Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	WP Smart Contracts
Website:	wpsmartcontracts.com
Platform:	Ethereum
Language:	Solidity
Date:	June 14th, 2022

Table of contents

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

Introduction

EtherAuthority was contracted by the WP Smart Contracts team to perform the Security audit of the 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 June 14th, 2022.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

The WP Smart Contracts provides the smart contract solutions to the wordpress users. They develop various WP plugins which lets WP websites use the smart contract deployment quickly. We audited their Azuki(ERC1155) and Ikasumi(ERC1155), YuzuFlattened(ERC1155) smart contracts.

Audit scope

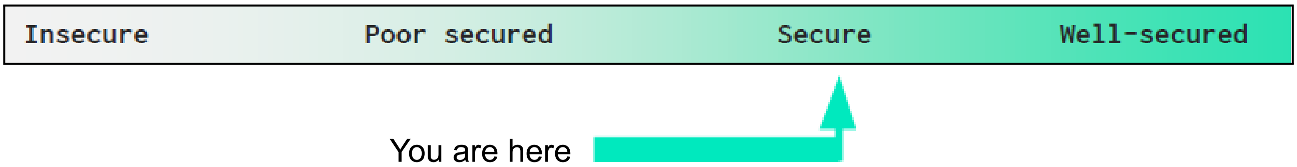
Name	Code Review and Security Analysis Report for WP Smart Contracts Protocol Smart Contracts
Platform	Ethereum / Solidity
File 1	Azuki.sol
File 1 MD5 Hash	93C8EDF0E49792E16DBBB875CD6129D9
Updated File 1 MD5 Hash	2FEE78B06749BFB03531E7BAA6543FDE
File 2	Ikasumi.sol
File 2 MD5 Hash	F49BC49A57F047FA20098CFFDC13B439
Updated File 2 MD5 Hash	BDBDDB3E992B94A0C1C7D80CB5BEFE8B
File 3	YuzuFlattened.sol
File 3 MD5 Hash	2B3052B1658BA3D9CDA293D2319773B1
Audit Date	June 14th, 2022
Revise Audit Date	June 17th, 2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 Azuki.sol - Signature Version: 1 - Signing Domain: ERC1155Azuki-Voucher	**YES, This is valid.**
File 2 Ikasumi.sol - Signature Version: 1 - Signing Domain: ERC1155Ikasumi-Voucher	**YES, This is valid.**
File 3 YuzuFlattened.sol - YuzuFlattened has functions like: autoMint, mint, etc.	**YES, This is valid.**

Audit Summary

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



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.

All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

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

All the issues have been resolved / acknowledged in the revised code.

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

Technical Quick Stats

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

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 3 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

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

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

Some code parts are not well commented on smart contracts. We suggest using Ethereum's NatSpec style for the commenting.

Documentation

We were given a WP Smart Contracts Protocol smart contract code in the form of an Etherscan 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.

Another source of information was its official website https://wpsmartcontracts.com which provided rich information about the project architecture and tokenomics.

Use of Dependencies

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

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

AS-IS overview

Azuki.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_domainSeparatorV4	internal	Passed	No Issue
3	_buildDomainSeparator	read	Passed	No Issue
4	_hashTypedDataV4	internal	Passed	No Issue
5	sell	write	Passed	No Issue
6	returnTheChange	internal	Passed	No Issue
7	buy	write	Passed	No Issue
8	cancelSale	external	Passed	No Issue
9	bid	external	Passed	No Issue
10	cancelBid	external	Passed	No Issue
11	acceptBid	external	Passed	No Issue
12	distributeFunds	write	Passed	No Issue
13	_mint	internal	Passed	No Issue
14	_mintBatch	internal	Passed	No Issue
15	callOptionalReturn	write	Passed	No Issue
16	updateAdmin	external	Passed	No Issue
17	updateOwner	external	Passed	No Issue
18	redeem	write	Passed	No Issue
19	domainSeparator	external	Passed	No Issue
20	verify	internal	Passed	No Issue
21	getChainId	read	Passed	No Issue

Ikasumi.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_domainSeparatorV4	internal	Passed	No Issue
3	_buildDomainSeparator	read	Passed	No Issue
4	_hashTypedDataV4	internal	Passed	No Issue
5	sell	write	Passed	No Issue
6	returnTheChange	internal	Passed	No Issue
7	buy	write	Passed	No Issue
8	cancelSale	external	Passed	No Issue
9	bid	external	Passed	No Issue
10	cancelBid	external	Passed	No Issue
11	acceptBid	external	Passed	No Issue
12	distributeFunds	write	Passed	No Issue
13	_mint	internal	Passed	No Issue
14	_mintBatch	internal	Passed	No Issue

15	callOptionalReturn	write	Passed	No Issue
16	updateAdmin	external	Passed	No Issue
17	updateOwner	external	Passed	No Issue
18	redeem	write	Passed	No Issue
19	domainSeparator	external	Passed	No Issue
20	verify	internal	Passed	No Issue
21	getChainId	read	Passed	No Issue

YuzuFlattened.sol

Functions

Sl.	**Functions**	**Type**	**Observation**	**Conclusion**
1	constructor	write	Passed	No Issue
2	onlyRole	modifier	Passed	No Issue
3	supportsInterface	read	Passed	No Issue
4	hasRole	read	Passed	No Issue
5	_checkRole	internal	Passed	No Issue
6	getRoleAdmin	read	Passed	No Issue
7	grantRole	write	access only Role	No Issue
8	revokeRole	write	access only Role	No Issue
9	renounceRole	write	Passed	No Issue
10	_setupRole	internal	Passed	No Issue
11	_setRoleAdmin	internal	Passed	No Issue
12	_grantRole	write	Passed	No Issue
13	_revokeRole	write	Passed	No Issue
14	totalSupply	read	Passed	No Issue
15	exists	read	Passed	No Issue
16	_mint	internal	Passed	No Issue
17	_mintBatch	internal	Passed	No Issue
18	name	read	Passed	No Issue
19	symbol	read	Passed	No Issue
20	autoMint	write	access only Minter	No Issue
21	autoMintBatch	write	Infinite loops possibility	Refer Audit Findings
22	mint	write	access only Minter	No Issue
23	mintBatch	write	Infinite loops possibility	Refer Audit Findings
24	supportsInterface	read	Passed	No Issue
25	addMinter	write	access only Minter	No Issue
26	isMinter	read	Passed	No Issue
27	canIMint	read	Passed	No Issue
28	onlyMinter	modifier	Passed	No Issue
29	supportsInterface	read	Passed	No Issue
30	uri	read	Passed	No Issue
31	balanceOf	read	Passed	No Issue
32	balanceOfBatch	read	Infinite loops possibility	Refer Audit Findings

33	setApprovalForAll	write	Passed	No Issue
34	isApprovedForAll	read	Passed	No Issue
35	safeTransferFrom	write	Passed	No Issue
36	safeBatchTransferFrom	write	Infinite loops possibility	Refer Audit Findings
37	_safeTransferFrom	internal	Passed	No Issue
38	_safeBatchTransferFrom	internal	Passed	No Issue
39	_setURI	internal	Passed	No Issue
40	_mint	internal	Passed	No Issue
41	_mintBatch	internal	Passed	No Issue
42	_doSafeTransferAcceptanceCheck	write	Passed	No Issue
43	_doSafeBatchTransferAcceptanceCheck	write	Passed	No Issue
44	asSingletonArray	write	Passed	No Issue

Severity Definitions

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

Audit Findings

Critical Severity

No critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

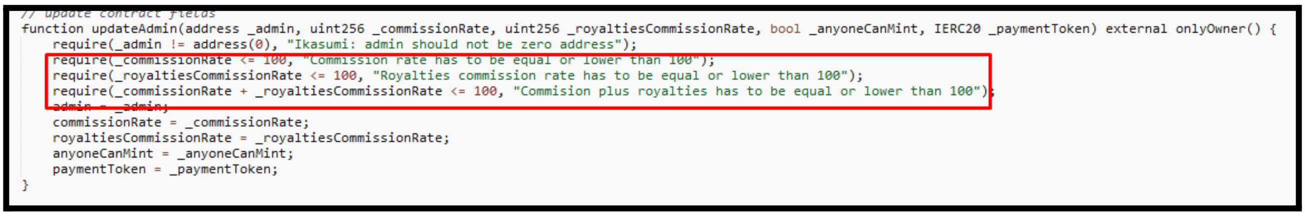
(1) Item creator can bid/buy his own item: **Ikasumi.sol, Azuki.sol**

Item creator can bid/buy his own item. This is meaningless.

Resolution: We suggest not allowing the item creator to bid/buy his own item. If this is a part of the plan then disregard this issue.

Status: Fixed

(2) Commission and Royalty rate can be 100%: **Ikasumi.sol, Azuki.sol**



```
// update contract fields
function updateAdmin(address _admin, uint256 _commissionRate, uint256 _royaltiesCommissionRate, bool _anyoneCanMint, IERC20 _paymentToken) external onlyOwner() {
    require(_admin != address(0), "Ikasumi: admin should not be zero address");
    require(_commissionRate <= 100, "Commission rate has to be equal or lower than 100");
    require(_royaltiesCommissionRate <= 100, "Royalties commission rate has to be equal or lower than 100");
    require(_commissionRate + _royaltiesCommissionRate <= 100, "Commision plus royalties has to be equal or lower than 100");
    admin = _admin;
    commissionRate = _commissionRate;
    royaltiesCommissionRate = _royaltiesCommissionRate;
    anyoneCanMint = _anyoneCanMint;
    paymentToken = _paymentToken;
}
```

The owner can set commission and royalty rate to 100%. Hence the bid owner gets 0 as payment.

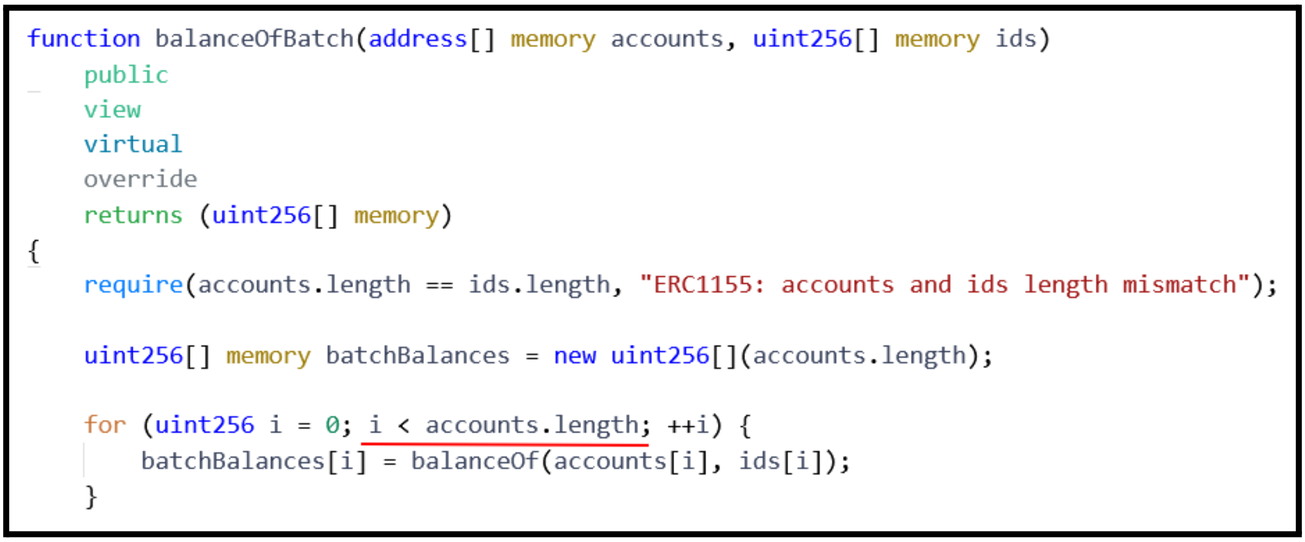
Resolution: We suggest setting some range below than 100% so that the bid owner will get some token as payment for sure.

Status: Fixed

Low

(1) Infinite loops possibility - **YuzuFlattened.sol**

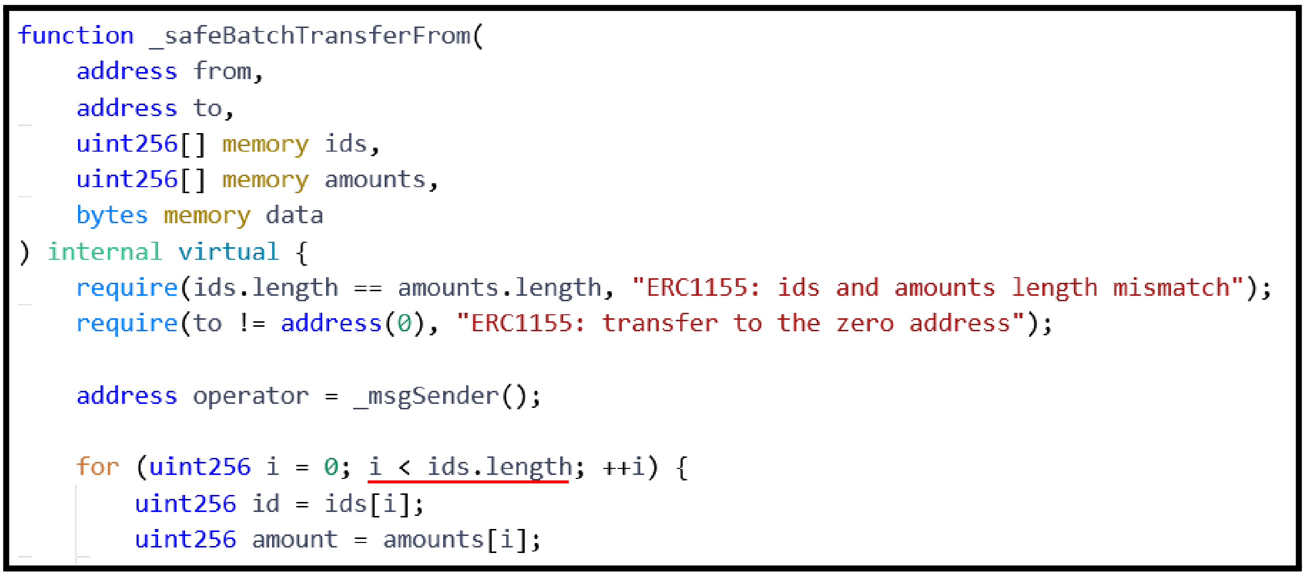
As array elements will increase, then it will cost more and more gas. And eventually, it will stop all the functionality. After several hundreds of transactions, all those functions depending on it will stop. We suggest avoiding loops. For example, use mapping to store the array index. And query that data directly, instead of looping through all the elements to find an element.



```
function balanceOfBatch(address[] memory accounts, uint256[] memory ids)
    public
    view
    virtual
    override
    returns (uint256[] memory)
{
    require(accounts.length == ids.length, "ERC1155: accounts and ids length mismatch");

    uint256[] memory batchBalances = new uint256[](accounts.length);

    for (uint256 i = 0; i < accounts.length; ++i) {
        batchBalances[i] = balanceOf(accounts[i], ids[i]);
    }
```



```
function _safeBatchTransferFrom(
    address from,
    address to,
    uint256[] memory ids,
    uint256[] memory amounts,
    bytes memory data
) internal virtual {
    require(ids.length == amounts.length, "ERC1155: ids and amounts length mismatch");
    require(to != address(0), "ERC1155: transfer to the zero address");

    address operator = _msgSender();

    for (uint256 i = 0; i < ids.length; ++i) {
        uint256 id = ids[i];
        uint256 amount = amounts[i];
```

Other owner functions are:

- mintBatch() -> _mintBatch() - ids.length.
- autoMintBatch() - amounts.length

Resolution: Adjust logic to replace loops with mapping or other code structure or validate for some length of array only.

Status: Acknowledged

(2) Critical operation lacks event log: **Ikasumi.sol, Azuki.sol**

Missing event log for:

- cancelSale

- cancelBid

Resolution: Write an event log for listed events.
Status: Fixed

Very Low / Informational / Best practices:

No Very Low severity vulnerabilities were found.

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:

- autoMint: YuzuFlattened minter can automatically mint tokens from an account.
- autoMintBatch: YuzuFlattened minter can automatically mint tokens from an account batch vise.
- mint: YuzuFlattened minter can mint a token.
- mintBatch: YuzuFlattened minter can mint a token batch vise.
- addMinter: YuzuFlattened minter can add minter address.

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 Rinkeby Etherscan weblink. And we have used all possible tests based on given objects as files. We had observed some issues in the smart contracts, and those issues have been resolved / acknowledged in the revised code. **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.

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

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

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

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.

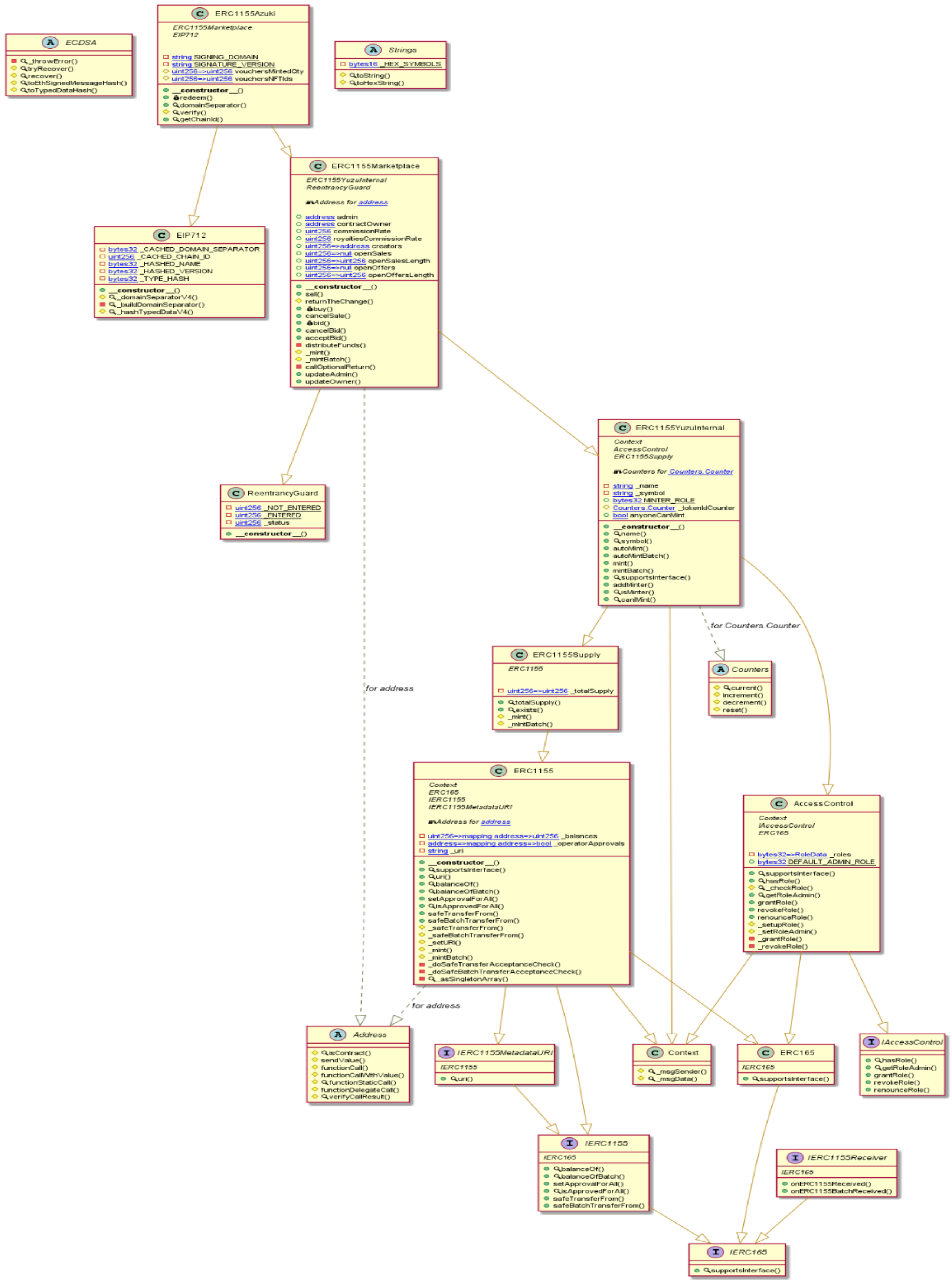
Technical Disclaimer

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.

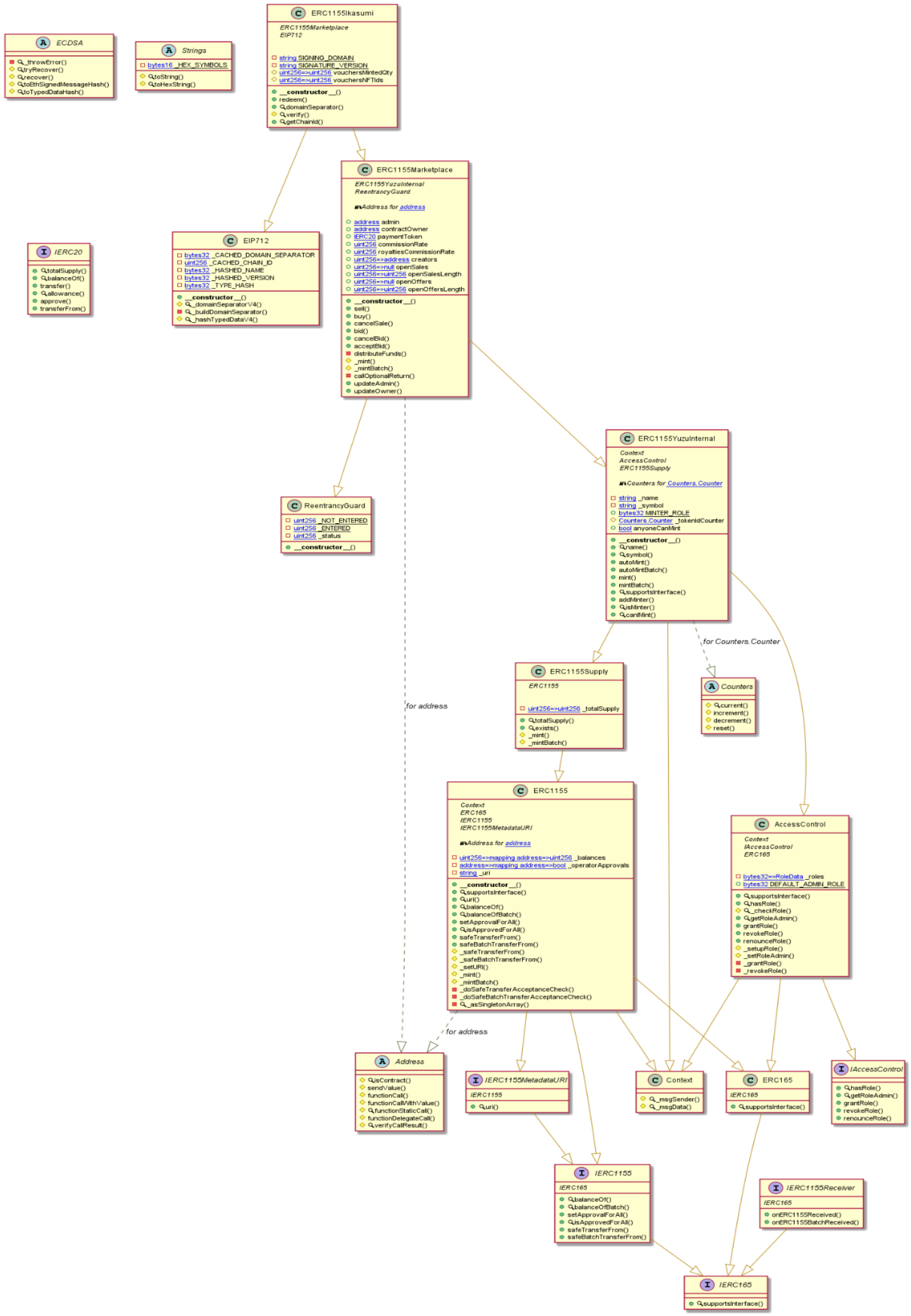
Appendix

Code Flow Diagram - WP Smart Contracts Protocol

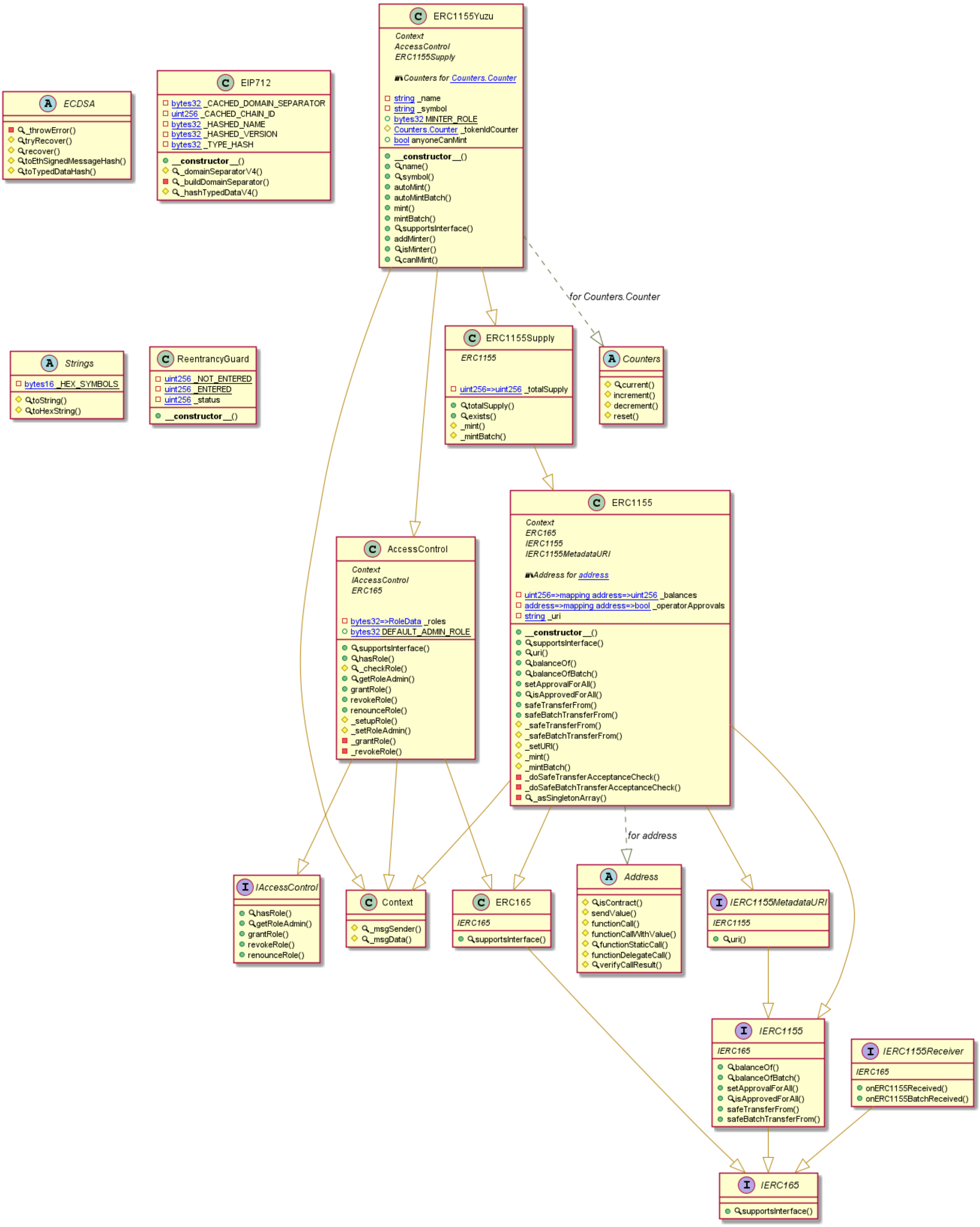
Azuki Diagram



Ikasumi Diagram

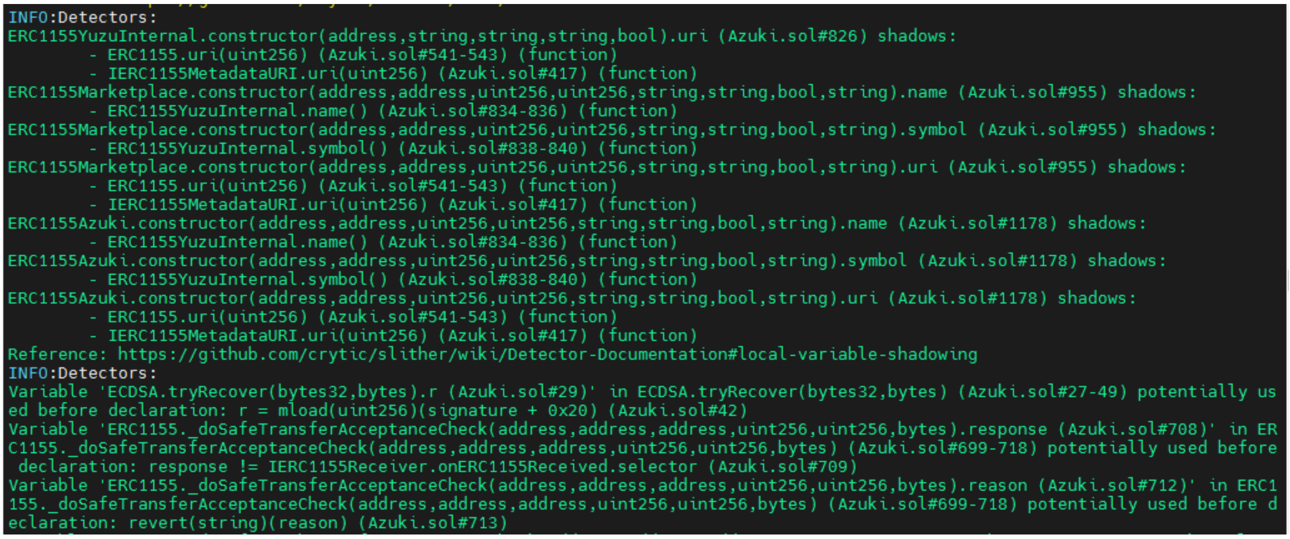


YuzuFlattened Diagram

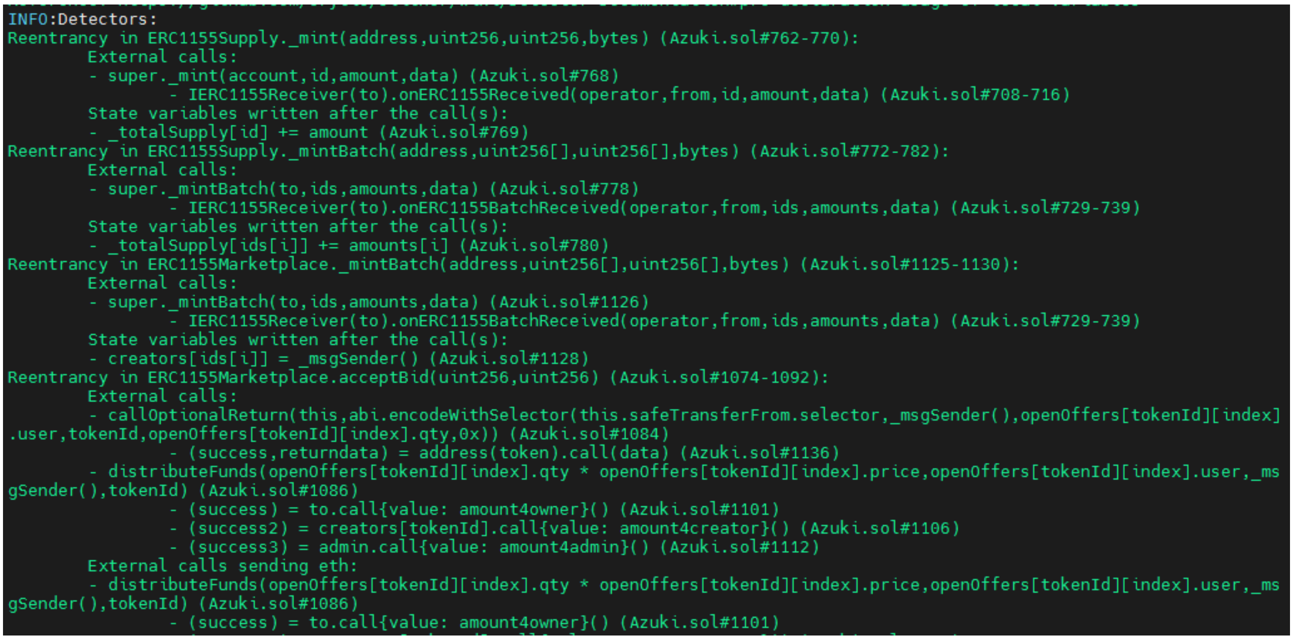


Slither Results Log

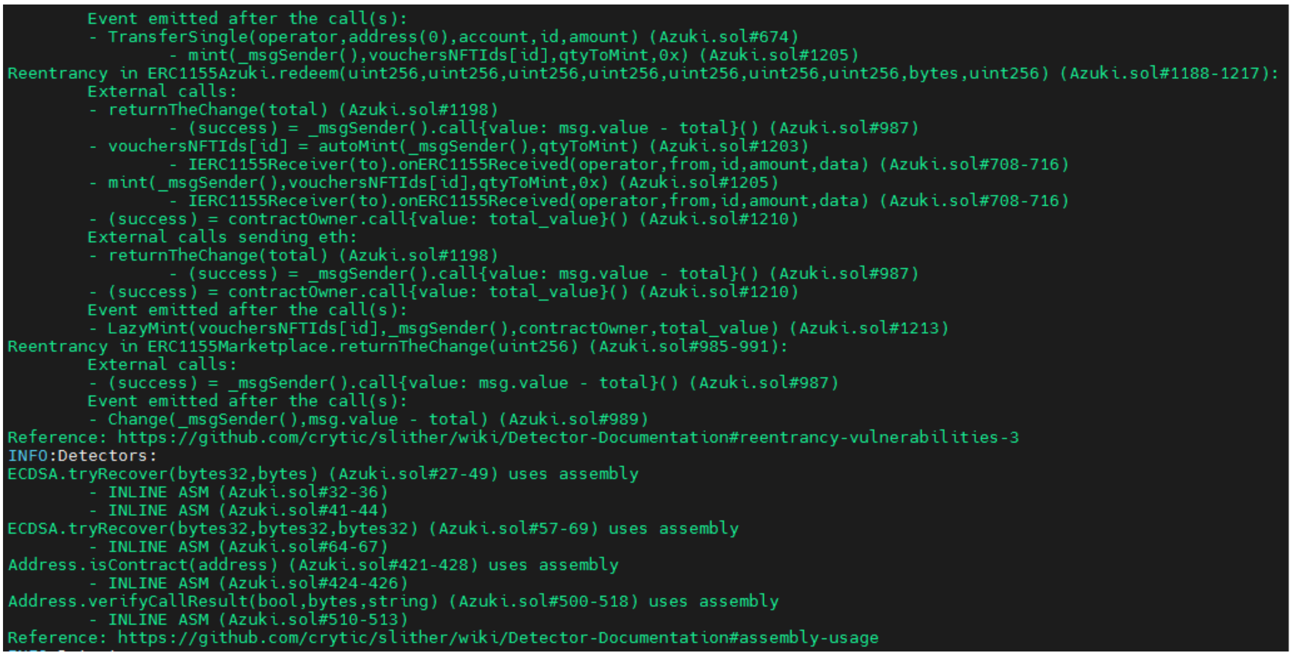
Slither log >> Azuki.sol



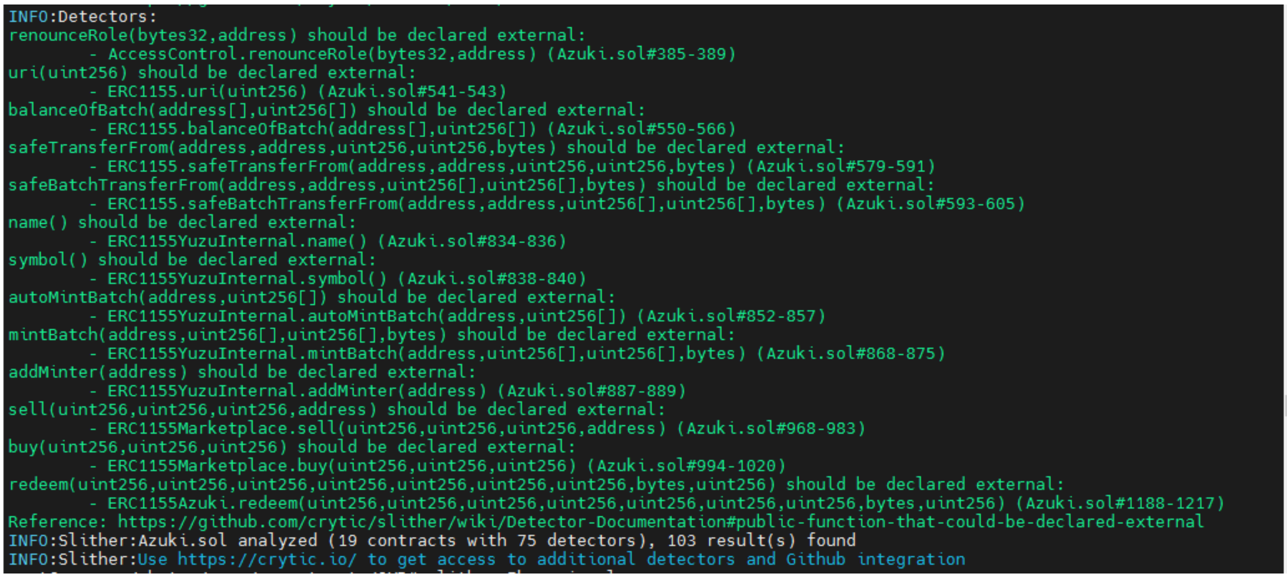
```
INFO:Detectors:
ERC1155YuzuInternal.constructor(address,string,string,string,bool).uri (Azuki.sol#826) shadows:
        - ERC1155.uri(uint256) (Azuki.sol#541-543) (function)
        - IERC1155MetadataURI.uri(uint256) (Azuki.sol#417) (function)
ERC1155Marketplace.constructor(address,address,uint256,uint256,string,string,bool,string).name (Azuki.sol#955) shadows:
        - ERC1155YuzuInternal.name() (Azuki.sol#834-836) (function)
ERC1155Marketplace.constructor(address,address,uint256,uint256,string,string,bool,string).symbol (Azuki.sol#955) shadows:
        - ERC1155YuzuInternal.symbol() (Azuki.sol#838-840) (function)
ERC1155Marketplace.constructor(address,address,uint256,uint256,string,string,bool,string).uri (Azuki.sol#955) shadows:
        - ERC1155.uri(uint256) (Azuki.sol#541-543) (function)
        - IERC1155MetadataURI.uri(uint256) (Azuki.sol#417) (function)
ERC1155Azuki.constructor(address,address,uint256,uint256,string,string,bool,string).name (Azuki.sol#1178) shadows:
        - ERC1155YuzuInternal.name() (Azuki.sol#834-836) (function)
ERC1155Azuki.constructor(address,address,uint256,uint256,string,string,bool,string).symbol (Azuki.sol#1178) shadows:
        - ERC1155YuzuInternal.symbol() (Azuki.sol#838-840) (function)
ERC1155Azuki.constructor(address,address,uint256,uint256,string,string,bool,string).uri (Azuki.sol#1178) shadows:
        - ERC1155.uri(uint256) (Azuki.sol#541-543) (function)
        - IERC1155MetadataURI.uri(uint256) (Azuki.sol#417) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Variable 'ECDSA.tryRecover(bytes32,bytes).r (Azuki.sol#29)' in ECDSA.tryRecover(bytes32,bytes) (Azuki.sol#27-49) potentially us
ed before declaration: r = mload(uint256)(signature + 0x20) (Azuki.sol#42)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (Azuki.sol#708)' in ER
C1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (Azuki.sol#699-718) potentially used before
 declaration: response != IERC1155Receiver.onERC1155Received.selector (Azuki.sol#709)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (Azuki.sol#712)' in ERC1
155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (Azuki.sol#699-718) potentially used before d
eclaration: revert(string)(reason) (Azuki.sol#713)
```



```
INFO:Detectors:
Reentrancy in ERC1155Supply._mint(address,uint256,uint256,bytes) (Azuki.sol#762-770):
        External calls:
        - super._mint(account,id,amount,data) (Azuki.sol#768)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (Azuki.sol#708-716)
        State variables written after the call(s):
        - _totalSupply[id] += amount (Azuki.sol#769)
Reentrancy in ERC1155Supply._mintBatch(address,uint256[],uint256[],bytes) (Azuki.sol#772-782):
        External calls:
        - super._mintBatch(to,ids,amounts,data) (Azuki.sol#778)
                - IERC1155Receiver(to).onERC1155BatchReceived(operator,from,ids,amounts,data) (Azuki.sol#729-739)
        State variables written after the call(s):
        - _totalSupply[ids[i]] += amounts[i] (Azuki.sol#780)
Reentrancy in ERC1155Marketplace._mintBatch(address,uint256[],uint256[],bytes) (Azuki.sol#1125-1130):
        External calls:
        - super._mintBatch(to,ids,amounts,data) (Azuki.sol#1126)
                - IERC1155Receiver(to).onERC1155BatchReceived(operator,from,ids,amounts,data) (Azuki.sol#729-739)
        State variables written after the call(s):
        - creators[ids[i]] = _msgSender() (Azuki.sol#1128)
Reentrancy in ERC1155Marketplace.acceptBid(uint256,uint256) (Azuki.sol#1074-1092):
        External calls:
        - callOptionalReturn(this,abi.encodeWithSelector(this.safeTransferFrom.selector,_msgSender(),openOffers[tokenId][index]
.user,tokenId,openOffers[tokenId][index].qty,0x)) (Azuki.sol#1084)
                - (success,returndata) = address(token).call(data) (Azuki.sol#1136)
        - distributeFunds(openOffers[tokenId][index].qty * openOffers[tokenId][index].price,openOffers[tokenId][index].user,_ms
gSender(),tokenId) (Azuki.sol#1086)
                - (success) = to.call{value: amount4owner}() (Azuki.sol#1101)
                - (success2) = creators[tokenId].call{value: amount4creator}() (Azuki.sol#1106)
                - (success3) = admin.call{value: amount4admin}() (Azuki.sol#1112)
        External calls sending eth:
        - distributeFunds(openOffers[tokenId][index].qty * openOffers[tokenId][index].price,openOffers[tokenId][index].user,_ms
gSender(),tokenId) (Azuki.sol#1086)
                - (success) = to.call{value: amount4owner}() (Azuki.sol#1101)
```

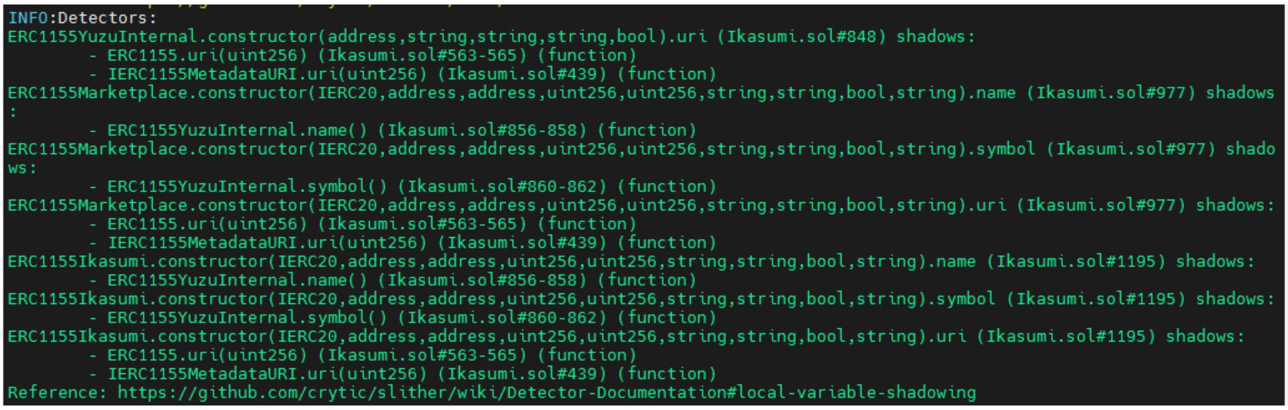


```
        Event emitted after the call(s):
        - TransferSingle(operator,address(0),account,id,amount) (Azuki.sol#674)
                - mint(_msgSender(),vouchersNFTIds[id],qtyToMint,0x) (Azuki.sol#1205)
Reentrancy in ERC1155Azuki.redeem(uint256,uint256,uint256,uint256,uint256,uint256,uint256,bytes,uint256) (Azuki.sol#1188-1217):
        External calls:
        - returnTheChange(total) (Azuki.sol#1198)
                - (success) = _msgSender().call{value: msg.value - total}() (Azuki.sol#987)
        - vouchersNFTIds[id] = autoMint(_msgSender(),qtyToMint) (Azuki.sol#1203)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (Azuki.sol#708-716)
        - mint(_msgSender(),vouchersNFTIds[id],qtyToMint,0x) (Azuki.sol#1205)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (Azuki.sol#708-716)
        - (success) = contractOwner.call{value: total_value}() (Azuki.sol#1210)
        External calls sending eth:
        - returnTheChange(total) (Azuki.sol#1198)
                - (success) = _msgSender().call{value: msg.value - total}() (Azuki.sol#987)
        - (success) = contractOwner.call{value: total_value}() (Azuki.sol#1210)
        Event emitted after the call(s):
        - LazyMint(vouchersNFTIds[id],_msgSender(),contractOwner,total_value) (Azuki.sol#1213)
Reentrancy in ERC1155Marketplace.returnTheChange(uint256) (Azuki.sol#985-991):
        External calls:
        - (success) = _msgSender().call{value: msg.value - total}() (Azuki.sol#987)
        Event emitted after the call(s):
        - Change(_msgSender(),msg.value - total) (Azuki.sol#989)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ECDSA.tryRecover(bytes32,bytes) (Azuki.sol#27-49) uses assembly
        - INLINE ASM (Azuki.sol#32-36)
        - INLINE ASM (Azuki.sol#41-44)
ECDSA.tryRecover(bytes32,bytes32,bytes32) (Azuki.sol#57-69) uses assembly
        - INLINE ASM (Azuki.sol#64-67)
Address.isContract(address) (Azuki.sol#421-428) uses assembly
        - INLINE ASM (Azuki.sol#424-426)
Address.verifyCallResult(bool,bytes,string) (Azuki.sol#500-518) uses assembly
        - INLINE ASM (Azuki.sol#510-513)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

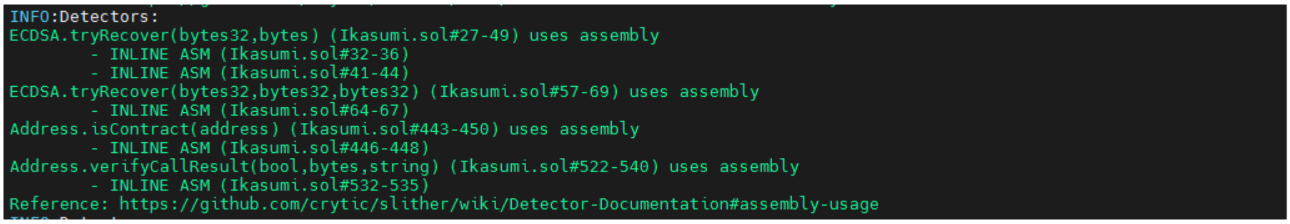


```
INFO:Detectors:
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (Azuki.sol#385-389)
uri(uint256) should be declared external:
        - ERC1155.uri(uint256) (Azuki.sol#541-543)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155.balanceOfBatch(address[],uint256[]) (Azuki.sol#550-566)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (Azuki.sol#579-591)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (Azuki.sol#593-605)
name() should be declared external:
        - ERC1155YuzuInternal.name() (Azuki.sol#834-836)
symbol() should be declared external:
        - ERC1155YuzuInternal.symbol() (Azuki.sol#838-840)
autoMintBatch(address,uint256[]) should be declared external:
        - ERC1155YuzuInternal.autoMintBatch(address,uint256[]) (Azuki.sol#852-857)
mintBatch(address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155YuzuInternal.mintBatch(address,uint256[],uint256[],bytes) (Azuki.sol#868-875)
addMinter(address) should be declared external:
        - ERC1155YuzuInternal.addMinter(address) (Azuki.sol#887-889)
sell(uint256,uint256,uint256,address) should be declared external:
        - ERC1155Marketplace.sell(uint256,uint256,uint256,address) (Azuki.sol#968-983)
buy(uint256,uint256,uint256) should be declared external:
        - ERC1155Marketplace.buy(uint256,uint256,uint256) (Azuki.sol#994-1020)
redeem(uint256,uint256,uint256,uint256,uint256,uint256,uint256,bytes,uint256) should be declared external:
        - ERC1155Azuki.redeem(uint256,uint256,uint256,uint256,uint256,uint256,uint256,bytes,uint256) (Azuki.sol#1188-1217)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Azuki.sol analyzed (19 contracts with 75 detectors), 103 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

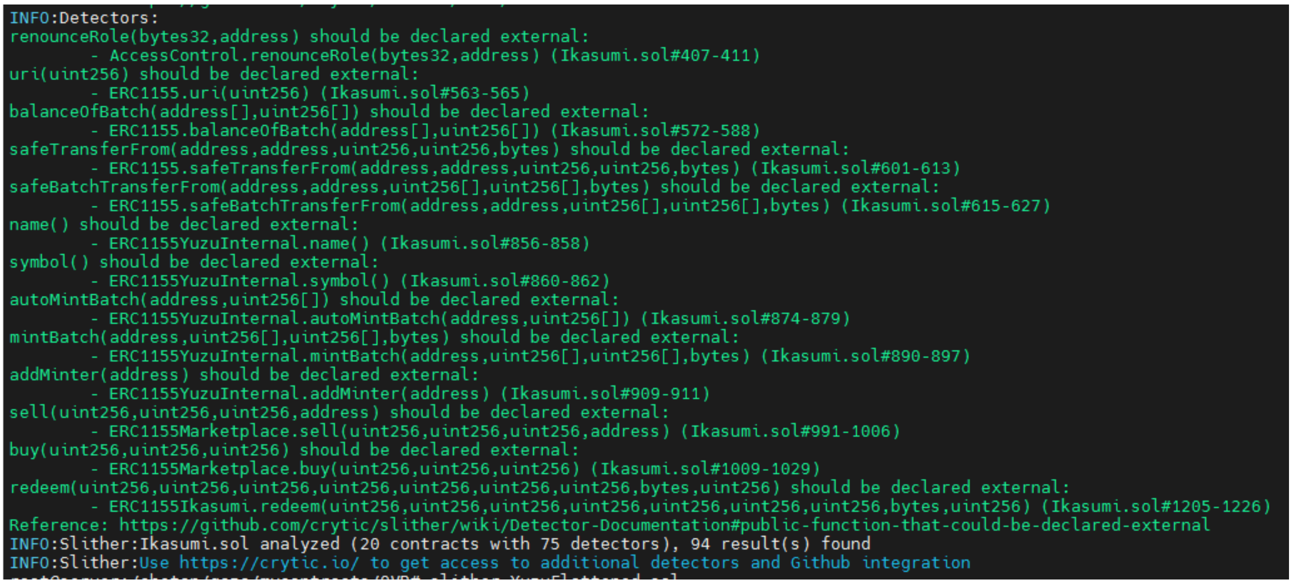
Slither log >> Ikasumi.sol



```
INFO:Detectors:
ERC1155YuzuInternal.constructor(address,string,string,string,bool).uri (Ikasumi.sol#848) shadows:
        - ERC1155.uri(uint256) (Ikasumi.sol#563-565) (function)
        - IERC1155MetadataURI.uri(uint256) (Ikasumi.sol#439) (function)
ERC1155Marketplace.constructor(IERC20,address,address,uint256,uint256,string,string,bool,string).name (Ikasumi.sol#977) shadows
:
        - ERC1155YuzuInternal.name() (Ikasumi.sol#856-858) (function)
ERC1155Marketplace.constructor(IERC20,address,address,uint256,uint256,string,string,bool,string).symbol (Ikasumi.sol#977) shado
ws:
        - ERC1155YuzuInternal.symbol() (Ikasumi.sol#860-862) (function)
ERC1155Marketplace.constructor(IERC20,address,address,uint256,uint256,string,string,bool,string).uri (Ikasumi.sol#977) shadows:
        - ERC1155.uri(uint256) (Ikasumi.sol#563-565) (function)
        - IERC1155MetadataURI.uri(uint256) (Ikasumi.sol#439) (function)
ERC1155Ikasumi.constructor(IERC20,address,address,uint256,uint256,string,string,bool,string).name (Ikasumi.sol#1195) shadows:
        - ERC1155YuzuInternal.name() (Ikasumi.sol#856-858) (function)
ERC1155Ikasumi.constructor(IERC20,address,address,uint256,uint256,string,string,bool,string).symbol (Ikasumi.sol#1195) shadows:
        - ERC1155YuzuInternal.symbol() (Ikasumi.sol#860-862) (function)
ERC1155Ikasumi.constructor(IERC20,address,address,uint256,uint256,string,string,bool,string).uri (Ikasumi.sol#1195) shadows:
        - ERC1155.uri(uint256) (Ikasumi.sol#563-565) (function)
        - IERC1155MetadataURI.uri(uint256) (Ikasumi.sol#439) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

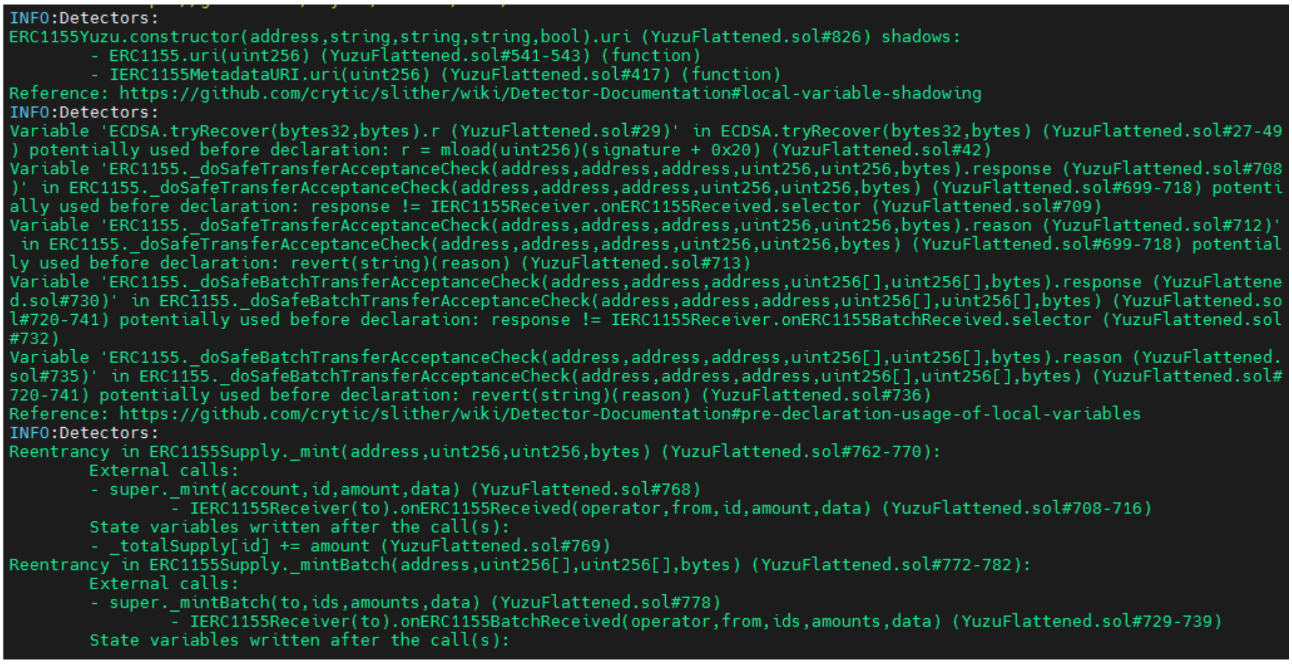


```
INFO:Detectors:
ECDSA.tryRecover(bytes32,bytes) (Ikasumi.sol#27-49) uses assembly
        - INLINE ASM (Ikasumi.sol#32-36)
        - INLINE ASM (Ikasumi.sol#41-44)
ECDSA.tryRecover(bytes32,bytes32,bytes32) (Ikasumi.sol#57-69) uses assembly
        - INLINE ASM (Ikasumi.sol#64-67)
Address.isContract(address) (Ikasumi.sol#443-450) uses assembly
        - INLINE ASM (Ikasumi.sol#446-448)
Address.verifyCallResult(bool,bytes,string) (Ikasumi.sol#522-540) uses assembly
        - INLINE ASM (Ikasumi.sol#532-535)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

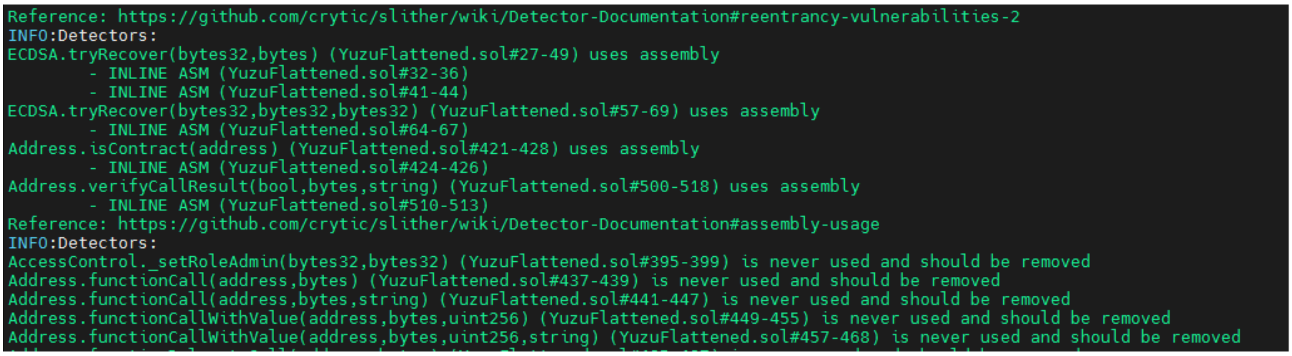


```
INFO:Detectors:
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (Ikasumi.sol#407-411)
uri(uint256) should be declared external:
        - ERC1155.uri(uint256) (Ikasumi.sol#563-565)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155.balanceOfBatch(address[],uint256[]) (Ikasumi.sol#572-588)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (Ikasumi.sol#601-613)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (Ikasumi.sol#615-627)
name() should be declared external:
        - ERC1155YuzuInternal.name() (Ikasumi.sol#856-858)
symbol() should be declared external:
        - ERC1155YuzuInternal.symbol() (Ikasumi.sol#860-862)
autoMintBatch(address,uint256[]) should be declared external:
        - ERC1155YuzuInternal.autoMintBatch(address,uint256[]) (Ikasumi.sol#874-879)
mintBatch(address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155YuzuInternal.mintBatch(address,uint256[],uint256[],bytes) (Ikasumi.sol#890-897)
addMinter(address) should be declared external:
        - ERC1155YuzuInternal.addMinter(address) (Ikasumi.sol#909-911)
sell(uint256,uint256,uint256,address) should be declared external:
        - ERC1155Marketplace.sell(uint256,uint256,uint256,address) (Ikasumi.sol#991-1006)
buy(uint256,uint256,uint256) should be declared external:
        - ERC1155Marketplace.buy(uint256,uint256,uint256) (Ikasumi.sol#1009-1029)
redeem(uint256,uint256,uint256,uint256,uint256,uint256,uint256,bytes,uint256) should be declared external:
        - ERC1155Ikasumi.redeem(uint256,uint256,uint256,uint256,uint256,uint256,uint256,bytes,uint256) (Ikasumi.sol#1205-1226)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Ikasumi.sol analyzed (20 contracts with 75 detectors), 94 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

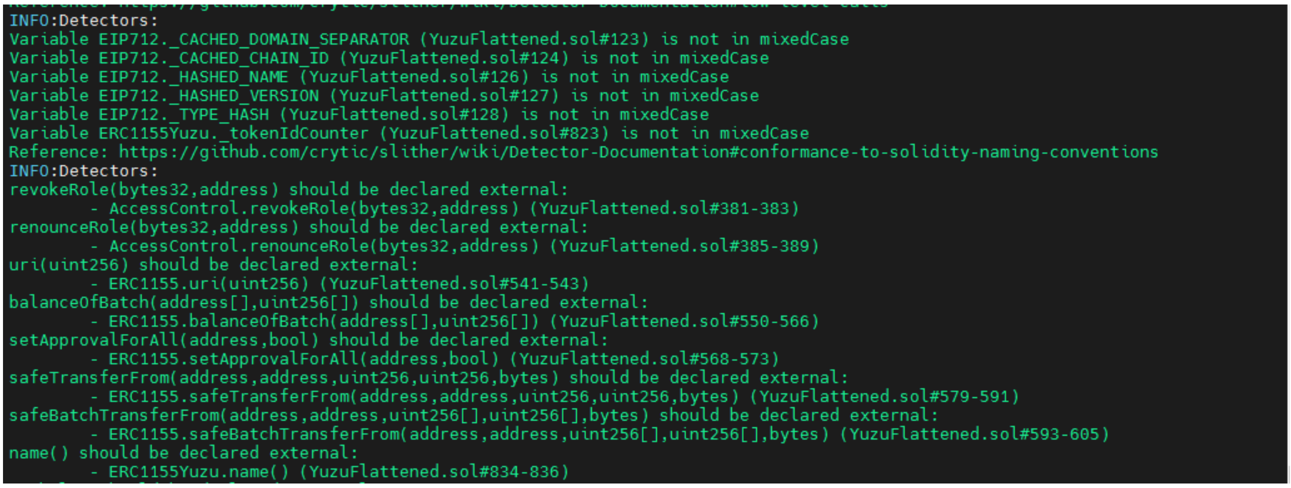
Slither log >> YuzuFlattened.sol



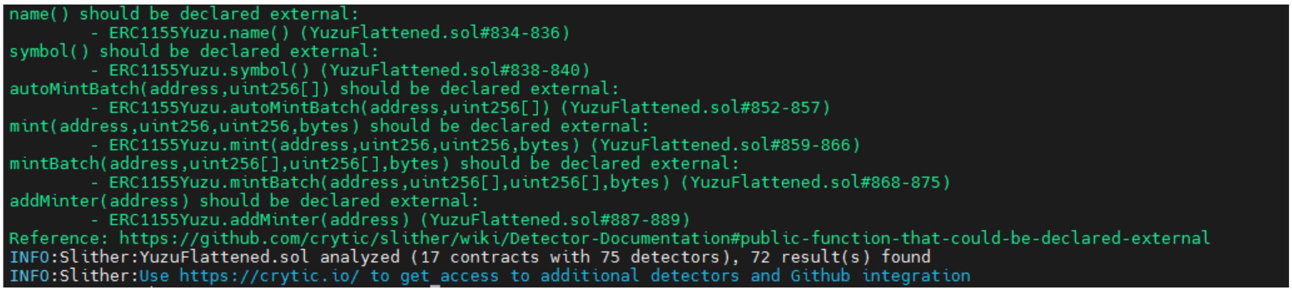
```
INFO:Detectors:
ERC1155Yuzu.constructor(address,string,string,string,bool).uri (YuzuFlattened.sol#826) shadows:
        - ERC1155.uri(uint256) (YuzuFlattened.sol#541-543) (function)
        - IERC1155MetadataURI.uri(uint256) (YuzuFlattened.sol#417) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Variable 'ECDSA.tryRecover(bytes32,bytes).r (YuzuFlattened.sol#29)' in ECDSA.tryRecover(bytes32,bytes) (YuzuFlattened.sol#27-49
) potentially used before declaration: r = mload(uint256)(signature + 0x20) (YuzuFlattened.sol#42)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (YuzuFlattened.sol#708
)' in ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (YuzuFlattened.sol#699-718) potenti
ally used before declaration: response != IERC1155Receiver.onERC1155Received.selector (YuzuFlattened.sol#709)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (YuzuFlattened.sol#712)'
 in ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (YuzuFlattened.sol#699-718) potential
ly used before declaration: revert(string)(reason) (YuzuFlattened.sol#713)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (YuzuFlattene
d.sol#730)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (YuzuFlattened.so
l#720-741) potentially used before declaration: response != IERC1155Receiver.onERC1155BatchReceived.selector (YuzuFlattened.sol
#732)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (YuzuFlattened.
sol#735)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (YuzuFlattened.sol#
720-741) potentially used before declaration: revert(string)(reason) (YuzuFlattened.sol#736)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Reentrancy in ERC1155Supply._mint(address,uint256,uint256,bytes) (YuzuFlattened.sol#762-770):
        External calls:
        - super._mint(account,id,amount,data) (YuzuFlattened.sol#768)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (YuzuFlattened.sol#708-716)
        State variables written after the call(s):
        - _totalSupply[id] += amount (YuzuFlattened.sol#769)
Reentrancy in ERC1155Supply._mintBatch(address,uint256[],uint256[],bytes) (YuzuFlattened.sol#772-782):
        External calls:
        - super._mintBatch(to,ids,amounts,data) (YuzuFlattened.sol#778)
                - IERC1155Receiver(to).onERC1155BatchReceived(operator,from,ids,amounts,data) (YuzuFlattened.sol#729-739)
        State variables written after the call(s):
```



```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
ECDSA.tryRecover(bytes32,bytes) (YuzuFlattened.sol#27-49) uses assembly
        - INLINE ASM (YuzuFlattened.sol#32-36)
        - INLINE ASM (YuzuFlattened.sol#41-44)
ECDSA.tryRecover(bytes32,bytes32,bytes32) (YuzuFlattened.sol#57-69) uses assembly
        - INLINE ASM (YuzuFlattened.sol#64-67)
Address.isContract(address) (YuzuFlattened.sol#421-428) uses assembly
        - INLINE ASM (YuzuFlattened.sol#424-426)
Address.verifyCallResult(bool,bytes,string) (YuzuFlattened.sol#500-518) uses assembly
        - INLINE ASM (YuzuFlattened.sol#510-513)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AccessControl._setRoleAdmin(bytes32,bytes32) (YuzuFlattened.sol#395-399) is never used and should be removed
Address.functionCall(address,bytes) (YuzuFlattened.sol#437-439) is never used and should be removed
Address.functionCall(address,bytes,string) (YuzuFlattened.sol#441-447) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (YuzuFlattened.sol#449-455) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (YuzuFlattened.sol#457-468) is never used and should be removed
```



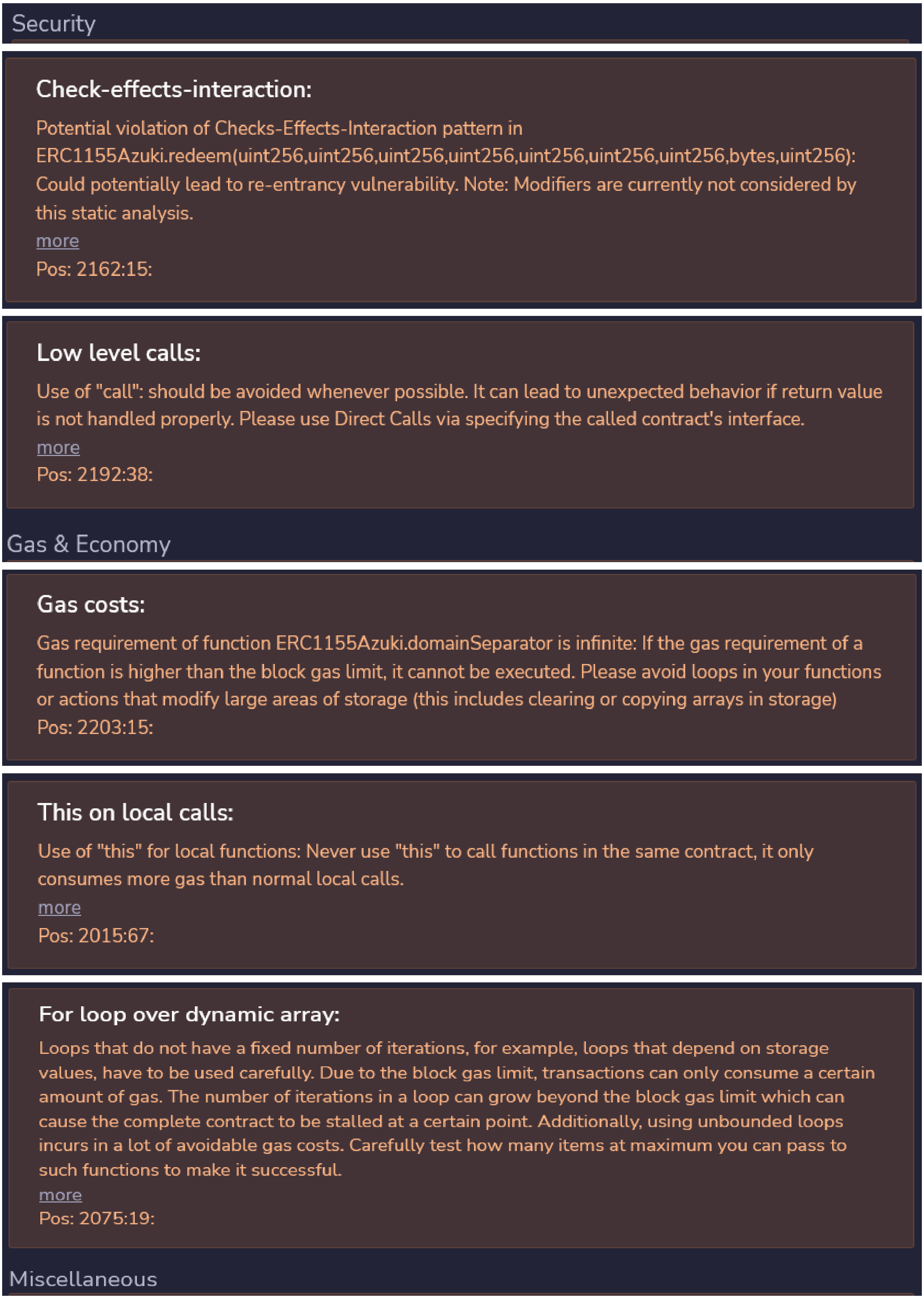
```
INFO:Detectors:
Variable EIP712._CACHED_DOMAIN_SEPARATOR (YuzuFlattened.sol#123) is not in mixedCase
Variable EIP712._CACHED_CHAIN_ID (YuzuFlattened.sol#124) is not in mixedCase
Variable EIP712._HASHED_NAME (YuzuFlattened.sol#126) is not in mixedCase
Variable EIP712._HASHED_VERSION (YuzuFlattened.sol#127) is not in mixedCase
Variable EIP712._TYPE_HASH (YuzuFlattened.sol#128) is not in mixedCase
Variable ERC1155Yuzu._tokenIdCounter (YuzuFlattened.sol#823) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (YuzuFlattened.sol#381-383)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (YuzuFlattened.sol#385-389)
uri(uint256) should be declared external:
        - ERC1155.uri(uint256) (YuzuFlattened.sol#541-543)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155.balanceOfBatch(address[],uint256[]) (YuzuFlattened.sol#550-566)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155.setApprovalForAll(address,bool) (YuzuFlattened.sol#568-573)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (YuzuFlattened.sol#579-591)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (YuzuFlattened.sol#593-605)
name() should be declared external:
        - ERC1155Yuzu.name() (YuzuFlattened.sol#834-836)
```



```
name() should be declared external:
        - ERC1155Yuzu.name() (YuzuFlattened.sol#834-836)
symbol() should be declared external:
        - ERC1155Yuzu.symbol() (YuzuFlattened.sol#838-840)
autoMintBatch(address,uint256[]) should be declared external:
        - ERC1155Yuzu.autoMintBatch(address,uint256[]) (YuzuFlattened.sol#852-857)
mint(address,uint256,uint256,bytes) should be declared external:
        - ERC1155Yuzu.mint(address,uint256,uint256,bytes) (YuzuFlattened.sol#859-866)
mintBatch(address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155Yuzu.mintBatch(address,uint256[],uint256[],bytes) (YuzuFlattened.sol#868-875)
addMinter(address) should be declared external:
        - ERC1155Yuzu.addMinter(address) (YuzuFlattened.sol#887-889)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:YuzuFlattened.sol analyzed (17 contracts with 75 detectors), 72 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

Azuki.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ERC1155Azuki.redeem(uint256,uint256,uint256,uint256,uint256,uint256,uint256,bytes,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 2162:15:

Low level calls:

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

more

Pos: 2192:38:

Gas & Economy

Gas costs:

Gas requirement of function ERC1155Azuki.domainSeparator 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: 2203:15:

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: 2015:67:

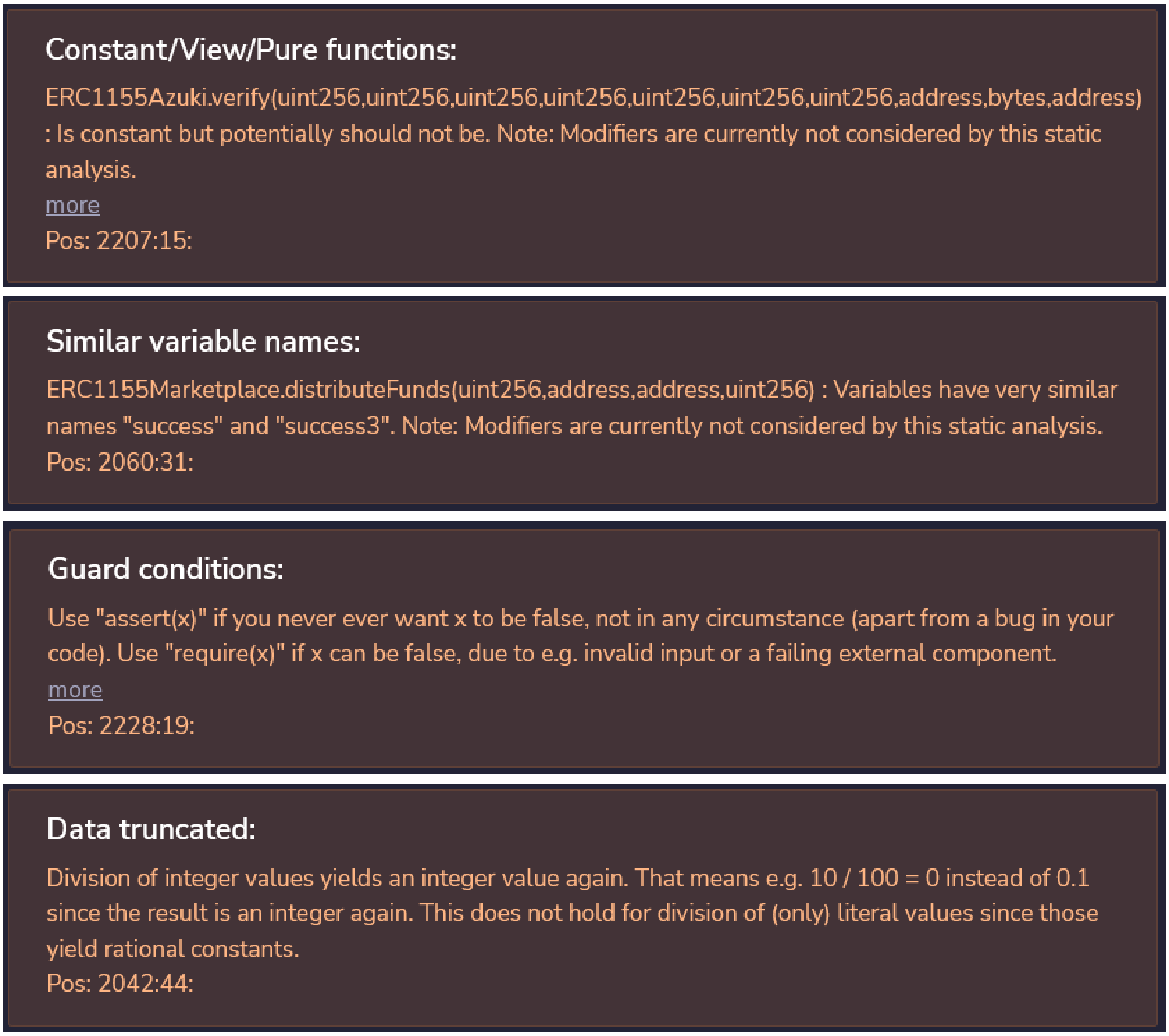
For loop over dynamic array:

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: 2075:19:

Miscellaneous



Constant/View/Pure functions:

ERC1155Azuki.verify(uint256,uint256,uint256,uint256,uint256,uint256,uint256,address,bytes,address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2207:15:

Similar variable names:

ERC1155Marketplace.distributeFunds(uint256,address,address,uint256) : Variables have very similar names "success" and "success3". Note: Modifiers are currently not considered by this static analysis.
Pos: 2060:31:

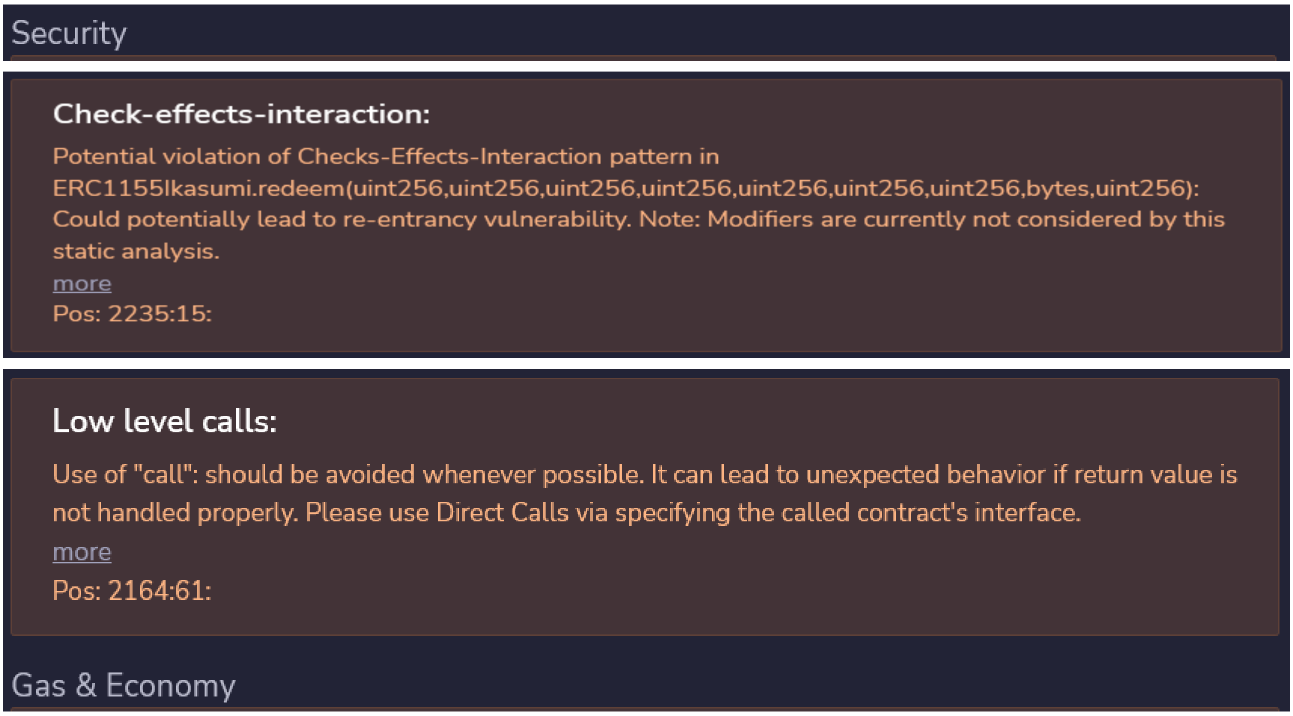
Guard conditions:

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: 2228:19:

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: 2042:44:

Ikasumi.sol



Security

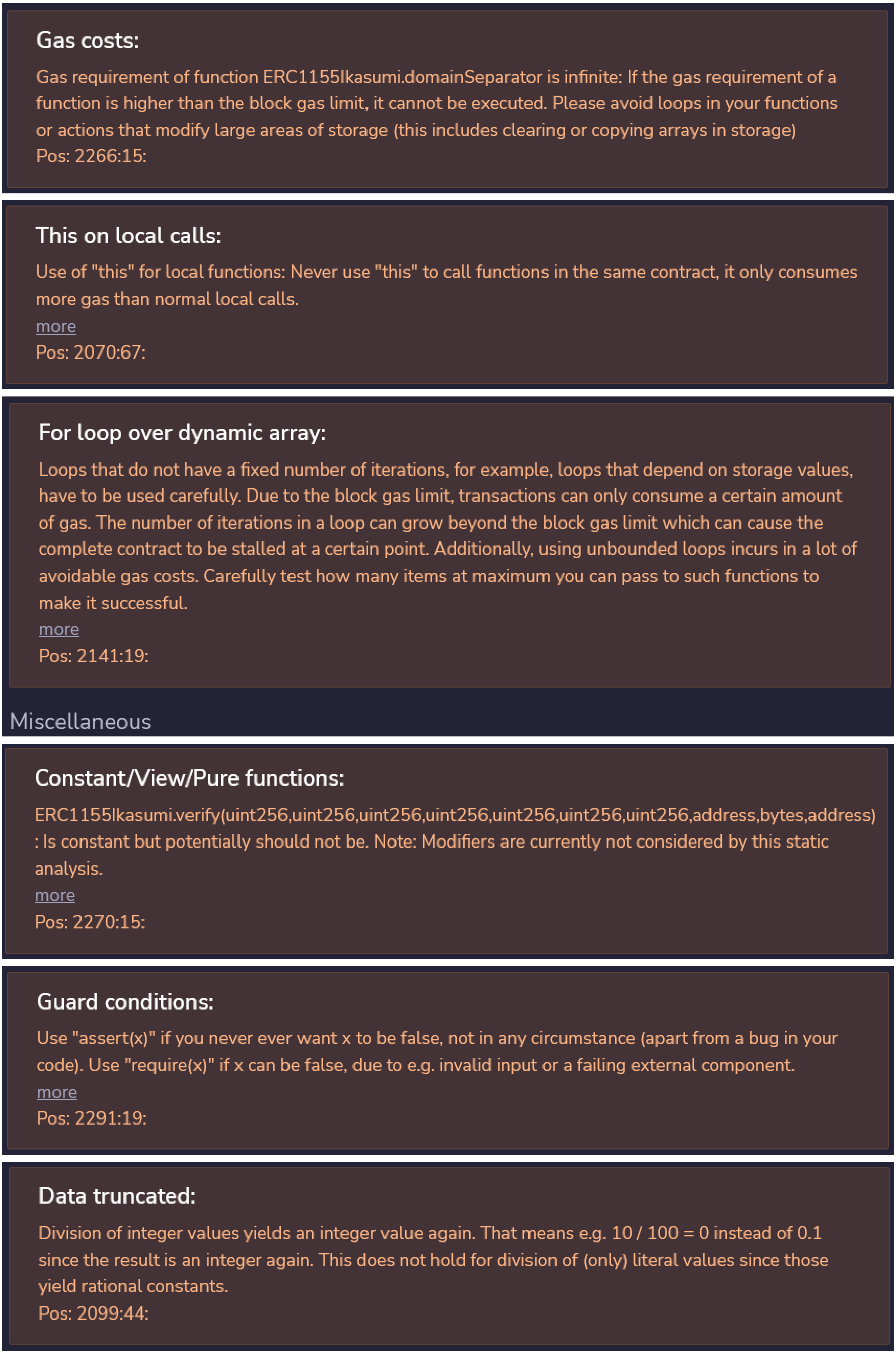
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ERC1155Ikasumi.redeem(uint256,uint256,uint256,uint256,uint256,uint256,uint256,bytes,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2235:15:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 2164:61:

Gas & Economy



Gas costs:

Gas requirement of function ERC1155Ikasumi.domainSeparator 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: 2266:15:

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: 2070:67:

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: 2141:19:

Miscellaneous

Constant/View/Pure functions:

ERC1155Ikasumi.verify(uint256,uint256,uint256,uint256,uint256,uint256,uint256,address,bytes,address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2270:15:

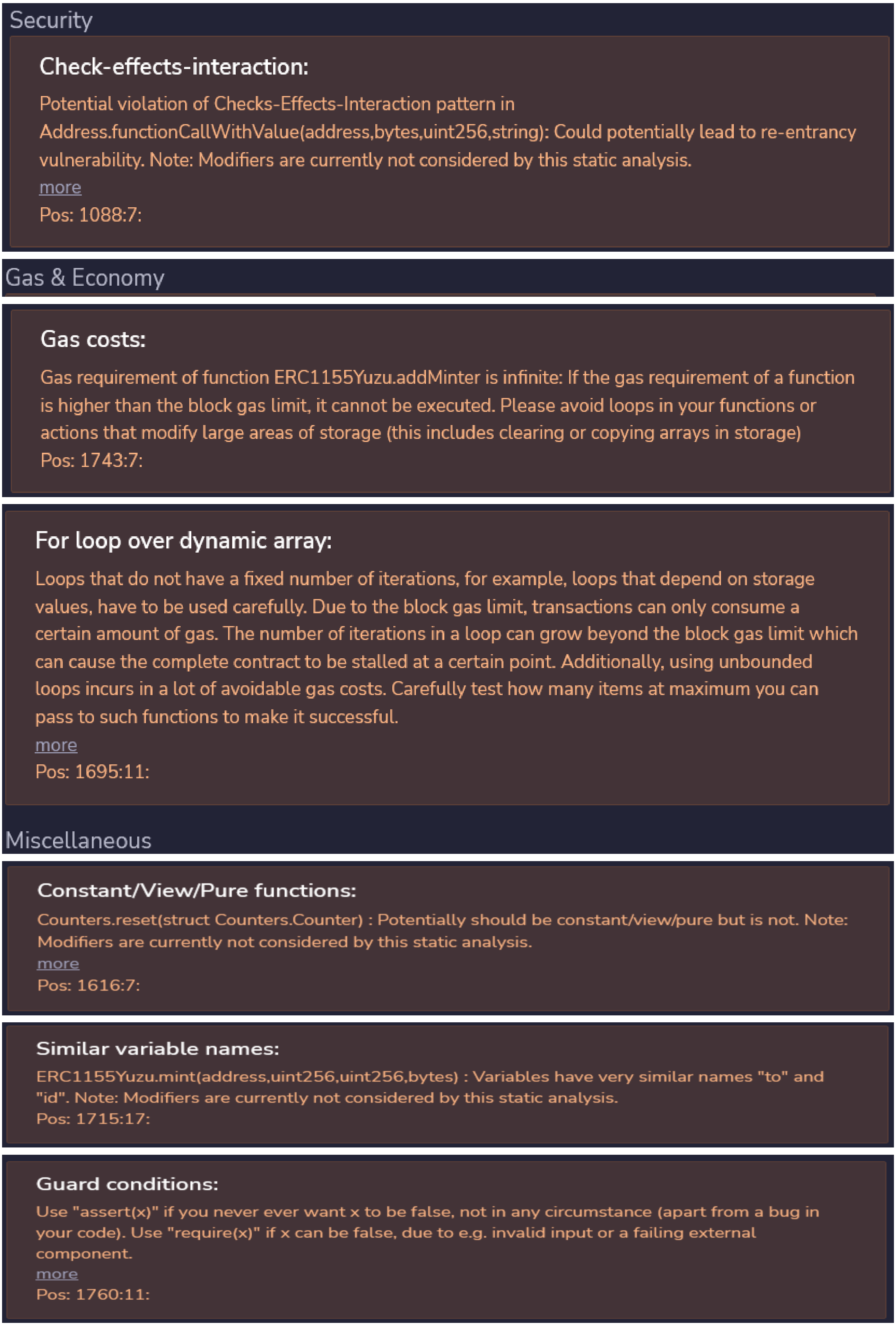
Guard conditions:

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: 2291:19:

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: 2099:44:

YuzuFlattened.sol



Security

Check-effects-interaction:

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

more

Pos: 1088:7:

Gas & Economy

Gas costs:

Gas requirement of function ERC1155Yuzu.addMinter 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: 1743:7:

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: 1695:11:

Miscellaneous

Constant/View/Pure functions:

Counters.reset(struct Counters.Counter) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1616:7:

Similar variable names:

ERC1155Yuzu.mint(address,uint256,uint256,bytes) : Variables have very similar names "to" and "id". Note: Modifiers are currently not considered by this static analysis.

Pos: 1715:17:

Guard conditions:

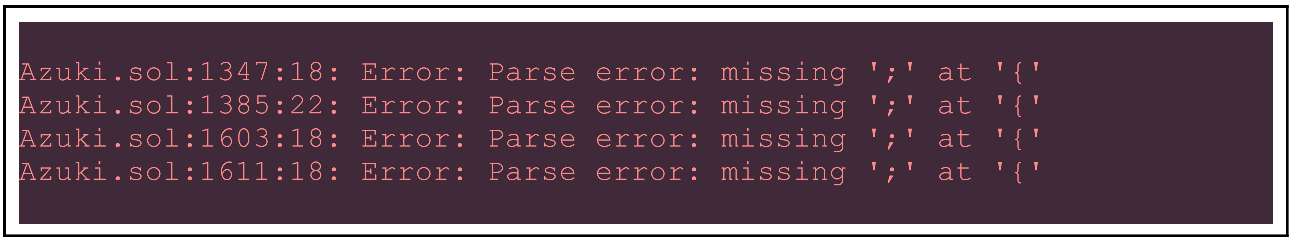
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 1760:11:

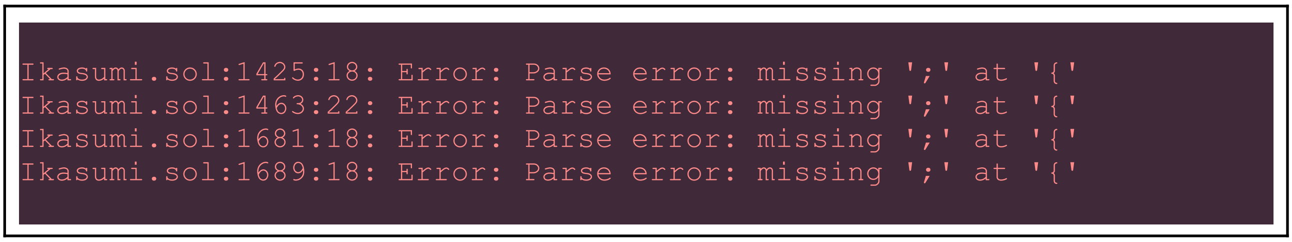
Solhint Linter

Azuki.sol



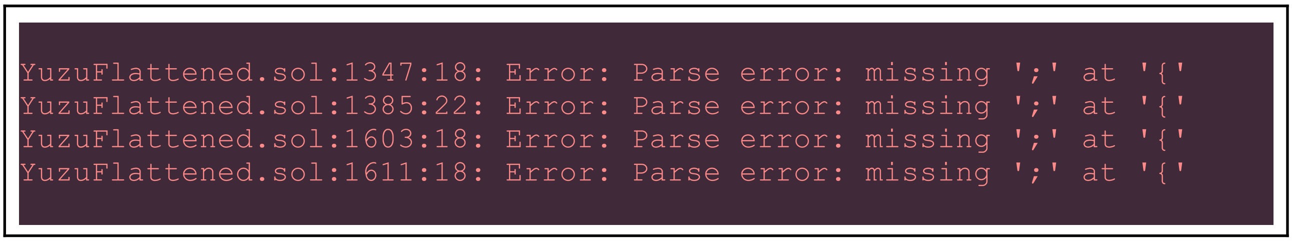
```
Azuki.sol:1347:18: Error: Parse error: missing ';' at '{'
Azuki.sol:1385:22: Error: Parse error: missing ';' at '{'
Azuki.sol:1603:18: Error: Parse error: missing ';' at '{'
Azuki.sol:1611:18: Error: Parse error: missing ';' at '{'
```

Ikasumi.sol



```
Ikasumi.sol:1425:18: Error: Parse error: missing ';' at '{'
Ikasumi.sol:1463:22: Error: Parse error: missing ';' at '{'
Ikasumi.sol:1681:18: Error: Parse error: missing ';' at '{'
Ikasumi.sol:1689:18: Error: Parse error: missing ';' at '{'
```

YuzuFlattened.sol



```
YuzuFlattened.sol:1347:18: Error: Parse error: missing ';' at '{'
YuzuFlattened.sol:1385:22: Error: Parse error: missing ';' at '{'
YuzuFlattened.sol:1603:18: Error: Parse error: missing ';' at '{'
YuzuFlattened.sol:1611:18: Error: Parse error: missing ';' at '{'
```

Software analysis result:

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