Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: IVY Defi Protocol
Website: www.ivydefi.vip
Platform: Binance Smart Chain
Language: Solidity
Date: June 30th, 2022

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 IVY Defi to perform the Security audit of the IVY Defi Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 30th, 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

- IVY, all transactions, mortgages and governance on the platform are transparent and on-chain. It is established on BSC, the largest decentralized application ecosystem, aiming to provide global users with a set of easy-to-use and highly transparent financial services.
- IVY Defi Protocol is a smart contract having functions like: mint, burn, mintIvy, transferIvyFrom, transferIvy, stake, unstake, mintSIvy, registerPool, etc.

Audit scope

Name	**Code Review and Security Analysis Report for IVY Defi Protocol Smart Contracts**
Platform	**Binance Chain / Solidity**
File 1	AccessControl.sol
File 1 MD5 Hash	3741F006AAB540B961C5A379E460FA70
File 2	ERC20.sol
File 2 MD5 Hash	42F9A10ADF073571D6B9AC3E1B660925
File 3	EscrowedIvyERC20.sol
File 3 MD5 Hash	F3C5999E8BDDB4F827B80BB0ACEC509B
File 4	IvyAware.sol

File 4 MD5 Hash	7D714F63921C19213DE47FD7F1E4E414
File 5	IvyCorePool.sol
File 5 MD5 Hash	395AD9C73CAEA3479D04C18453716AD8
File 6	IvyERC20.sol
File 6 MD5 Hash	642C609D7F77E4E0F9D1D61C00ED74D2
File 7	IvyPoolBase.sol
File 7 MD5 Hash	FB17C3801D75659010ADF8F0D74F9503
File 8	IvyPoolFactory.sol
File 8 MD5 Hash	1FE83385AFEC30778571F1AFF4F011B3
File 9	Ownable.sol
File 9 MD5 Hash	57F6D8C093C639C358D33A7357DE96CB
File 10	ReentrancyGuard.sol
File 10 MD5 Hash	B955F5BBF6FBD7698BD823D04DA7C4E1
Audit Date	June 30th,2022

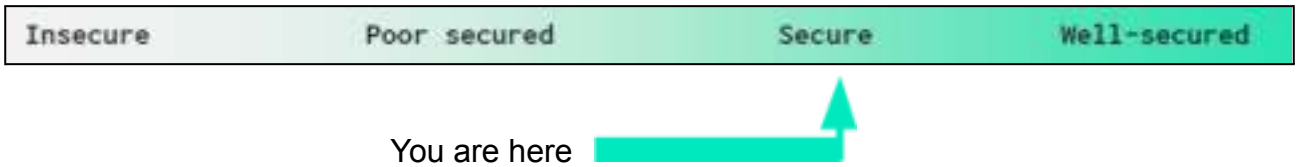
Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 AccessControl.sol • AccessControl has functions like: features, updateRole, etc.	**YES, This is valid.**
File 2 ERC20.sol • ERC20 has functions like: allowance, etc.	**YES, This is valid.**
File 3 EscrowedIvyERC20.sol • Name: Escrowed Ivy • Symbol: eIVY	**YES, This is valid.**
File 4 IvyAware.sol • IvyAware has functions like: transferIvy, etc.	**YES, This is valid.**
File 5 IvyCorePool.sol • IvyCorePool owner can execute Set the vault. • IvyCorePool owner can execute by the vault to transfer vault rewards IVY from the vault into the pool.	**YES, This is valid.**
File 6 IvyERC20.sol • Name: Ivy • Symbol: IVY • Decimals: 18 • Initial Supply: 4000 • Max Supply: 10000	**YES, This is valid.**
File 7 IvyPoolBase.sol • IvyPoolBase has functions like: pendingYieldRewards, getDeposit, etc.	**YES, This is valid.**
File 8 IvyPoolFactory.sol • IvyPoolFactory can create a core pool	**YES, This is valid.**

(IvyCorePool) and register it within the factory. • IvyPoolFactory can set the end block when necessary.	
File 9 Ownable.sol • Ownable can renounce Ownership. • Ownable can transfer ownership of the contract to a new account.	**YES, This is valid.**
File 10 ReentrancyGuard.sol • ReentrancyGuard contract module that helps prevent reentrant calls to a function.	**YES, This is valid.**

Audit Summary

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



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

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

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

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Moderated
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	Moderated
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 10 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 IVY Defi 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 IVY Defi Protocol.

The IVY Defi 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 well commented on smart contracts. We suggest using Ethereum's NatSpec style for the commenting.

Documentation

We were given an IVY Defi Protocol smart contract code in the form of a Github web link. The hash of that code is mentioned above in the table.

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

Another source of information was its official website https://www.ivydefi.vip/ which provided rich information about the project architecture.

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

AccessControl.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	features	read	Passed	No Issue
3	updateFeatures	write	Passed	No Issue
4	updateRole	write	Passed	No Issue
5	evaluateBy	read	Passed	No Issue
6	isFeatureEnabled	read	Passed	No Issue
7	isSenderInRole	read	Passed	No Issue
8	isOperatorInRole	read	Passed	No Issue
9	__hasRole	internal	Passed	No Issue

ERC20.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	name	read	Passed	No Issue
3	symbol	read	Passed	No Issue
4	decimals	read	Passed	No Issue
5	totalSupply	read	Passed	No Issue
6	balanceOf	read	Passed	No Issue
7	transfer	write	Passed	No Issue
8	allowance	read	Passed	No Issue
9	approve	write	Passed	No Issue
10	transferFrom	write	Passed	No Issue
11	increaseAllowance	write	Passed	No Issue
12	decreaseAllowance	write	Passed	No Issue
13	_transfer	internal	Passed	No Issue
14	_mint	internal	Passed	No Issue
15	_burn	internal	Passed	No Issue
16	_approve	internal	Passed	No Issue
17	_setupDecimals	internal	Passed	No Issue
18	_beforeTokenTransfer	internal	Passed	No Issue

EscrowedIvyERC20.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue

2	mint	external	Passed	No Issue
3	burn	external	Passed	No Issue
4	name	read	Passed	No Issue
5	symbol	read	Passed	No Issue
6	decimals	read	Passed	No Issue
7	totalSupply	read	Passed	No Issue
8	balanceOf	read	Passed	No Issue
9	transfer	write	Passed	No Issue
10	allowance	read	Passed	No Issue
11	approve	write	Passed	No Issue
12	transferFrom	write	Passed	No Issue
13	increaseAllowance	write	Passed	No Issue
14	decreaseAllowance	write	Passed	No Issue
15	_transfer	internal	Passed	No Issue
16	_mint	internal	Unlimited Minting	Refer Audit Findings
17	_burn	internal	Passed	No Issue
18	_approve	internal	Passed	No Issue
19	_setupDecimals	internal	Passed	No Issue
20	_beforeTokenTransfer	internal	Passed	No Issue

IvyAware.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	transferIvy	internal	Passed	No Issue
3	transferIvyFrom	internal	Passed	No Issue
4	mintIvy	internal	Passed	No Issue

IvyCorePool.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	mintIvy	internal	Passed	No Issue
3	pendingYieldRewards	external	Passed	No Issue
4	balanceOf	external	Passed	No Issue
5	getDeposit	external	Passed	No Issue
6	getDepositsLength	external	Passed	No Issue
7	stake	external	Passed	No Issue
8	unstake	external	Passed	No Issue
9	updateStakeLock	external	Passed	No Issue
10	sync	external	Passed	No Issue
11	processRewards	external	Passed	No Issue

12	setWeight	external	Passed	No Issue
13	_pendingYieldRewards	internal	Passed	No Issue
14	_stake	internal	Passed	No Issue
15	_unstake	internal	Passed	No Issue
16	_sync	internal	Passed	No Issue
17	_processRewards	internal	Passed	No Issue
18	_updateStakeLock	internal	Passed	No Issue
19	weightToReward	write	Passed	No Issue
20	rewardToWeight	write	Passed	No Issue
21	blockNumber	read	Passed	No Issue
22	now256	read	Passed	No Issue
23	mintSIvy	write	Passed	No Issue
24	transferPoolToken	internal	Passed	No Issue
25	transferPoolTokenFrom	internal	Passed	No Issue
26	pendingVaultRewards	read	Passed	No Issue
27	setVault	external	Passed	No Issue
28	processRewards	external	Passed	No Issue
29	receiveVaultRewards	external	Passed	No Issue
30	stakeAsPool	external	Unused function	Refer Audit Findings
31	_stake	internal	Passed	No Issue
32	_unstake	internal	Passed	No Issue
33	_processRewards	internal	Passed	No Issue
34	_processVaultRewards	write	Passed	No Issue

IvyERC20.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	features	read	Passed	No Issue
3	updateFeatures	write	Passed	No Issue
4	updateRole	write	Passed	No Issue
5	evaluateBy	read	Passed	No Issue
6	isFeatureEnabled	read	Passed	No Issue
7	isSenderInRole	read	Passed	No Issue
8	isOperatorInRole	read	Passed	No Issue
9	__hasRole	internal	Passed	No Issue
10	balanceOf	read	Passed	No Issue
11	transfer	write	Passed	No Issue
12	transferFrom	write	Passed	No Issue
13	safeTransferFrom	write	Passed	No Issue
14	unsafeTransferFrom	write	Passed	No Issue
15	approve	write	Passed	No Issue
16	allowance	read	Passed	No Issue
17	increaseAllowance	write	Passed	No Issue
18	decreaseAllowance	write	Passed	No Issue

19	mint	write	Passed	No Issue
20	burn	write	Passed	No Issue
21	getVotingPower	read	Passed	No Issue
22	getVotingPowerAt	read	Passed	No Issue
23	getVotingPowerHistory	read	Passed	No Issue
24	getVotingPowerHistoryLength	read	Passed	No Issue
25	delegate	write	Passed	No Issue
26	delegateWithSig	write	Passed	No Issue
27	__delegate	write	Passed	No Issue
28	__moveVotingPower	write	Passed	No Issue
29	__updateVotingPower	write	Passed	No Issue
30	__binaryLookup	read	Passed	No Issue

IvyPoolBase.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	transferIvy	internal	Passed	No Issue
3	transferIvyFrom	internal	Passed	No Issue
4	mintIvy	internal	Passed	No Issue
5	pendingYieldRewards	external	Passed	No Issue
6	balanceOf	external	Passed	No Issue
7	getDeposit	external	Passed	No Issue
8	getDepositsLength	external	Passed	No Issue
9	stake	external	Passed	No Issue
10	unstake	external	Passed	No Issue
11	updateStakeLock	external	Passed	No Issue
12	sync	external	Passed	No Issue
13	processRewards	external	Passed	No Issue
14	setWeight	external	Passed	No Issue
15	_pendingYieldRewards	internal	Passed	No Issue
16	_stake	internal	Passed	No Issue
17	_unstake	internal	Passed	No Issue
18	_sync	internal	Passed	No Issue
19	_processRewards	internal	Passed	No Issue
20	_updateStakeLock	internal	Passed	No Issue
21	weightToReward	write	Passed	No Issue
22	rewardToWeight	write	Passed	No Issue
23	blockNumber	read	Passed	No Issue
24	now256	read	Passed	No Issue
25	mintSIvy	write	Unused function	Refer Audit Findings
26	transferPoolToken	internal	Passed	No Issue
27	transferPoolTokenFrom	internal	Passed	No Issue
28	nonReentrant	modifier	Passed	No Issue

IvyPoolFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	transferIvy	internal	Passed	No Issue
3	transferIvyFrom	internal	Passed	No Issue
4	mintIvy	internal	Passed	No Issue
5	owner	read	Passed	No Issue
6	onlyOwner	modifier	Passed	No Issue
7	renounceOwnership	write	access only Owner	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	getPoolAddress	external	Passed	No Issue
10	getPoolData	read	Passed	No Issue
11	shouldUpdateRatio	read	Passed	No Issue
12	createPool	external	access only Owner	No Issue
13	setEndBlock	external	access only Owner	No Issue
14	setIvyPerBlock	external	access only Owner	No Issue
15	registerPool	write	access only Owner	No Issue
16	updateIVYPerBlock	external	Passed	No Issue
17	mintYieldTo	external	Passed	No Issue
18	changePoolWeight	external	Passed	No Issue
19	blockNumber	read	Passed	No Issue

Ownable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue

ReentrancyGuard.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	nonReentrant	modifier	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

No Medium severity vulnerabilities were found.

Low

No Low severity vulnerabilities were found.

Very Low / Informational / Best practices:

(1) Unlimited Minting:- **EscrowedIvyERC20.sol**

Token creators can mint unlimited tokens.

Token minting without any maximum limit is considered inappropriate for tokenomics.

Resolution: We recommend placing some limit on token minting to mitigate this issue.

(2) Unused interface / function / variables:

IvyPoolBase.sol

ICorePool.sol has been imported but not used.

mintSIvy is defined as an internal function, but not used.

sIVY variable has been defined and set while deploying, but not used in code.

IvyCorePool.sol

StakeAsPool is an external function but executed only by pool address added into the factory. But given poolbase has not used this function.

IvyERC20.sol

FEATURE_TRANSFERS, FEATURE_TRANSFERS_ON_BEHALF, ERC20_RECEIVED are unused variables.

Resolution: We suggest removing unused interface / functions / variables.

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:

- setVault: IvyCorePool owner can execute to Set the vault.
- receiveVaultRewards: IvyCorePool owner can execute by the vault to transfer vault rewards IVY from the vault into the pool.
- setWeight: IvyPoolBase owner can set weight.
- createPool: IvyPoolFactory owner can create a core pool (IvyCorePool) and register it within the factory.
- setEndBlock: IvyPoolFactory owner can set end block when necessary.
- setIvyPerBlock: IvyPoolFactory owner can set ivy per block when necessary.
- registerPool: IvyPoolFactory owner can register an already deployed pool instance within the factory.
- changePoolWeight: IvyPoolFactory owner can change the weight of the pool.
- renounceOwnership: Ownable can renounce new ownership.
- transferOwnership: Ownable can transfer ownership of the contract to a new account (`newOwner`).

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 Github weblink. And we have used all possible tests based on given objects as files. We have not observed any major issues in the smart contracts. **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 - IVY Defi Protocol

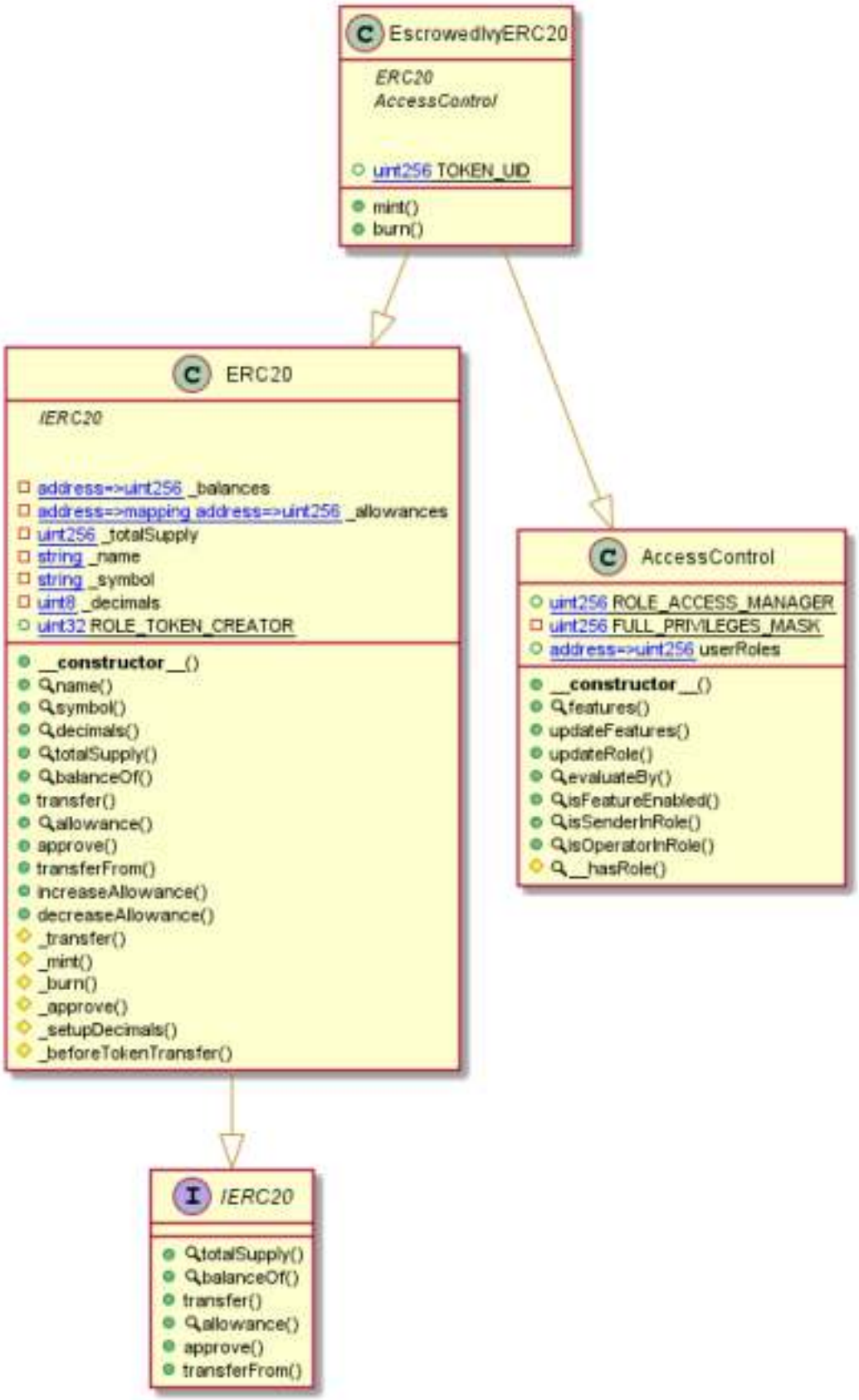
AccessControl Diagram

Ownable Diagram

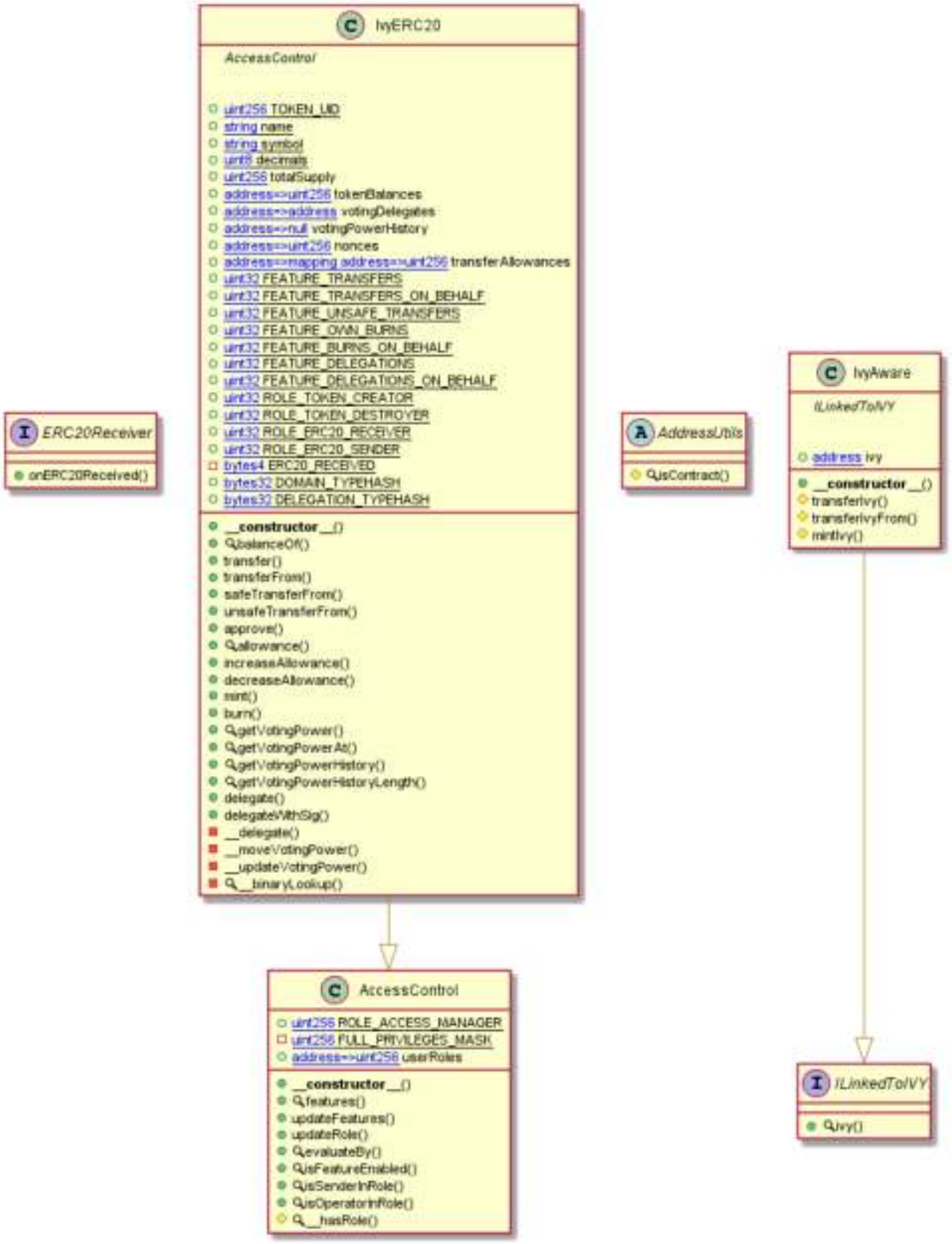
ReentrancyGuard Diagram

ERC20 Diagram

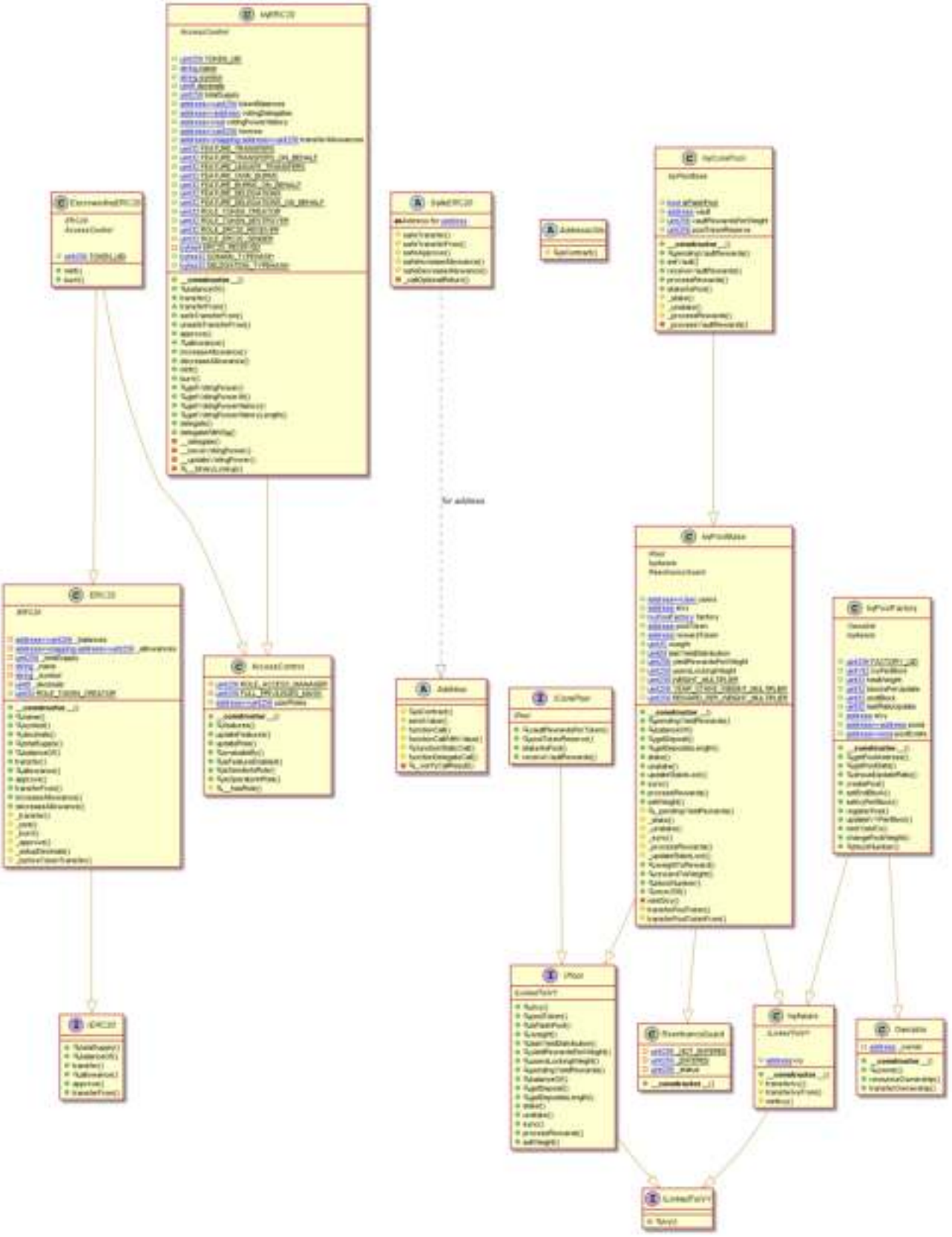
EscrowedIvyERC20 Diagram



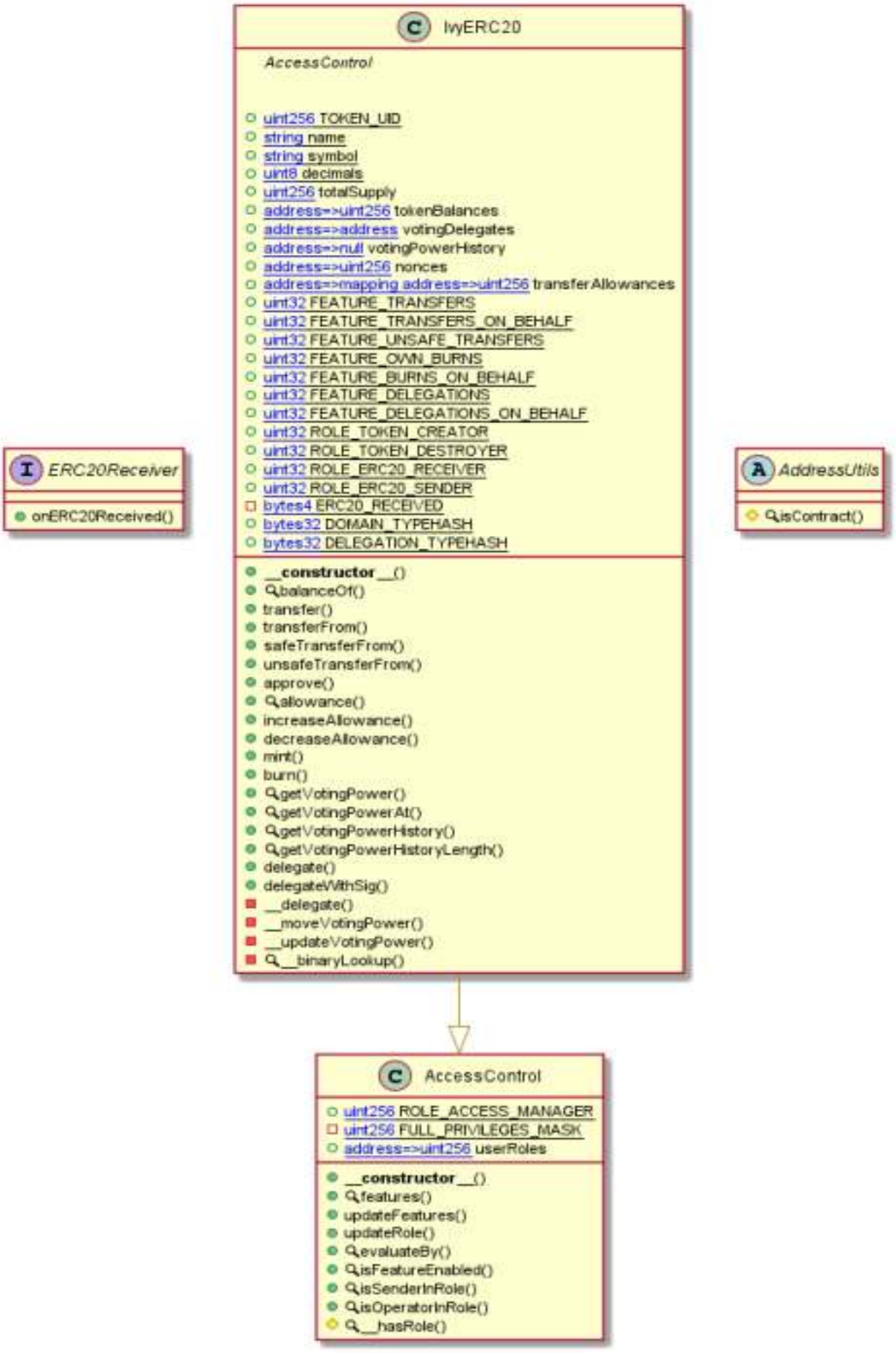
IvyAware Diagram



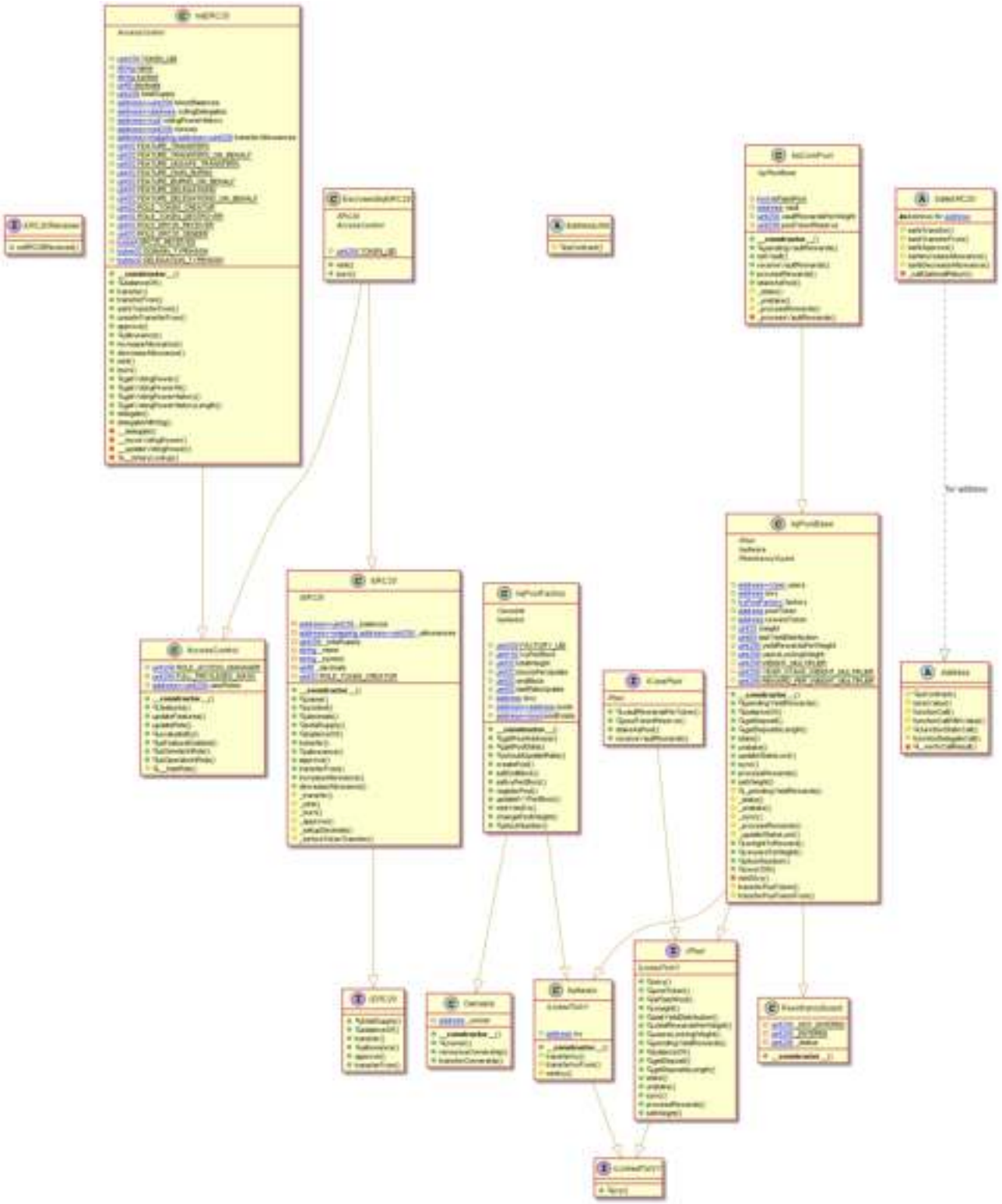
IvyCorePool Diagram



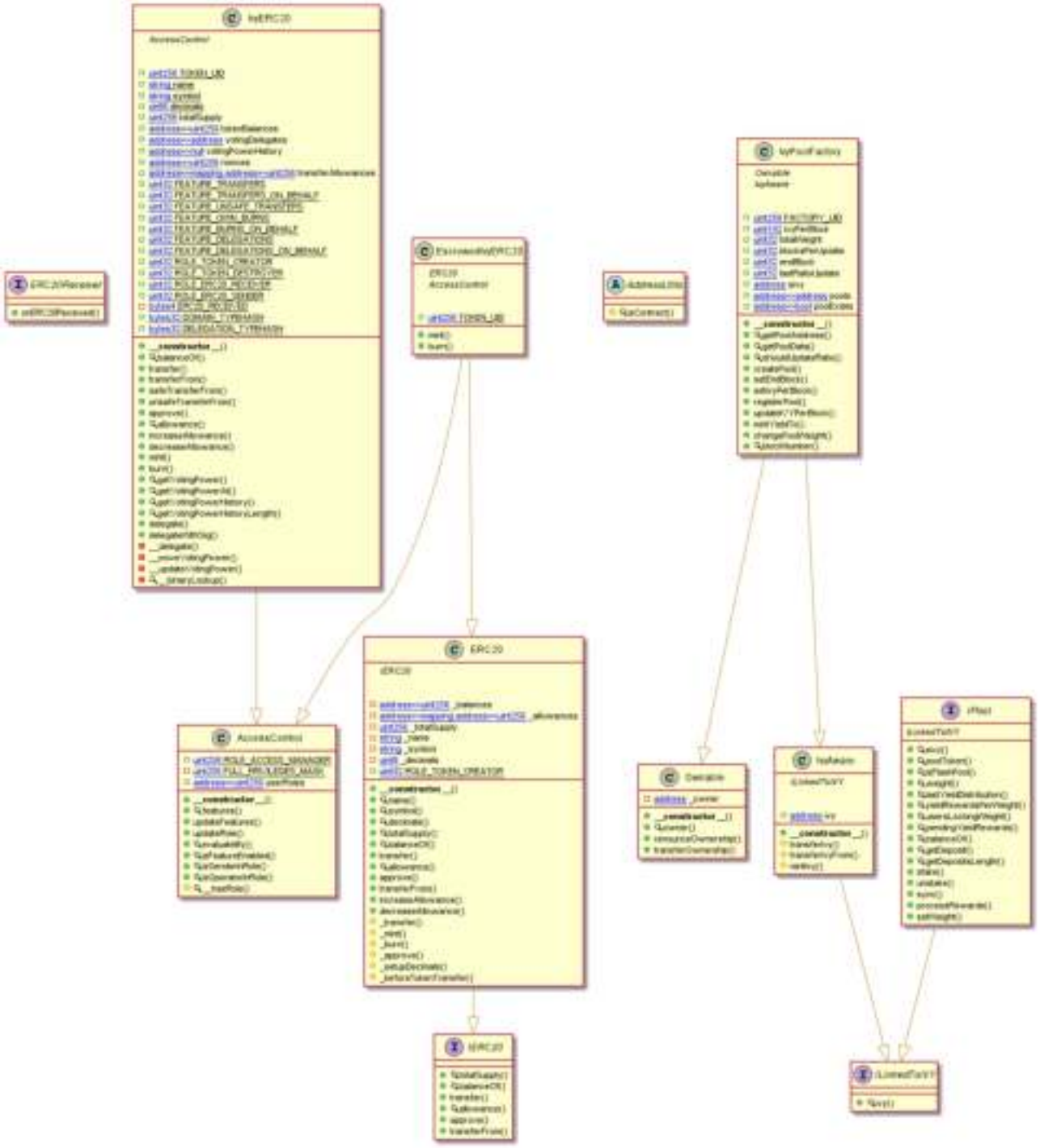
IvyERC20 Diagram



IvyPoolBase Diagram



IvyPoolFactory Diagram



Slither Results Log

Slither log >> AccessControl.sol

```
INFO:Detectors:
Pragma version0.8.0 (AccessControl.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.
6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter AccessControl.updateFeatures(uint256)._mask (AccessControl.sol#21) is not in mixedCase
Function AccessControl.__hasRole(uint256,uint256) (AccessControl.sol#55-57) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
AccessControl.slitherConstructorConstantVariables() (AccessControl.sol#4-59) uses literals with too many digits:
        - ROLE_ACCESS_MANAGER = 0x8000000000000000000000000000000000000000000000000000000000000000 (AccessControl.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
updateFeatures(uint256) should be declared external:
        - AccessControl.updateFeatures(uint256) (AccessControl.sol#21-23)
isFeatureEnabled(uint256) should be declared external:
        - AccessControl.isFeatureEnabled(uint256) (AccessControl.sol#42-44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:AccessControl.sol analyzed (1 contracts with 75 detectors), 7 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ERC20.sol

```
INFO:Detectors:
ERC20._burn(address,uint256) (ERC20.sol#126-134) is never used and should be removed
ERC20._mint(address,uint256) (ERC20.sol#116-124) is never used and should be removed
ERC20._setupDecimals(uint8) (ERC20.sol#148-150) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.0 (ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (ERC20.sol#46-48)
symbol() should be declared external:
        - ERC20.symbol() (ERC20.sol#50-52)
decimals() should be declared external:
        - ERC20.decimals() (ERC20.sol#54-56)
totalSupply() should be declared external:
        - ERC20.totalSupply() (ERC20.sol#58-60)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (ERC20.sol#62-64)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (ERC20.sol#66-69)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (ERC20.sol#71-73)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (ERC20.sol#75-78)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (ERC20.sol#80-88)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (ERC20.sol#90-93)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (ERC20.sol#95-98)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ERC20.sol analyzed (2 contracts with 75 detectors), 16 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> EscrowedIvyERC20.sol

```
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (EscrowedIvyERC20.sol#45-47)
symbol() should be declared external:
        - ERC20.symbol() (EscrowedIvyERC20.sol#49-51)
decimals() should be declared external:
        - ERC20.decimals() (EscrowedIvyERC20.sol#53-55)
totalSupply() should be declared external:
        - ERC20.totalSupply() (EscrowedIvyERC20.sol#57-59)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (EscrowedIvyERC20.sol#61-63)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (EscrowedIvyERC20.sol#65-68)
allowance(address,address) should be declared external:
```

```
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (EscrowedIvyERC20.sol#65-68)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (EscrowedIvyERC20.sol#70-72)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (EscrowedIvyERC20.sol#74-77)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (EscrowedIvyERC20.sol#79-87)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (EscrowedIvyERC20.sol#89-92)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (EscrowedIvyERC20.sol#94-97)
updateFeatures(uint256) should be declared external:
        - AccessControl.updateFeatures(uint256) (EscrowedIvyERC20.sol#175-177)
isFeatureEnabled(uint256) should be declared external:
        - AccessControl.isFeatureEnabled(uint256) (EscrowedIvyERC20.sol#196-198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:EscrowedIvyERC20.sol analyzed (4 contracts with 75 detectors), 19 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> IvyAware.sol

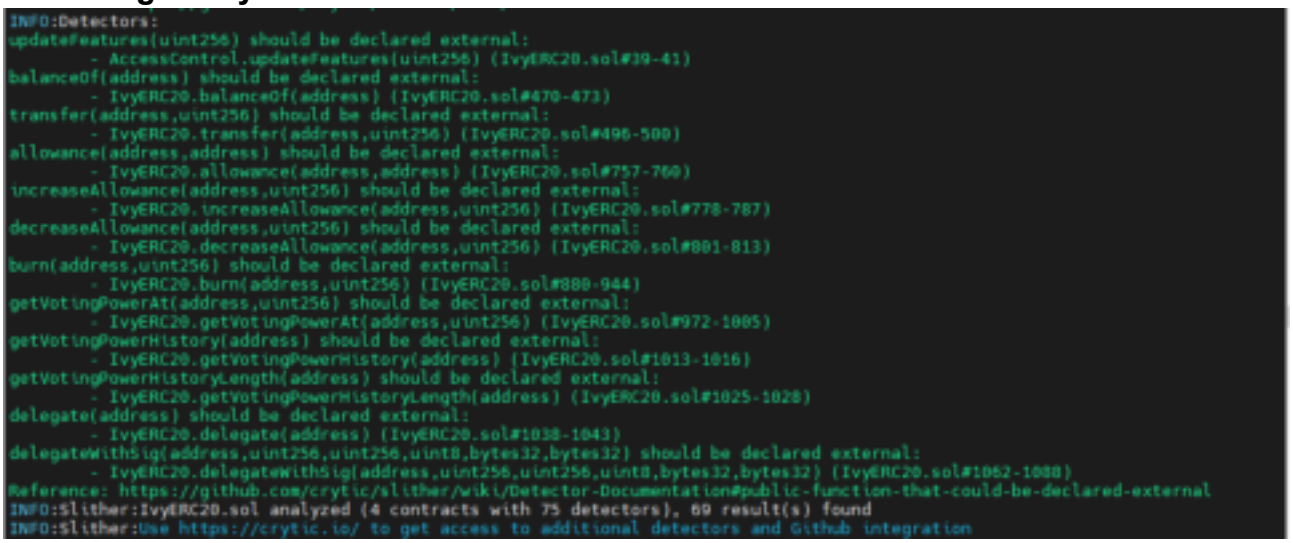
```
INFO:Detectors:
updateFeatures(uint256) should be declared external:
        - AccessControl.updateFeatures(uint256) (IvyAware.sol#40-42)
balanceOf(address) should be declared external:
        - IvyERC20.balanceOf(address) (IvyAware.sol#471-474)
transfer(address,uint256) should be declared external:
        - IvyERC20.transfer(address,uint256) (IvyAware.sol#497-501)
allowance(address,address) should be declared external:
        - IvyERC20.allowance(address,address) (IvyAware.sol#750-761)
increaseAllowance(address,uint256) should be declared external:
        - IvyERC20.increaseAllowance(address,uint256) (IvyAware.sol#779-788)
decreaseAllowance(address,uint256) should be declared external:
        - IvyERC20.decreaseAllowance(address,uint256) (IvyAware.sol#802-814)
burn(address,uint256) should be declared external:
        - IvyERC20.burn(address,uint256) (IvyAware.sol#881-945)
getVotingPowerAt(address,uint256) should be declared external:
        - IvyERC20.getVotingPowerAt(address,uint256) (IvyAware.sol#973-1006)
getVotingPowerHistory(address) should be declared external:
        - IvyERC20.getVotingPowerHistory(address) (IvyAware.sol#1014-1017)
getVotingPowerHistoryLength(address) should be declared external:
        - IvyERC20.getVotingPowerHistoryLength(address) (IvyAware.sol#1026-1029)
delegate(address) should be declared external:
        - IvyERC20.delegate(address) (IvyAware.sol#1038-1044)
delegateWithSig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - IvyERC20.delegateWithSig(address,uint256,uint256,uint8,bytes32,bytes32) (IvyAware.sol#1063-1089)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:IvyAware.sol analyzed (6 contracts with 75 detectors), 80 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> IvyCorePool.sol

```
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (IvyCorePool.sol#538-540)
symbol() should be declared external:
        - ERC20.symbol() (IvyCorePool.sol#542-544)
decimals() should be declared external:
        - ERC20.decimals() (IvyCorePool.sol#546-548)
totalSupply() should be declared external:
        - ERC20.totalSupply() (IvyCorePool.sol#550-552)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (IvyCorePool.sol#554-556)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (IvyCorePool.sol#558-561)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (IvyCorePool.sol#563-565)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (IvyCorePool.sol#567-570)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (IvyCorePool.sol#572-580)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (IvyCorePool.sol#582-585)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (IvyCorePool.sol#587-590)
updateFeatures(uint256) should be declared external:
        - AccessControl.updateFeatures(uint256) (IvyCorePool.sol#607-609)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (IvyCorePool.sol#754-757)
```

```
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (IvyCorePool.sol#754-757)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (IvyCorePool.sol#763-767)
balanceOf(address) should be declared external:
        - IvyERC20.balanceOf(address) (IvyCorePool.sol#1133-1136)
transfer(address,uint256) should be declared external:
        - IvyERC20.transfer(address,uint256) (IvyCorePool.sol#1159-1163)
allowance(address,address) should be declared external:
        - IvyERC20.allowance(address,address) (IvyCorePool.sol#1393-1396)
increaseAllowance(address,uint256) should be declared external:
        - IvyERC20.increaseAllowance(address,uint256) (IvyCorePool.sol#1414-1423)
decreaseAllowance(address,uint256) should be declared external:
        - IvyERC20.decreaseAllowance(address,uint256) (IvyCorePool.sol#1437-1449)
burn(address,uint256) should be declared external:
        - IvyERC20.burn(address,uint256) (IvyCorePool.sol#1516-1580)
getVotingPowerAt(address,uint256) should be declared external:
        - IvyERC20.getVotingPowerAt(address,uint256) (IvyCorePool.sol#1608-1641)
getVotingPowerHistory(address) should be declared external:
        - IvyERC20.getVotingPowerHistory(address) (IvyCorePool.sol#1649-1652)
getVotingPowerHistoryLength(address) should be declared external:
        - IvyERC20.getVotingPowerHistoryLength(address) (IvyCorePool.sol#1661-1664)
delegate(address) should be declared external:
        - IvyERC20.delegate(address) (IvyCorePool.sol#1674-1679)
delegateWithSig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - IvyERC20.delegateWithSig(address,uint256,uint256,uint8,bytes32,bytes32) (IvyCorePool.sol#1698-1724)
getPoolData(address) should be declared external:
        - IvyPoolFactory.getPoolData(address) (IvyCorePool.sol#2097-2112)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:IvyCorePool.sol analyzed (17 contracts with 75 detectors), 177 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> IvyERC20.sol



Slither log >> IvyPoolFactory.sol

```
INFO:Detectors:
IvyERC20.slitherConstructorConstantVariables() (IvyPoolFactory.sol#158-1295) uses literals with too many digits:
        - ROLE_ACCESS_MANAGER = 0x8000000000000000000000000000000000000000000000000000000000000000 (IvyPoolFactory.sol#23)
EscrowedIvyERC20.slitherConstructorConstantVariables() (IvyPoolFactory.sol#1515-1526) uses literals with too many digits:
        - ROLE_ACCESS_MANAGER = 0x8000000000000000000000000000000000000000000000000000000000000000 (IvyPoolFactory.sol#23)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
IvyERC20.ERC20_RECEIVED (IvyPoolFactory.sol#304) is never used in IvyERC20 (IvyPoolFactory.sol#158-1295)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
updateFeatures(uint256) should be declared external:
        - AccessControl.updateFeatures(uint256) (IvyPoolFactory.sol#39-41)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (IvyPoolFactory.sol#143-146)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (IvyPoolFactory.sol#152-156)
balanceOf(address) should be declared external:
        - IvyERC20.balanceOf(address) (IvyPoolFactory.sol#522-525)
transfer(address,uint256) should be declared external:
        - IvyERC20.transfer(address,uint256) (IvyPoolFactory.sol#548-552)
allowance(address,address) should be declared external:
        - IvyERC20.allowance(address,address) (IvyPoolFactory.sol#809-812)
increaseAllowance(address,uint256) should be declared external:
        - IvyERC20.increaseAllowance(address,uint256) (IvyPoolFactory.sol#830-839)
decreaseAllowance(address,uint256) should be declared external:
        - IvyERC20.decreaseAllowance(address,uint256) (IvyPoolFactory.sol#853-865)
burn(address,uint256) should be declared external:
        - IvyERC20.burn(address,uint256) (IvyPoolFactory.sol#932-996)
getVotingPowerAt(address,uint256) should be declared external:
        - IvyERC20.getVotingPowerAt(address,uint256) (IvyPoolFactory.sol#1024-1057)
```

```
getVotingPowerHistory(address) should be declared external:
        - IvyERC20.getVotingPowerHistory(address) (IvyPoolFactory.sol#1065-1068)
getVotingPowerHistoryLength(address) should be declared external:
        - IvyERC20.getVotingPowerHistoryLength(address) (IvyPoolFactory.sol#1077-1080)
delegate(address) should be declared external:
        - IvyERC20.delegate(address) (IvyPoolFactory.sol#1090-1095)
delegateWithSig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - IvyERC20.delegateWithSig(address,uint256,uint256,uint8,bytes32,bytes32) (IvyPoolFactory.sol#1114-1140)
name() should be declared external:
        - ERC20.name() (IvyPoolFactory.sol#1401-1403)
symbol() should be declared external:
        - ERC20.symbol() (IvyPoolFactory.sol#1405-1407)
decimals() should be declared external:
        - ERC20.decimals() (IvyPoolFactory.sol#1409-1411)
totalSupply() should be declared external:
        - ERC20.totalSupply() (IvyPoolFactory.sol#1413-1415)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (IvyPoolFactory.sol#1417-1419)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (IvyPoolFactory.sol#1421-1424)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (IvyPoolFactory.sol#1426-1428)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (IvyPoolFactory.sol#1430-1433)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (IvyPoolFactory.sol#1435-1443)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (IvyPoolFactory.sol#1445-1448)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (IvyPoolFactory.sol#1450-1453)
getPoolData(address) should be declared external:
        - IvyPoolFactory.getPoolData(address) (IvyPoolFactory.sol#1761-1776)
registerPool(address) should be declared external:
        - IvyPoolFactory.registerPool(address) (IvyPoolFactory.sol#1851-1869)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:IvyPoolFactory.sol analyzed (12 contracts with 75 detectors), 104 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> IvyPoolBase.sol

```
INFO:Detectors:
IvyERC20.slitherConstructorConstantVariables() (IvyPoolBase.sol#159-1296) uses literals with too many digits:
        - ROLE_ACCESS_MANAGER = 0x8000000000000000000000000000000000000000000000000000000000000000 (IvyPoolBase.sol#24)
EscrowedIvyERC20.slitherConstructorConstantVariables() (IvyPoolBase.sol#1517-1528) uses literals with too many digits:
        - ROLE_ACCESS_MANAGER = 0x8000000000000000000000000000000000000000000000000000000000000000 (IvyPoolBase.sol#24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
IvyERC20.ERC20_RECEIVED (IvyPoolBase.sol#395) is never used in IvyERC20 (IvyPoolBase.sol#159-1296)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
updateFeatures(uint256) should be declared external:
        - AccessControl.updateFeatures(uint256) (IvyPoolBase.sol#40-42)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (IvyPoolBase.sol#144-147)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (IvyPoolBase.sol#153-157)
balanceOf(address) should be declared external:
        - IvyERC20.balanceOf(address) (IvyPoolBase.sol#523-526)
transfer(address,uint256) should be declared external:
        - IvyERC20.transfer(address,uint256) (IvyPoolBase.sol#549-553)
allowance(address,address) should be declared external:
        - IvyERC20.allowance(address,address) (IvyPoolBase.sol#810-813)
increaseAllowance(address,uint256) should be declared external:
        - IvyERC20.increaseAllowance(address,uint256) (IvyPoolBase.sol#831-840)
decreaseAllowance(address,uint256) should be declared external:
        - IvyERC20.decreaseAllowance(address,uint256) (IvyPoolBase.sol#854-866)
burn(address,uint256) should be declared external:
        - IvyERC20.burn(address,uint256) (IvyPoolBase.sol#933-997)
getVotingPowerAt(address,uint256) should be declared external:
        - IvyERC20.getVotingPowerAt(address,uint256) (IvyPoolBase.sol#1025-1058)
getVotingPowerHistory(address) should be declared external:
        - IvyERC20.getVotingPowerHistory(address) (IvyPoolBase.sol#1060-1069)
getVotingPowerHistoryLength(address) should be declared external:
        - IvyERC20.getVotingPowerHistoryLength(address) (IvyPoolBase.sol#1070-1081)
delegate(address) should be declared external:
        - IvyERC20.delegate(address) (IvyPoolBase.sol#1091-1096)
```

```
delegate(address) should be declared external:
        - IvyERC20.delegate(address) (IvyPoolBase.sol#1091-1096)
delegateWithSig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - IvyERC20.delegateWithSig(address,uint256,uint256,uint8,bytes32,bytes32) (IvyPoolBase.sol#1115-1141)
name() should be declared external:
        - ERC20.name() (IvyPoolBase.sol#1402-1404)
symbol() should be declared external:
        - ERC20.symbol() (IvyPoolBase.sol#1406-1408)
decimals() should be declared external:
        - ERC20.decimals() (IvyPoolBase.sol#1410-1412)
totalSupply() should be declared external:
        - ERC20.totalSupply() (IvyPoolBase.sol#1414-1416)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (IvyPoolBase.sol#1418-1420)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (IvyPoolBase.sol#1422-1425)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (IvyPoolBase.sol#1427-1429)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (IvyPoolBase.sol#1431-1434)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (IvyPoolBase.sol#1436-1444)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (IvyPoolBase.sol#1446-1449)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (IvyPoolBase.sol#1451-1454)
getPoolData(address) should be declared external:
        - IvyPoolFactory.getPoolData(address) (IvyPoolBase.sol#2113-2128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:IvyPoolBase.sol analyzed (18 contracts with 75 detectors), 177 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Ownable.sol

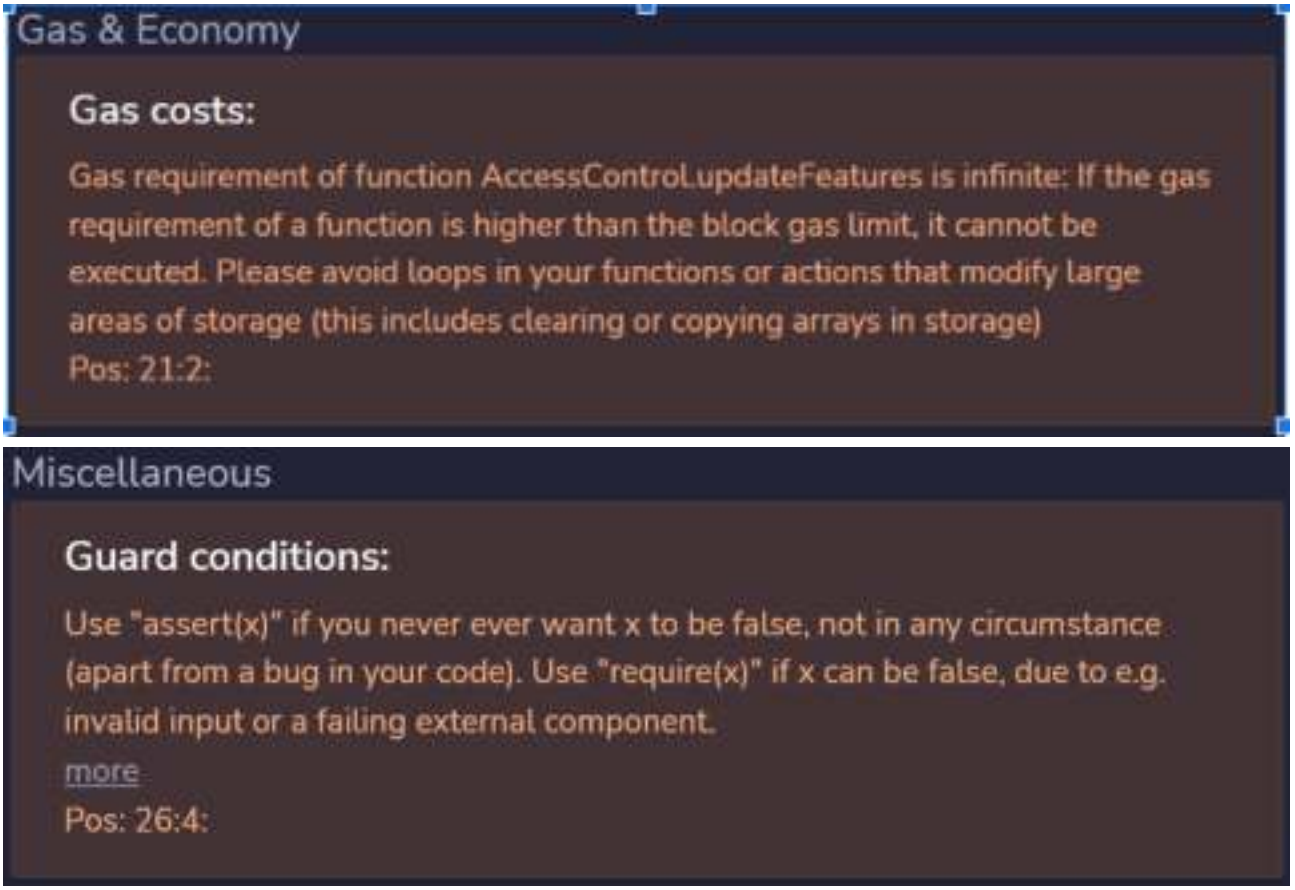
```
INFO:Detectors:
Pragma version0.8.0 (Ownable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Ownable.sol#52-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Ownable.sol#61-65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Ownable.sol analyzed (1 contracts with 75 detectors), 4 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ReentrancyGuard.sol

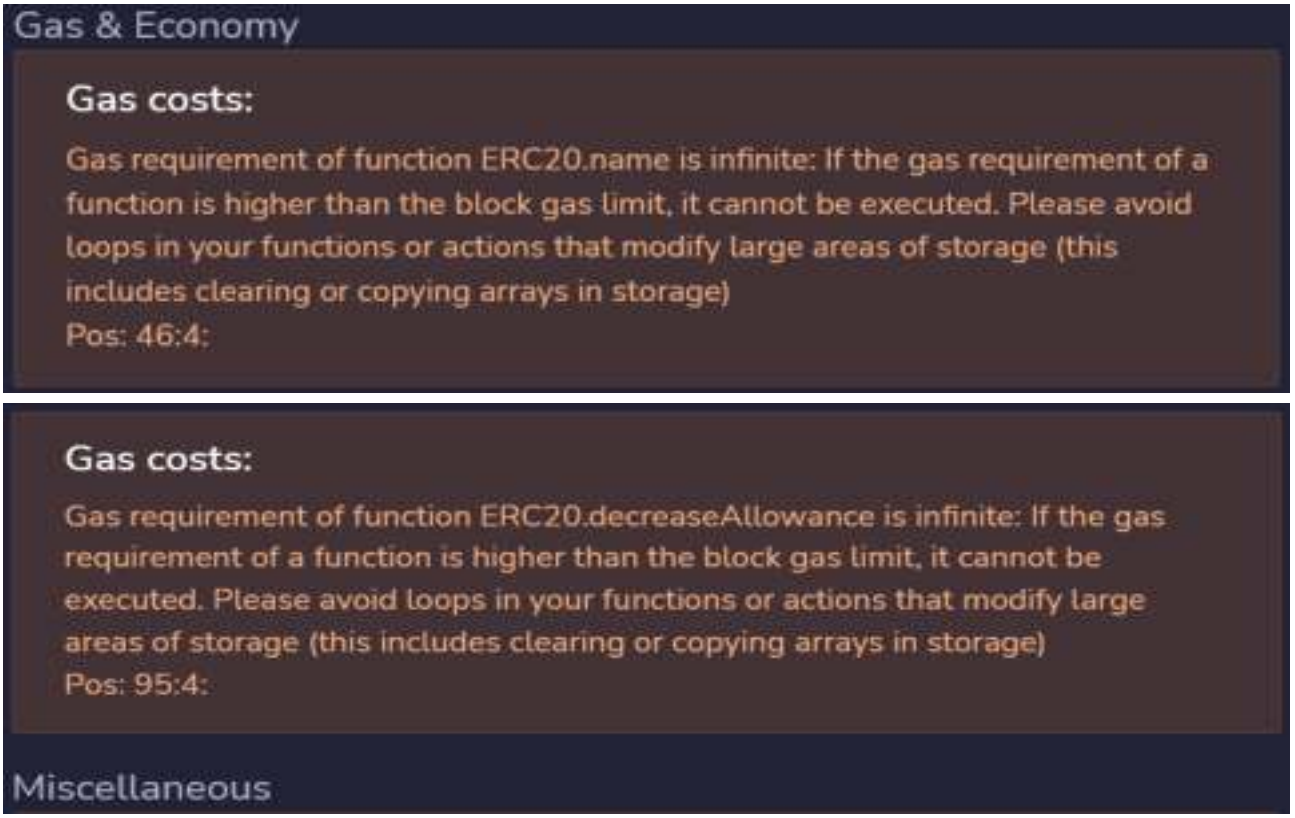
```
INFO:Detectors:
Pragma version0.8.0 (ReentrancyGuard.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.
7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Slither:ReentrancyGuard.sol analyzed (1 contracts with 75 detectors), 2 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

AccessControl.sol



ERC20.sol



Constant/View/Pure functions:

ERC20._beforeTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 152:4:

Similar variable names:

ERC20.(string,string) : Variables have very similar names "_name" and "name_".
Pos: 41:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 142:8:

EscrowedIvyERC20.sol

Gas & Economy

Gas costs:

Gas requirement of function EscrowedIvyERC20.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: 222:2:

Miscellaneous

Constant/View/Pure functions:

ERC20._beforeTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 151:4:

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: 218:4:

IvyAware.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in IvyAware.(address): Could potentially lead to re-entrancy vulnerability.

more

Pos: 1265:6:

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: 1082:16:

Gas & Economy

Gas costs:

Gas requirement of function IvyERC20.delegateWithSig 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: 1063:2:

Miscellaneous

Guard conditions:

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

more

Pos: 1268:8:

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: 1216:26:

IvyCorePool.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in IvyCorePool._processVaultRewards(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 3055:8:

Gas & Economy

Gas costs:

Gas requirement of function IvyCorePool.stakeAsPool 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: 2961:8:

Miscellaneous

Constant/View/Pure functions:

IvyPoolBase.pendingYieldRewards(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2376:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 3061:12:

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: 2586:16:

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: 2768:19:

IvyERC20.sol

Security

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: 1081:16:

Gas & Economy

Gas costs:

Gas requirement of function IvyERC20.delegateWithSig 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: 1062:2:

Miscellaneous

Guard conditions:

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

more

Pos: 1081:8:

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: 1215:26:

IvyPoolBase.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in IvyPoolBase._stake(address,uint256,uint64,bool,bool): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 2493:8:

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: 2794:19:

Gas & Economy

Gas costs:

Gas requirement of function IvyCorePool.stake 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: 2434:8:

Miscellaneous

Constant/View/Pure functions:

IvyPoolBase.pendingYieldRewards(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2392:8:

Similar variable names:

IvyPoolBase.pendingYieldRewards(address) : Variables have very similar names "user" and "users". Note: Modifiers are currently not considered by this static analysis.
Pos: 2413:31:

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: 2502:12:

IvyPoolFactory.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in IvyPoolFactory.changePoolWeight(address,uint32): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1912:8:

Gas & Economy

Gas costs:

Gas requirement of function IvyPoolFactory.updateIVYPerBlock 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: 1875:8:

Miscellaneous

Constant/View/Pure functions:

IvyPoolFactory.createPool(address,address,uint64,uint32) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1805:8:

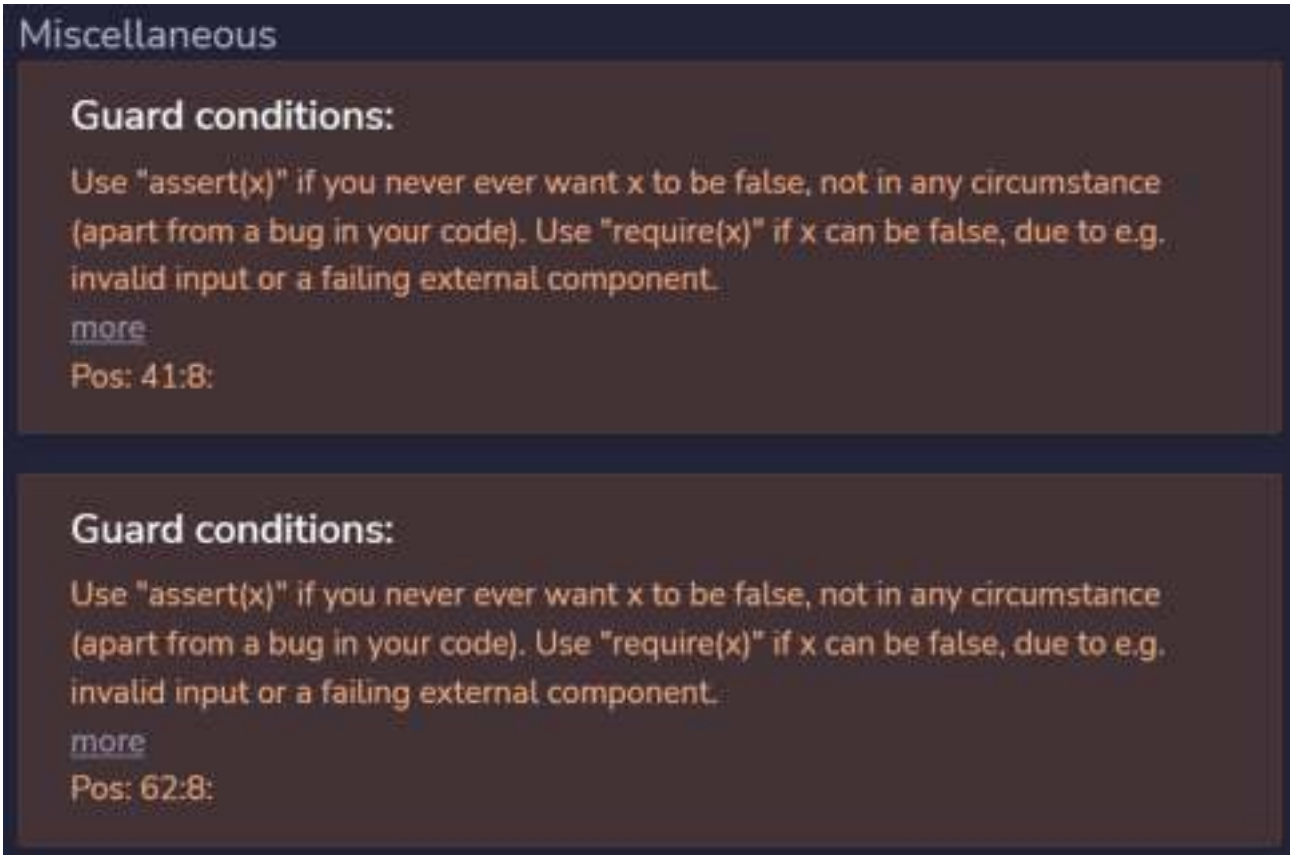
Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1914:12:

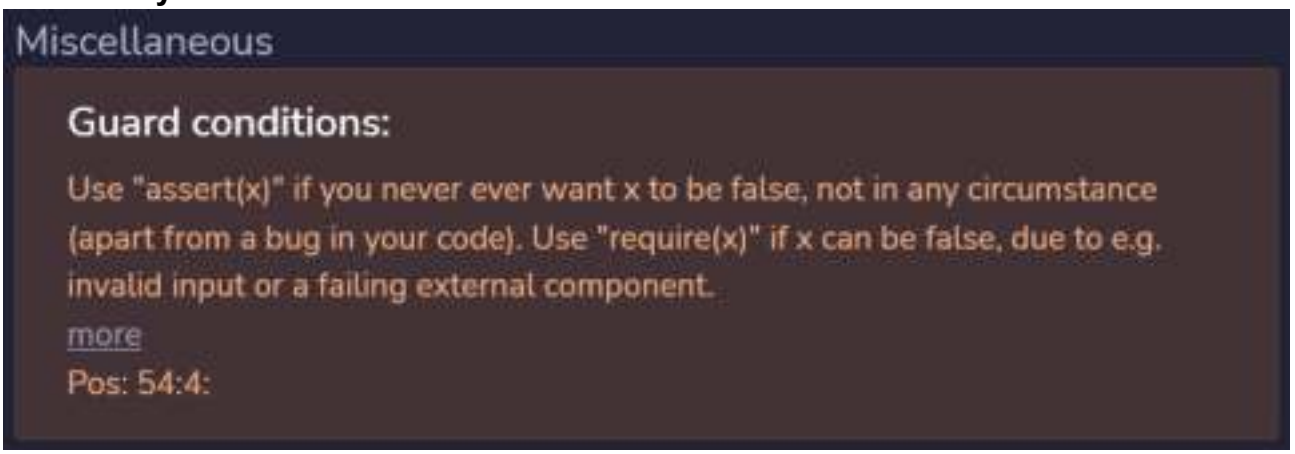
Data truncated:

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: 1879:26:

Ownable.sol



ReentrancyGuard.sol



Solhint Linter

AccessControl.sol

```
AccessControl.sol:2:1: Error: Compiler version 0.8.0 does not satisfy
the r semver requirement
AccessControl.sol:13:3: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

ERC20.sol

```
ERC20.sol:3:1: Error: Compiler version 0.8.0 does not satisfy the r
semver requirement
ERC20.sol:40:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
ERC20.sol:156:24: Error: Code contains empty blocks
```

EscrowedIvyERC20.sol

```
EscrowedIvyERC20.sol:2:1: Error: Compiler version 0.8.0 does not
satisfy the r semver requirement
EscrowedIvyERC20.sol:39:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
EscrowedIvyERC20.sol:155:24: Error: Code contains empty blocks
EscrowedIvyERC20.sol:167:3: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
```

IvyAware.sol

```
IvyAware.sol:3:1: Error: Compiler version >=0.8.0 does not satisfy
the r semver requirement
IvyAware.sol:32:3: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
IvyAware.sol:97:5: Error: Avoid using inline assembly. It is
acceptable only in rare cases
IvyAware.sol:126:26: Error: Constant name must be in capitalized
SNAKE_CASE
IvyAware.sol:138:26: Error: Constant name must be in capitalized
SNAKE_CASE
IvyAware.sol:153:25: Error: Constant name must be in capitalized
SNAKE_CASE
IvyAware.sol:453:3: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
IvyAware.sol:592:38: Error: Code contains empty blocks
```

```
IvyAware.sol:582:73: Error: Variable "_data" is unused
IvyAware.sol:1082:13: Error: Avoid to make time-based decisions in
your business logic
IvyAware.sol:1265:3: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

IvyCorePool.sol

```
IvyCorePool.sol:2:1: Error: Compiler version 0.8.0 does not satisfy
the r semver requirement
IvyCorePool.sol:308:5: Error: Avoid using inline assembly. It is
acceptable only in rare casesIvyCorePool.sol:659:3: Error: Explicitly
mark visibility in function (Set ignoreConstructors to true if using
solidity >=0.7.0)
IvyCorePool.sol:726:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
IvyCorePool.sol:815:25: Error: Constant name must be in capitalized
SNAKE_CASE
IvyCorePool.sol:1115:3: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity
>=0.7.0)IvyCorePool.sol:1217:73: Error: Variable "_data" is unused
IvyCorePool.sol:1717:13: Error: Avoid to make time-based decisions in
your business logic
IvyCorePool.sol:2336:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
IvyCorePool.sol:2778:16: Error: Avoid to make time-based decisions in
your business logic
IvyCorePool.sol:2803:35: Error: Constant name must be in capitalized
SNAKE_CASE
IvyCorePool.sol:2855:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
IvyCorePool.sol:2863:89: Error: Code contains empty blocks
```

IvyERC20.sol

```
IvyERC20.sol:2:1: Error: Compiler version >=0.8.0 does not satisfy
the r semver requirement
IvyERC20.sol:31:3: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
IvyERC20.sol:96:5: Error: Avoid using inline assembly. It is
acceptable only in rare cases
IvyERC20.sol:152:25: Error: Constant name must be in capitalized
SNAKE_CASE
IvyERC20.sol:452:3: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
IvyERC20.sol:591:38: Error: Code contains empty blocks
IvyERC20.sol:581:73: Error: Variable "_data" is unused
IvyERC20.sol:1081:13: Error: Avoid to make time-based decisions in
your business logic
```

IvyPoolBase.sol

```
IvyPoolBase.sol:2:1: Error: Compiler version 0.8.0 does not satisfy
the r semver requirement
IvyPoolBase.sol:32:3: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
IvyPoolBase.sol:97:5: Error: Avoid using inline assembly. It is
acceptable only in rare cases
IvyPoolBase.sol:116:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
IvyPoolBase.sol:205:25: Error: Constant name must be in capitalized
SNAKE_CASE
IvyPoolBase.sol:505:3: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
IvyPoolBase.sol:644:38: Error: Code contains empty blocks
IvyPoolBase.sol:634:73: Error: Variable "_data" is unused
IvyPoolBase.sol:1134:13: Error: Avoid to make time-based decisions in
your business logic
IvyPoolBase.sol:2352:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
IvyPoolBase.sol:2794:16: Error: Avoid to make time-based decisions in
your business logic
IvyPoolBase.sol:2819:35: Error: Constant name must be in capitalized
SNAKE_CASE
IvyPoolBase.sol:2871:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
IvyPoolBase.sol:2879:89: Error: Code contains empty blocks
```

IvyPoolFactory.sol

```
IvyPoolFactory.sol:2:1: Error: Compiler version 0.8.0 does not
satisfy the r semver requirement

IvyPoolFactory.sol:31:3: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)

IvyPoolFactory.sol:96:5: Error: Avoid using inline assembly. It is
acceptable only in rare cases

IvyPoolFactory.sol:115:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)

IvyPoolFactory.sol:204:25: Error: Constant name must be in
capitalized SNAKE_CASE

IvyPoolFactory.sol:504:3: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)

IvyPoolFactory.sol:643:38: Error: Code contains empty blocks

IvyPoolFactory.sol:633:73: Error: Variable "_data" is unused

IvyPoolFactory.sol:1133:13: Error: Avoid to make time-based decisions
in your business logic
```

```
IvyPoolFactory.sol:1708:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)

IvyPoolFactory.sol:1810:34: Error: Code contains empty blocks
```

Ownable.sol

```
Ownable.sol:2:1: Error: Compiler version 0.8.0 does not satisfy the r
semver requirement
Ownable.sol:24:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
```

ReentrancyGuard.sol

```
ReentrancyGuard.sol:6:1: Error: Compiler version 0.8.0 does not
satisfy the r semver requirement
ReentrancyGuard.sol:41:3: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
```

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