (BodyToken.sol#1192)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (BodyToken.sol#986)'
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (BodyToken.sol#977-996) potentially used
fore declaration: response != IERC1155Receiver.onERC1155Received.selector (BodyToken.sol#987)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (BodyToken.sol#990)' in
C1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (BodyToken.sol#977-996) potentially used b
re declaration: revert(string)(reason) (BodyToken.sol#991)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (BodyToken.s
1007)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (BodyToken.sol#998-10
 potentially used before declaration: response != IERC1155Receiver.onERC1155BatchReceived.selector (BodyToken.sol#1008)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (BodyToken.sol
11)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (BodyToken.sol#998-1017
otentially used before declaration: revert(string)(reason) (BodyToken.sol#1012)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Address._revert(bytes,string) (BodyToken.sol#582-594) uses assembly
        - INLINE ASM (BodyToken.sol#587-590)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (BodyToken.sol#693-717) compares to a boolean constant:
        -require(bool,string)(from == msg.sender || _operatorApprovals[from][msg.sender] == true,ERC1155: Need operator approv
for 3rd party transfers.) (BodyToken.sol#701-704)
ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (BodyToken.sol#722-754) compares to a boolean constan
        -require(bool,string)(from == msg.sender || _operatorApprovals[from][msg.sender] == true,ERC1155: Need operator approv
for 3rd party transfers.) (BodyToken.sol#731-734)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
Address._revert(bytes,string) (BodyToken.sol#582-594) is never used and should be removed
Address.functionCall(address,bytes) (BodyToken.sol#440-442) is never used and should be removed
Address.functionCall(address,bytes,string) (BodyToken.sol#450-456) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (BodyToken.sol#469-471) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (BodyToken.sol#479-488) is never used and should be removed
Address.functionDelegateCall(address,bytes) (BodyToken.sol#521-523) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (BodyToken.sol#531-538) is never used and should be removed
Address.functionStaticCall(address,bytes) (BodyToken.sol#496-498) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (BodyToken.sol#506-513) is never used and should be removed
Address.sendValue(address,uint256) (BodyToken.sol#415-420) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (BodyToken.sol#570-580) is never used and should be removed
Address.verifyCallResultFromTarget(address,bool,bytes,string) (BodyToken.sol#546-562) is never used and should be removed
CitizenChildToken._beforeTokenTransfer(address,address,address,uint256[],uint256[],bytes) (BodyToken.sol#1260-1282) is never u
 and should be removed
Context._msgData() (BodyToken.sol#286-288) is never used and should be removed
ERC1155._burn(address,uint256,uint256) (BodyToken.sol#866-883) is never used and should be removed
ERC1155._burnBatch(address,uint256[],uint256[]) (BodyToken.sol#894-917) is never used and should be removed
ERC1155._mint(address,uint256,uint256,bytes) (BodyToken.sol#805-821) is never used and should be removed
ERC1155._mintBatch(address,uint256[],uint256[],bytes) (BodyToken.sol#834-854) is never used and should be removed
ERC2981._deleteDefaultRoyalty() (BodyToken.sol#1097-1099) is never used and should be removed
ERC2981._resetTokenRoyalty(uint256) (BodyToken.sol#1119-1121) is never used and should be removed
ERC2981._setDefaultRoyalty(address,uint96) (BodyToken.sol#1087-1092) is never used and should be removed
ERC2981._setTokenRoyalty(uint256,address,uint96) (BodyToken.sol#1109-1114) is never used and should be removed
SafeMath.div(uint256,uint256) (BodyToken.sol#19-22) is never used and should be removed
SafeMath.mul(uint256,uint256) (BodyToken.sol#15-17) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.1 (BodyToken.sol#3) allows old versions
solc-0.8.1 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Low level call in Address.functionStaticCall(address,bytes,string) (BodyToken.sol#506-513):
        - (success,returndata) = target.staticcall(data) (BodyToken.sol#511)
Low level call in Address.functionDelegateCall(address,bytes,string) (BodyToken.sol#531-538):
        - (success,returndata) = target.delegatecall(data) (BodyToken.sol#536)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Variable ERC1155._balances (BodyToken.sol#603) is not in mixedCase
Variable ERC1155._operatorApprovals (BodyToken.sol#606) is not in mixedCase
Parameter CitizenChildToken.totalSupply(uint256)._tokenId (BodyToken.sol#1175) is not in mixedCase
Parameter CitizenChildToken.uri(uint256)._tokenId (BodyToken.sol#1179) is not in mixedCase
Parameter CitizenChildToken.setCitizenTokenAddress(address)._citizenAddress (BodyToken.sol#1191) is not in mixedCase
Parameter CitizenChildToken.createNewToken(uint256,string,string)._tokenId (BodyToken.sol#1200) is not in mixedCase
Parameter CitizenChildToken.createNewToken(uint256,string,string)._tokenSymbol (BodyToken.sol#1201) is not in mixedCase
Parameter CitizenChildToken.createNewToken(uint256,string,string)._tokenUri (BodyToken.sol#1202) is not in mixedCase
Parameter CitizenChildToken.updateTokenUri(uint256,string)._tokenId (BodyToken.sol#1222) is not in mixedCase
Parameter CitizenChildToken.updateTokenUri(uint256,string)._tokenUri (BodyToken.sol#1222) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in CitizenChildToken.receive() (BodyToken.sol#1163-1166):
        External calls:
        - address(owner()).transfer(msg.value) (BodyToken.sol#1164)
        Event emitted after the call(s):
        - RoyaltyPaymentReceived(msg.sender,msg.value) (BodyToken.sol#1165)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable CitizenChildToken._createNewToken(uint256,string,string)._tokenSymbol (BodyToken.sol#1209) is too similar to CitizenC
dToken.tokenSymbols (BodyToken.sol#1131)
Variable CitizenChildToken.createNewToken(uint256,string,string)._tokenSymbol (BodyToken.sol#1201) is too similar to CitizenCh
Token.tokenSymbols (BodyToken.sol#1131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
BodyToken.sol analyzed (13 contracts with 84 detectors), 57 result(s) found
```

Slither log >> CitizenToken.sol

```
CitizenToken.constructor(string,bytes32,uint256,address)._ipfsCID (CitizenToken.sol#1737) shadows:
        - ERC998ERC1155TopDown._ipfsCID (CitizenToken.sol#1319) (state variable)
CitizenToken.modifySku(uint256,uint256).owner (CitizenToken.sol#1938) shadows:
        - Ownable.owner() (CitizenToken.sol#616-618) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

CitizenToken.mint(address,uint256[],uint256[]) (CitizenToken.sol#1848-1866) should emit an event for:
        - _totalSupply = _totalSupply.add(_citizenCount) (CitizenToken.sol#1861)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

CitizenToken.constructor(string,bytes32,uint256,address).particleToken (CitizenToken.sol#1739) lacks a zero-check on :
                - particleTokenAddress = particleToken (CitizenToken.sol#1742)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

CitizenToken.burn(uint256) (CitizenToken.sol#1902-1916) has external calls inside a loop: ICitizenChildToken(childContracts[i]
urnBatch(address(this),childTokens,amounts) (CitizenToken.sol#1908)
CitizenToken.mintChildren(uint256,uint256) (CitizenToken.sol#1876-1887) has external calls inside a loop: ICitizenChildToken(c
dAddress).mint(address(this),childTokenId,childAmount,abi.encodePacked(skuId)) (CitizenToken.sol#1885)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (CitizenToken.sol#125
 in ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (CitizenToken.sol#1245-1264) potenti
y used before declaration: response != IERC1155Receiver.onERC1155Received.selector (CitizenToken.sol#1255)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (CitizenToken.sol#1258)
n ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (CitizenToken.sol#1245-1264) potential
used before declaration: revert(string)(reason) (CitizenToken.sol#1259)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (CitizenToke
ol#1275)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (CitizenToken.sol#
6-1285) potentially used before declaration: response != IERC1155Receiver.onERC1155BatchReceived.selector (CitizenToken.sol#12
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (CitizenToken.
#1279)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (CitizenToken.sol#12
1285) potentially used before declaration: revert(string)(reason) (CitizenToken.sol#1280)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```

```
Reentrancy in CitizenToken.burn(uint256) (CitizenToken.sol#1902-1916):
        External calls:
        - ICitizenChildToken(childContracts[i]).burnBatch(address(this),childTokens,amounts) (CitizenToken.sol#1908)
        State variables written after the call(s):
        - removeTokenId(getSkuOwner(skuId),skuId) (CitizenToken.sol#1913)
                - _ownersData[from][i] = _ownersData[from][_ownersData[from].length - 1] (CitizenToken.sol#1450)
                - _ownersData[from].pop() (CitizenToken.sol#1451)
        - _totalSupplyById[_tokenData[skuId].citizenId] = _totalSupplyById[_tokenData[skuId].citizenId].sub(1) (CitizenToken.s
1911)
Reentrancy in CitizenToken.burn(uint256) (CitizenToken.sol#1902-1916):
        External calls:
        - ICitizenChildToken(childContracts[i]).burnBatch(address(this),childTokens,amounts) (CitizenToken.sol#1908)
        - _burn(getSkuOwner(skuId),_tokenData[skuId].citizenId,1) (CitizenToken.sol#1914)
                - ICitizenChildToken(childAddress).mint(address(this),childTokenId,childAmount,abi.encodePacked(skuId)) (Citiz
oken.sol#1885)
        State variables written after the call(s):
        - _totalSupply -- (CitizenToken.sol#1915)
Reentrancy in CitizenToken.modifySku(uint256,uint256) (CitizenToken.sol#1936-1946):
        External calls:
        - super._burn(owner,_tokenData[skuId].citizenId,1) (CitizenToken.sol#1939)
                - ICitizenChildToken(childAddress).mint(address(this),childTokenId,childAmount,abi.encodePacked(skuId)) (Citiz
oken.sol#1885)
        State variables written after the call(s):
        - _totalSupplyById[_tokenData[skuId].citizenId] = _totalSupplyById[_tokenData[skuId].citizenId].sub(1) (CitizenToken.s
1940)
        - _totalSupplyById[_tokenData[skuId].citizenId] = _totalSupplyById[_tokenData[skuId].citizenId].add(1) (CitizenToken.s
1944)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Reentrancy in ERC998ERC1155TopDown.safeBatchTransferChildFrom(uint256,address,address,uint256[],uint256[],bytes) (CitizenToken
l#1494-1517):
        External calls:
        - ERC1155(childContract).safeBatchTransferFrom(address(this),to,childTokenIds,amounts,data) (CitizenToken.sol#1515)
        Event emitted after the call(s):
        - TransferBatchChild(_tokenData[uId].citizenId,to,childContract,childTokenIds,amounts) (CitizenToken.sol#1516)
Reentrancy in ERC998ERC1155TopDown.safeTransferChildFrom(uint256,address,address,uint256,uint256,bytes) (CitizenToken.sol#1460
89):
        External calls:
        - ERC1155(childContract).safeTransferFrom(address(this),to,childTokenId,amount,data) (CitizenToken.sol#1487)
        Event emitted after the call(s):
        - TransferSingleChild(_tokenData[uId].citizenId,to,childContract,childTokenId,amount) (CitizenToken.sol#1488)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
Address._revert(bytes,string) (CitizenToken.sol#420-432) uses assembly
        - INLINE ASM (CitizenToken.sol#425-428)
EnumerableSet.values(EnumerableSet.Bytes32Set) (CitizenToken.sol#499-508) uses assembly
        - INLINE ASM (CitizenToken.sol#503-505)
EnumerableSet.values(EnumerableSet.AddressSet) (CitizenToken.sol#535-544) uses assembly
        - INLINE ASM (CitizenToken.sol#539-541)
EnumerableSet.values(EnumerableSet.UintSet) (CitizenToken.sol#571-580) uses assembly
        - INLINE ASM (CitizenToken.sol#575-577)
ERC998ERC1155TopDown.uri(uint256) (CitizenToken.sol#1332-1342) uses assembly
        - INLINE ASM (CitizenToken.sol#1335-1337)
ERC998ERC1155TopDown.onERC1155Received(address,address,uint256,uint256,bytes) (CitizenToken.sol#1523-1551) uses assembly
        - INLINE ASM (CitizenToken.sol#1543-1545)
ERC998ERC1155TopDown.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes) (CitizenToken.sol#1557-1579) uses assem
        - INLINE ASM (CitizenToken.sol#1570-1572)
CitizenToken.getChildrenMetadata(uint256,uint8) (CitizenToken.sol#1889-1896) uses assembly
        - INLINE ASM (CitizenToken.sol#1892-1894)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (CitizenToken.sol#961-985) compares to a boolean constant:
        -require(bool,string)(from == msg.sender || _operatorApprovals[from][msg.sender] == true,ERC1155: Need operator approv
for 3rd party transfers.) (CitizenToken.sol#969-972)
ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (CitizenToken.sol#990-1022) compares to a boolean con
nt:
        -require(bool,string)(from == msg.sender || _operatorApprovals[from][msg.sender] == true,ERC1155: Need operator approv
for 3rd party transfers.) (CitizenToken.sol#999-1002)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

CitizenToken.generateUniqueId() (CitizenToken.sol#1868-1870) has costly operations inside a loop:
        - ++ _sku (CitizenToken.sol#1869)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
```

```
Pragma version^0.8.1 (CitizenToken.sol#2) allows old versions
solc-0.8.1 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (CitizenToken.sol#253-258):
        - (success) = recipient.call{value: amount}() (CitizenToken.sol#256)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (CitizenToken.sol#317-326):
        - (success,returndata) = target.call{value: value}(data) (CitizenToken.sol#324)
Low level call in Address.functionStaticCall(address,bytes,string) (CitizenToken.sol#344-351):
        - (success,returndata) = target.staticcall(data) (CitizenToken.sol#349)
Low level call in Address.functionDelegateCall(address,bytes,string) (CitizenToken.sol#369-376):
        - (success,returndata) = target.delegatecall(data) (CitizenToken.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter Base58.bytes32ToHexString(bytes32)._bytes32 (CitizenToken.sol#47) is not in mixedCase
Variable ERC1155._balances (CitizenToken.sol#871) is not in mixedCase
Variable ERC1155._operatorApprovals (CitizenToken.sol#874) is not in mixedCase
Parameter ERC998ERC1155TopDown.uri(uint256)._tokenId (CitizenToken.sol#1332) is not in mixedCase
Parameter ERC998ERC1155TopDown.createNewUniqueIds(address,uint256,uint256)._to (CitizenToken.sol#1425) is not in mixedCase
Parameter ERC998ERC1155TopDown.createNewUniqueIds(address,uint256,uint256)._id (CitizenToken.sol#1426) is not in mixedCase
Parameter ERC998ERC1155TopDown.createNewUniqueIds(address,uint256,uint256)._amount (CitizenToken.sol#1427) is not in mixedCase
Variable ERC998ERC1155TopDown._tokenData (CitizenToken.sol#1314) is not in mixedCase
Variable ERC998ERC1155TopDown._ownersData (CitizenToken.sol#1317) is not in mixedCase
Variable ERC998ERC1155TopDown._ipfsCID (CitizenToken.sol#1319) is not in mixedCase
Parameter CitizenToken.skusByWallet(address)._tokenOwner (CitizenToken.sol#1774) is not in mixedCase
Parameter CitizenToken.tokensByWallet(address)._tokenOwner (CitizenToken.sol#1778) is not in mixedCase
Parameter CitizenToken.citizenBalance(address)._tokenOwner (CitizenToken.sol#1786) is not in mixedCase
Parameter CitizenToken.getSkuOwner(uint256)._skuId (CitizenToken.sol#1790) is not in mixedCase
Parameter CitizenToken.getCitizenId(uint256)._skuId (CitizenToken.sol#1794) is not in mixedCase
Parameter CitizenToken.increaseMaxSupply(uint256)._additionalSupply (CitizenToken.sol#1798) is not in mixedCase
Parameter CitizenToken.mint(address,uint256[],uint256[])._to (CitizenToken.sol#1849) is not in mixedCase
Parameter CitizenToken.mint(address,uint256[],uint256[])._citizenIds (CitizenToken.sol#1850) is not in mixedCase
Parameter CitizenToken.mint(address,uint256[],uint256[])._amounts (CitizenToken.sol#1851) is not in mixedCase
Parameter CitizenToken.getChildBalances(uint256,address,uint256[])._childContract (CitizenToken.sol#1920) is not in mixedCase
Parameter CitizenToken.getChildBalances(uint256,address,uint256[])._childTokens (CitizenToken.sol#1921) is not in mixedCase
```

```
Parameter CitizenToken.modifySku(uint256,uint256)._newCitizenId (CitizenToken.sol#1936) is not in mixedCase
Parameter CitizenToken.modifySkusInBatch(uint256[],uint256[])._newCitizenIds (CitizenToken.sol#1948) is not in mixedCase
Variable CitizenToken._totalSupply (CitizenToken.sol#1718) is not in mixedCase
Variable CitizenToken._totalSupplyById (CitizenToken.sol#1721) is not in mixedCase
Variable CitizenToken._sku (CitizenToken.sol#1723) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in CitizenToken.receive() (CitizenToken.sol#1750-1753):
        External calls:
        - address(owner()).transfer(msg.value) (CitizenToken.sol#1751)
        Event emitted after the call(s):
        - RoyaltyPaymentReceived(msg.sender,msg.value) (CitizenToken.sol#1752)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable CitizenToken.getChildBalances(uint256,address,uint256[])._childContract (CitizenToken.sol#1920) is too similar to IER
8ERC1155TopDown.childContractsFor(uint256).childContracts (CitizenToken.sol#168)
Variable CitizenToken.getChildBalances(uint256,address,uint256[])._childContract (CitizenToken.sol#1920) is too similar to ERC
ERC1155TopDown.childContractsFor(uint256).childContracts (CitizenToken.sol#1359)
Variable CitizenToken.getChildBalances(uint256,address,uint256[])._childContract (CitizenToken.sol#1920) is too similar to Cit
nToken.burn(uint256).childContracts (CitizenToken.sol#1904)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

CitizenToken.particleTokenAddress (CitizenToken.sol#1731) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
CitizenToken.sol analyzed (17 contracts with 84 detectors), 111 result(s) found
```

Slither log >> GlyphToken.sol

```
CitizenChildToken.setCitizenTokenAddress(address)._citizenAddress (GlyphToken.sol#1190) lacks a zero-check on :
                - citizenTokenContract = _citizenAddress (GlyphToken.sol#1191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (GlyphToken.sol#985)'
 ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (GlyphToken.sol#976-995) potentially us
before declaration: response != IERC1155Receiver.onERC1155Received.selector (GlyphToken.sol#986)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (GlyphToken.sol#989)' i
RC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (GlyphToken.sol#976-995) potentially used
fore declaration: revert(string)(reason) (GlyphToken.sol#990)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (GlyphToken.
#1006)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (GlyphToken.sol#997-
6) potentially used before declaration: response != IERC1155Receiver.onERC1155BatchReceived.selector (GlyphToken.sol#1007)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (GlyphToken.so
010)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (GlyphToken.sol#997-10
 potentially used before declaration: revert(string)(reason) (GlyphToken.sol#1011)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Address._revert(bytes,string) (GlyphToken.sol#581-593) uses assembly
        - INLINE ASM (GlyphToken.sol#586-589)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (GlyphToken.sol#692-716) compares to a boolean constant:
        -require(bool,string)(from == msg.sender || _operatorApprovals[from][msg.sender] == true,ERC1155: Need operator approv
for 3rd party transfers.) (GlyphToken.sol#700-703)
ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (GlyphToken.sol#721-753) compares to a boolean consta
        -require(bool,string)(from == msg.sender || _operatorApprovals[from][msg.sender] == true,ERC1155: Need operator approv
for 3rd party transfers.) (GlyphToken.sol#730-733)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
Pragma version^0.8.1 (GlyphToken.sol#2) allows old versions
solc-0.8.1 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (GlyphToken.sol#414-419):
        - (success) = recipient.call{value: amount}() (GlyphToken.sol#417)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (GlyphToken.sol#478-487):
        - (success,returndata) = target.call{value: value}(data) (GlyphToken.sol#485)
Low level call in Address.functionStaticCall(address,bytes,string) (GlyphToken.sol#505-512):
        - (success,returndata) = target.staticcall(data) (GlyphToken.sol#510)
Low level call in Address.functionDelegateCall(address,bytes,string) (GlyphToken.sol#530-537):
        - (success,returndata) = target.delegatecall(data) (GlyphToken.sol#535)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable ERC1155._balances (GlyphToken.sol#602) is not in mixedCase
Variable ERC1155._operatorApprovals (GlyphToken.sol#605) is not in mixedCase
Parameter CitizenChildToken.totalSupply(uint256)._tokenId (GlyphToken.sol#1174) is not in mixedCase
Parameter CitizenChildToken.uri(uint256)._tokenId (GlyphToken.sol#1178) is not in mixedCase
Parameter CitizenChildToken.setCitizenTokenAddress(address)._citizenAddress (GlyphToken.sol#1190) is not in mixedCase
Parameter CitizenChildToken.createNewToken(uint256,string,string)._tokenId (GlyphToken.sol#1199) is not in mixedCase
Parameter CitizenChildToken.createNewToken(uint256,string,string)._tokenSymbol (GlyphToken.sol#1200) is not in mixedCase
Parameter CitizenChildToken.createNewToken(uint256,string,string)._tokenUri (GlyphToken.sol#1201) is not in mixedCase
Parameter CitizenChildToken.updateTokenUri(uint256,string)._tokenId (GlyphToken.sol#1221) is not in mixedCase
Parameter CitizenChildToken.updateTokenUri(uint256,string)._tokenUri (GlyphToken.sol#1221) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
Reentrancy in CitizenChildToken.receive() (GlyphToken.sol#1162-1165):
        External calls:
        - address(owner()).transfer(msg.value) (GlyphToken.sol#1163)
        Event emitted after the call(s):
        - RoyaltyPaymentReceived(msg.sender,msg.value) (GlyphToken.sol#1164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable CitizenChildToken._createNewToken(uint256,string,string)._tokenSymbol (GlyphToken.sol#1208) is too similar to Citizen
ldToken.tokenSymbols (GlyphToken.sol#1130)
Variable CitizenChildToken.createNewToken(uint256,string,string)._tokenSymbol (GlyphToken.sol#1200) is too similar to CitizenC
dToken.tokenSymbols (GlyphToken.sol#1130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
GlyphToken.sol analyzed (13 contracts with 84 detectors), 57 result(s) found
```

Slither log >> ParticleToken.sol

```
ParticleToken.constructor(string,string,uint256,address)._uri (ParticleToken.sol#994) shadows:
        - ERC1155._uri (ParticleToken.sol#500) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

ParticleToken.bidParticle(uint256,uint256,uint256,uint256,bytes) (ParticleToken.sol#1016-1035) should emit an event for:
        - _totalSupply = _totalSupply.add(_amount) (ParticleToken.sol#1023)
ParticleToken.burn(address,uint256) (ParticleToken.sol#1045-1049) should emit an event for:
        - _totalSupply = _totalSupply.sub(_amount) (ParticleToken.sol#1048)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

ParticleToken.constructor(string,string,uint256,address)._citizenContract (ParticleToken.sol#997) lacks a zero-check on :
                - citizenTokenContract = _citizenContract (ParticleToken.sol#1002)
ParticleToken.setCitizenTokenAddress(address).tokenAddress (ParticleToken.sol#1041) lacks a zero-check on :
                - citizenTokenContract = tokenAddress (ParticleToken.sol#1042)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

ParticleToken.bidParticle(uint256,uint256,uint256,uint256,bytes) (ParticleToken.sol#1016-1035) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= _startDate && block.timestamp <= _endDate,PARTICLE: Mint window closed) (Par
ticleToken.sol#1025)
        - require(bool,string)(block.timestamp <= _expiryDate,PARTICLE: Signature Expired) (ParticleToken.sol#1026)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address._revert(bytes,string) (ParticleToken.sol#226-238) uses assembly
        - INLINE ASM (ParticleToken.sol#231-234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (ParticleToken.sol#584-608) compares to a boolean constant:
        -require(bool,string)(from == msg.sender || _operatorApprovals[from][msg.sender] == true,ERC1155: Need operator approv
al for 3rd party transfers.) (ParticleToken.sol#592-595)
ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (ParticleToken.sol#613-645) compares to a boolean con
stant:
        -require(bool,string)(from == msg.sender || _operatorApprovals[from][msg.sender] == true,ERC1155: Need operator approv
```

```
        -require(bool,string)(from == msg.sender || _operatorApprovals[from][msg.sender] == true,ERC1155: Need operator approv
al for 3rd party transfers.) (ParticleToken.sol#622-625)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Address._revert(bytes,string) (ParticleToken.sol#226-238) is never used and should be removed
Address.functionCall(address,bytes) (ParticleToken.sol#84-86) is never used and should be removed
Address.functionCall(address,bytes,string) (ParticleToken.sol#94-100) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (ParticleToken.sol#113-115) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (ParticleToken.sol#123-132) is never used and should be removed
Address.functionDelegateCall(address,bytes) (ParticleToken.sol#165-167) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (ParticleToken.sol#175-182) is never used and should be removed
Address.functionStaticCall(address,bytes) (ParticleToken.sol#140-142) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (ParticleToken.sol#150-157) is never used and should be removed
Address.sendValue(address,uint256) (ParticleToken.sol#59-64) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (ParticleToken.sol#214-224) is never used and should be removed
Address.verifyCallResultFromTarget(address,bool,bytes,string) (ParticleToken.sol#190-206) is never used and should be removed
Base58.bytes32ToHexString(bytes32) (ParticleToken.sol#284-296) is never used and should be removed
Base58.encodeCID(bytes32) (ParticleToken.sol#249-282) is never used and should be removed
Base58.toByte(uint8) (ParticleToken.sol#298-303) is never used and should be removed
Context._msgData() (ParticleToken.sol#905-907) is never used and should be removed
ERC1155._burnBatch(address,uint256[],uint256[]) (ParticleToken.sol#785-808) is never used and should be removed
ERC1155._mintBatch(address,uint256[],uint256[],bytes) (ParticleToken.sol#725-745) is never used and should be removed
SafeMath.mul(uint256,uint256) (ParticleToken.sol#455-457) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.1 (ParticleToken.sol#2) allows old versions
solc-0.8.1 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Pragma version^0.8.1 (ParticleToken.sol#2) allows old versions
solc-0.8.1 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (ParticleToken.sol#59-64):
        - (success) = recipient.call{value: amount}() (ParticleToken.sol#62)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (ParticleToken.sol#123-132):
        - (success,returndata) = target.call{value: value}(data) (ParticleToken.sol#130)
Low level call in Address.functionStaticCall(address,bytes,string) (ParticleToken.sol#150-157):
        - (success,returndata) = target.staticcall(data) (ParticleToken.sol#155)
Low level call in Address.functionDelegateCall(address,bytes,string) (ParticleToken.sol#175-182):
        - (success,returndata) = target.delegatecall(data) (ParticleToken.sol#180)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter Base58.bytes32ToHexString(bytes32)._bytes32 (ParticleToken.sol#284) is not in mixedCase
Variable ERC1155._balances (ParticleToken.sol#494) is not in mixedCase
Variable ERC1155._operatorApprovals (ParticleToken.sol#497) is not in mixedCase
Parameter ParticleToken.bidParticle(uint256,uint256,uint256,uint256,bytes)._amount (ParticleToken.sol#1017) is not in mixedCas
e
Parameter ParticleToken.bidParticle(uint256,uint256,uint256,uint256,bytes)._startDate (ParticleToken.sol#1018) is not in mixed
Case
Parameter ParticleToken.bidParticle(uint256,uint256,uint256,uint256,bytes)._endDate (ParticleToken.sol#1019) is not in mixedCa
se
Parameter ParticleToken.bidParticle(uint256,uint256,uint256,uint256,bytes)._expiryDate (ParticleToken.sol#1020) is not in mixe
dCase
Parameter ParticleToken.increaseMaxSupply(uint256)._additionalSupply (ParticleToken.sol#1037) is not in mixedCase
Parameter ParticleToken.burn(address,uint256)._from (ParticleToken.sol#1045) is not in mixedCase
Parameter ParticleToken.burn(address,uint256)._amount (ParticleToken.sol#1045) is not in mixedCase
Constant ParticleToken.tokenId (ParticleToken.sol#979) is not in UPPER_CASE_WITH_UNDERSCORES
Variable ParticleToken._totalSupply (ParticleToken.sol#985) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
Reentrancy in ParticleToken.bidParticle(uint256,uint256,uint256,uint256,bytes) (ParticleToken.sol#1016-1035):
        External calls:
        - address(owner()).transfer(msg.value) (ParticleToken.sol#1033)
        State variables written after the call(s):
        - _mint(msg.sender,tokenId,_amount,) (ParticleToken.sol#1034)
                - _balances[id][to] = amount.add(_balances[id][to]) (ParticleToken.sol#706)
        Event emitted after the call(s):
        - TransferSingle(msg.sender,address(0),to,id,amount) (ParticleToken.sol#707)
                - _mint(msg.sender,tokenId,_amount,) (ParticleToken.sol#1034)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

ParticleToken.symbol (ParticleToken.sol#981) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
ParticleToken.sol analyzed (10 contracts with 84 detectors), 49 result(s) found
```

Solidity Static Analysis

BodyToken.sol

Check-effects-interaction:

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

more

Pos: 1163:2:

Inline assembly:

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

more

Pos: 587:12:

Low level calls:

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

more

Pos: 536:50:

Gas costs:

Gas requirement of function BodyToken.uri 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: 635:2:

Email: audit@EtherAuthority.io

For loop over dynamic array:

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

more

Pos: 13:4:

For loop over dynamic array:

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

more

Pos: 1278:6:

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: 1272:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.

more

Pos: 1120:8:

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: 1065:32:

CitizenToken.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in CitizenToken.(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 48:2:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in CitizenToken.mint(address,uint256[],uint256[]): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 146:2:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 287:4:

Gas costs:

Gas requirement of function CitizenToken.tokensByWallet 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: 76:2:

Gas costs:

Gas requirement of function CitizenToken.mint is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 146:2:

This on local calls:

Use of "this" for local functions: Never use "this" to call functions in the same contract, it only consumes more gas than normal local calls.
more
Pos: 267:11:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 78:4:

Constant/View/Pure functions:

CitizenToken.tokensByWallet(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 76:2:

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: 247:4:

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: 18:16:

GlyphToken.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in CitizenChildToken.(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1163:2:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 587:12:

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 536:50:

Gas costs:

Gas requirement of function GlyphToken.uri 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: 635:2:

Check-effects-interaction:

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

more

Pos: 1163:2:

Low level calls:

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

more

Pos: 536:50:

Gas costs:

Gas requirement of function GlyphToken.uri 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: 635:2:

Gas costs:

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

Pos: 1244:2:

For loop over dynamic array:

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

more

Pos: 845:4:

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: 1272:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 1120:8:

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: 1065:32:

ParticleToken.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ParticleToken.bidParticle(uint256,uint256,uint256,uint256,bytes): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 49:2:

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: 59:12:

Gas costs:

Gas requirement of function ParticleToken.uri 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: 48:2:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 82:4:

Constant/View/Pure functions:

ParticleToken.safeTransferFrom(address,address,uint256,uint256,bytes) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 84:2:

Constant/View/Pure functions:

ParticleToken.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 94:2:

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: 14:4:

Solhint Linter

BodyToken.sol

```
Compiler version ^0.8.1 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:2
global import of path CitizenChildToken.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:4
Use double quotes for string literals
Pos: 8:4
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 3:7
Use double quotes for string literals
Pos: 76:13
Variable "_uri" is unused
Pos: 5:8
```

CitizenToken.sol

```
Compiler version ^0.8.1 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path IParticle.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:3
Use double quotes for string literals
Pos: 8:3
global import of path ICitizenChildToken.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:4
Use double quotes for string literals
Pos: 8:4
global import of path ERC998ERC1155TopDown.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:5
Use double quotes for string literals
Pos: 8:5
global import of path
@openzeppelin/contracts/token/common/ERC2981.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:7
Use double quotes for string literals
Pos: 8:7
```

```
global import of path
@openzeppelin/contracts/utils/structs/EnumerableSet.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
Pos: 1:8
Use double quotes for string literals
Pos: 8:8
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 3:32
Code contains empty blocks
Pos: 73:99
Error message for require is too long
Pos: 5:113
Use double quotes for string literals
Pos: 55:113
Error message for require is too long
Pos: 5:114
Use double quotes for string literals
Pos: 41:114
Use double quotes for string literals
Pos: 29:115
Use double quotes for string literals
Pos: 61:134
Error message for require is too long
Pos: 5:135
Use double quotes for string literals
Pos: 81:157
Use double quotes for string literals
Pos: 40:159
Use double quotes for string literals
Pos: 44:161
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 5:189
Use double quotes for string literals
Pos: 52:200
Use double quotes for string literals
Pos: 52:234
```

GlyphToken.sol

```
Compiler version ^0.8.1 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:2
global import of path CitizenChildToken.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:4
Use double quotes for string literals
Pos: 8:4
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 3:7
Use double quotes for string literals
Pos: 76:13
```

```
Variable "_uri" is unused
Pos: 5:8
```

ParticleToken.sol

```
Compiler version ^0.8.1 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path SafeMath.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:3
Use double quotes for string literals
Pos: 8:3
global import of path ERC1155.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:5
Use double quotes for string literals
Pos: 8:5
global import of path @openzeppelin/contracts/access/Ownable.sol is
not allowed. Specify names to import individually or bind all exports
of the module into a name (import "path" as Name)
Pos: 1:6
Use double quotes for string literals
Pos: 8:6
Constant name must be in capitalized SNAKE_CASE
Pos: 3:11
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 3:25
Use double quotes for string literals
Pos: 41:56
Avoid making time-based decisions in your business logic
Pos: 13:58
Use double quotes for string literals
Pos: 45:58
Use double quotes for string literals
Pos: 41:66
Variable "_signature" is unused
Pos: 5:53
Variable "payloadHash" is unused
Pos: 5:61
Error message for require is too long
Pos: 5:78
Use double quotes for string literals
Pos: 49:78
```

Software analysis result:

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

Email: audit@EtherAuthority.io