INSPECTOR
LOVELY

SMART CONTRACT

SECURITY AUDIT

MOBLAND

inspector.lovely.finance

TABLE OF CONTENTS

DISCLAIMER

This is a comprehensive report based on our automated and manual examination of cybersecurity vulnerabilities and framework flaws of the project's smart contract. Reading the full analysis report is essential to build your understanding of project's security level. It is crucial to take note, though we have done our best to perform this analysis and report, that you should not rely on the our research and cannot claim what it states or how we created it. Before making any judgments, you have to conduct your own independent research. We will discuss this in more depth in the following disclaimer - please read it fully. DISCLAIMER: You agree to the terms of this disclaimer by reading this report or any portion thereof. Please stop reading this report and remove and delete any copies of this report that you download and/or print if you do not agree to these conditions. Scan and verify report's presence in the GitHub repository by a qr-code on the title page. This report is for non-reliability information only and does not represent investment advice. No one shall be entitled to depend on the report or its contents, and Inspector Lovely and its affiliates shall not be held responsible to you or anyone else, nor shall Inspector Lovely provide any guarantee or representation to any person with regard to the accuracy or integrity of the report. Without any terms, warranties or other conditions other than as set forth in that exclusion and Inspector Lovely excludes hereby all representations, warrants, conditions and other terms (including, without limitation, guarantees implied by the law of satisfactory quality, fitness for purposes and the use of reasonable care and skills). The report is provided as "as is" and does not contain any terms and conditions. Except as legally banned, Inspector Lovely disclaims all responsibility and responsibilities and no claim against Inspector Lovely is made to any amount or type of loss or damages (without limitation, direct, indirect, special, punitive, consequential or pure economic loses or losses) that may be caused by you or any other person, or any damages or damages, including without limitations (whether innocent or negligent). Security analysis is based only on the smart contracts. No applications or operations were reviewed for security. No product code has been reviewed.

AUDIT SCOPE

Name	Code Review and Security Analysis Report fo Mobland Protocol Smart Contracts
Platform	BSC / Solidity
File 1	NftFactory.sol
File 2	SuperpowerNFT.sol
File 3	SuperpowerNFTBase.soll
File 4	WhitelistSlot.sol
File 5	Farm.sol
File 6	FarmBridged.sol
File 7	Turf.sol
File 8	TurfBridged.sol
File 9	WormholeCommon.sol
File 10	Wormhole721.sol
File 11	WormholeTunnel.sol
File 12	GamePool.sol
File 13	Signable.sol
File 14	SignableStakes.sol
File 15	SideToken.sol
File 16	BudToken.sol
File 17	SeedToken.sol
Audit Date	February 6th,2023
Revision Date	February 8th, 2023

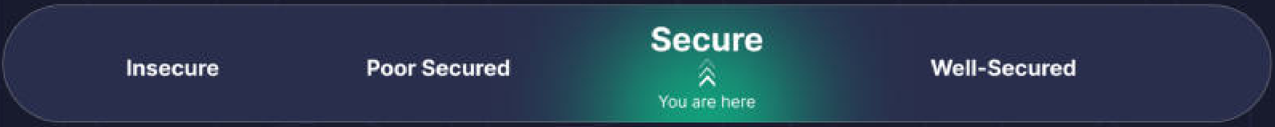
PROPOSED SMART CONTRACT FEATURES

Claimed Feature Detail	Our Observation
File 1 NftFactory.sol • Owner can set a whitelist address. • Owner can Withdraw proceeds. • Owner can update the prices of an existing running Sale. • Owner can create a new Sale for an NFT and update an existing Sale.	Validated
File 2 SuperpowerNFT.sol • Owner can set the maximum supply. • Owner can mint tokens.	Validated
File 3 SuperpowerNFTBase.sol • Owner can set the game address. • Owner can set the locker address and remove the locker address. • Owner can freeze the token URI.	Validated
File 4 WhitelistSlot.sol • Owner can set a new URI	Validated
File 5 Farm.sol • Name: MOBLAND Farm • Symbol: mFARM	Validated
File 6 FarmBridged.sol • Name: MOBLAND Farm • Symbol: mFARM	Validated
File 7 Turf.sol • Name: MOBLAND Turf • Symbol: mTURF	Validated

Claimed Feature Detail	Our Observation
File 8 TurfBridged.sol • Name: MOBLAND Turf • Symbol: mTURF	Validated
File 9 WormholeCommon.sol • WormholeCommon can check if the transfer is Completed or not.	Validated
File 10 WormholeTunnel.sol • Owner can set wormhole register contract address	Validated
File 11 Wormhole721.sol • Complete a transfer from Wormhole.	Validated
File 12 GamePool.sol • The Owner can withdraw an amount of funds in SEEDS or BUDS, or all of them i the amount is 0. • he Owner can initialize the attributes of a turf token and farm token..	Validated
File 13 Signable.sol • The Owner can set a signable address.	Validated
File 14 SignableStakes.sol • SignableStakes contract can check hash unstake.	Validated
File 15 SideToken.sol • Minter can mint amounts. • The Owner can set a minter address.	Validated
File 16 BudToken.sol • Name: Mobland Bud Token • Symbol: BUD • Decimals: 18 • Version: 1	Validated
File 17 SeedToken.sol • Name: Mobland Seed Token • Symbol: SEED • Decimals: 18 • Version: 1	Validated

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, 1 high, 0 medium and 2 low and some very low level issues.

All issues have been fixed / 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.

KEY TECHNICAL METRICS

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	Moderated
	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 17 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 Mobland 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 Mobland Protocol.

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

Code parts are **not well c**ommented on smart contracts.

DOCUMENTATION

We were given a Mobland Protocol smart contract code in the form of a github link. The links of that code are 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://mob.land 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

NftFactory.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	initialize	write	Passed	No Issue
3	setWl	external	access only Owner	No Issue
4	setPaymentToken	external	access only Owner	No Issue
5	setNewNft	external	access only Owner	No Issue
6	removeNewNft	external	access only Owner	No Issue
7	getNftIdByAddress	external	Passed	No Issue
8	getNftAddressById	external	Passed	No Issue
9	getPaymentTokenSymbol	external	Passed	No Issue
10	newSale	external	Infinite loops possibility	Refer Audit Findings
11	updateSale	external	access only Owner	No Issue
12	endSale	external	access only Owner	No Issue
13	updatePrice	external	access only Owner	No Issue
14	getSale	external	Passed	No Issue
15	getPrice	read	Passed	No Issue
16	getWlPrice	read	Passed	No Issue
17	buyTokens	external	Passed	No Issue
18	withdrawProceeds	write	access only Owner	No Issue

SuperpowerNFT.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	onlyFactory	modifier	Passed	No Issue
3	canMint	modifier	Passed	No Issue
4	setDefaultPlayer	external	access only Owner	No Issue
5	setMaxSupply	external	access only Owner	No Issue
6	setFactory	external	access only Owner	No Issue
7	isFactory	read	Passed	No Issue
8	hasFactories	read	Passed	No Issue

9	canMintAmount	read	Passed	No Issue
10	mint	write	access only Factory	No Issue
11	endMinting	external	access only Owner	No Issue
12	mintEnded	external	Passed	No Issue
13	maxSupply	external	Passed	No Issue
14	nextTokenId	external	Passed	No Issue

SuperpowerNFTBase.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	onlyLocker	modifier	Passed	No Issue
3	onlyGame	modifier	Passed	No Issue
4	tokenExists	modifier	Passed	No Issue
5	__SuperpowerNFTBase_init	internal	initializer	No Issue
6	_beforeTokenTransfer	internal	Passed	No Issue
7	preInitializeAttributesFor	external	access only Owner	No Issue
8	attributesOf	external	Passed	No Issue
9	initializeAttributesFor	external	Passed	No Issue
10	updateAttributes	external	Passed	No Issue
11	supportsInterface	read	Passed	No Issue
12	_baseURI	internal	Passed	No Issue
13	updateTokenURI	external	access only Owner	No Issue
14	freezeTokenURI	external	access only Owner	No Issue
15	contractURI	read	Passed	No Issue
16	setGame	external	access only Owner	No Issue
17	locked	read	Passed	No Issue
18	lockerOf	external	Passed	No Issue
19	isLocker	read	Passed	No Issue
20	setLocker	external	Locker contract not	No Issue
21	removeLocker	external	set	No Issue
22	hasLocks	read	access only Owner	No Issue
23	lock	external	Passed	No Issue
24	unlock	external	access only Locker	No Issue

25	unlockIfRemovedLocker	external	access only Owner	No Issue
26	approve	write	Passed	No Issue
27	getApproved	read	Passed	No Issue
28	setApprovalForAll	write	Passed	No Issue
29	isApprovedForAll	read	Passed	No Issue
30	wormholeTransfer	write	Passed	No Issue

WhitelistSlot.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	setURI	write	access only Owner	No Issue
3	setBurner	write	access only Owner	No Issue
4	mintBatch	write	access only Owner	No Issue
5	mintMany	write	Infinite loops possibility	Refer Audit Findings
6	burn	write	Passed	No Issue

Farm.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	onlyFactory	modifier	Passed	No Issue
3	canMint	modifier	Passed	No Issue
4	setDefaultPlayer	external	access only Owner	No Issue
5	setMaxSupply	external	access only Owner	No Issue
6	setFactory	external	access only Owner	No Issue
7	isFactory	read	Passed	No Issue
8	hasFactories	read	Passed	No Issue
9	canMintAmount	read	Passed	No Issue
10	mint	write	access only Factory	No Issue
11	endMinting	external	access only Owner	No Issue
12	mintEnded	external	Passed	No Issue
13	maxSupply	external	Passed	No Issue
14	nextTokenId	external	Passed	No Issue

FarmBridged.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	onlyLocker	modifier	Passed	No Issue
3	onlyGame	modifier	Passed	No Issue
4	tokenExists	modifier	Passed	No Issue
5	__SuperpowerNFTBase_init	internal	initializer	No Issue
6	_beforeTokenTransfer	internal	Passed	No Issue
7	preInitializeAttributesFor	external	access only Owner	No Issue
8	attributesOf	external	Passed	No Issue
9	initializeAttributesFor	external	Passed	No Issue
10	updateAttributes	external	Passed	No Issue
11	supportsInterface	read	Passed	No Issue
12	_baseURI	internal	Passed	No Issue
13	updateTokenURI	external	access only Owner	No Issue
14	freezeTokenURI	external	access only Owner	No Issue
15	contractURI	read	Passed	No Issue
16	setGame	external	access only Owner	No Issue
17	locked	read	Passed	No Issue
18	lockerOf	external	Passed	No Issue
19	isLocker	read	Passed	No Issue
20	setLocker	external	access only Owner	No Issue
21	removeLocker	external	access only Owner	No Issue
22	hasLocks	read	Passed	No Issue
23	lock	external	access only Locker	No Issue
24	unlock	external	access only Locker	No Issue
25	unlockIfRemovedLocker	external	access only Owner	No Issue
26	approve	write	Passed	No Issue
27	getApproved	read	Passed	No Issue
28	setApprovalForAll	write	Passed	No Issue
29	isApprovedForAll	read	Passed	No Issue
30	wormholeTransfer	write	Passed	No Issue

Turf.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	onlyFactory	modifier	Passed	No Issue
3	canMint	modifier	Passed	No Issue
4	setDefaultPlayer	external	access only Owner	No Issue
5	setMaxSupply	external	access only Owner	No Issue
6	setFactory	external	access only Owner	No Issue
7	isFactory	read	Passed	No Issue
8	hasFactories	read	Passed	No Issue
9	canMintAmount	read	Passed	No Issue
10	mint	write	access only Factory	No Issue
11	endMinting	external	access only Owner	No Issue
12	mintEnded	external	Passed	No Issue
13	maxSupply	external	Passed	No Issue
14	nextTokenId	external	Passed	No Issue

TurfBridged.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	onlyLocker	modifier	Passed	No Issue
3	onlyGame	modifier	Passed	No Issue
4	tokenExists	modifier	Passed	No Issue
5	__SuperpowerNFTBase_init	internal	initializer	No Issue
6	_beforeTokenTransfer	internal	Passed	No Issue
7	preInitializeAttributesFor	external	access only Owner	No Issue
8	attributesOf	external	Passed	No Issue
9	initializeAttributesFor	external	Passed	No Issue
10	updateAttributes	external	Passed	No Issue
11	supportsInterface	read	Passed	No Issue
12	_baseURI	internal	Passed	No Issue
13	updateTokenURI	external	access only Owner	No Issue
14	freezeTokenURI	external	access only Owner	No Issue

15	contractURI	read	Passed	No Issue
16	setGame	external	access only Owner	No Issue
17	locked	read	Passed	No Issue
18	lockerOf	external	Passed	No Issue
19	isLocker	read	Passed	No Issue
20	setLocker	external	access only Owner	No Issue
21	removeLocker	external	access only Owner	No Issue
22	hasLocks	read	Passed	No Issue
23	lock	external	access only Locker	No Issue
24	unlock	external	access only Locker	No Issue
25	unlockIfRemovedLocker	external	access only Owner	No Issue
26	approve	write	Passed	No Issue
27	getApproved	read	Passed	No Issue
28	setApprovalForAll	write	Passed	No Issue
29	isApprovedForAll	read	Passed	No Issue
30	wormholeTransfer	write	Passed	No Issue

Wormhole721.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	supportsInterface	read	Passed	No Issue
3	wormholeTransfer	write	Passed	No Issue
4	wormholeCompleteTransfer	write	Passed	No Issue

WormholeCommon.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	isTransferCompleted	read	Passed	No Issue
3	contractByChainId	read	Passed	No Issue
4	wormhole	read	Passed	No Issue
5	chainId	read	Passed	No Issue

6	_setWormhole	internal	Passed	No Issue
7	_setChainId	internal	Passed	No Issue
8	_setTransferCompleted	internal	Passed	No Issue
9	_setContract	internal	Passed	No Issue
10	_wormholeCompleteTransfer	internal	Passed	No Issue
11	_wormholeTransferWithValue	internal	Passed	No Issue
12	_logTransfer	internal	Passed	No Issue
13	_verifyContractVM	internal	Passed	No Issue
14	_encodeTransfer	internal	Passed	No Issue
15	_parseTransfer	internal	Passed	No Issue

GamePool.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	_equalString	internal	Passed	No Issue
3	initialize	write	initializer	No Issue
4	setConf	external	access only Owner	No Issue
5	stakeAsset	external	Passed	No Issue
6	unstakeAsset	external	Passed	No Issue
7	_checkStakeState	internal	Passed	No Issue
8	getStakeIndexByTokenId	read	Passed	No Issue
9	getStakeByIndex	external	Passed	No Issue
10	getNumberOfStakes	external	Passed	No Issue
11	getUserDeposits	external	Passed	No Issue
12	getUserStakes	external	Passed	No Issue
13	_saveSignatureAsUsed	internal	Passed	No Issue
14	depositSeed	external	Passed	No Issue
15	depositBud	external	Passed	No Issue
16	depositSeedAndPayOtherUser	external	Passed	No Issue
17	_depositFT	internal	Passed	No Issue
18	depositByIndex	read	Passed	No Issue
19	numberOfDeposits	external	Passed	No Issue
20	depositById	external	Passed	No Issue
21	depositByIdAndUser	external	Passed	No Issue
22	harvest	external	Passed	No Issue

23	withdrawFT	external	access only Owner	No Issue
24	initializeTurf	external	access only Owner	No Issue
25	updateTurfAttributes	external	Passed	No Issue
26	getTurfAttributes	external	Passed	No Issue
27	initializeFarm	external	access only Owner	No Issue
28	updateFarmAttributes	external	Passed	No Issue
29	getFarmAttributes	external	Passed	No Issue
30	attributesOf	external	Passed	No Issue
31	hashDeposit	read	Passed	No Issue
32	hashDepositAndPay	read	Passed	No Issue
33	hashHarvesting	read	Passed	No Issue
34	hashFarmAttributes	read	Passed	No Issue
35	hashTurfAttributes	read	Passed	No Issue

Signable.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	access only Initializing	No Issue
3	__Ownable_init_unchained	internal	initializer	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	owner	read	Passed	No Issue
6	_checkOwner	internal	Passed	No Issue
7	renounceOwnership	write	access only Owner	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	_transferOwnership	internal	Passed	No Issue
10	__Signable_init	internal	access only Owner	No Issue
11	setValidator	external	access only Owner	No Issue
12	getValidator	external	Passed	No Issue
13	isValidator	external	Passed	No Issue
14	isSignedByValidator	read	Passed	No Issue
15	isSignedByAValidator	read	Passed	No Issue

SignableStakes.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	hashUnstake	read	Passed	No Issue
3	setValidator	external	access only Owner	No Issue
4	getValidator	external	Passed	No Issue
5	isValidator	external	Passed	No Issue
6	isSignedByValidator	read	Passed	No Issue
7	isSignedByAValidator	read	Passed	No Issue

SeedToken.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	initialize	write	initializer	No Issue
3	onlyMinter	modifier	Passed	No Issue
4	__SideToken_init	internal	initializer	No Issue
5	mint	write	access only Minter	No Issue
6	setMinter	external	access only Owner	No Issue
7	__UUPSUpgradableTemplate_init	internal	initializer	No Issue
8	_authorizeUpgrade	internal	access only Owner	No Issue

SideToken.sol

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	onlyMinter	modifier	Passed	No Issue
3	__SideToken_init	internal	initializer	No Issue
4	mint	write	access only Minter	No Issue
5	setMinter	external	Minter contract not set	No Issue

BudToken.sol

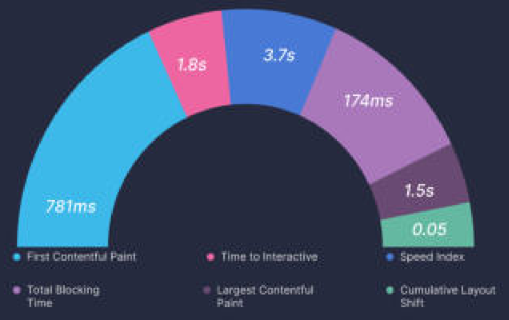
Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	initialize	write	initializer	No Issue
3	__UUPSUpgradableTemplate_init	internal	initializer	No Issue
4	_authorizeUpgrade	internal	access only Owner	No Issue
5	onlyMinter	modifier	Passed	No Issue
6	__SideToken_init	internal	initializer	No Issue
7	mint	write	access only Minter	No Issue

INSPECTOR
LOVELY

PROJECT WEBSITE PERFORMANCE AUDIT

Performance Metrics



Browser Timings

● Redirect Duration 0ms	● Connection Duration 263ms	● Backend Duration 74ms
● Time to First Byte 337ms	● First Paint 356ms	● DOM Interactive Time 782ms
● DOM Content Loaded 1.2s	● Onload Time 1.2s	● Fully Loaded Time 1.8s

Grade

D	Performance 81%	Structure 79%

Web Vitals

LCP 1.5s	TBT 174ms	CLS 0.05

Top Issues

IMPECT	AUDIT	
High	Avoid enormous network payloads **(LCP)**	Total size was 5.20MB

URL	SIZE
https://static.mob.land/videos/header-gangster.mp4	2.70MB
https://mob.land/bundle.min.740731643facf09bee8e.js	1.17MB
https://mob.land/80295d731c7b17b94205.webp	335KB
https://mob.land/096c2ad6e1d5eac1f58a.webp	109KB
https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.3.0/webfonts/fa-brands-400.woff2	106KB
https://mob.land/54fdc54b405b267f9a84.webp	96.6KB
https://mob.land/babda2e78cd8974da999.webp	72.0KB
https://mob.land/e69ed207d6d6f0e6a8aa.webp	67.8KB
https://mob.land/e73e5b53d5145eefb8f9.webp	59.1KB
https://mob.land/7d621e223bb9aa68fbdf.ttf	50.1KB

Level of Criticality

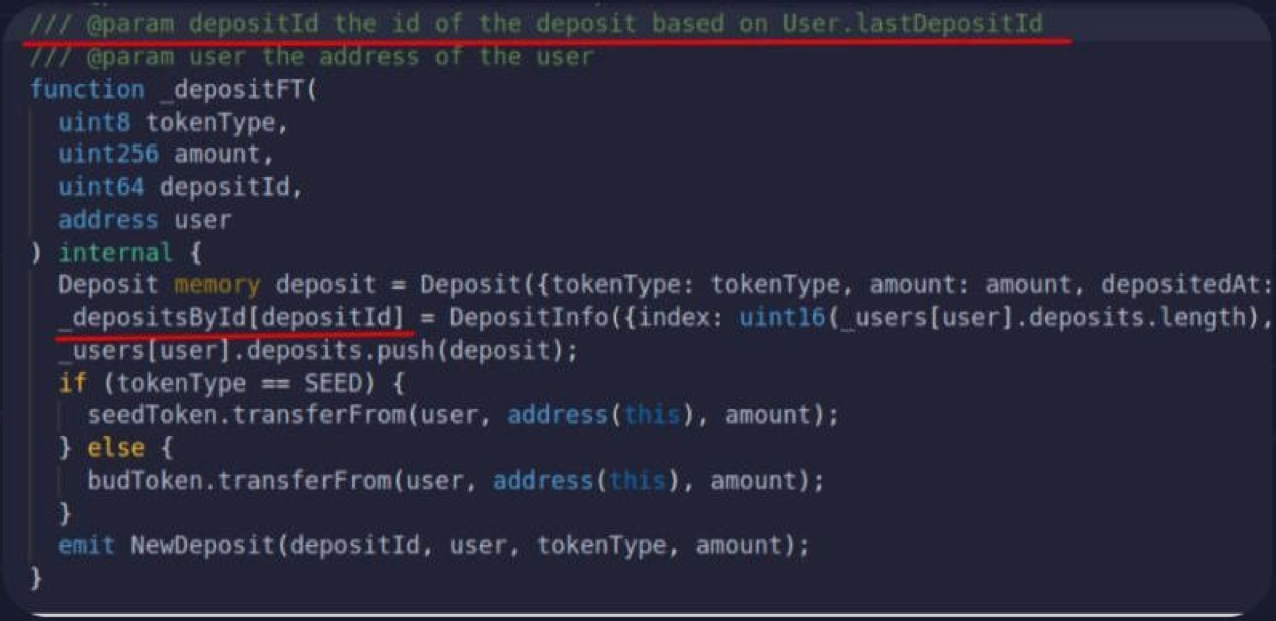
RISK LEVEL	DESCRIPTION
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Med	Medium-level 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 a 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

1. Deposit id override by any depositor: - GamePool.sol



```
/// @param depositId the id of the deposit based on User.lastDepositId
/// @param user the address of the user
function _depositFT(
  uint8 tokenType,
  uint256 amount,
  uint64 depositId,
  address user
) internal {
  Deposit memory deposit = Deposit({tokenType: tokenType, amount: amount, depositedAt:
  _depositsById[depositId] = DepositInfo({index: uint16(_users[user].deposits.length),
  _users[user].deposits.push(deposit);
  if (tokenType == SEED) {
    seedToken.transferFrom(user, address(this), amount);
  } else {
    budToken.transferFrom(user, address(this), amount);
  }
  emit NewDeposit(depositId, user, tokenType, amount);
}
```

Functions depositSeed(), depositBud(), depositSeedAndPayOtherUser() are called internal functions ""_depositFT()".

- DepositId not checked duplicate in _depositsById mapping
- DepositId has sequence issue
- "_depositFT" internal function comment says - "depositId the id of the deposit based on User.lastDepositId".

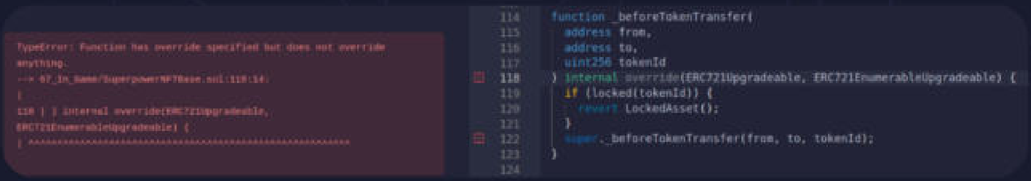
Resolution: DepositId should be auto-incremented or check duplicate deposited from mapping "_depositsById".

Status: This issue is fixed in the revised contract code.

MED	No Medium severity vulnerabilities were found.

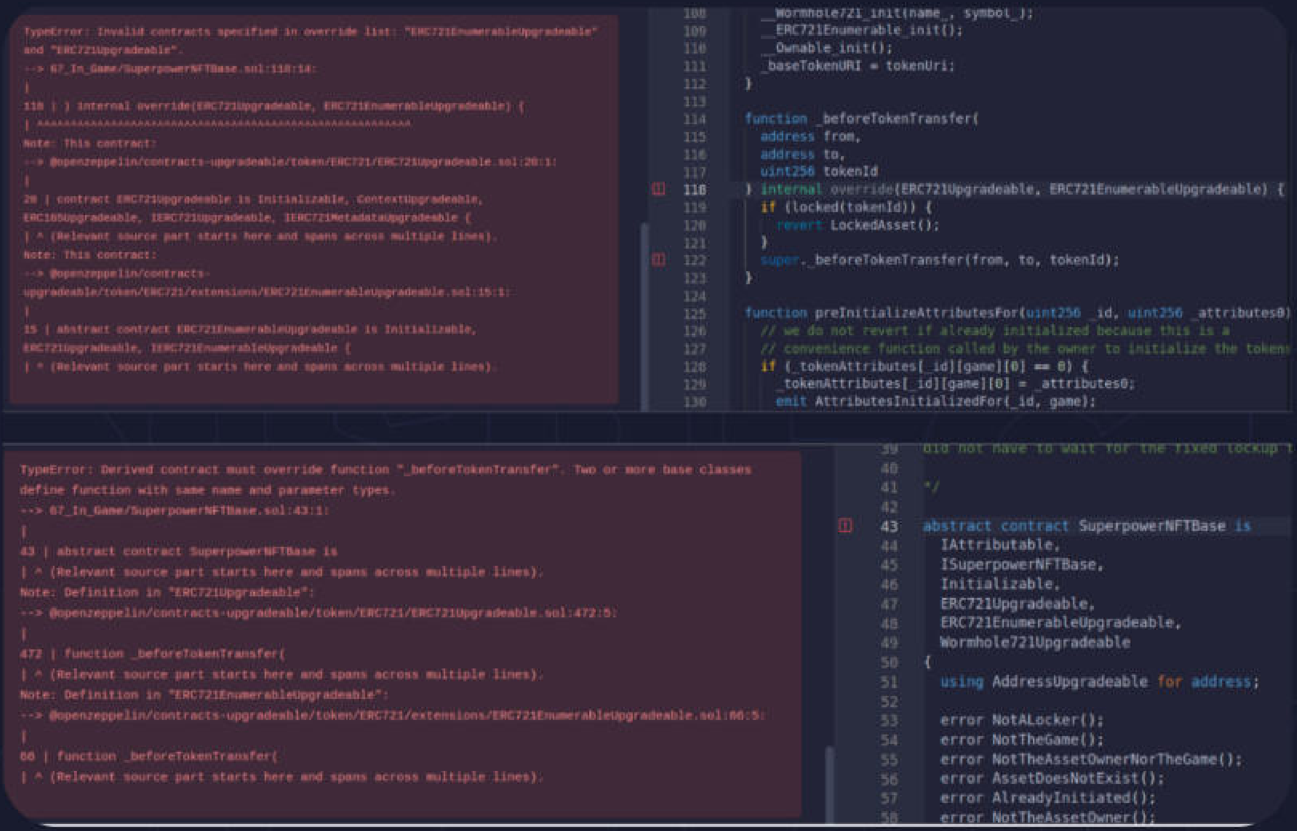
LOW

1. Compile time error: SuperpowerNFTBase.sol



```
TypeError: Function has override specified but does not override
anything.
--> 67_In_Game/SuperpowerNFTBase.sol:118:14:
|
118 | ) internal override(ERC721Upgradeable,
ERC721EnumerableUpgradeable) {
| ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
function _beforeTokenTransfer(
  address from,
  address to,
  uint256 tokenId
) internal override(ERC721Upgradeable, ERC721EnumerableUpgradeable) {
  if (locked(tokenId)) {
    revert LockedAsset();
  }
  super._beforeTokenTransfer(from, to, tokenId);
}
```



```
TypeError: Invalid contracts specified in override list: "ERC721EnumerableUpgradeable"
and "ERC721Upgradeable".
--> 67_In_Game/SuperpowerNFTBase.sol:118:14:
|
118 | ) internal override(ERC721Upgradeable, ERC721EnumerableUpgradeable) {
| ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: This contract:
--> @openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol:28:1:
|
28 | contract ERC721Upgradeable is Initializable, ContextUpgradeable,
ERC165Upgradeable, IERC721Upgradeable, IERC721MetadataUpgradeable {
| ^ (Relevant source part starts here and spans across multiple lines).
Note: This contract:
--> @openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol:15:1:
|
15 | abstract contract ERC721EnumerableUpgradeable is Initializable,
ERC721Upgradeable, IERC721EnumerableUpgradeable {
| ^ (Relevant source part starts here and spans across multiple lines).
```

```
      __Wormhole721_init(name_, symbol_);
      __ERC721Enumerable_init();
      __Ownable_init();
      _baseTokenURI = tokenUri;
    }

    function _beforeTokenTransfer(
      address from,
      address to,
      uint256 tokenId
    ) internal override(ERC721Upgradeable, ERC721EnumerableUpgradeable) {
      if (locked(tokenId)) {
        revert LockedAsset();
      }
      super._beforeTokenTransfer(from, to, tokenId);
    }

    function preInitializeAttributesFor(uint256 _id, uint256 _attributes0)
      // we do not revert if already initialized because this is a
      // convenience function called by the owner to initialize the tokens
      if (_tokenAttributes[_id][game][0] == 0) {
        _tokenAttributes[_id][game][0] = _attributes0;
        emit AttributesInitializedFor(_id, game);
```

```
TypeError: Derived contract must override function "_beforeTokenTransfer". Two or more base classes
define function with same name and parameter types.
--> 67_In_Game/SuperpowerNFTBase.sol:43:1:
|
43 | abstract contract SuperpowerNFTBase is
| ^ (Relevant source part starts here and spans across multiple lines).
Note: Definition in "ERC721Upgradeable":
--> @openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol:472:5:
|
472 | function _beforeTokenTransfer(
| ^ (Relevant source part starts here and spans across multiple lines).
Note: Definition in "ERC721EnumerableUpgradeable":
--> @openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol:66:5:
|
66 | function _beforeTokenTransfer(
| ^ (Relevant source part starts here and spans across multiple lines).
```

```
did not have to wait for the fixed lockup t

*/

abstract contract SuperpowerNFTBase is
  IAttributable,
  ISuperpowerNFTBase,
  Initializable,
  ERC721Upgradeable,
  ERC721EnumerableUpgradeable,
  Wormhole721Upgradeable
{
  using AddressUpgradeable for address;

  error NotALocker();
  error NotTheGame();
  error NotTheAssetOwnerNorTheGame();
  error AssetDoesNotExist();
  error AlreadyInitiated();
  error NotTheAssetOwner();
```

Function has override specified but does not override anything.

Resolution: Add uint256 4th function parameter to avoid this error.

Status: This issue is acknowledged in the revision of the contract code.

2. Critical operation lacks event log: - GamePool.sol

Missing event log for:

- stakeAsset()
- unstakeAsset()

Resolution: Please write an event log for listed events.

Status: This issue is fixed in the revised contract code

VERY LOW / INFORMATIONAL / BEST PRACTICES:

1. Unused Events, Errors, modifier, mappings :

Events are defined but not used in code.

NftFactory.sol

- FactorySetFor
- FactoryRemovedFor

Errors are defined but not used in code.

NftFactory.sol

-
- NotAFactoryForThisNFT
- FactoryNotFound
- InsufficientPayment

SuperpowerNFT.sol

- NotEnoughWLSlots
- InvalidDeadline
- WhitelistNotSetYet

SuperpowerNFTBase.sol

- AlreadyInitiated
- NotTheAssetOwner
- GamePool.sol
- onlyOnTestnet

A modifier is defined but not used.

SuperpowerNFTBase.sol

- onlyGame()

A Mappings is defined but not used

GamePool.sol

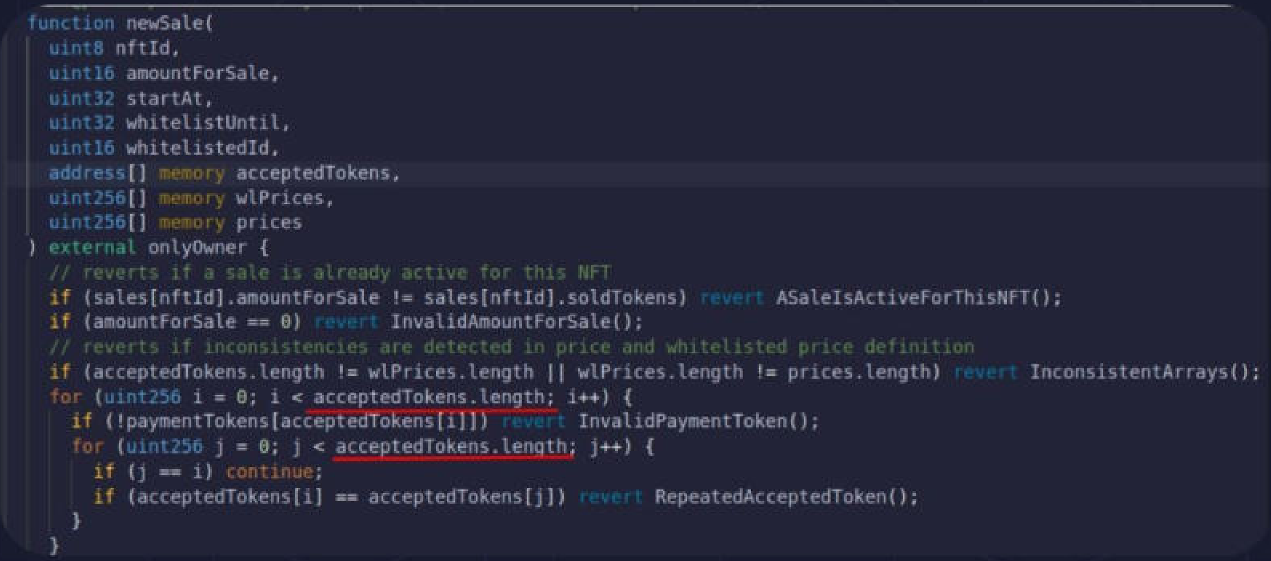
- _stakedByTokenId

Resolution: We suggest removing unused events, modifiers, mappings and errors

Status: This issue is fixed in the revised contract code

2. Infinite loops possibility:

NftFactory.sol: newSale()



```
function newSale(
  uint8 nftId,
  uint16 amountForSale,
  uint32 startAt,
  uint32 whitelistUntil,
  uint16 whitelistedId,
  address[] memory acceptedTokens,
  uint256[] memory wlPrices,
  uint256[] memory prices
) external onlyOwner {
  // reverts if a sale is already active for this NFT
  if (sales[nftId].amountForSale != sales[nftId].soldTokens) revert ASaleIsActiveForThisNFT();
  if (amountForSale == 0) revert InvalidAmountForSale();
  // reverts if inconsistencies are detected in price and whitelisted price definition
  if (acceptedTokens.length != wlPrices.length || wlPrices.length != prices.length) revert InconsistentArrays();
  for (uint256 i = 0; i < acceptedTokens.length; i++) {
    if (!paymentTokens[acceptedTokens[i]]) revert InvalidPaymentToken();
    for (uint256 j = 0; j < acceptedTokens.length; j++) {
      if (j == i) continue;
      if (acceptedTokens[i] == acceptedTokens[j]) revert RepeatedAcceptedToken();
    }
  }
```

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.

Resolution: Adjust logic to replace loops with mapping or other code structure.

NftFactory.sol

- newSale() - acceptedTokens.length

WhitelistSlot.sol

- mintMany() - ids.length

Status: This issue is fixed in the revised contract code

3. Minter contract not set: - GamePool.sol

There is a SideToken contract checking whether the minter is a Contract or not in the setMinter() function, so the minter contract is not in scope.

Resolution: A minter contract is not provided, if you provide a minter contract in the future, make sure this contract is fully secure

Status: This issue is acknowledged in the contract code.

4. Locker contract not set: - GamePool.sol

There is a TurfToken contract checking whether the locker is a Contract or not in the setLocker() function, so the Locker contract is not in scope.
This function is defined in this file - contracts/SuperpowerNFTBase.sol.

Resolution: A locker contract is not provided, if you set a locker contract in the future, make sure this contract is fully secure

Status: This issue is acknowledged in the contract code.

CENTRALIZATION

This smart contract has some functions that can be executed by the Admin (Owner) only. If the admin wallet's private key is compromised, then it would create trouble. Following are Admin functions:

NftFactory.so

- setWl: Owner can set a whitelist address.
- setPaymentToken: Owner can set, activate, or deactivate a payment token address.
- setNewNft: Owner can set a new NFT for sale.
- removeNewNft: Owner can remove an NFT from the sale.
- newSalet: Owner can create a new Sale for an NFT.
- updateSale: Owner can update an existing Sale.
- endSale: Owner can end (remove) an existing Sale.
- updatePrice: Owner can update the prices of an existing running Sale.
- withdrawProceeds: Owner can Withdraw the proceeds.

SuperpowerNFT.sol

- setDefaultPlayer: Owner can set the default player address.
- setMaxSupply: Owner can set the maximum supply.
- setFactory: Owner can set the factory address.
- mint: Owner can mint an amount.
- endMinting: Owner can handle end minting.

SuperpowerNFTBase.sol

- preInitializeAttributesFor: Owner can pre initialize attributes.
- updateTokenURI: Owner can update the token URI.
- freezeTokenURI: Owner can freeze the token URI.
- setGame: Owner can set the game address.
- setLocker: Owner can set the locker address.
- removeLocker: Owner can remove the locker address.
- unlockIfRemovedLocker: Owner can emergency unlock in case a compromised locker is removed.

WhitelistSlot.sol

- setURI: Owner can set a new URI.

Farm.sol

- setBurner: Owner can set a new burner address.
- mintBatch: Owner can mint the Batch addresses.
- mintMany: Owner can mint the many addresses.

WormholeTunnel.sol

- wormholeInit: Owner can wormhole initialize.
- wormholeRegisterContract: Owner can set a wormhole register contract address.

Signable.sol

- setValidator: Owner can set a signable validator address.

GamePool.sol

- setConf: Owner can set burning points .
- withdrawFT: Owner can withdraw an amount of funds in SEEDS or BUDS, or all of them if amount is 0.
- initializeTurf: Owner can initialize the attributes of a turf token.
- initializeFarm: Owner can initialize the attributes of a farm.

SideToken.sol

- setMinter: Owner can set a minter address.

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

CONCLUSION

We were given a contract code in the form of a github link. And we have used all possible tests based on given objects as files. We observed 1 high severity issue, 2 low severity issues, and some Informational severity issues in the smart contracts. All issues have been fixed/acknowledged in the 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 the 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.

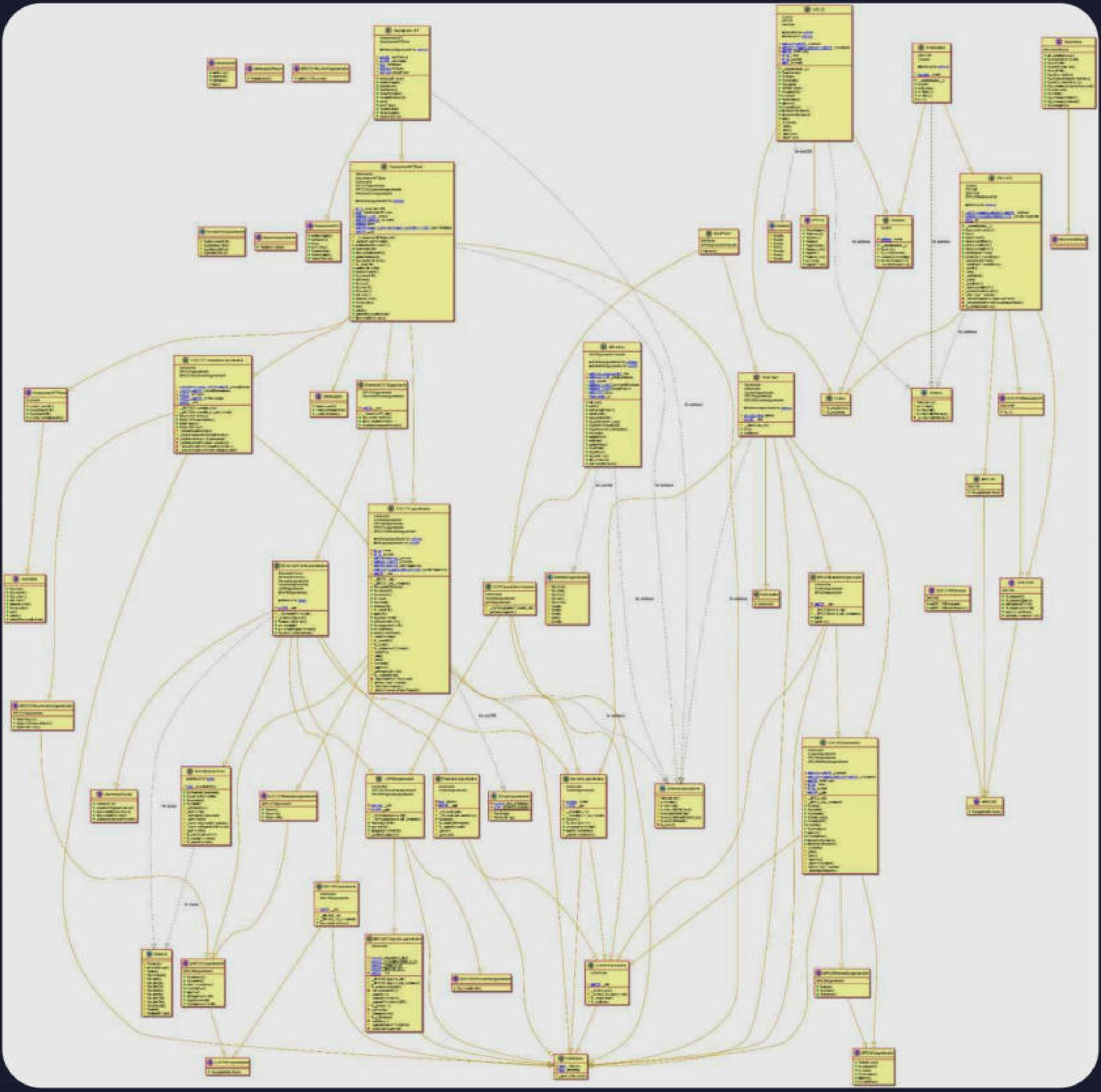
Audit report contains all found security vulnerabilities and other issues in the reviewed code.

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

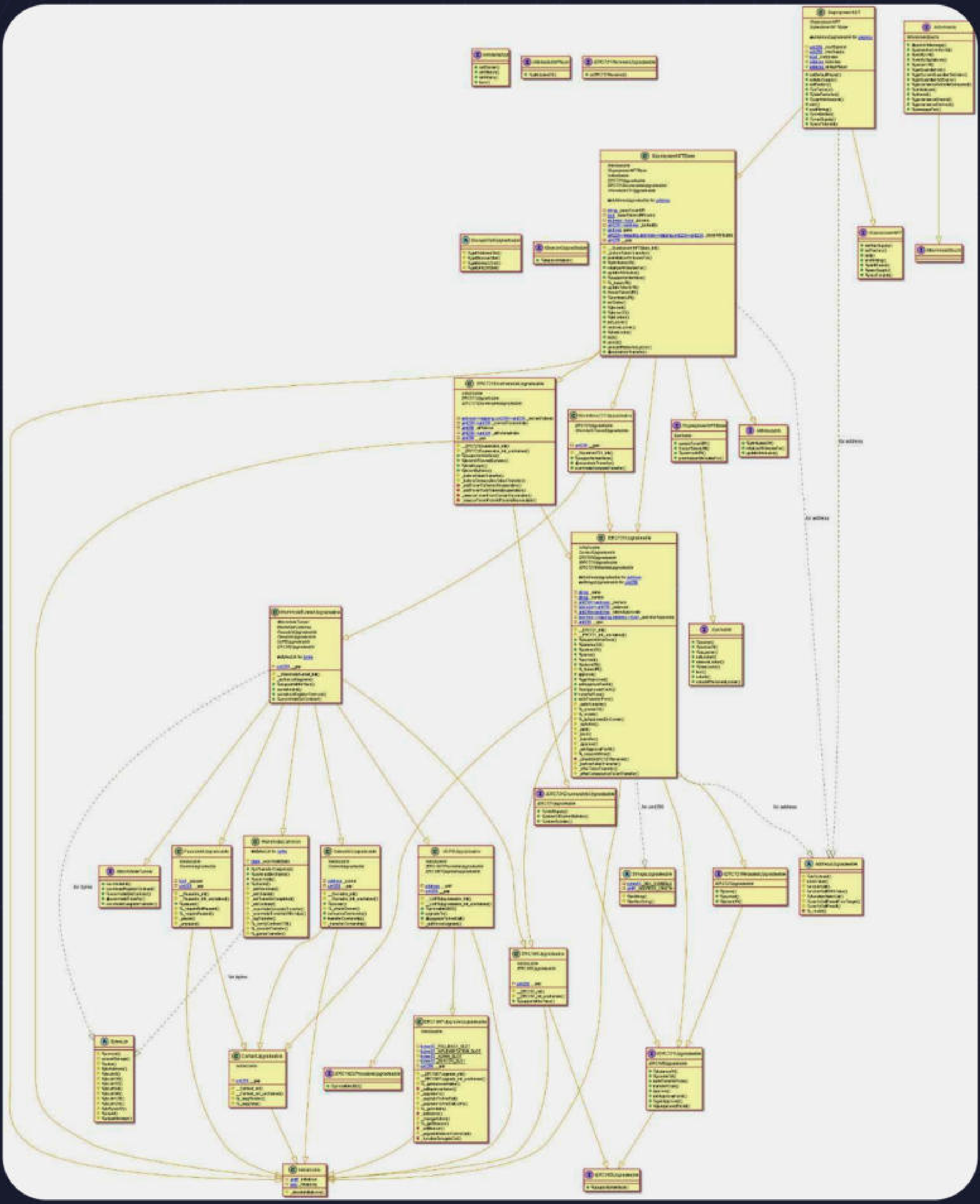
ADDENDUM

Code Flow Diagram - Mobland

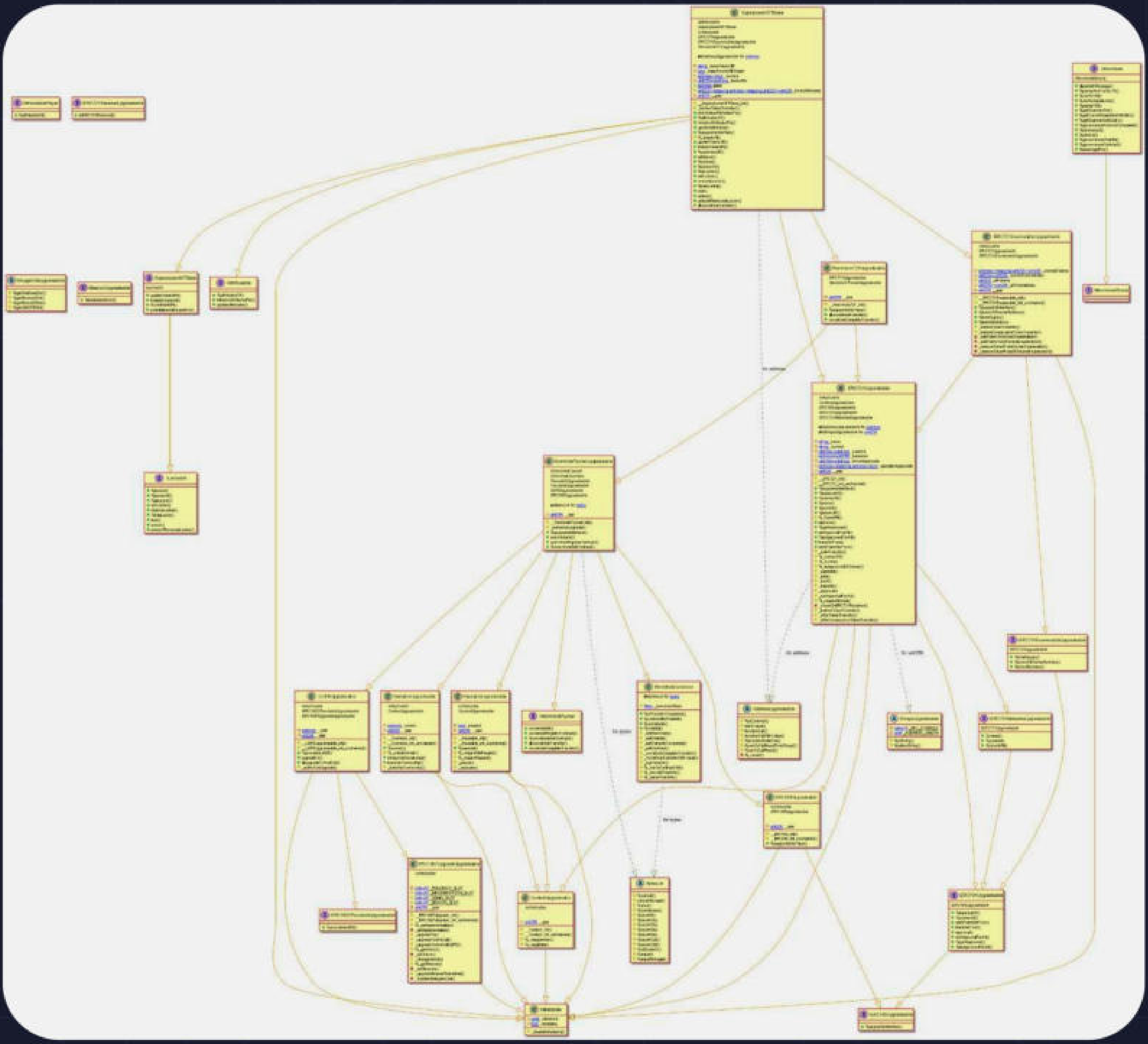
NftFactory Diagram



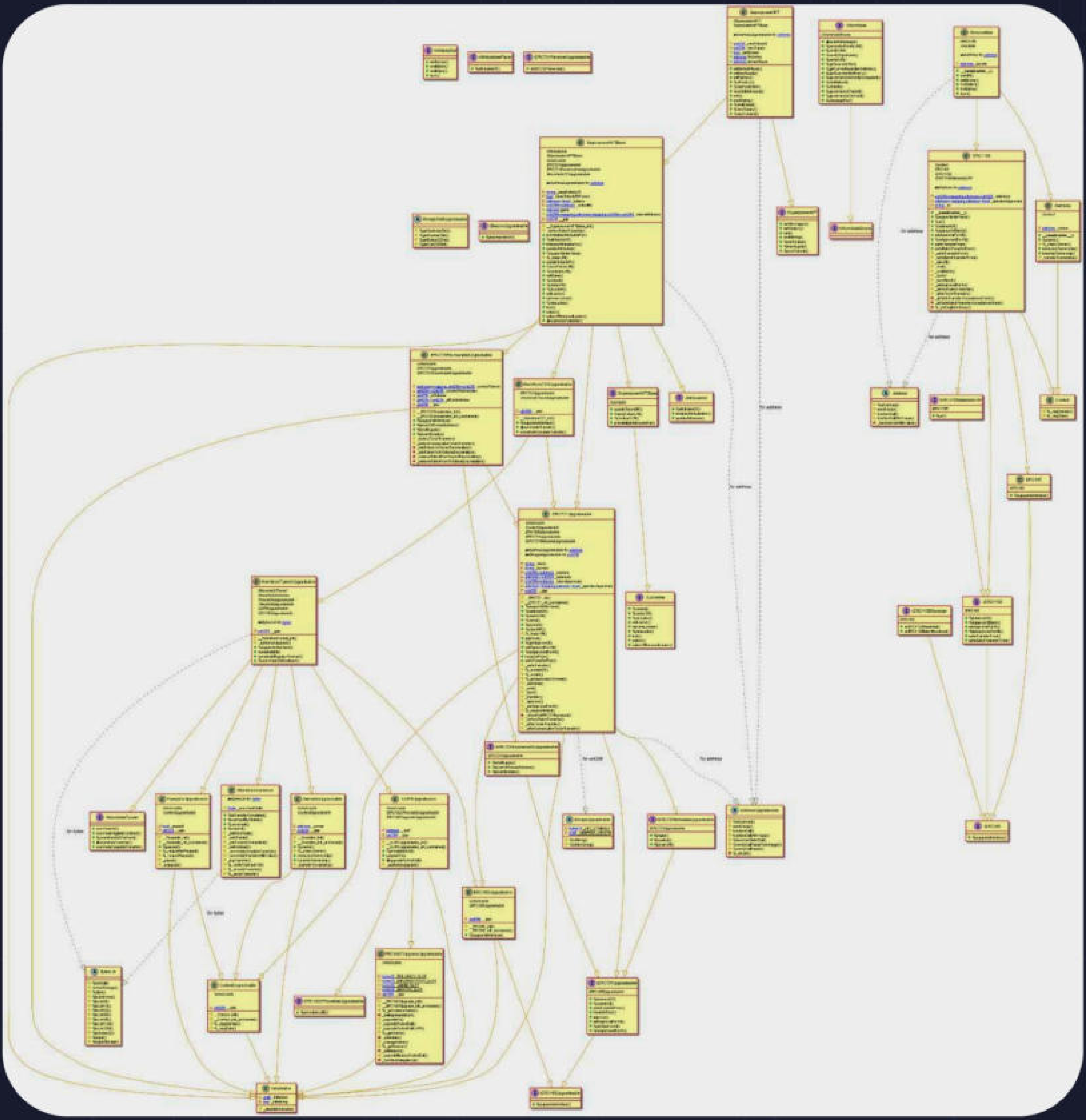
SuperpowerNFT Diagram



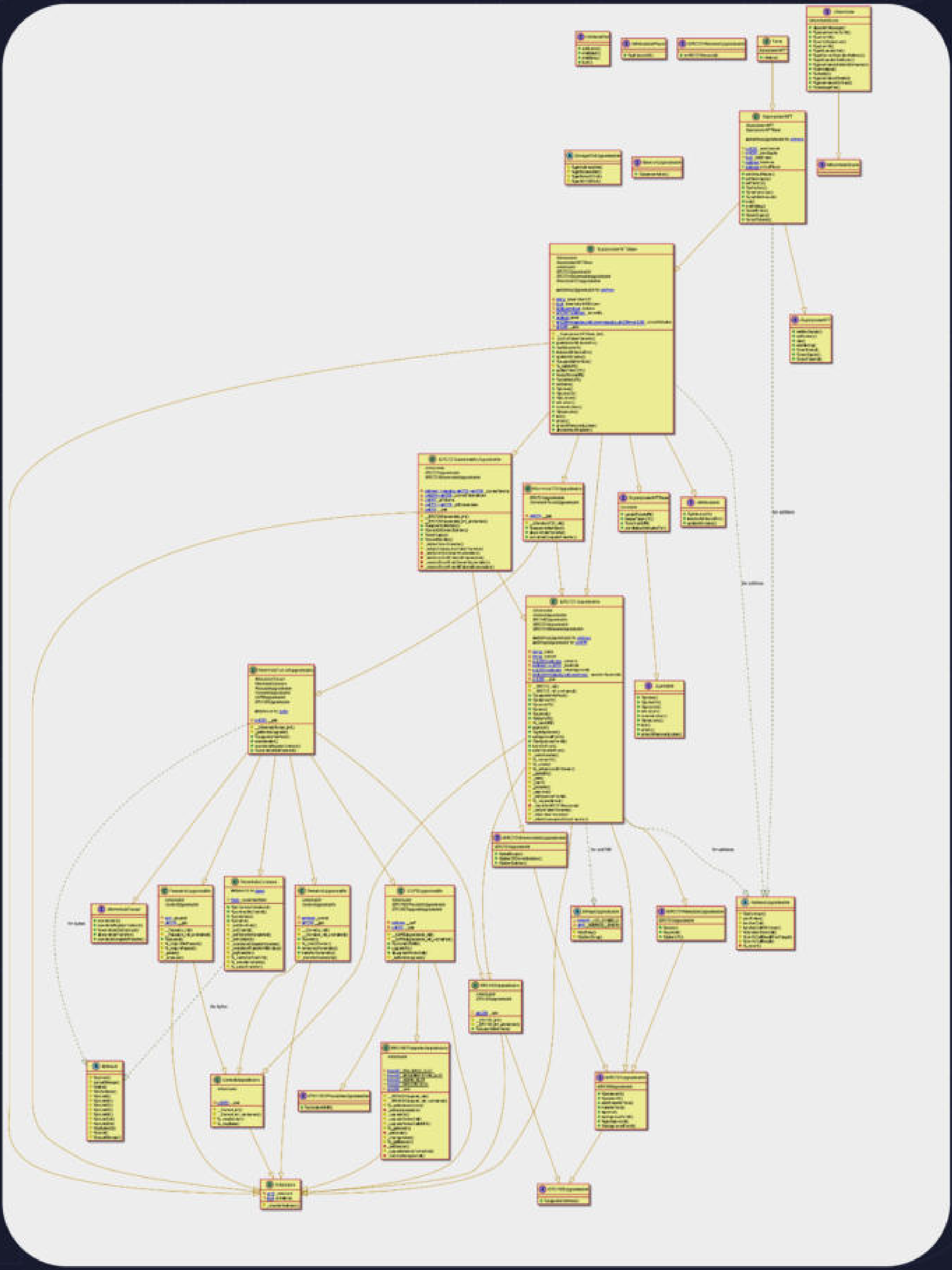
SuperpowerNFTBase Diagram



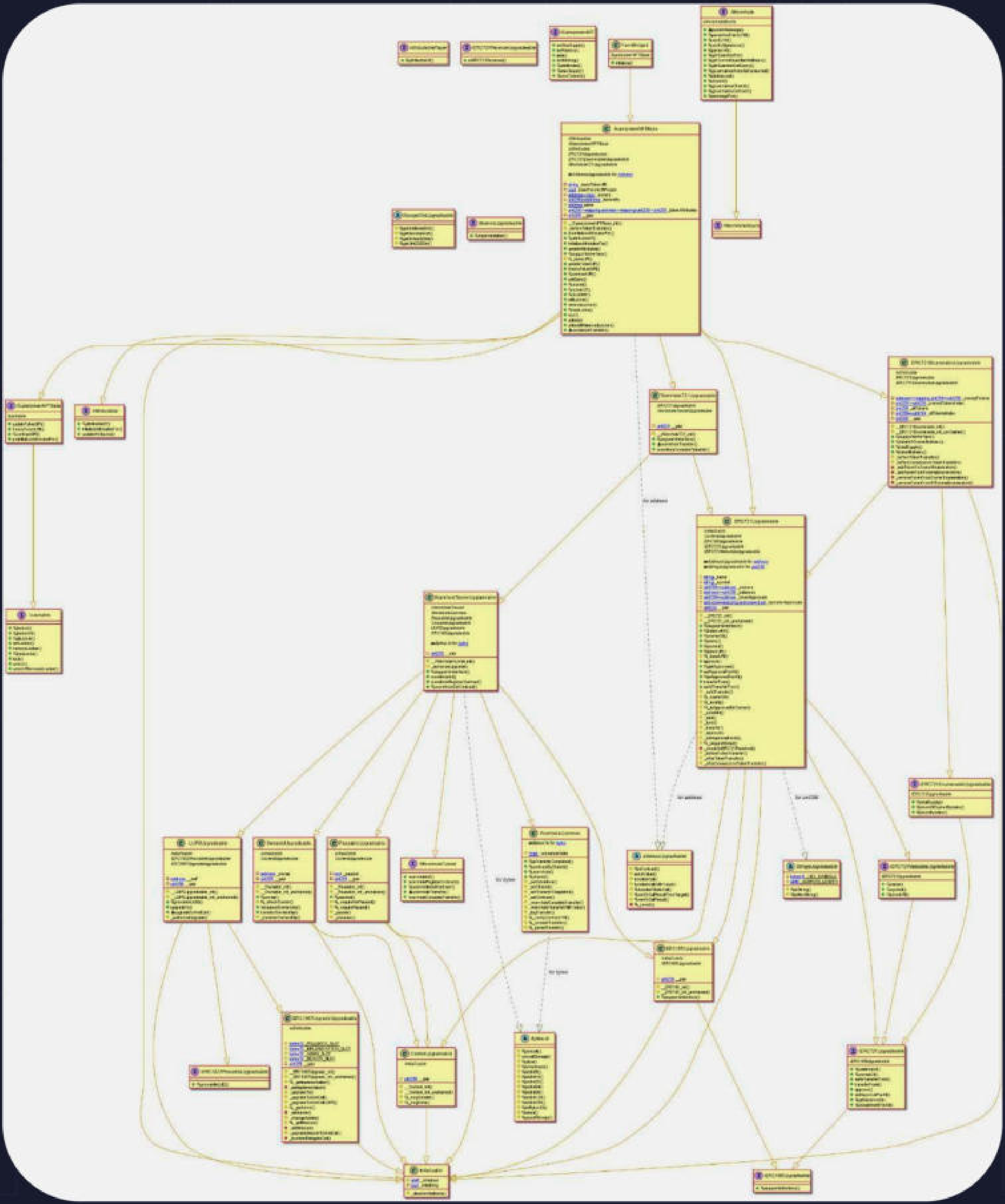
WhitelistSlot Diagram



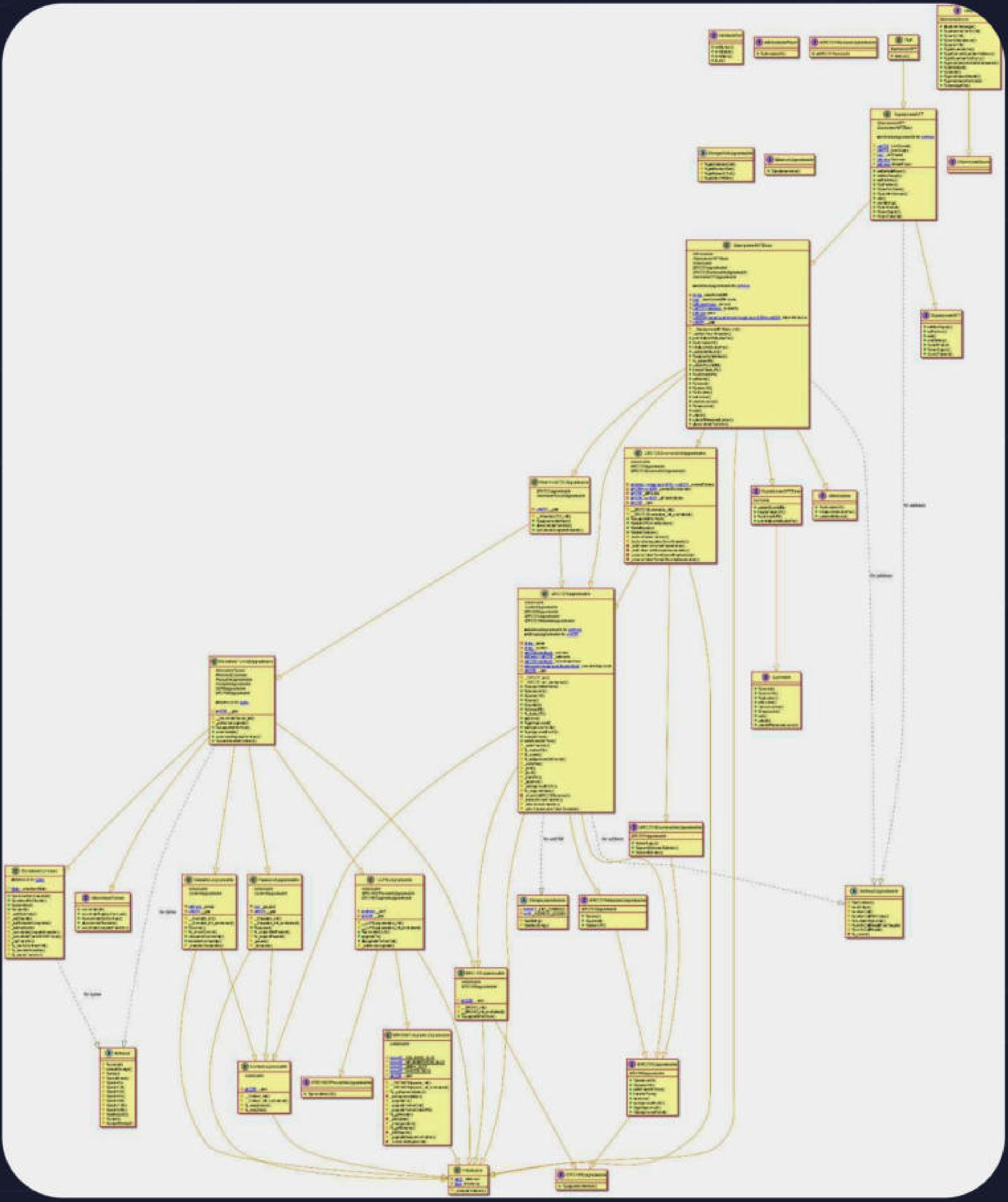
Farm Diagram



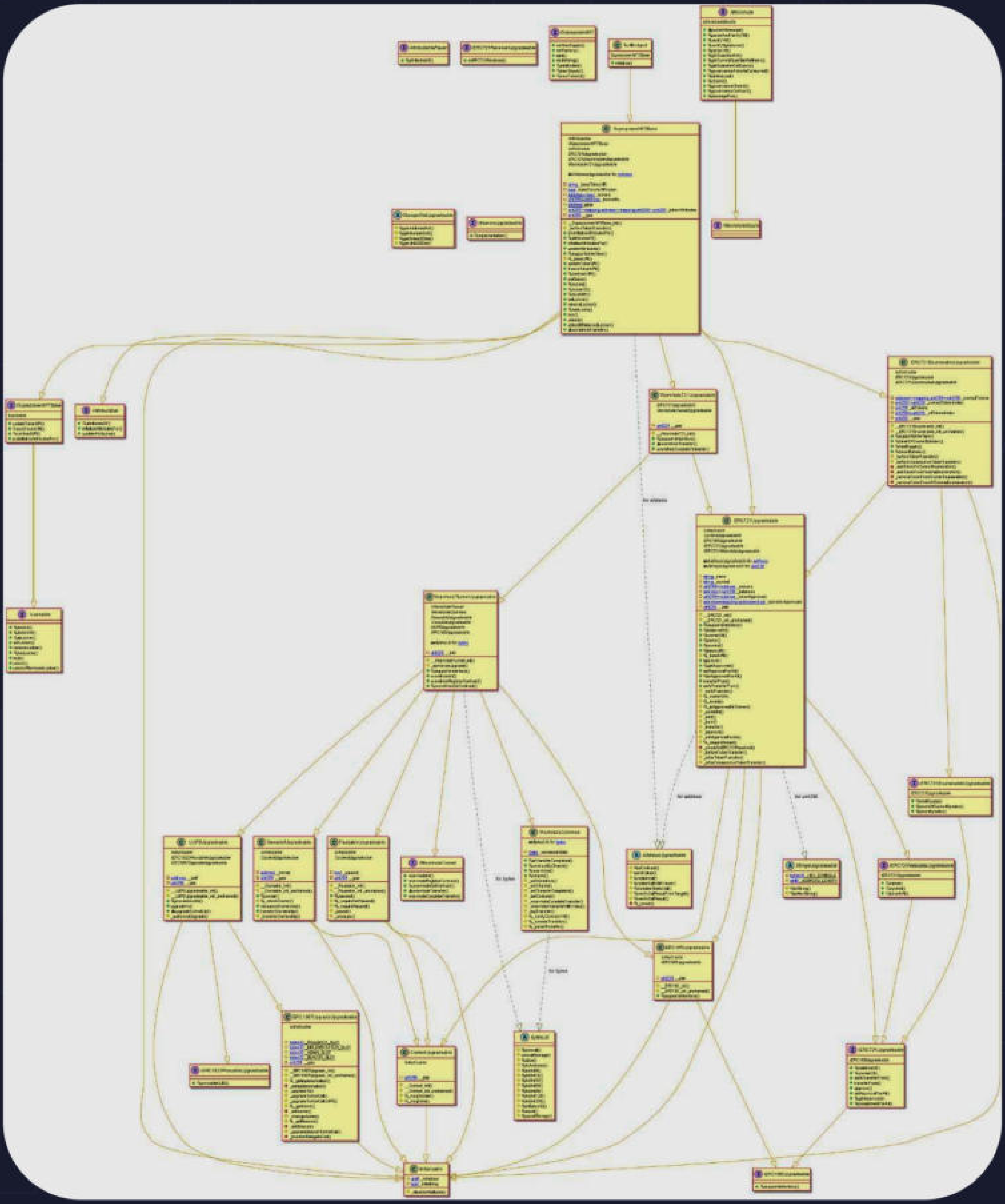
FarmBridged Diagram



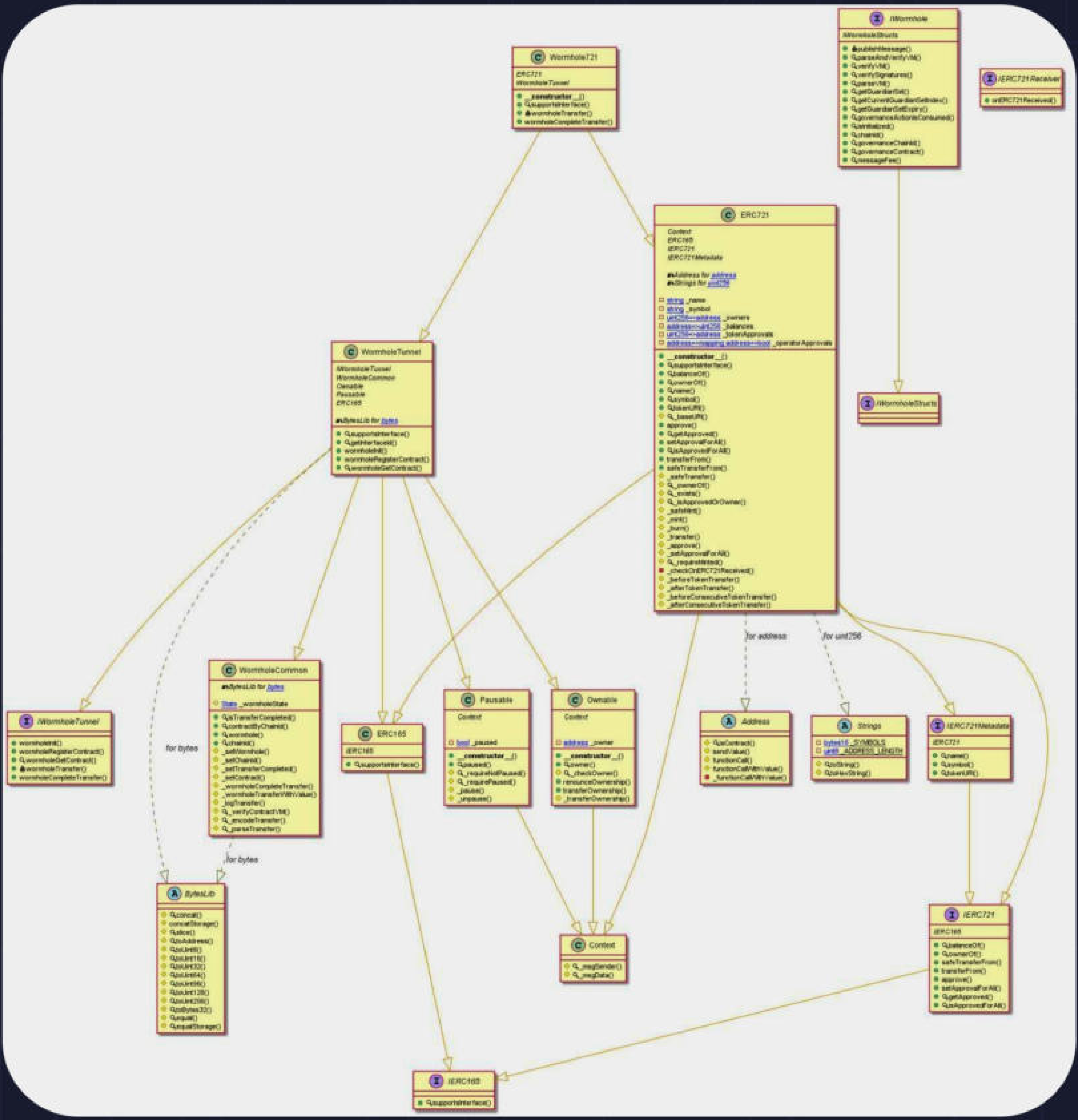
Turf Diagram



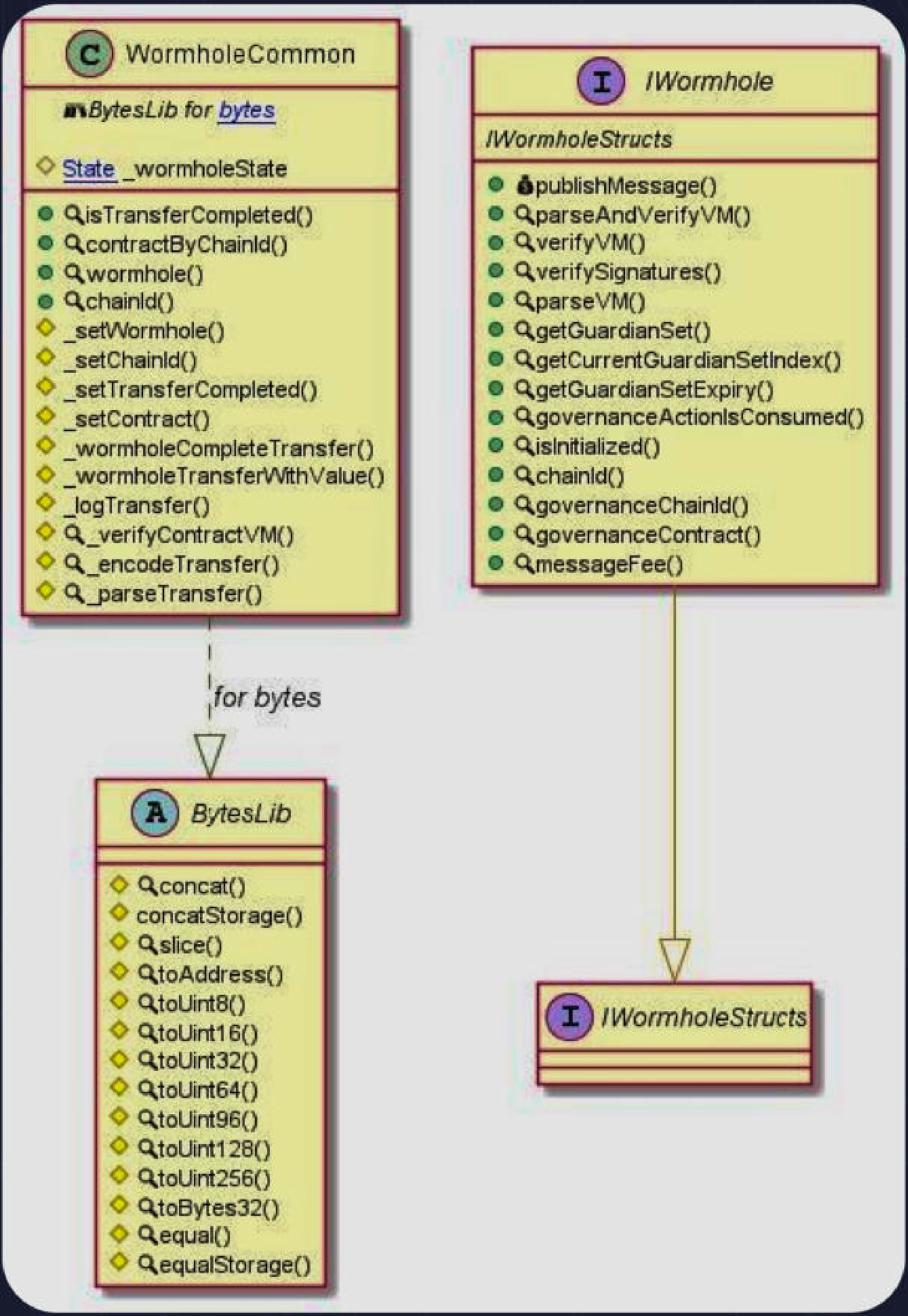
TurfBridged Diagram



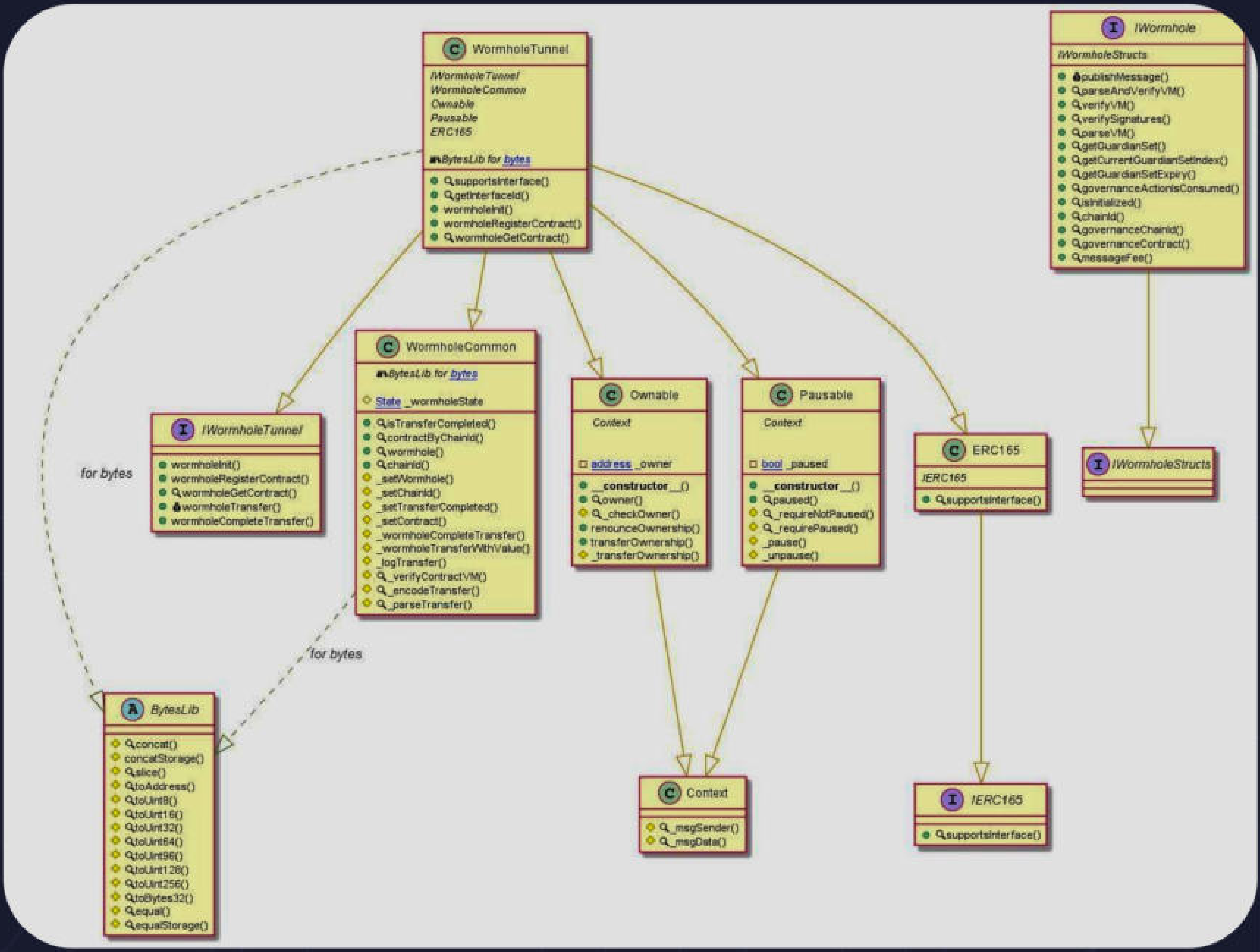
Wormhole721 Diagram



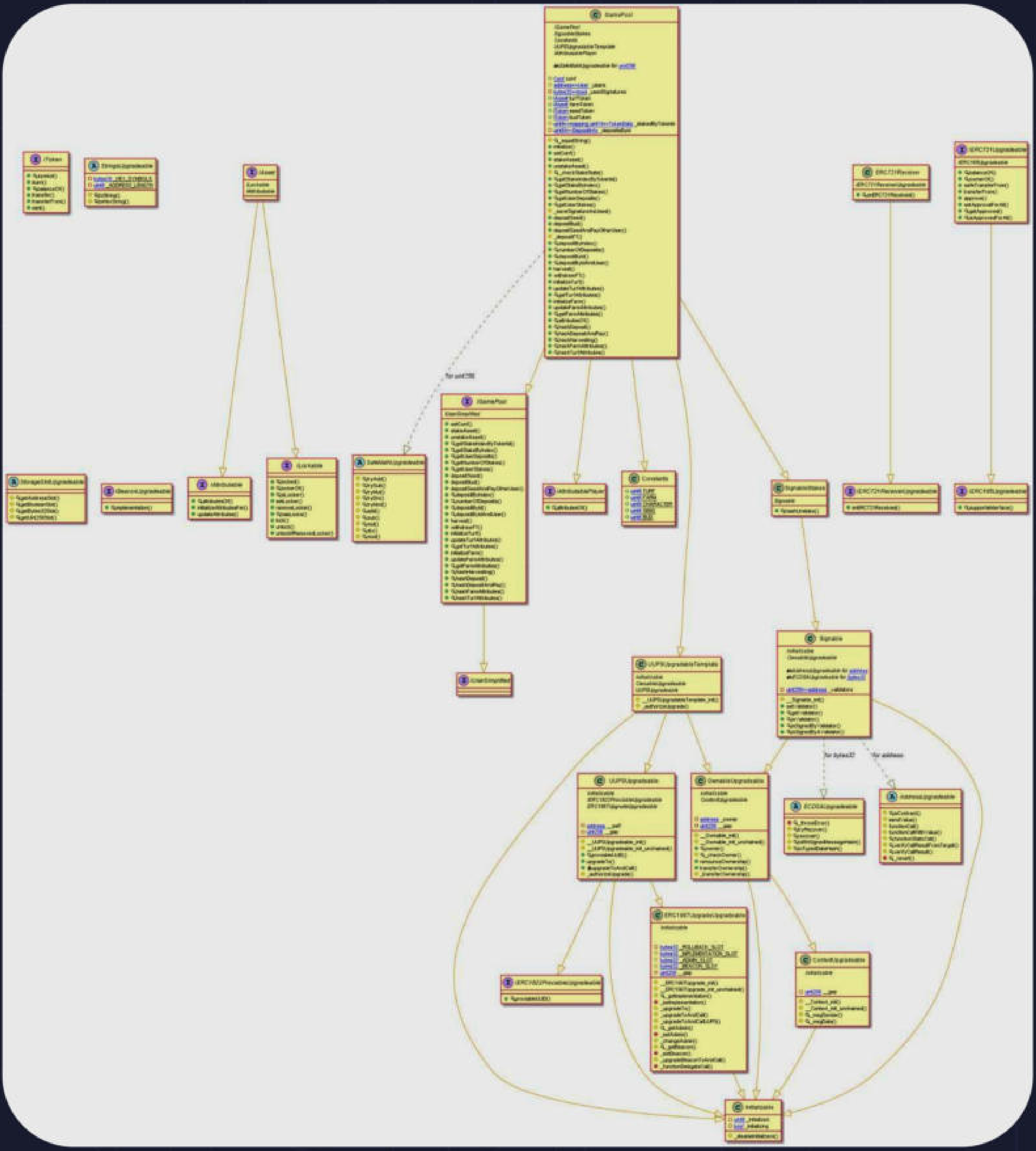
WormholeCommon Diagram



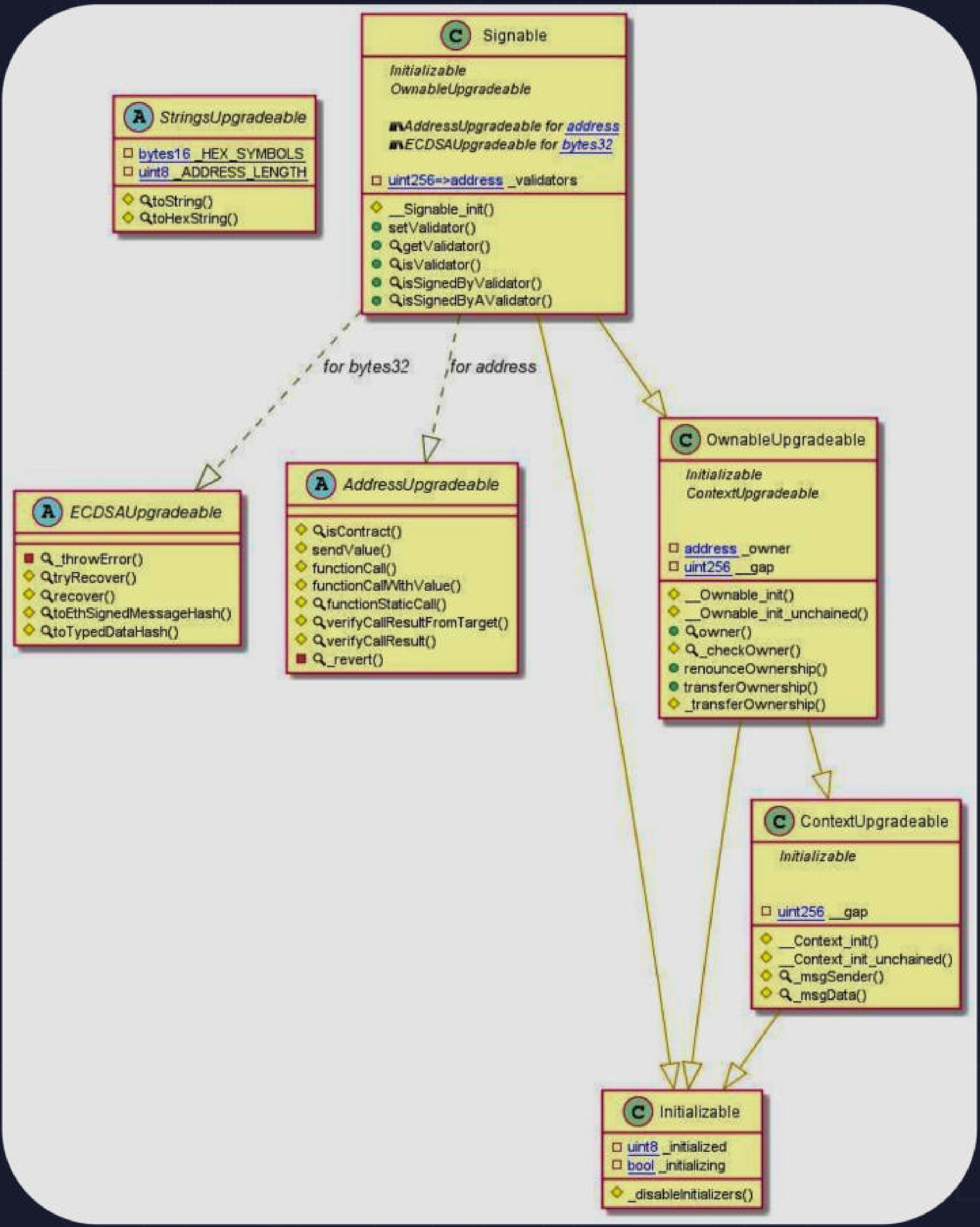
WormholeTunnel Diagram



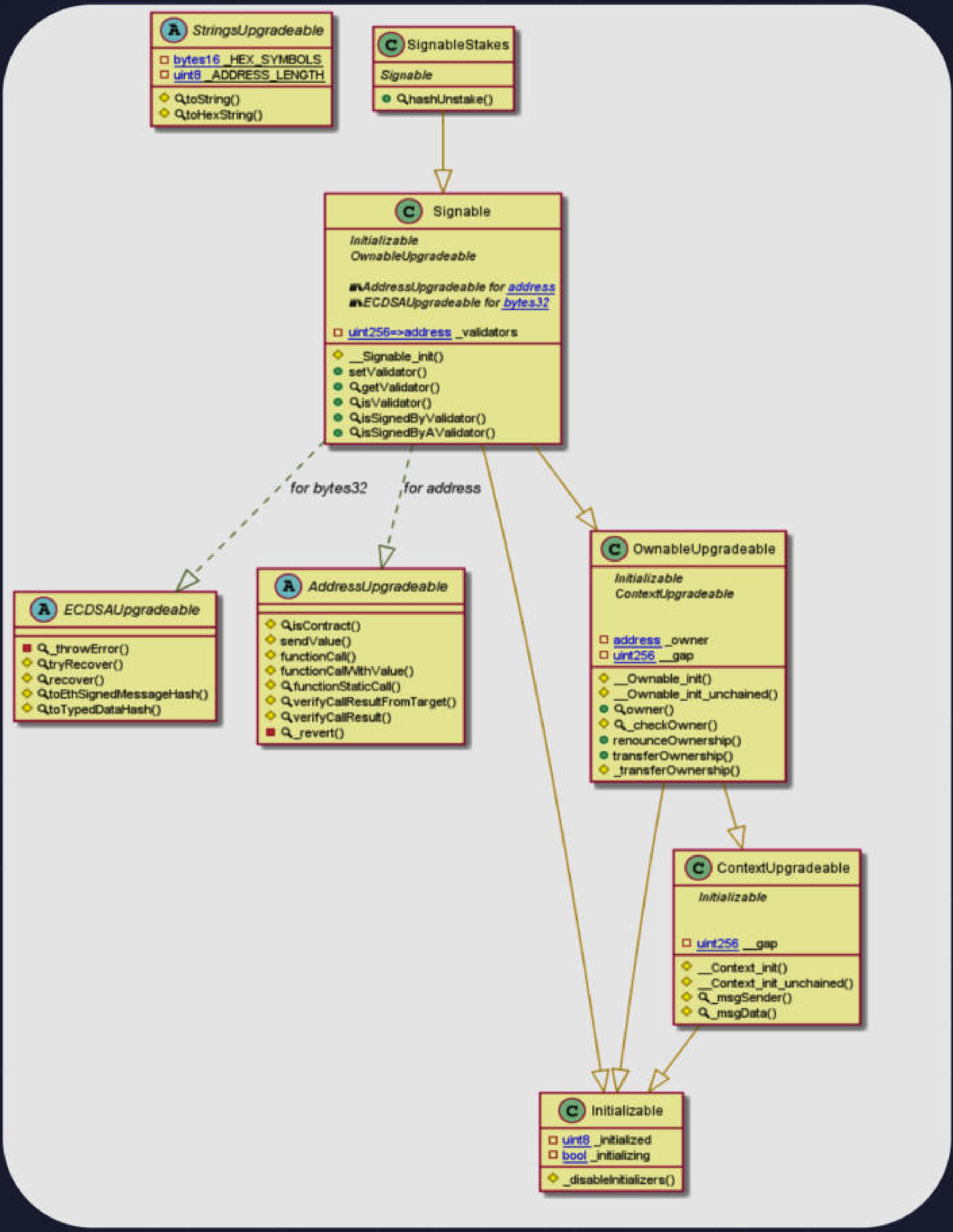
GamePool Diagram



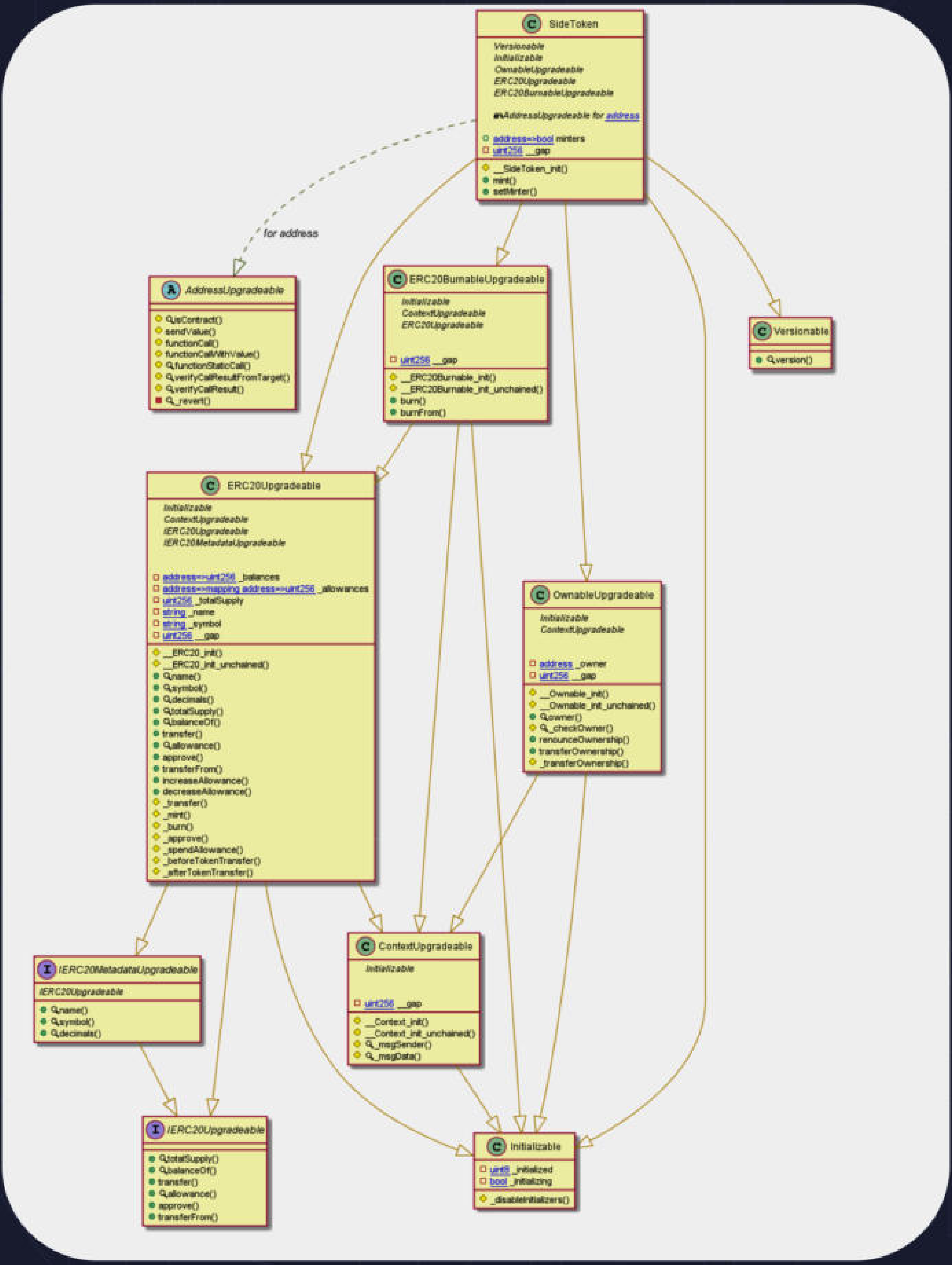
Signable Diagram



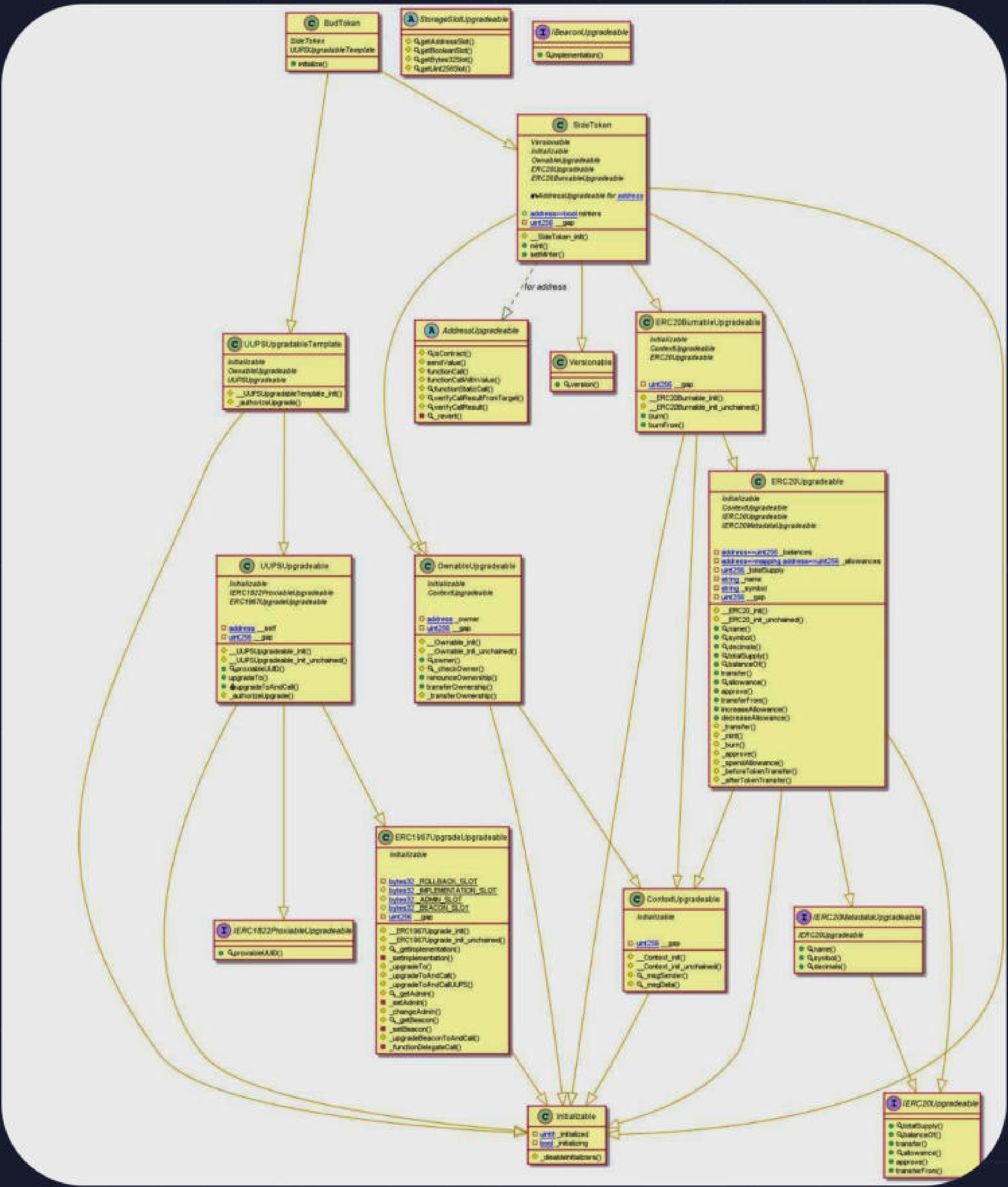
SignableStakes Diagram



SideToken Diagram

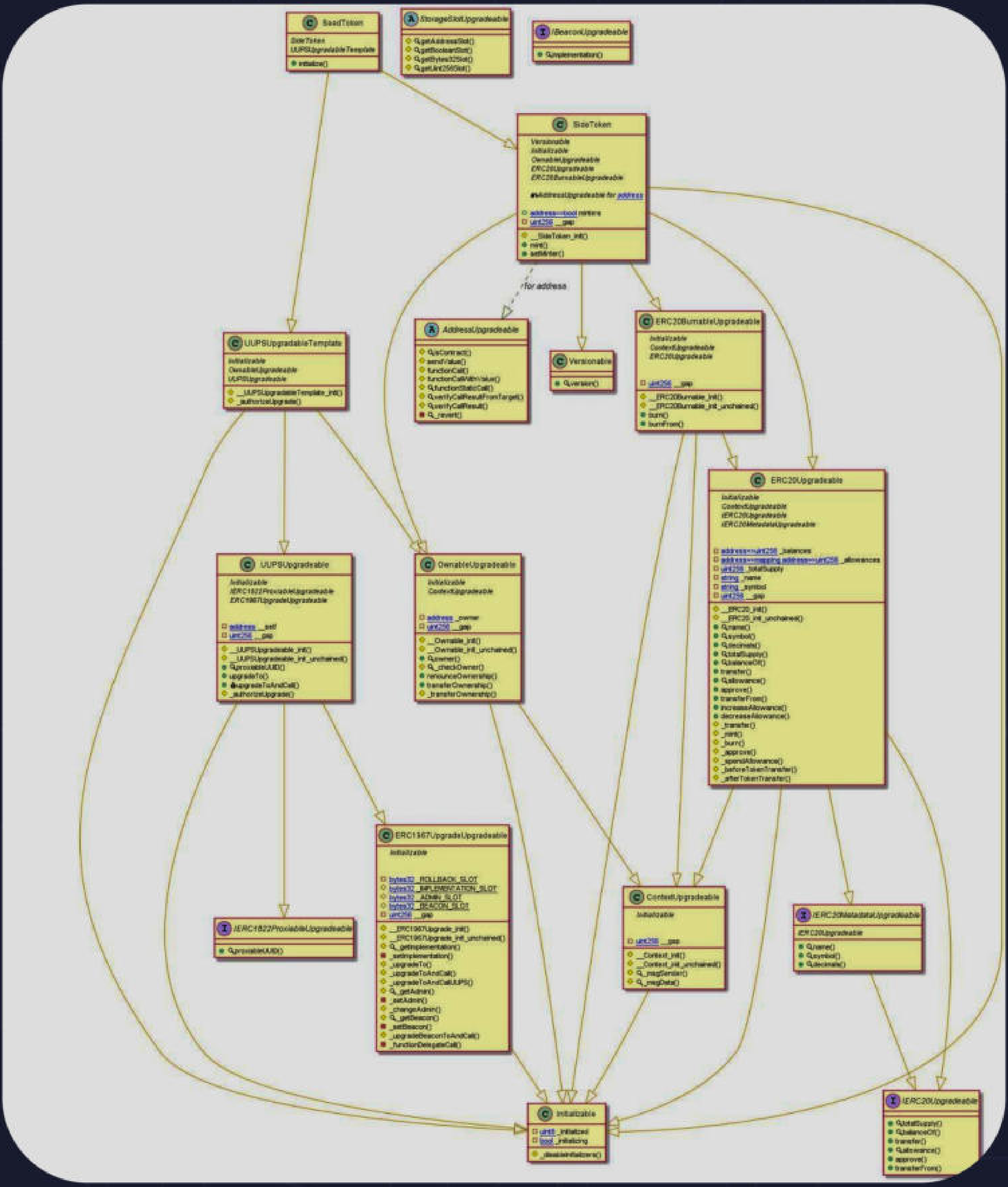


BudToken Diagram



)

SeedToken Diagram

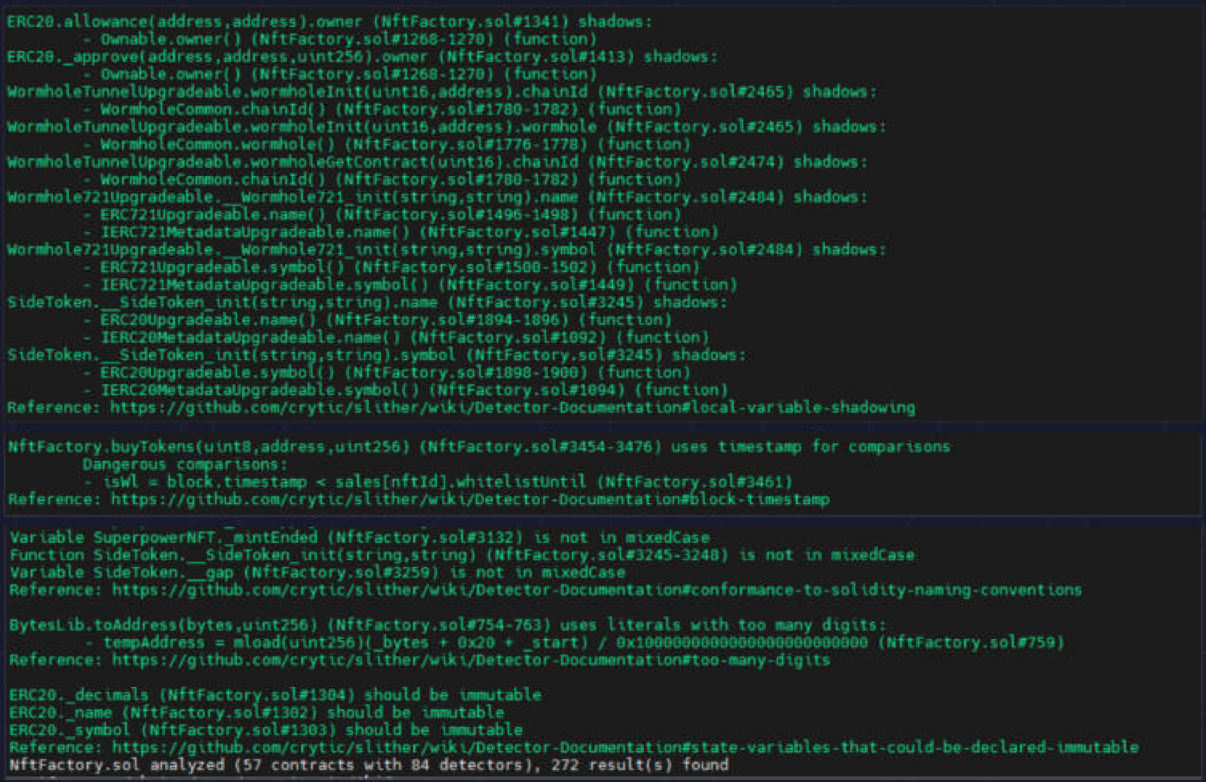


SECURITY ASSESSMENT REPORT

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

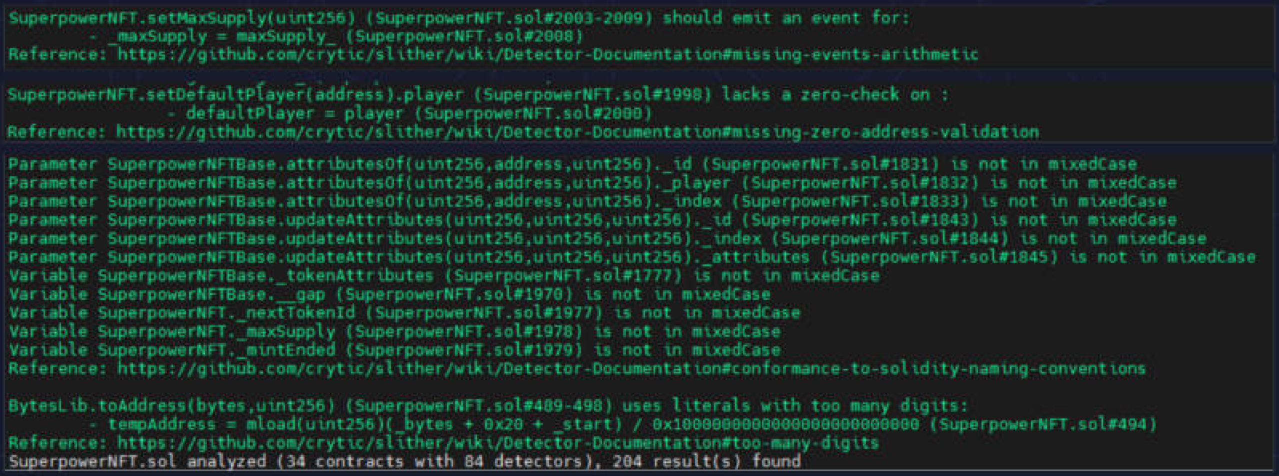
We did the analysis of the project altogether. Below are the results.

Slither log >> NftFactory.sol



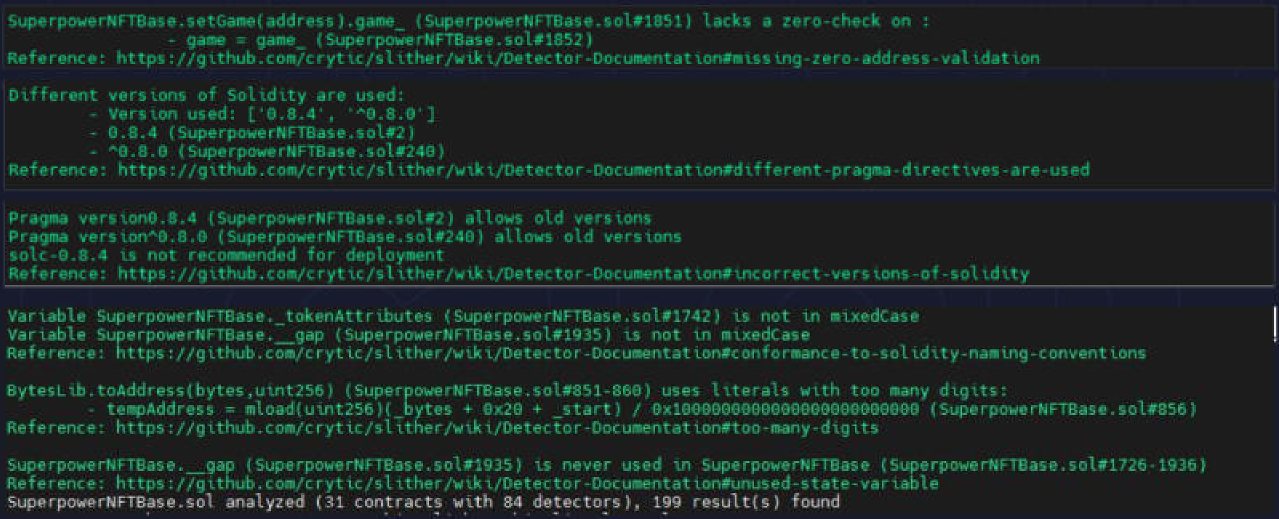
```
ERC20.allowance(address,address).owner (NftFactory.sol#1341) shadows:
        - Ownable.owner() (NftFactory.sol#1268-1270) (function)
ERC20._approve(address,address,uint256).owner (NftFactory.sol#1413) shadows:
        - Ownable.owner() (NftFactory.sol#1268-1270) (function)
WormholeTunnelUpgradeable.wormholeInit(uint16,address).chainId (NftFactory.sol#2465) shadows:
        - WormholeCommon.chainId() (NftFactory.sol#1780-1782) (function)
WormholeTunnelUpgradeable.wormholeInit(uint16,address).wormhole (NftFactory.sol#2465) shadows:
        - WormholeCommon.wormhole() (NftFactory.sol#1776-1778) (function)
WormholeTunnelUpgradeable.wormholeGetContract(uint16).chainId (NftFactory.sol#2474) shadows:
        - WormholeCommon.chainId() (NftFactory.sol#1780-1782) (function)
Wormhole721Upgradeable.__Wormhole721_init(string,string).name (NftFactory.sol#2484) shadows:
        - ERC721Upgradeable.name() (NftFactory.sol#1496-1498) (function)
        - IERC721MetadataUpgradeable.name() (NftFactory.sol#1447) (function)
Wormhole721Upgradeable.__Wormhole721_init(string,string).symbol (NftFactory.sol#2484) shadows:
        - ERC721Upgradeable.symbol() (NftFactory.sol#1500-1502) (function)
        - IERC721MetadataUpgradeable.symbol() (NftFactory.sol#1449) (function)
SideToken.__SideToken_init(string,string).name (NftFactory.sol#3245) shadows:
        - ERC20Upgradeable.name() (NftFactory.sol#1894-1896) (function)
        - IERC20MetadataUpgradeable.name() (NftFactory.sol#1092) (function)
SideToken.__SideToken_init(string,string).symbol (NftFactory.sol#3245) shadows:
        - ERC20Upgradeable.symbol() (NftFactory.sol#1898-1900) (function)
        - IERC20MetadataUpgradeable.symbol() (NftFactory.sol#1094) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

NftFactory.buyTokens(uint8,address,uint256) (NftFactory.sol#3454-3476) uses timestamp for comparisons
        Dangerous comparisons:
        - isWl = block.timestamp < sales[nftId].whitelistUntil (NftFactory.sol#3461)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Variable SuperpowerNFT._mintEnded (NftFactory.sol#3132) is not in mixedCase
Function SideToken.__SideToken_init(string,string) (NftFactory.sol#3245-3248) is not in mixedCase
Variable SideToken.__gap (NftFactory.sol#3259) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BytesLib.toAddress(bytes,uint256) (NftFactory.sol#754-763) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (NftFactory.sol#759)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

ERC20._decimals (NftFactory.sol#1304) should be immutable
ERC20._name (NftFactory.sol#1302) should be immutable
ERC20._symbol (NftFactory.sol#1303) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
NftFactory.sol analyzed (57 contracts with 84 detectors), 272 result(s) found
```

Slither log >> SuperpowerNFT.sol



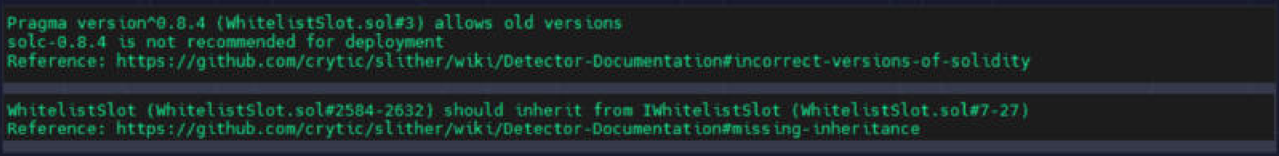
```
SuperpowerNFT.setMaxSupply(uint256) (SuperpowerNFT.sol#2003-2009) should emit an event for:
        - _maxSupply = maxSupply_ (SuperpowerNFT.sol#2008)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

SuperpowerNFT.setDefaultPlayer(address).player (SuperpowerNFT.sol#1998) lacks a zero-check on :
                - defaultPlayer = player (SuperpowerNFT.sol#2000)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._id (SuperpowerNFT.sol#1831) is not in mixedCase
Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._player (SuperpowerNFT.sol#1832) is not in mixedCase
Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._index (SuperpowerNFT.sol#1833) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._id (SuperpowerNFT.sol#1843) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._index (SuperpowerNFT.sol#1844) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._attributes (SuperpowerNFT.sol#1845) is not in mixedCase
Variable SuperpowerNFTBase._tokenAttributes (SuperpowerNFT.sol#1777) is not in mixedCase
Variable SuperpowerNFTBase.__gap (SuperpowerNFT.sol#1970) is not in mixedCase
Variable SuperpowerNFT._nextTokenId (SuperpowerNFT.sol#1977) is not in mixedCase
Variable SuperpowerNFT._maxSupply (SuperpowerNFT.sol#1978) is not in mixedCase
Variable SuperpowerNFT._mintEnded (SuperpowerNFT.sol#1979) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BytesLib.toAddress(bytes,uint256) (SuperpowerNFT.sol#489-498) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (SuperpowerNFT.sol#494)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
SuperpowerNFT.sol analyzed (34 contracts with 84 detectors), 204 result(s) found
```

Slither log >> SuperpowerNFTBase.sol

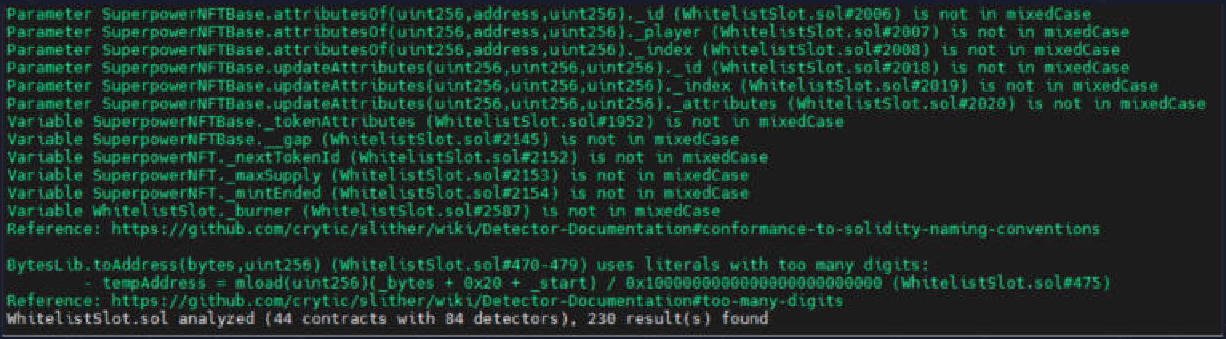


```
SuperpowerNFTBase.setGame(address).game_ (SuperpowerNFTBase.sol#1851) lacks a zero-check on :
                - game = game_ (SuperpowerNFTBase.sol#1852)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Different versions of Solidity are used:
        - Version used: ['0.8.4', '^0.8.0']
        - 0.8.4 (SuperpowerNFTBase.sol#2)
        - ^0.8.0 (SuperpowerNFTBase.sol#240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Pragma version0.8.4 (SuperpowerNFTBase.sol#2) allows old versions
Pragma version^0.8.0 (SuperpowerNFTBase.sol#240) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable SuperpowerNFTBase._tokenAttributes (SuperpowerNFTBase.sol#1742) is not in mixedCase
Variable SuperpowerNFTBase.__gap (SuperpowerNFTBase.sol#1935) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BytesLib.toAddress(bytes,uint256) (SuperpowerNFTBase.sol#851-860) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (SuperpowerNFTBase.sol#856)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

SuperpowerNFTBase.__gap (SuperpowerNFTBase.sol#1935) is never used in SuperpowerNFTBase (SuperpowerNFTBase.sol#1726-1936)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
SuperpowerNFTBase.sol analyzed (31 contracts with 84 detectors), 199 result(s) found
```

Slither log >> WhitelistSlot.sol

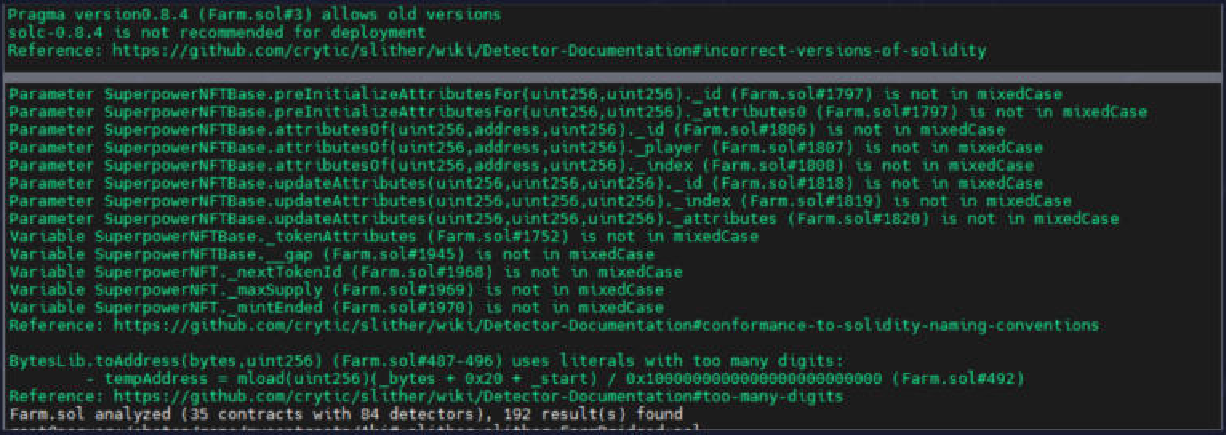


```
Pragma version^0.8.4 (WhitelistSlot.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

WhitelistSlot (WhitelistSlot.sol#2584-2632) should inherit from IWhitelistSlot (WhitelistSlot.sol#7-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
```



```
Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._id (WhitelistSlot.sol#2006) is not in mixedCase
Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._player (WhitelistSlot.sol#2007) is not in mixedCase
Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._index (WhitelistSlot.sol#2008) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._id (WhitelistSlot.sol#2018) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._index (WhitelistSlot.sol#2019) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._attributes (WhitelistSlot.sol#2020) is not in mixedCase
Variable SuperpowerNFTBase._tokenAttributes (WhitelistSlot.sol#1952) is not in mixedCase
Variable SuperpowerNFTBase.__gap (WhitelistSlot.sol#2145) is not in mixedCase
Variable SuperpowerNFT._nextTokenId (WhitelistSlot.sol#2152) is not in mixedCase
Variable SuperpowerNFT._maxSupply (WhitelistSlot.sol#2153) is not in mixedCase
Variable SuperpowerNFT._mintEnded (WhitelistSlot.sol#2154) is not in mixedCase
Variable WhitelistSlot._burner (WhitelistSlot.sol#2587) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BytesLib.toAddress(bytes,uint256) (WhitelistSlot.sol#470-479) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (WhitelistSlot.sol#475)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
WhitelistSlot.sol analyzed (44 contracts with 84 detectors), 230 result(s) found
```

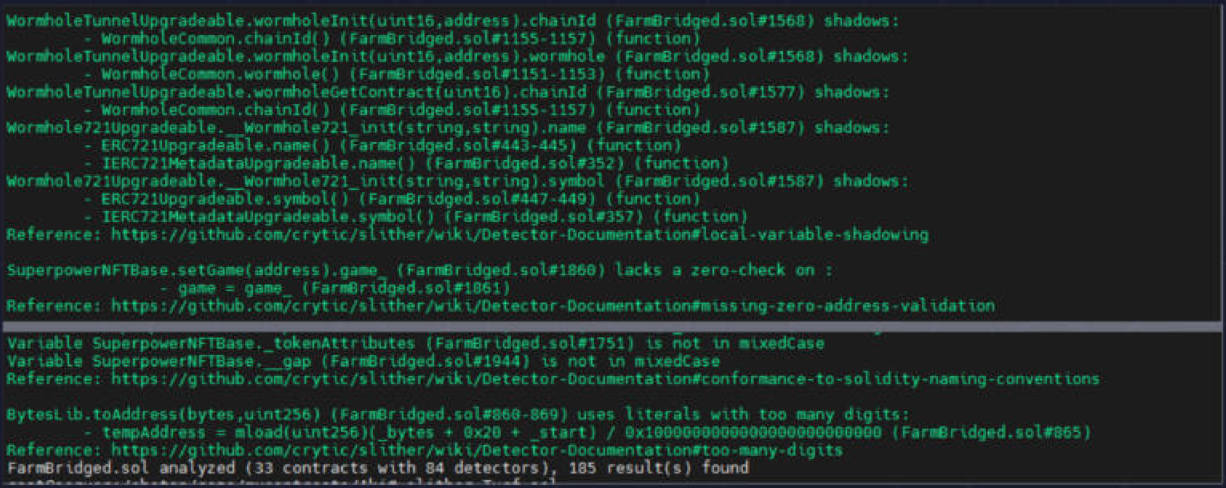
Slither log >> Farm.sol



```
Pragma version0.8.4 (Farm.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Parameter SuperpowerNFTBase.preInitializeAttributesFor(uint256,uint256)._id (Farm.sol#1797) is not in mixedCase
Parameter SuperpowerNFTBase.preInitializeAttributesFor(uint256,uint256)._attributes0 (Farm.sol#1797) is not in mixedCase
Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._id (Farm.sol#1806) is not in mixedCase
Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._player (Farm.sol#1807) is not in mixedCase
Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._index (Farm.sol#1808) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._id (Farm.sol#1818) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._index (Farm.sol#1819) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._attributes (Farm.sol#1820) is not in mixedCase
Variable SuperpowerNFTBase._tokenAttributes (Farm.sol#1752) is not in mixedCase
Variable SuperpowerNFTBase.__gap (Farm.sol#1945) is not in mixedCase
Variable SuperpowerNFT._nextTokenId (Farm.sol#1968) is not in mixedCase
Variable SuperpowerNFT._maxSupply (Farm.sol#1969) is not in mixedCase
Variable SuperpowerNFT._mintEnded (Farm.sol#1970) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BytesLib.toAddress(bytes,uint256) (Farm.sol#487-496) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (Farm.sol#492)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
Farm.sol analyzed (35 contracts with 84 detectors), 192 result(s) found
```

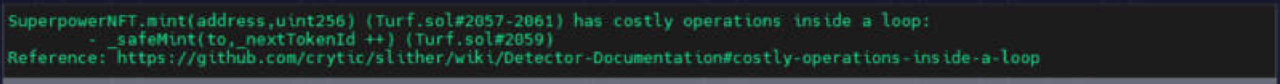
Slither log >> FarmBridged.sol



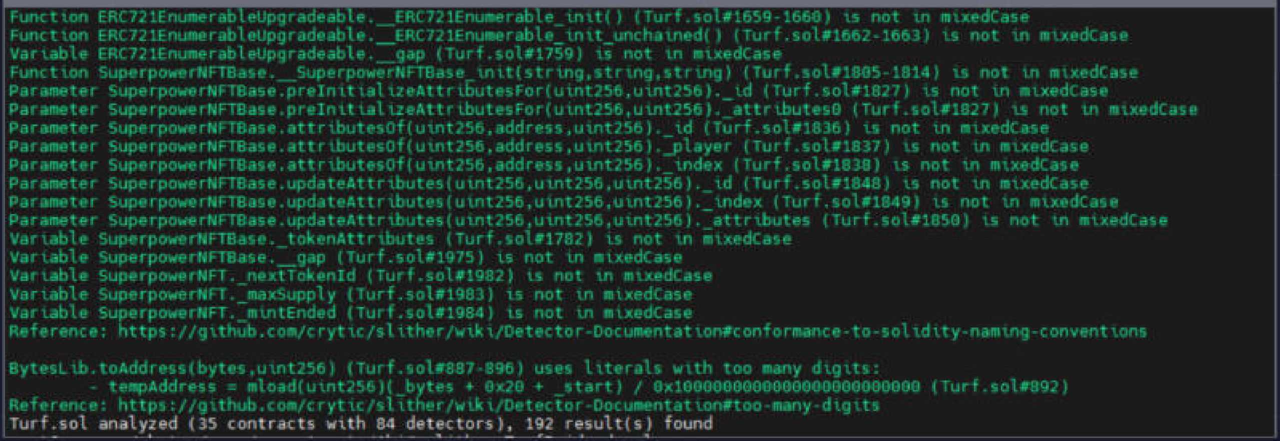
```
WormholeTunnelUpgradeable.wormholeInit(uint16,address).chainId (FarmBridged.sol#1568) shadows:
        - WormholeCommon.chainId() (FarmBridged.sol#1155-1157) (function)
WormholeTunnelUpgradeable.wormholeInit(uint16,address).wormhole (FarmBridged.sol#1568) shadows:
        - WormholeCommon.wormhole() (FarmBridged.sol#1151-1153) (function)
WormholeTunnelUpgradeable.wormholeGetContract(uint16).chainId (FarmBridged.sol#1577) shadows:
        - WormholeCommon.chainId() (FarmBridged.sol#1155-1157) (function)
Wormhole721Upgradeable.__Wormhole721_init(string,string).name (FarmBridged.sol#1587) shadows:
        - ERC721Upgradeable.name() (FarmBridged.sol#443-445) (function)
        - IERC721MetadataUpgradeable.name() (FarmBridged.sol#352) (function)
Wormhole721Upgradeable.__Wormhole721_init(string,string).symbol (FarmBridged.sol#1587) shadows:
        - ERC721Upgradeable.symbol() (FarmBridged.sol#447-449) (function)
        - IERC721MetadataUpgradeable.symbol() (FarmBridged.sol#357) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SuperpowerNFTBase.setGame(address).game_ (FarmBridged.sol#1860) lacks a zero-check on :
                - game = game_ (FarmBridged.sol#1861)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Variable SuperpowerNFTBase._tokenAttributes (FarmBridged.sol#1751) is not in mixedCase
Variable SuperpowerNFTBase.__gap (FarmBridged.sol#1944) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BytesLib.toAddress(bytes,uint256) (FarmBridged.sol#860-869) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (FarmBridged.sol#865)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
FarmBridged.sol analyzed (33 contracts with 84 detectors), 185 result(s) found
```

Slither log >> Turf.sol

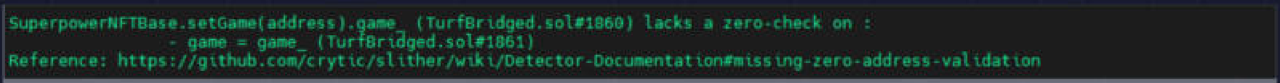


```
SuperpowerNFT.mint(address,uint256) (Turf.sol#2057-2061) has costly operations inside a loop:
        - _safeMint(to,_nextTokenId ++) (Turf.sol#2059)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
```

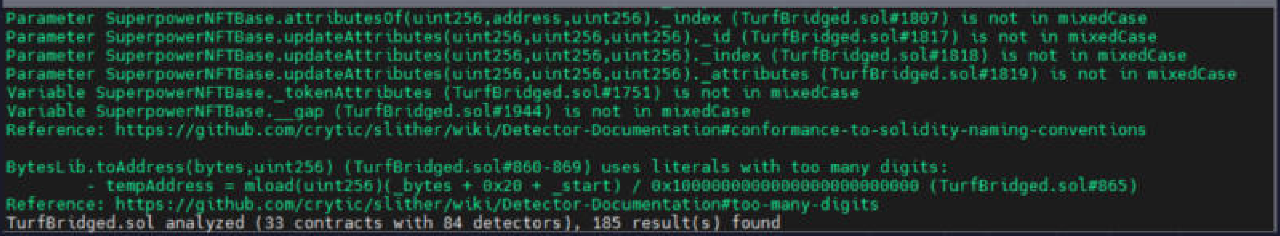


```
Function ERC721EnumerableUpgradeable.__ERC721Enumerable_init() (Turf.sol#1659-1660) is not in mixedCase
Function ERC721EnumerableUpgradeable.__ERC721Enumerable_init_unchained() (Turf.sol#1662-1663) is not in mixedCase
Variable ERC721EnumerableUpgradeable.__gap (Turf.sol#1759) is not in mixedCase
Function SuperpowerNFTBase.__SuperpowerNFTBase_init(string,string,string) (Turf.sol#1805-1814) is not in mixedCase
Parameter SuperpowerNFTBase.preInitializeAttributesFor(uint256,uint256)._id (Turf.sol#1827) is not in mixedCase
Parameter SuperpowerNFTBase.preInitializeAttributesFor(uint256,uint256)._attributes0 (Turf.sol#1827) is not in mixedCase
Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._id (Turf.sol#1836) is not in mixedCase
Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._player (Turf.sol#1837) is not in mixedCase
Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._index (Turf.sol#1838) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._id (Turf.sol#1848) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._index (Turf.sol#1849) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._attributes (Turf.sol#1850) is not in mixedCase
Variable SuperpowerNFTBase._tokenAttributes (Turf.sol#1782) is not in mixedCase
Variable SuperpowerNFTBase.__gap (Turf.sol#1975) is not in mixedCase
Variable SuperpowerNFT._nextTokenId (Turf.sol#1982) is not in mixedCase
Variable SuperpowerNFT._maxSupply (Turf.sol#1983) is not in mixedCase
Variable SuperpowerNFT._mintEnded (Turf.sol#1984) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BytesLib.toAddress(bytes,uint256) (Turf.sol#887-896) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (Turf.sol#892)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
Turf.sol analyzed (35 contracts with 84 detectors), 192 result(s) found
```

Slither log >> TurfBridged.sol

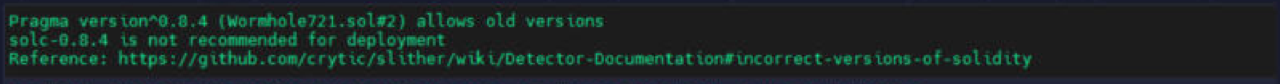


```
SuperpowerNFTBase.setGame(address).game_ (TurfBridged.sol#1860) lacks a zero-check on :
                - game = game_ (TurfBridged.sol#1861)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

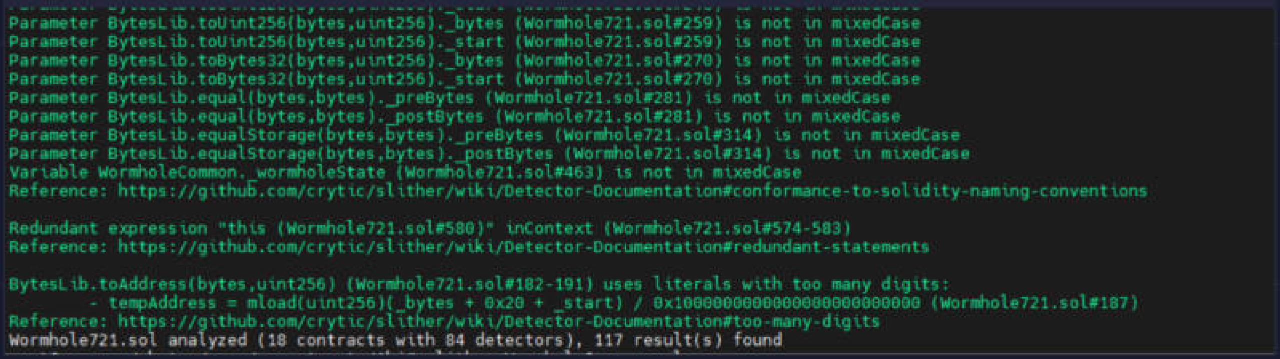


```
Parameter SuperpowerNFTBase.attributesOf(uint256,address,uint256)._index (TurfBridged.sol#1807) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._id (TurfBridged.sol#1817) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._index (TurfBridged.sol#1818) is not in mixedCase
Parameter SuperpowerNFTBase.updateAttributes(uint256,uint256,uint256)._attributes (TurfBridged.sol#1819) is not in mixedCase
Variable SuperpowerNFTBase._tokenAttributes (TurfBridged.sol#1751) is not in mixedCase
Variable SuperpowerNFTBase.__gap (TurfBridged.sol#1944) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BytesLib.toAddress(bytes,uint256) (TurfBridged.sol#860-869) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (TurfBridged.sol#865)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
TurfBridged.sol analyzed (33 contracts with 84 detectors), 185 result(s) found
```

Slither log >> Wormhole721.sol

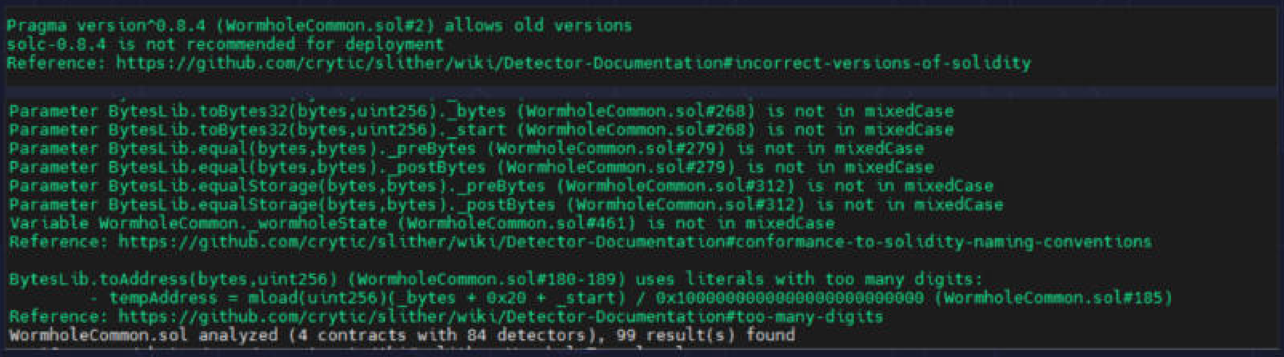


```
Pragma version^0.8.4 (Wormhole721.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```



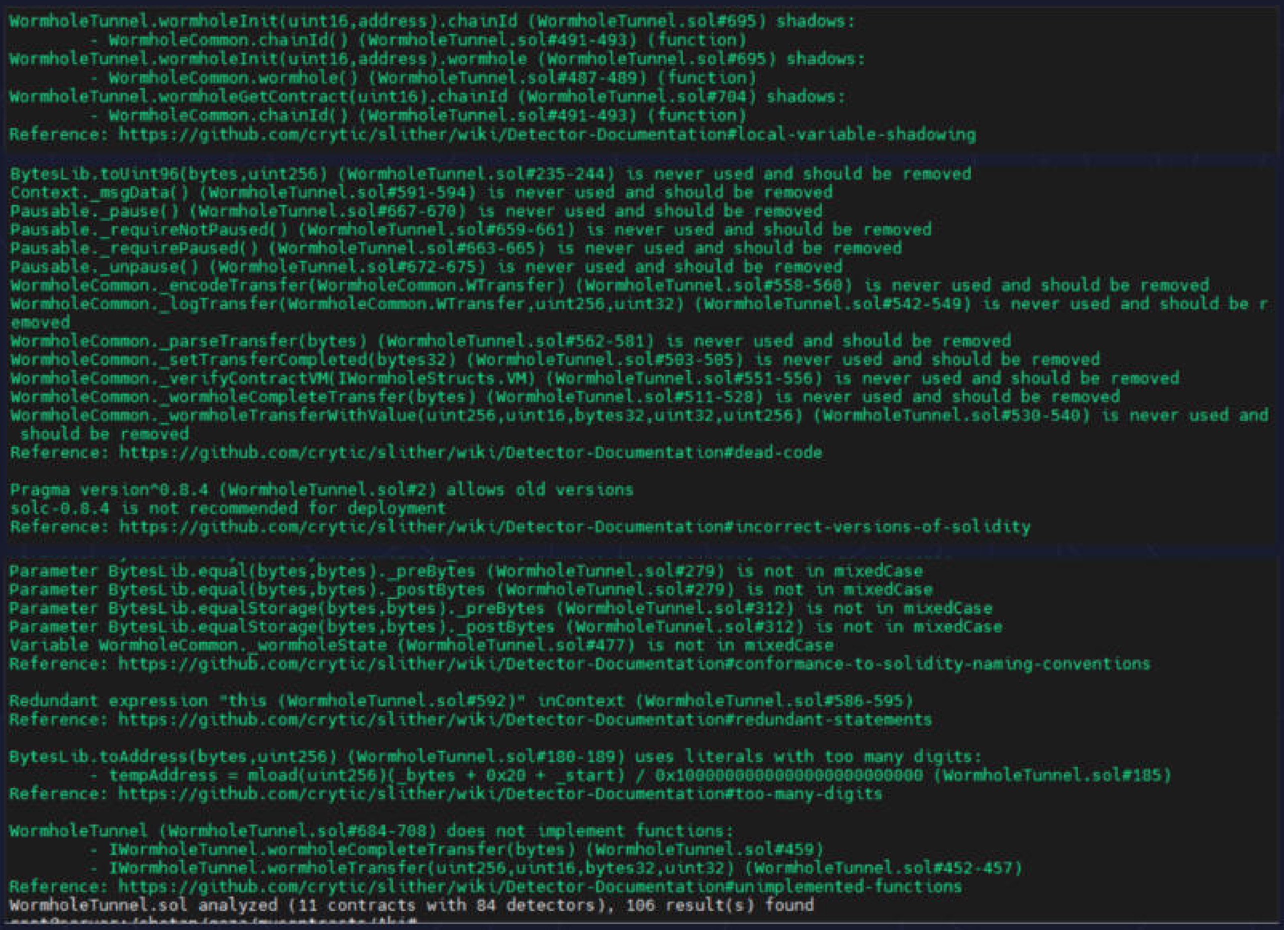
```
Parameter BytesLib.toUint256(bytes,uint256)._bytes (Wormhole721.sol#259) is not in mixedCase
Parameter BytesLib.toUint256(bytes,uint256)._start (Wormhole721.sol#259) is not in mixedCase
Parameter BytesLib.toBytes32(bytes,uint256)._bytes (Wormhole721.sol#270) is not in mixedCase
Parameter BytesLib.toBytes32(bytes,uint256)._start (Wormhole721.sol#270) is not in mixedCase
Parameter BytesLib.equal(bytes,bytes)._preBytes (Wormhole721.sol#281) is not in mixedCase
Parameter BytesLib.equal(bytes,bytes)._postBytes (Wormhole721.sol#281) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._preBytes (Wormhole721.sol#314) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._postBytes (Wormhole721.sol#314) is not in mixedCase
Variable WormholeCommon._wormholeState (Wormhole721.sol#463) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (Wormhole721.sol#580)" inContext (Wormhole721.sol#574-583)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

BytesLib.toAddress(bytes,uint256) (Wormhole721.sol#182-191) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (Wormhole721.sol#187)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
Wormhole721.sol analyzed (18 contracts with 84 detectors), 117 result(s) found
```

Slither log >> WormholeCommon.sol



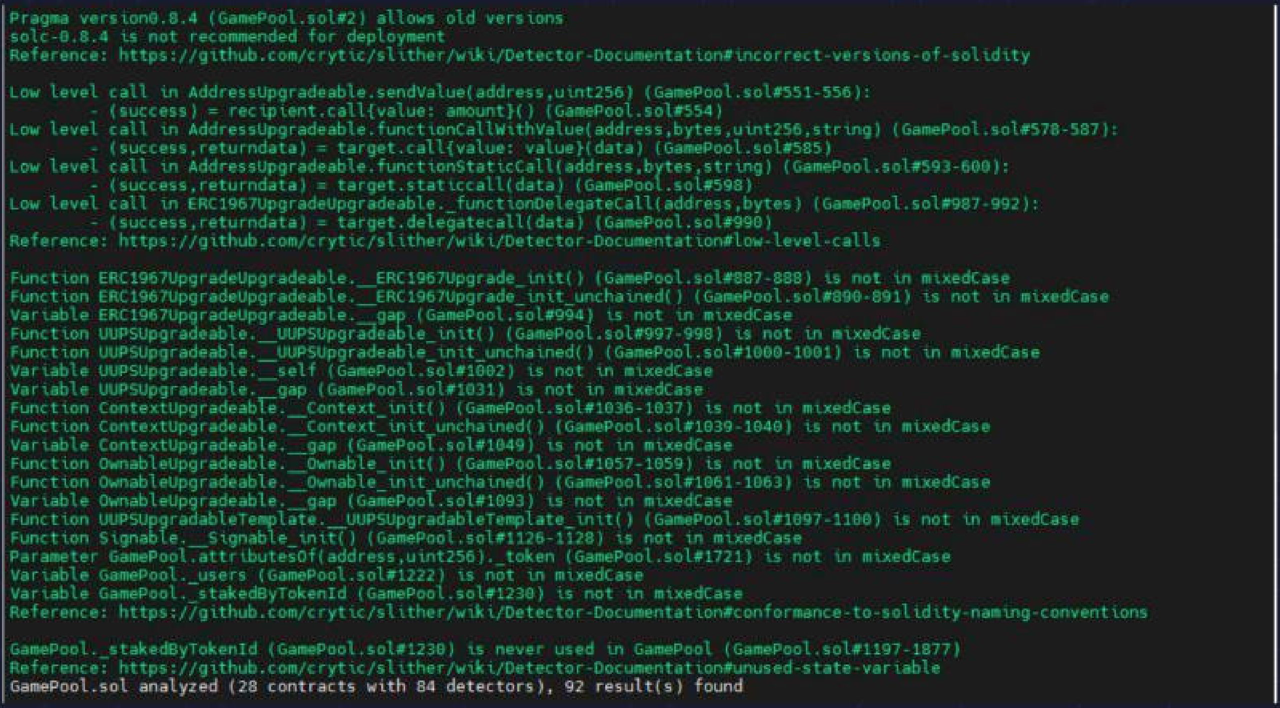
```
Pragma version^0.8.4 (WormholeCommon.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter BytesLib.toBytes32(bytes,uint256)._bytes (WormholeCommon.sol#268) is not in mixedCase
Parameter BytesLib.toBytes32(bytes,uint256)._start (WormholeCommon.sol#268) is not in mixedCase
Parameter BytesLib.equal(bytes,bytes)._preBytes (WormholeCommon.sol#279) is not in mixedCase
Parameter BytesLib.equal(bytes,bytes)._postBytes (WormholeCommon.sol#279) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._preBytes (WormholeCommon.sol#312) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._postBytes (WormholeCommon.sol#312) is not in mixedCase
Variable WormholeCommon._wormholeState (WormholeCommon.sol#461) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BytesLib.toAddress(bytes,uint256) (WormholeCommon.sol#180-189) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (WormholeCommon.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
WormholeCommon.sol analyzed (4 contracts with 84 detectors), 99 result(s) found
```

Slither log >> WormholeTunnel.sol



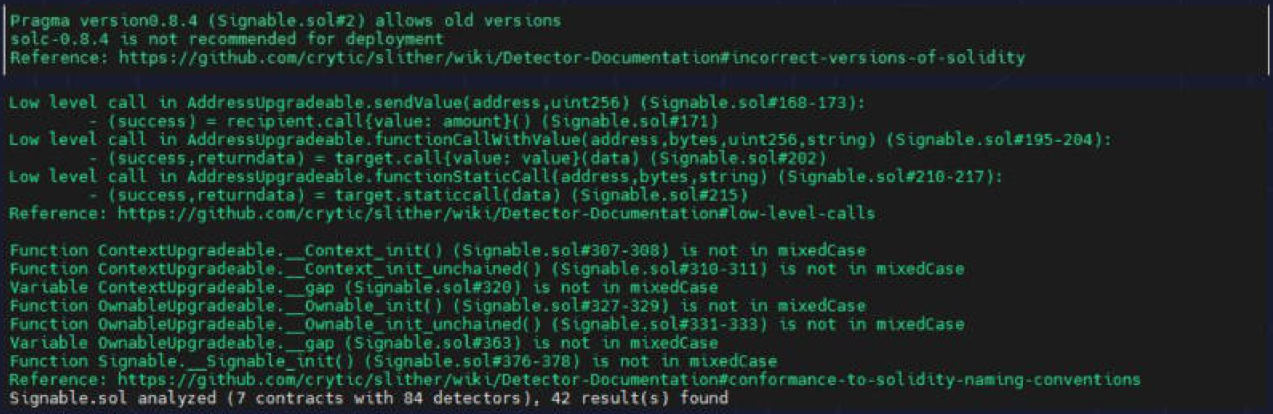
```
WormholeTunnel.wormholeInit(uint16,address).chainId (WormholeTunnel.sol#695) shadows:
        - WormholeCommon.chainId() (WormholeTunnel.sol#491-493) (function)
WormholeTunnel.wormholeInit(uint16,address).wormhole (WormholeTunnel.sol#695) shadows:
        - WormholeCommon.wormhole() (WormholeTunnel.sol#487-489) (function)
WormholeTunnel.wormholeGetContract(uint16).chainId (WormholeTunnel.sol#704) shadows:
        - WormholeCommon.chainId() (WormholeTunnel.sol#491-493) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

BytesLib.toUint96(bytes,uint256) (WormholeTunnel.sol#235-244) is never used and should be removed
Context._msgData() (WormholeTunnel.sol#591-594) is never used and should be removed
Pausable._pause() (WormholeTunnel.sol#667-670) is never used and should be removed
Pausable._requireNotPaused() (WormholeTunnel.sol#659-661) is never used and should be removed
Pausable._requirePaused() (WormholeTunnel.sol#663-665) is never used and should be removed
Pausable._unpause() (WormholeTunnel.sol#672-675) is never used and should be removed
WormholeCommon._encodeTransfer(WormholeCommon.WTransfer) (WormholeTunnel.sol#558-560) is never used and should be removed
WormholeCommon._logTransfer(WormholeCommon.WTransfer,uint256,uint32) (WormholeTunnel.sol#542-549) is never used and should be r
emoved
WormholeCommon._parseTransfer(bytes) (WormholeTunnel.sol#562-581) is never used and should be removed
WormholeCommon._setTransferCompleted(bytes32) (WormholeTunnel.sol#503-505) is never used and should be removed
WormholeCommon._verifyContractVM(IWormholeStructs.VM) (WormholeTunnel.sol#551-556) is never used and should be removed
WormholeCommon._wormholeCompleteTransfer(bytes) (WormholeTunnel.sol#511-528) is never used and should be removed
WormholeCommon._wormholeTransferWithValue(uint256,uint16,bytes32,uint32,uint256) (WormholeTunnel.sol#530-540) is never used and
 should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.4 (WormholeTunnel.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter BytesLib.equal(bytes,bytes)._preBytes (WormholeTunnel.sol#279) is not in mixedCase
Parameter BytesLib.equal(bytes,bytes)._postBytes (WormholeTunnel.sol#279) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._preBytes (WormholeTunnel.sol#312) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._postBytes (WormholeTunnel.sol#312) is not in mixedCase
Variable WormholeCommon._wormholeState (WormholeTunnel.sol#477) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (WormholeTunnel.sol#592)" inContext (WormholeTunnel.sol#586-595)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

BytesLib.toAddress(bytes,uint256) (WormholeTunnel.sol#180-189) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (WormholeTunnel.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

WormholeTunnel (WormholeTunnel.sol#684-708) does not implement functions:
        - IWormholeTunnel.wormholeCompleteTransfer(bytes) (WormholeTunnel.sol#459)
        - IWormholeTunnel.wormholeTransfer(uint256,uint16,bytes32,uint32) (WormholeTunnel.sol#452-457)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
WormholeTunnel.sol analyzed (11 contracts with 84 detectors), 106 result(s) found
```

Slither log >> GamePool.sol



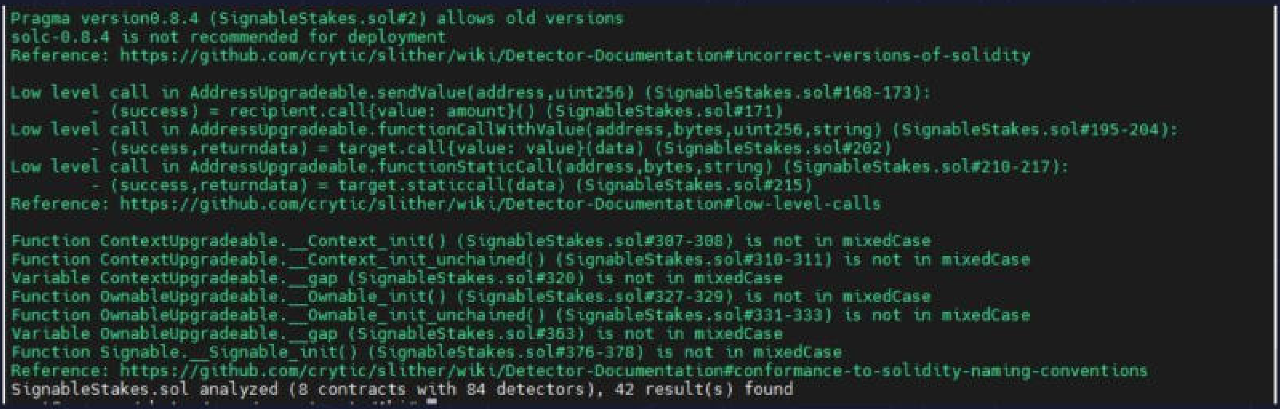
```
Pragma version0.8.4 (GamePool.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (GamePool.sol#551-556):
        - (success) = recipient.call{value: amount}() (GamePool.sol#554)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (GamePool.sol#578-587):
        - (success,returndata) = target.call{value: value}(data) (GamePool.sol#585)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (GamePool.sol#593-600):
        - (success,returndata) = target.staticcall(data) (GamePool.sol#598)
Low level call in ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (GamePool.sol#987-992):
        - (success,returndata) = target.delegatecall(data) (GamePool.sol#990)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init() (GamePool.sol#887-888) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init_unchained() (GamePool.sol#890-891) is not in mixedCase
Variable ERC1967UpgradeUpgradeable.__gap (GamePool.sol#994) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init() (GamePool.sol#997-998) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init_unchained() (GamePool.sol#1000-1001) is not in mixedCase
Variable UUPSUpgradeable.__self (GamePool.sol#1002) is not in mixedCase
Variable UUPSUpgradeable.__gap (GamePool.sol#1031) is not in mixedCase
Function ContextUpgradeable.__Context_init() (GamePool.sol#1036-1037) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (GamePool.sol#1039-1040) is not in mixedCase
Variable ContextUpgradeable.__gap (GamePool.sol#1049) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (GamePool.sol#1057-1059) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (GamePool.sol#1061-1063) is not in mixedCase
Variable OwnableUpgradeable.__gap (GamePool.sol#1093) is not in mixedCase
Function UUPSUpgradableTemplate.__UUPSUpgradableTemplate_init() (GamePool.sol#1097-1100) is not in mixedCase
Function Signable.__Signable_init() (GamePool.sol#1126-1128) is not in mixedCase
Parameter GamePool.attributesOf(address,uint256)._token (GamePool.sol#1721) is not in mixedCase
Variable GamePool._users (GamePool.sol#1222) is not in mixedCase
Variable GamePool._stakedByTokenId (GamePool.sol#1230) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

GamePool._stakedByTokenId (GamePool.sol#1230) is never used in GamePool (GamePool.sol#1197-1877)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
GamePool.sol analyzed (28 contracts with 84 detectors), 92 result(s) found
```

Slither log >> Signable.sol



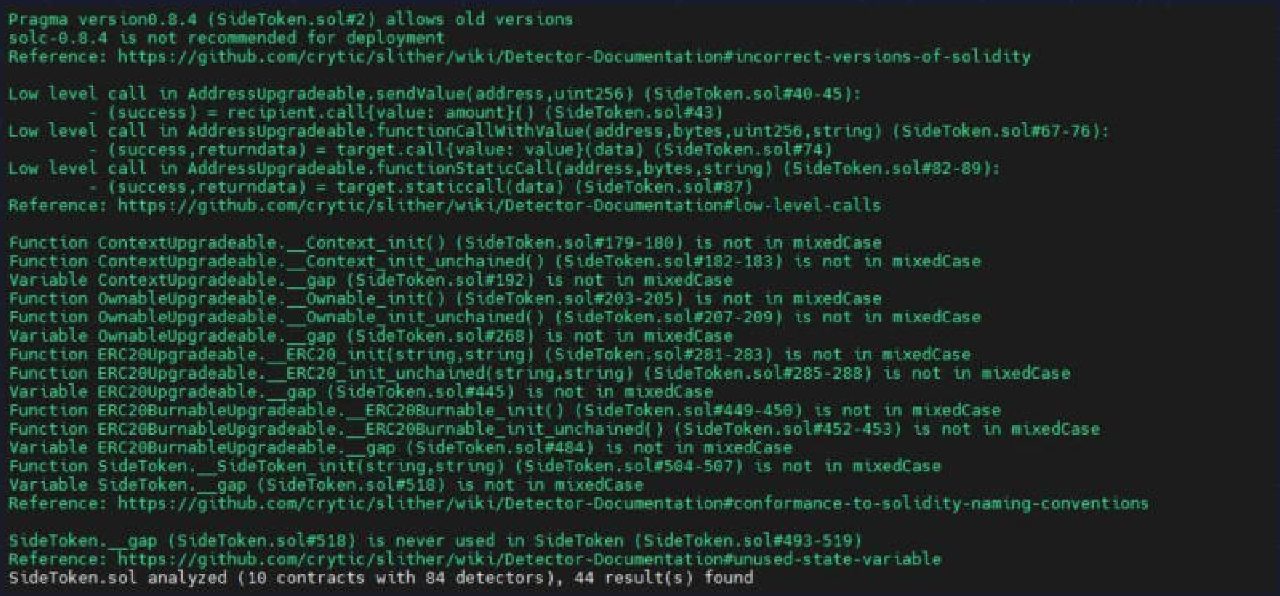
```
Pragma version0.8.4 (Signable.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (Signable.sol#168-173):
        - (success) = recipient.call{value: amount}() (Signable.sol#171)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (Signable.sol#195-204):
        - (success,returndata) = target.call{value: value}(data) (Signable.sol#202)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (Signable.sol#210-217):
        - (success,returndata) = target.staticcall(data) (Signable.sol#215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function ContextUpgradeable.__Context_init() (Signable.sol#307-308) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (Signable.sol#310-311) is not in mixedCase
Variable ContextUpgradeable.__gap (Signable.sol#320) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (Signable.sol#327-329) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (Signable.sol#331-333) is not in mixedCase
Variable OwnableUpgradeable.__gap (Signable.sol#363) is not in mixedCase
Function Signable.__Signable_init() (Signable.sol#376-378) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
Signable.sol analyzed (7 contracts with 84 detectors), 42 result(s) found
```

Slither log >> SignableStakes.sol



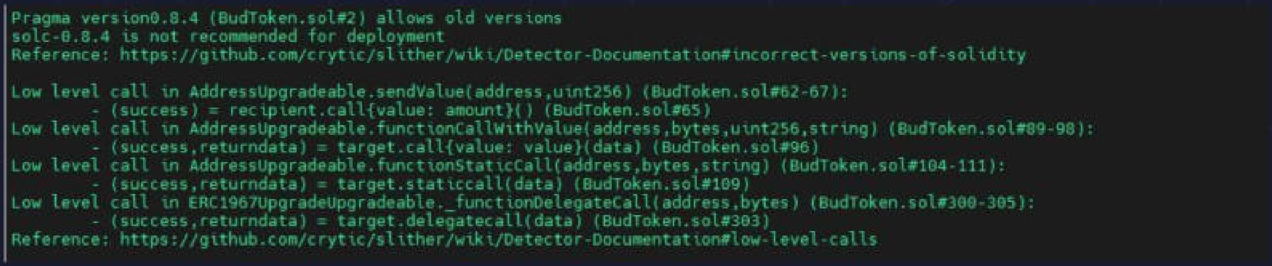
```
Pragma version0.8.4 (SignableStakes.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (SignableStakes.sol#168-173):
        - (success) = recipient.call{value: amount}() (SignableStakes.sol#171)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (SignableStakes.sol#195-204):
        - (success,returndata) = target.call{value: value}(data) (SignableStakes.sol#202)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (SignableStakes.sol#210-217):
        - (success,returndata) = target.staticcall(data) (SignableStakes.sol#215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function ContextUpgradeable.__Context_init() (SignableStakes.sol#307-308) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (SignableStakes.sol#310-311) is not in mixedCase
Variable ContextUpgradeable.__gap (SignableStakes.sol#320) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (SignableStakes.sol#327-329) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (SignableStakes.sol#331-333) is not in mixedCase
Variable OwnableUpgradeable.__gap (SignableStakes.sol#363) is not in mixedCase
Function Signable.__Signable_init() (SignableStakes.sol#376-378) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
SignableStakes.sol analyzed (8 contracts with 84 detectors), 42 result(s) found
```

Slither log >> SideToken.sol

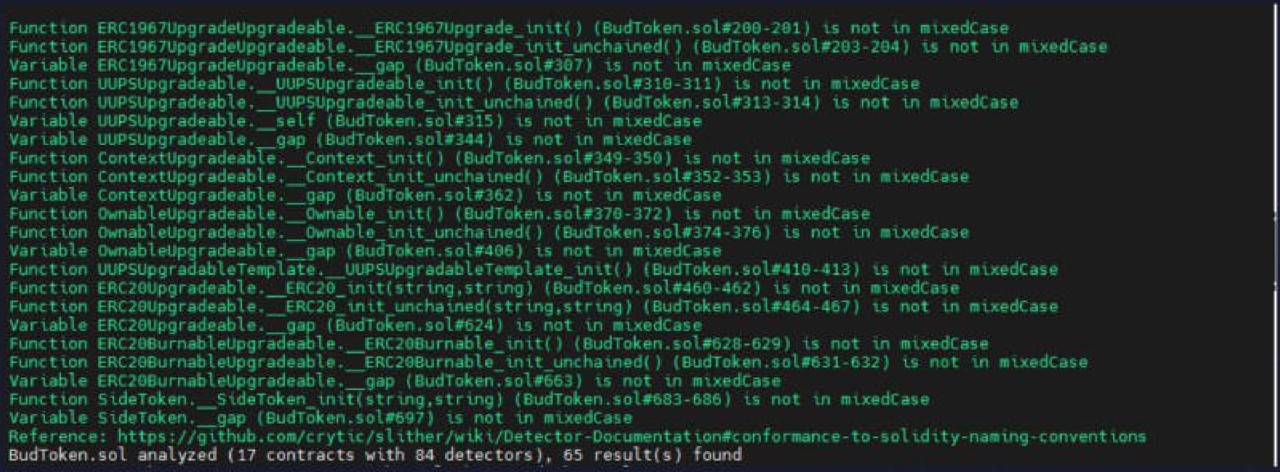


```
Pragma version0.8.4 (SideToken.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (SideToken.sol#40-45):
        - (success) = recipient.call{value: amount}() (SideToken.sol#43)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (SideToken.sol#67-76):
        - (success,returndata) = target.call{value: value}(data) (SideToken.sol#74)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (SideToken.sol#82-89):
        - (success,returndata) = target.staticcall(data) (SideToken.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function ContextUpgradeable.__Context_init() (SideToken.sol#179-180) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (SideToken.sol#182-183) is not in mixedCase
Variable ContextUpgradeable.__gap (SideToken.sol#192) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (SideToken.sol#203-205) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (SideToken.sol#207-209) is not in mixedCase
Variable OwnableUpgradeable.__gap (SideToken.sol#268) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (SideToken.sol#281-283) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (SideToken.sol#285-288) is not in mixedCase
Variable ERC20Upgradeable.__gap (SideToken.sol#445) is not in mixedCase
Function ERC20BurnableUpgradeable.__ERC20Burnable_init() (SideToken.sol#449-450) is not in mixedCase
Function ERC20BurnableUpgradeable.__ERC20Burnable_init_unchained() (SideToken.sol#452-453) is not in mixedCase
Variable ERC20BurnableUpgradeable.__gap (SideToken.sol#484) is not in mixedCase
Function SideToken.__SideToken_init(string,string) (SideToken.sol#504-507) is not in mixedCase
Variable SideToken.__gap (SideToken.sol#518) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SideToken.__gap (SideToken.sol#518) is never used in SideToken (SideToken.sol#493-519)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
SideToken.sol analyzed (10 contracts with 84 detectors), 44 result(s) found
```

Slither log >> BudToken.sol

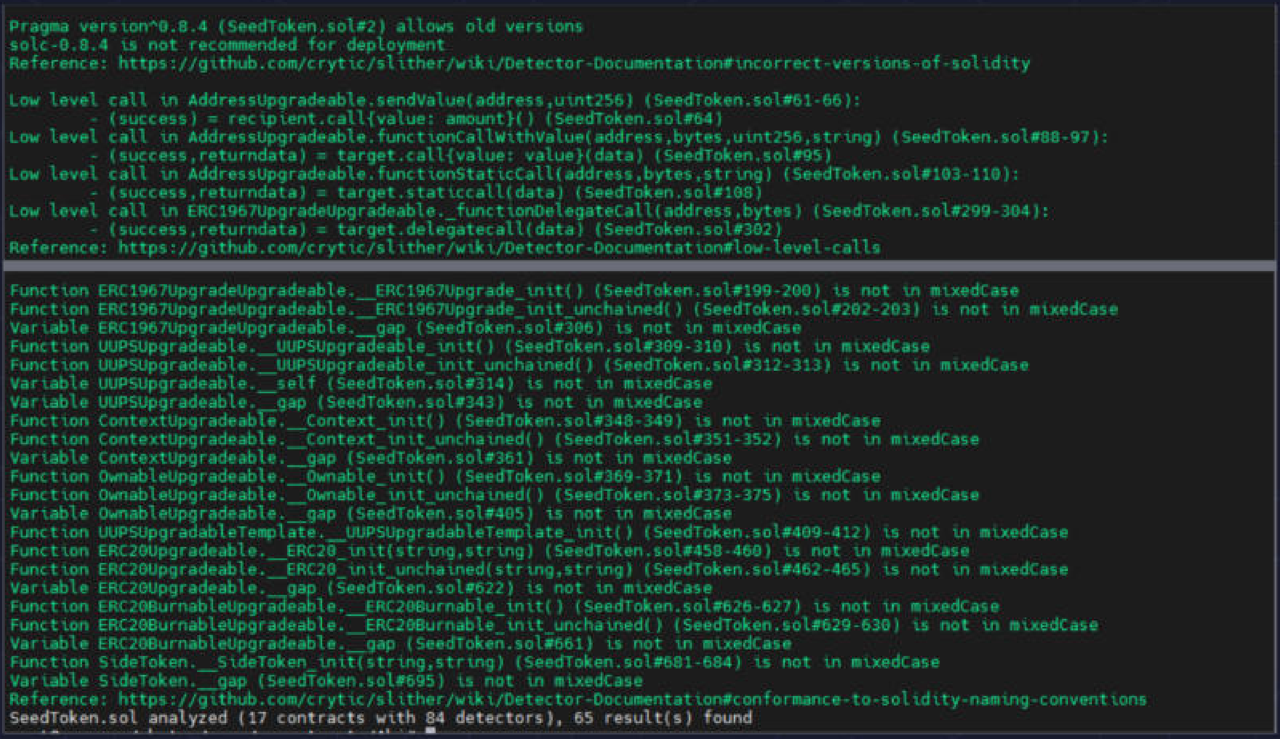


```
Pragma version0.8.4 (BudToken.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (BudToken.sol#62-67):
        - (success) = recipient.call{value: amount}() (BudToken.sol#65)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (BudToken.sol#89-98):
        - (success,returndata) = target.call{value: value}(data) (BudToken.sol#96)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (BudToken.sol#104-111):
        - (success,returndata) = target.staticcall(data) (BudToken.sol#109)
Low level call in ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (BudToken.sol#300-305):
        - (success,returndata) = target.delegatecall(data) (BudToken.sol#303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```



```
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init() (BudToken.sol#200-201) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init_unchained() (BudToken.sol#203-204) is not in mixedCase
Variable ERC1967UpgradeUpgradeable.__gap (BudToken.sol#307) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init() (BudToken.sol#310-311) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init_unchained() (BudToken.sol#313-314) is not in mixedCase
Variable UUPSUpgradeable.__self (BudToken.sol#315) is not in mixedCase
Variable UUPSUpgradeable.__gap (BudToken.sol#344) is not in mixedCase
Function ContextUpgradeable.__Context_init() (BudToken.sol#349-350) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (BudToken.sol#352-353) is not in mixedCase
Variable ContextUpgradeable.__gap (BudToken.sol#362) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (BudToken.sol#370-372) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (BudToken.sol#374-376) is not in mixedCase
Variable OwnableUpgradeable.__gap (BudToken.sol#406) is not in mixedCase
Function UUPSUpgradableTemplate.__UUPSUpgradableTemplate_init() (BudToken.sol#410-413) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (BudToken.sol#460-462) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (BudToken.sol#464-467) is not in mixedCase
Variable ERC20Upgradeable.__gap (BudToken.sol#624) is not in mixedCase
Function ERC20BurnableUpgradeable.__ERC20Burnable_init() (BudToken.sol#628-629) is not in mixedCase
Function ERC20BurnableUpgradeable.__ERC20Burnable_init_unchained() (BudToken.sol#631-632) is not in mixedCase
Variable ERC20BurnableUpgradeable.__gap (BudToken.sol#663) is not in mixedCase
Function SideToken.__SideToken_init(string,string) (BudToken.sol#683-686) is not in mixedCase
Variable SideToken.__gap (BudToken.sol#697) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
BudToken.sol analyzed (17 contracts with 84 detectors), 65 result(s) found
```

Slither log >> SeedToken.sol

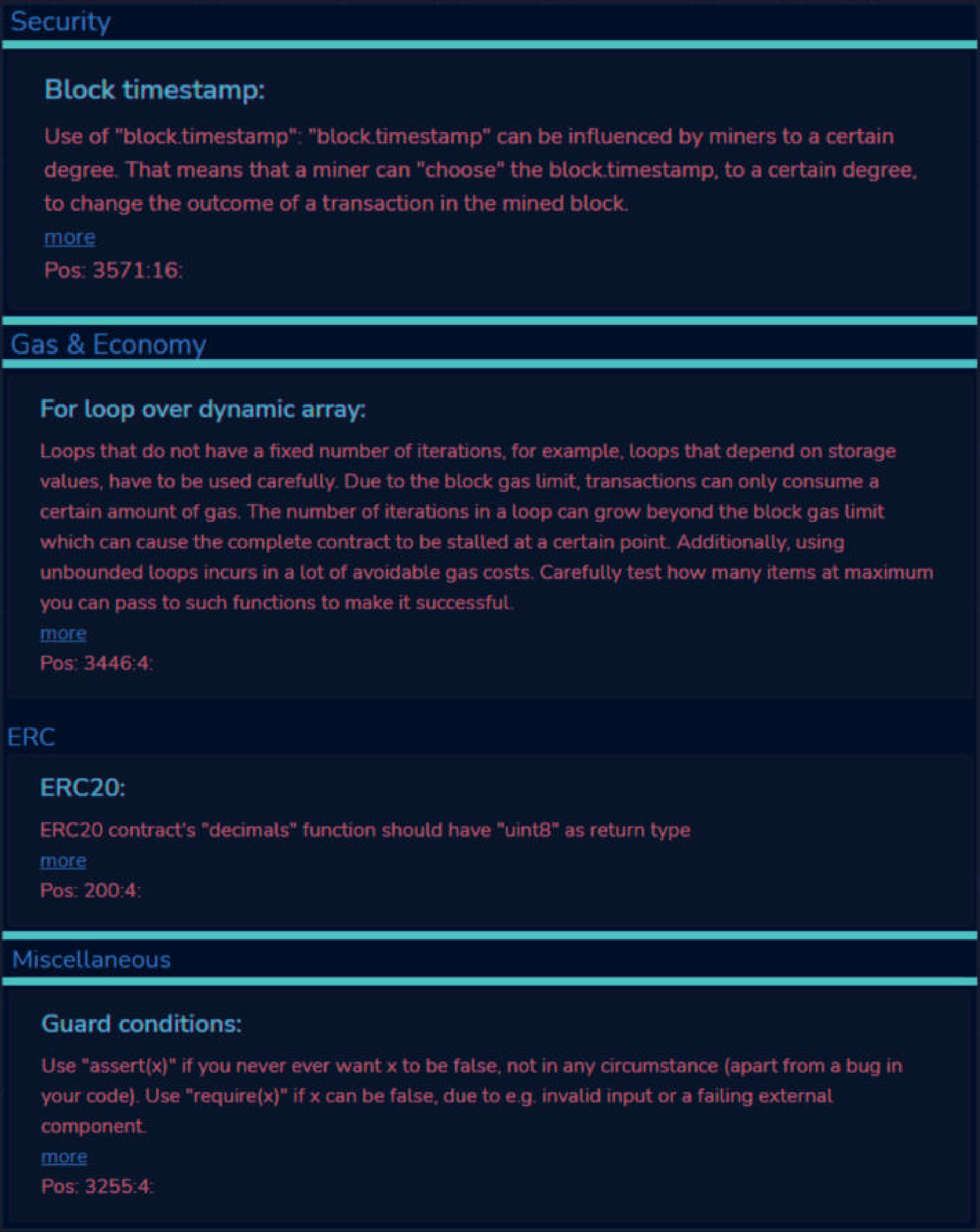


```
Pragma version^0.8.4 (SeedToken.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (SeedToken.sol#61-66):
        - (success) = recipient.call{value: amount}() (SeedToken.sol#64)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (SeedToken.sol#88-97):
        - (success,returndata) = target.call{value: value}(data) (SeedToken.sol#95)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (SeedToken.sol#103-110):
        - (success,returndata) = target.staticcall(data) (SeedToken.sol#108)
Low level call in ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (SeedToken.sol#299-304):
        - (success,returndata) = target.delegatecall(data) (SeedToken.sol#302)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init() (SeedToken.sol#199-200) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init_unchained() (SeedToken.sol#202-203) is not in mixedCase
Variable ERC1967UpgradeUpgradeable.__gap (SeedToken.sol#306) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init() (SeedToken.sol#309-310) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init_unchained() (SeedToken.sol#312-313) is not in mixedCase
Variable UUPSUpgradeable.__self (SeedToken.sol#314) is not in mixedCase
Variable UUPSUpgradeable.__gap (SeedToken.sol#343) is not in mixedCase
Function ContextUpgradeable.__Context_init() (SeedToken.sol#348-349) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (SeedToken.sol#351-352) is not in mixedCase
Variable ContextUpgradeable.__gap (SeedToken.sol#361) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (SeedToken.sol#369-371) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (SeedToken.sol#373-375) is not in mixedCase
Variable OwnableUpgradeable.__gap (SeedToken.sol#405) is not in mixedCase
Function UUPSUpgradableTemplate.__UUPSUpgradableTemplate_init() (SeedToken.sol#409-412) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (SeedToken.sol#458-460) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (SeedToken.sol#462-465) is not in mixedCase
Variable ERC20Upgradeable.__gap (SeedToken.sol#622) is not in mixedCase
Function ERC20BurnableUpgradeable.__ERC20Burnable_init() (SeedToken.sol#626-627) is not in mixedCase
Function ERC20BurnableUpgradeable.__ERC20Burnable_init_unchained() (SeedToken.sol#629-630) is not in mixedCase
Variable ERC20BurnableUpgradeable.__gap (SeedToken.sol#661) is not in mixedCase
Function SideToken.__SideToken_init(string,string) (SeedToken.sol#681-684) is not in mixedCase
Variable SideToken.__gap (SeedToken.sol#695) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
SeedToken.sol analyzed (17 contracts with 84 detectors), 65 result(s) found
```

SOLIDITY STATIC ANALYSIS

NftFactory.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: 3571:16:

Gas & Economy

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: 3446:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 200:4:

Miscellaneous

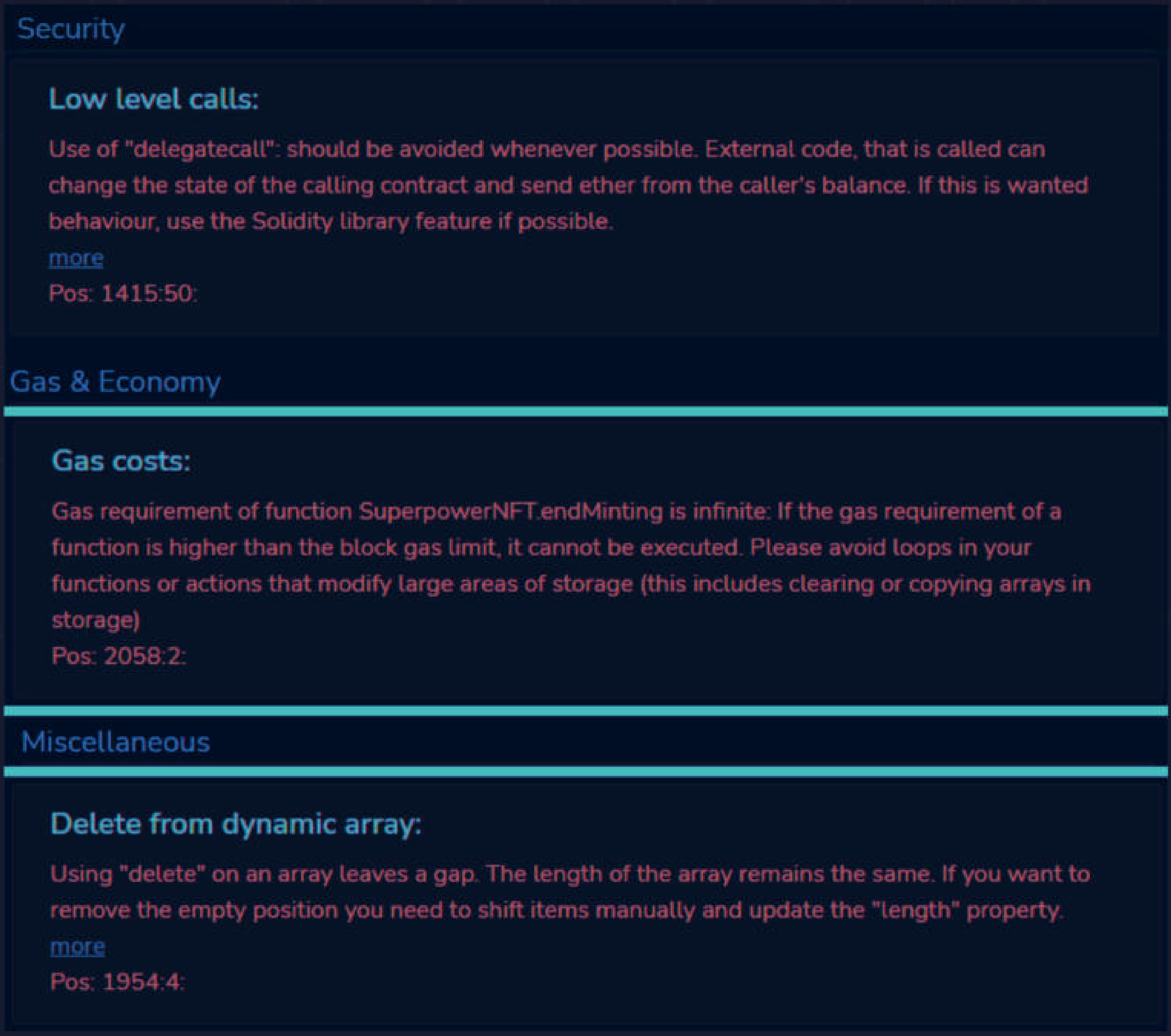
Guard conditions:

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: 3255:4:

SuperpowerNFT.sol



Security

Low level calls:

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

more

Pos: 1415:50:

Gas & Economy

Gas costs:

Gas requirement of function SuperpowerNFT.endMinting 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: 2058:2:

Miscellaneous

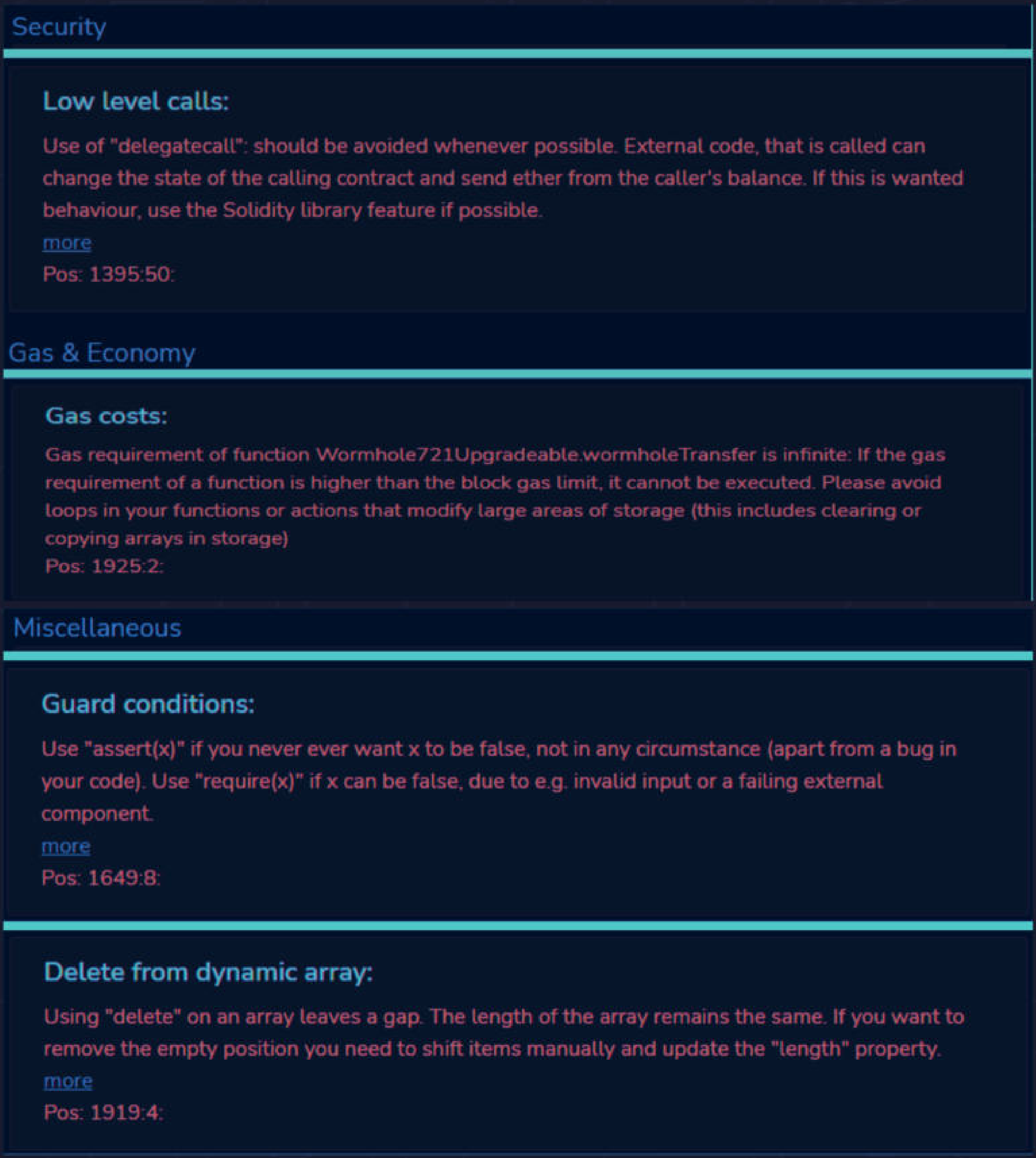
Delete from dynamic array:

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: 1954:4:

SuperpowerNFTBase.sol



Security

Low level calls:

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

more

Pos: 1395:50:

Gas & Economy

Gas costs:

Gas requirement of function Wormhole721Upgradeable.wormholeTransfer 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: 1925:2:

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: 1649:8:

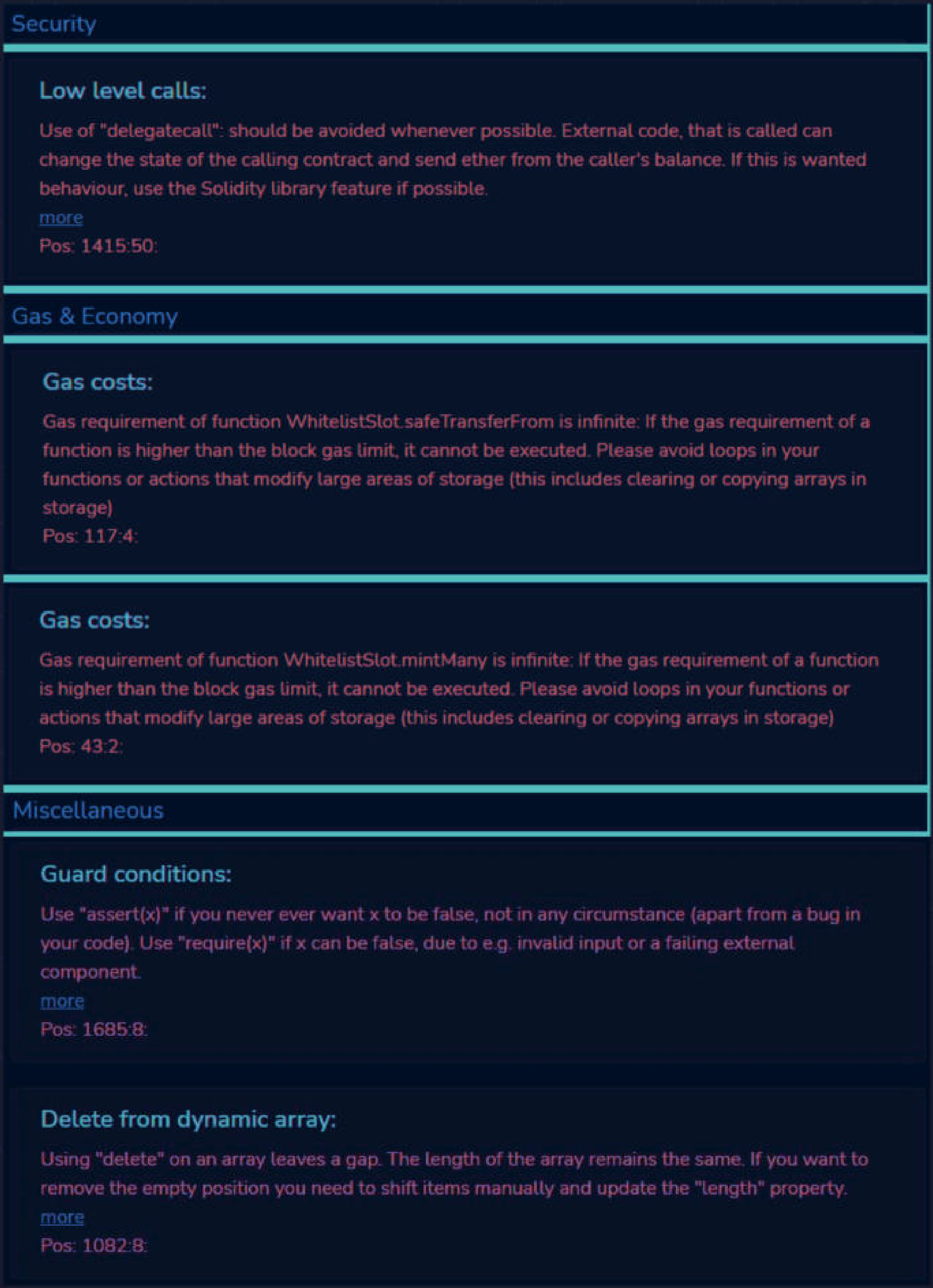
Delete from dynamic array:

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

more

Pos: 1919:4:

WhitelistSlot.sol



Security

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 1415:50:

Gas & Economy

Gas costs:

Gas requirement of function WhitelistSlot.safeTransferFrom 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: 117:4:

Gas costs:

Gas requirement of function WhitelistSlot.mintMany 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: 43:2:

Miscellaneous

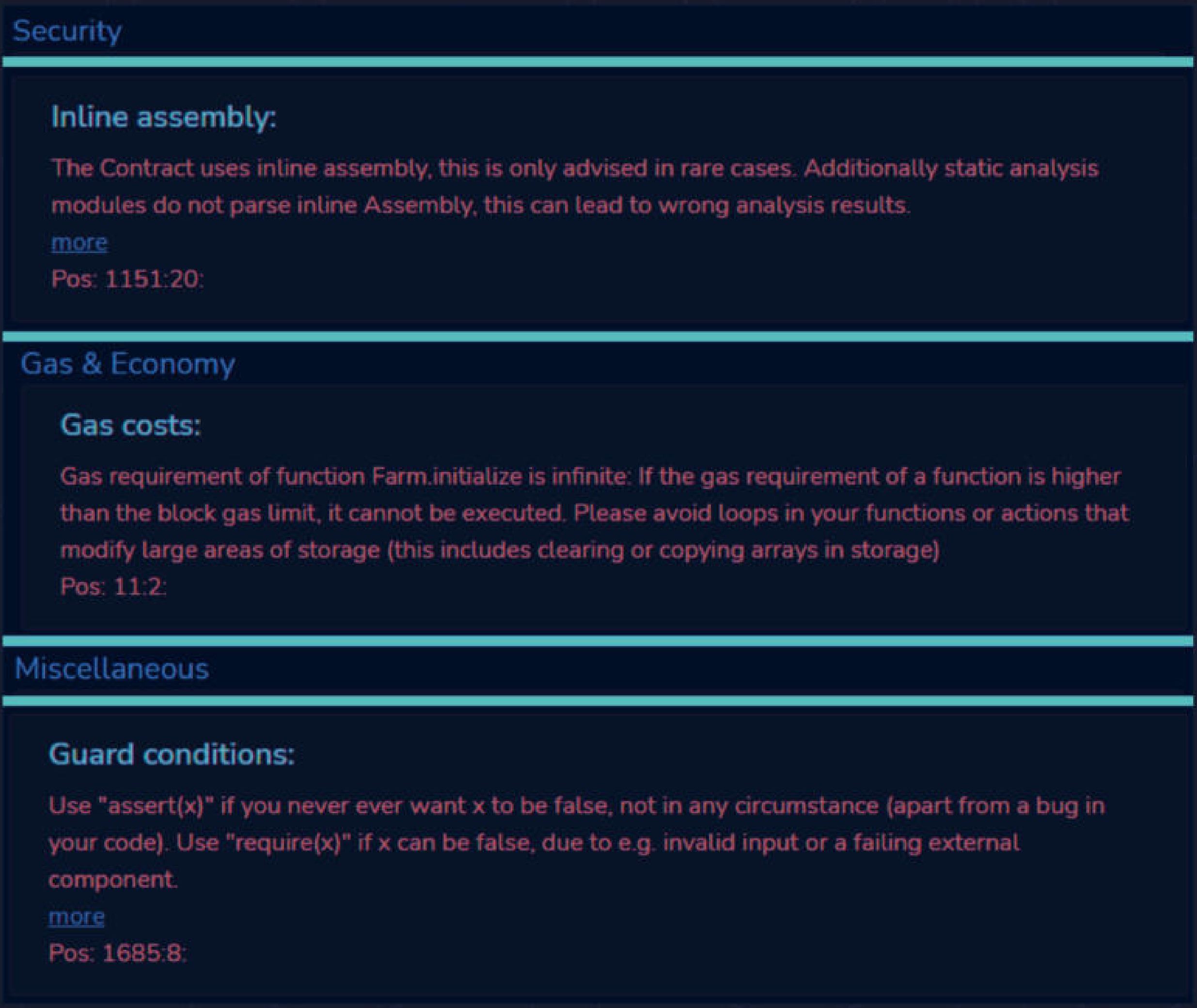
Guard conditions:

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: 1685:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 1082:8:

Farm.sol



Security

Inline assembly:

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

more

Pos: 1151:20:

Gas & Economy

Gas costs:

Gas requirement of function Farm.initialize 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: 11:2:

Miscellaneous

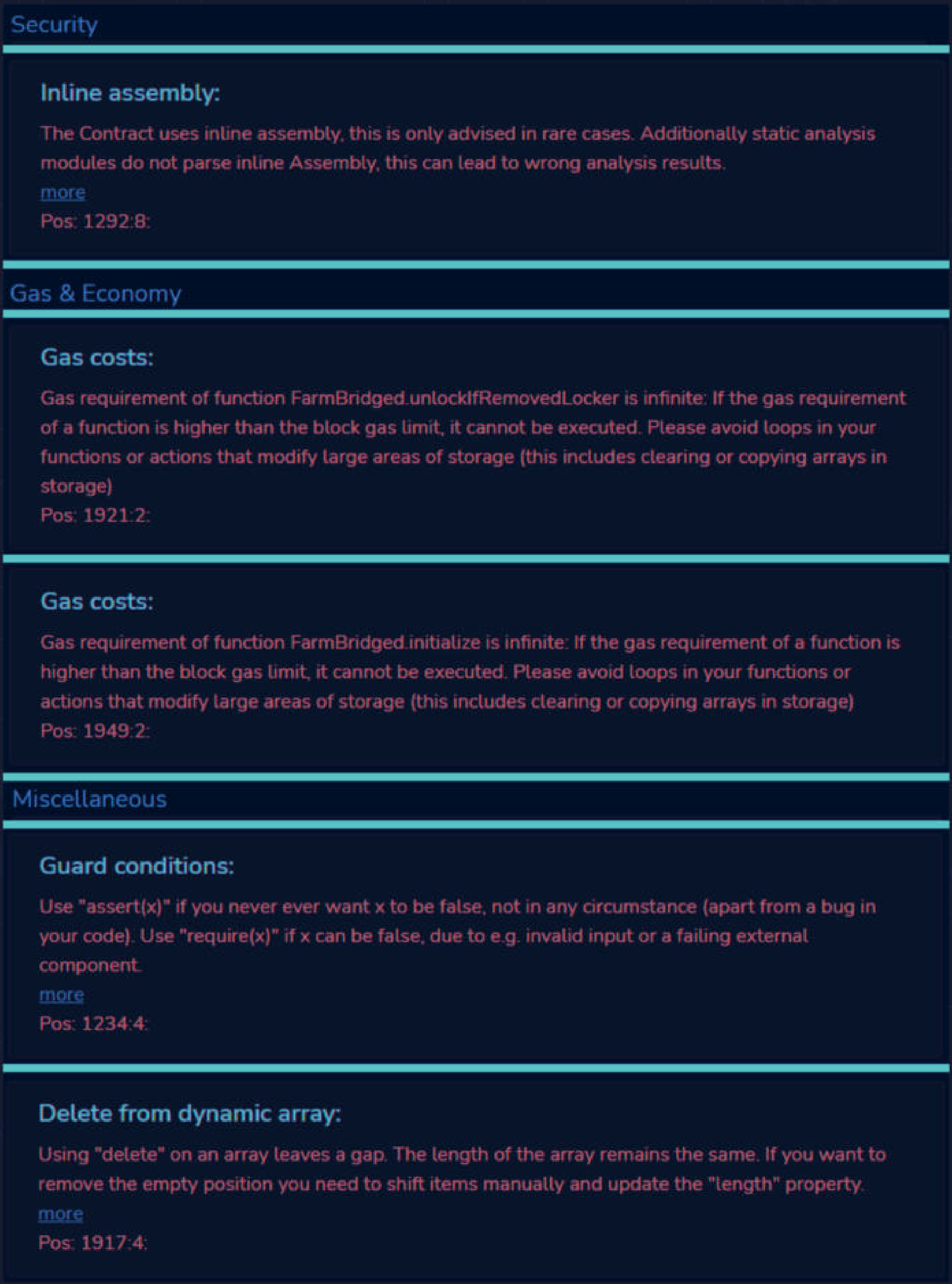
Guard conditions:

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: 1685:8:

FarmBridged.sol



Security

Inline assembly:

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

more

Pos: 1292:8:

Gas & Economy

Gas costs:

Gas requirement of function FarmBridged.unlockIfRemovedLocker 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: 1921:2:

Gas costs:

Gas requirement of function FarmBridged.initialize 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: 1949:2:

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: 1234:4:

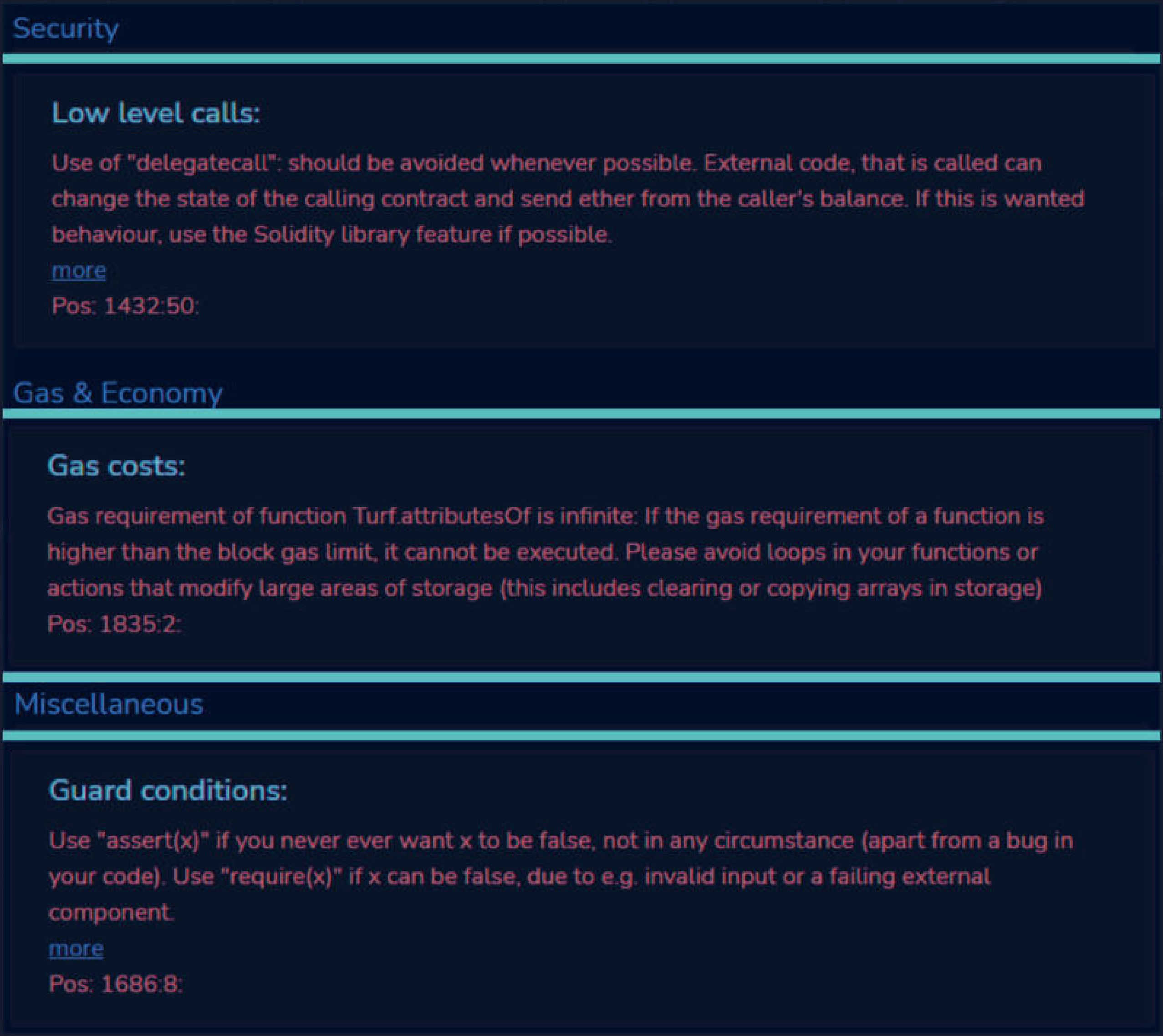
Delete from dynamic array:

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: 1917:4:

Turf.sol



Security

Low level calls:

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

more

Pos: 1432:50:

Gas & Economy

Gas costs:

Gas requirement of function Turf.attributesOf 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: 1835:2:

Miscellaneous

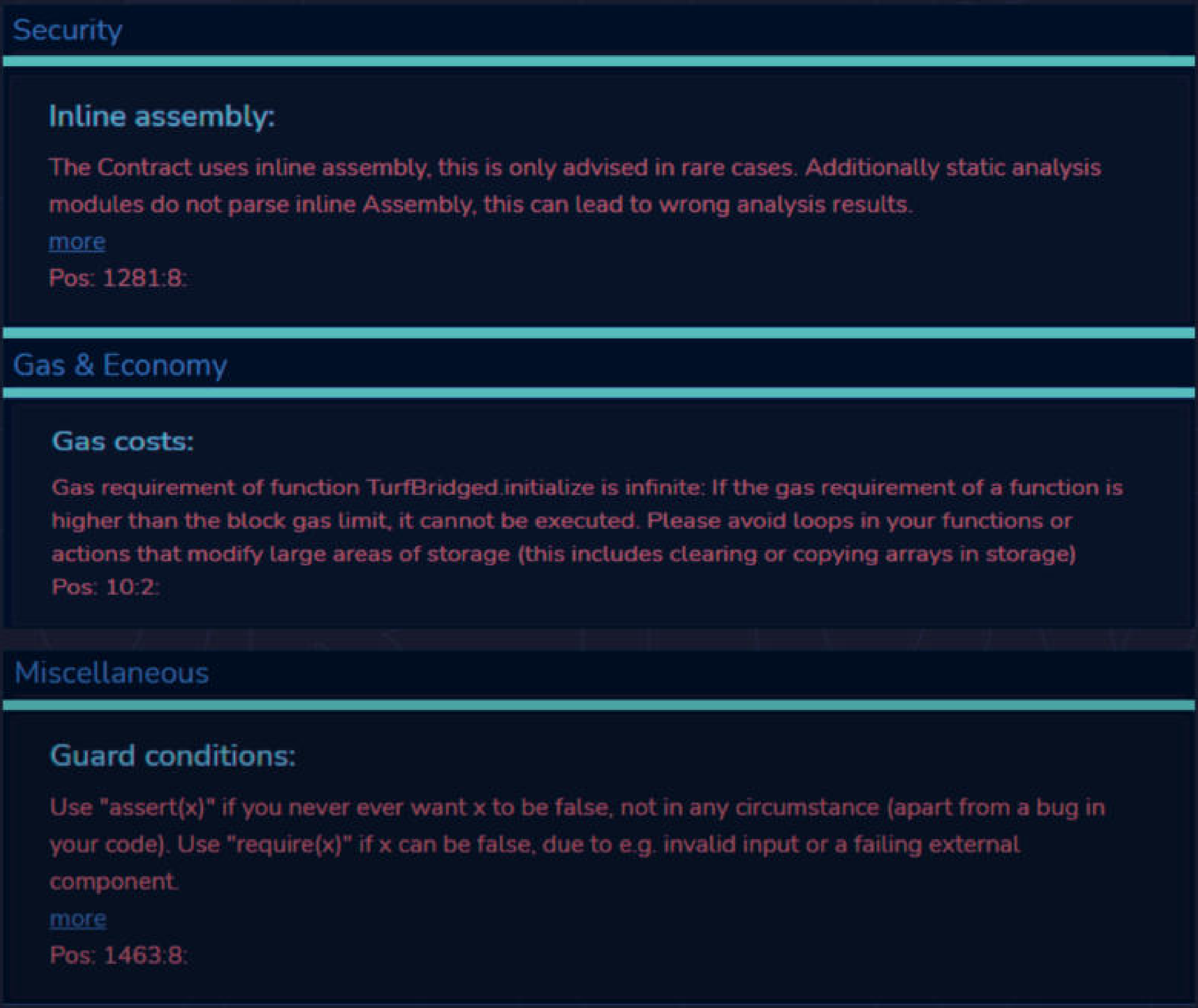
Guard conditions:

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: 1686:8:

TurfBridged.sol



Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 1281:8:

Gas & Economy

Gas costs:

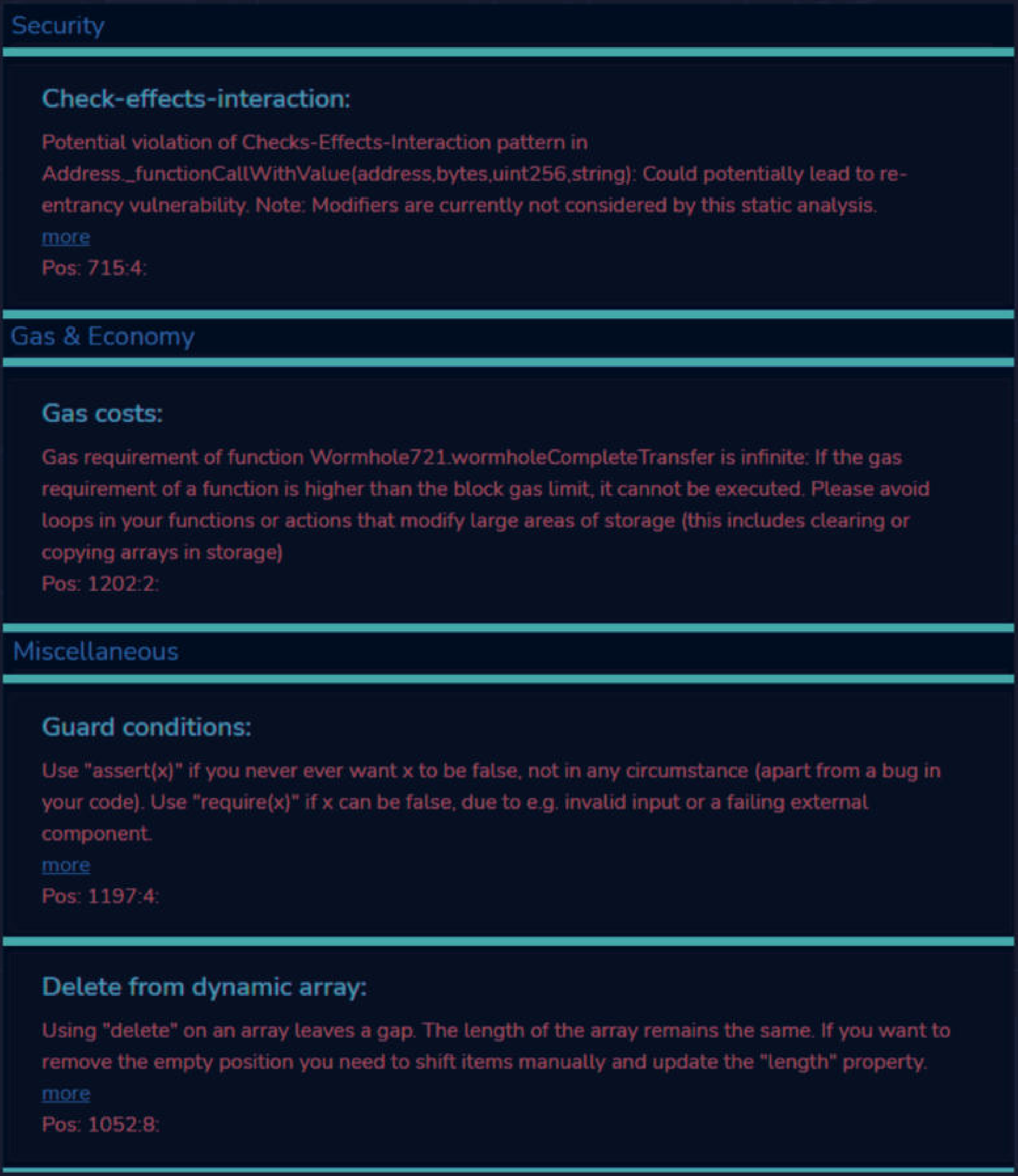
Gas requirement of function TurfBridged.initialize is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 10:2:

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: 1463:8:

Wormhole721.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: 715:4:

Gas & Economy

Gas costs:

Gas requirement of function Wormhole721.wormholeCompleteTransfer 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: 1202:2:

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: 1197:4:

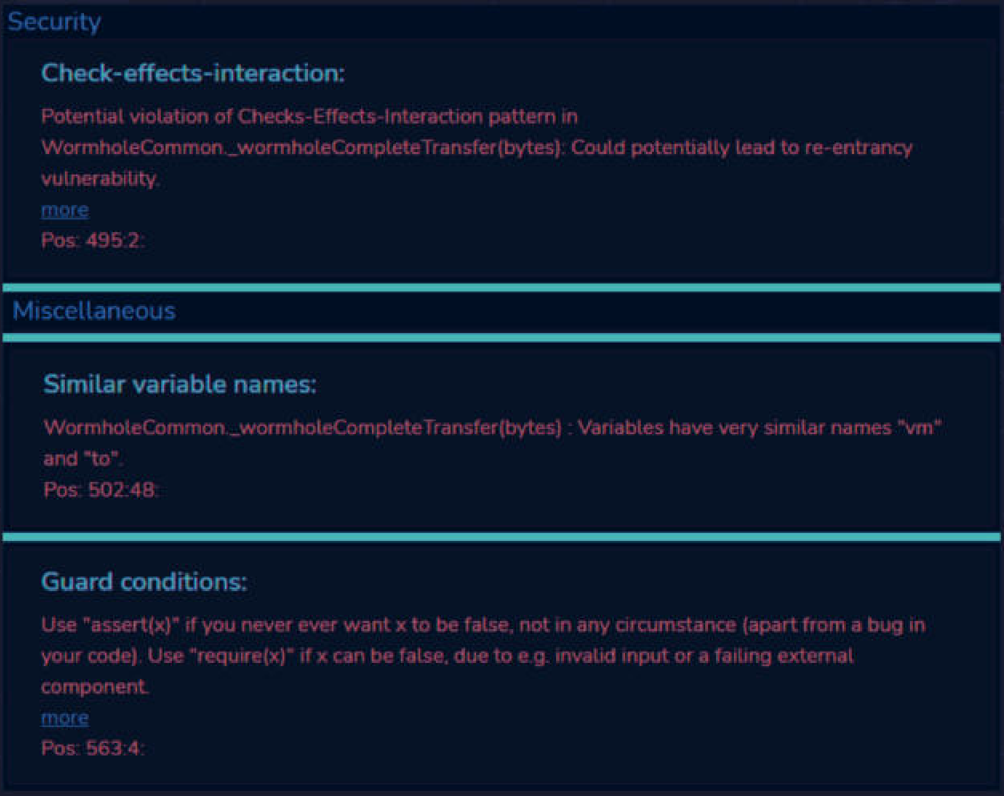
Delete from dynamic array:

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: 1052:8:

WormholeCommon.sol



Security

Check-effects-interaction:

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

more

Pos: 495:2:

Miscellaneous

Similar variable names:

WormholeCommon._wormholeCompleteTransfer(bytes) : Variables have very similar names "vm" and "to".

Pos: 502:48:

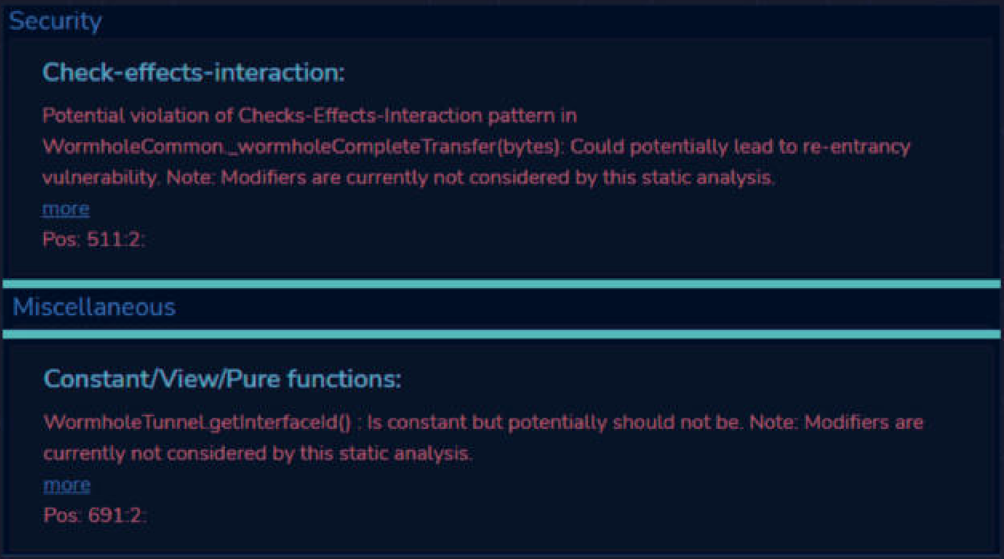
Guard conditions:

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: 563:4:

WormholeTunnel.sol



Security

Check-effects-interaction:

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

more

Pos: 511:2:

Miscellaneous

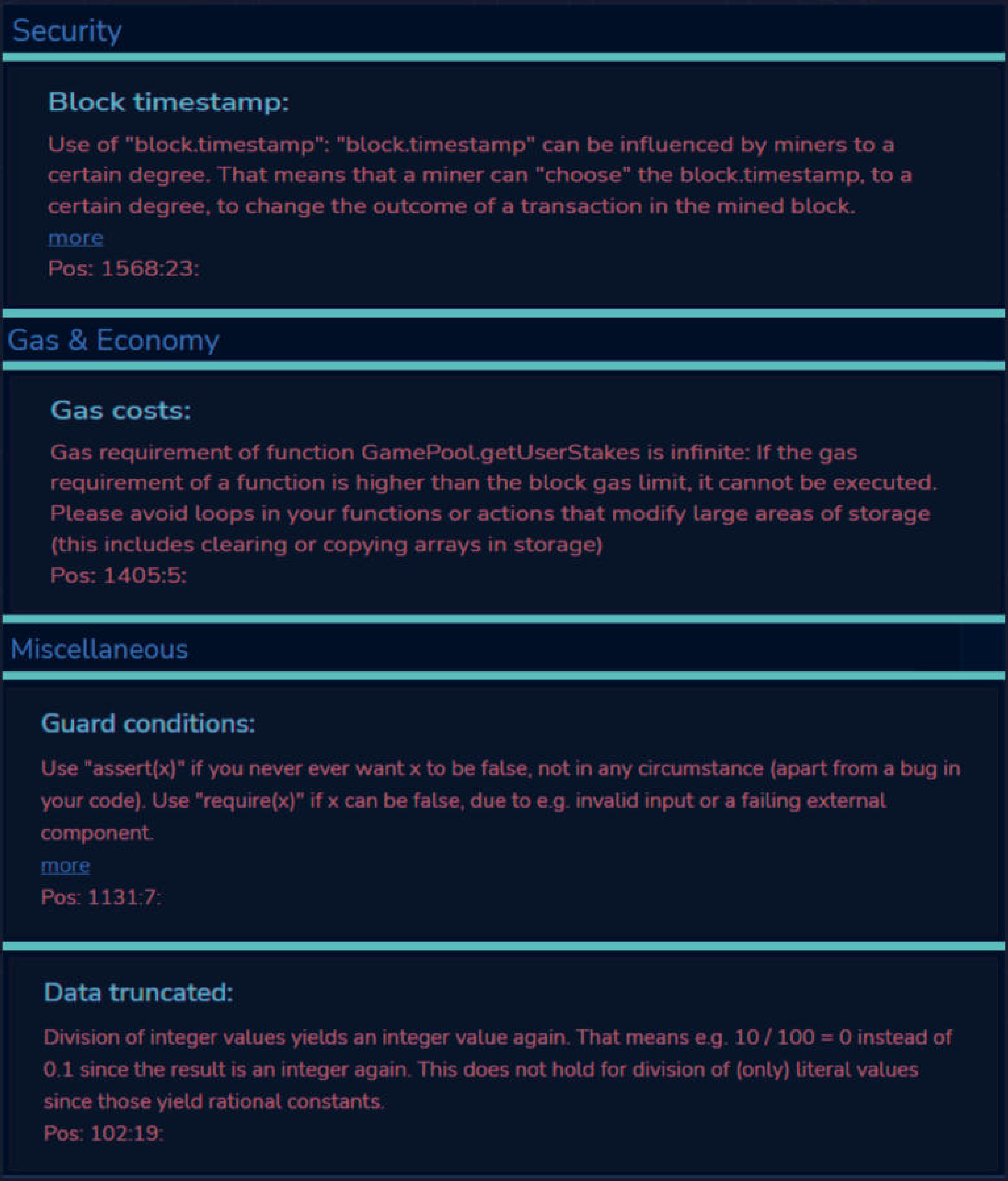
Constant/View/Pure functions:

WormholeTunnel.getInterfaceId() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 691:2:

WormholeTunnel.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: 1568:23:

Gas & Economy

Gas costs:

Gas requirement of function GamePool.getUserStakes 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: 1405:5:

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

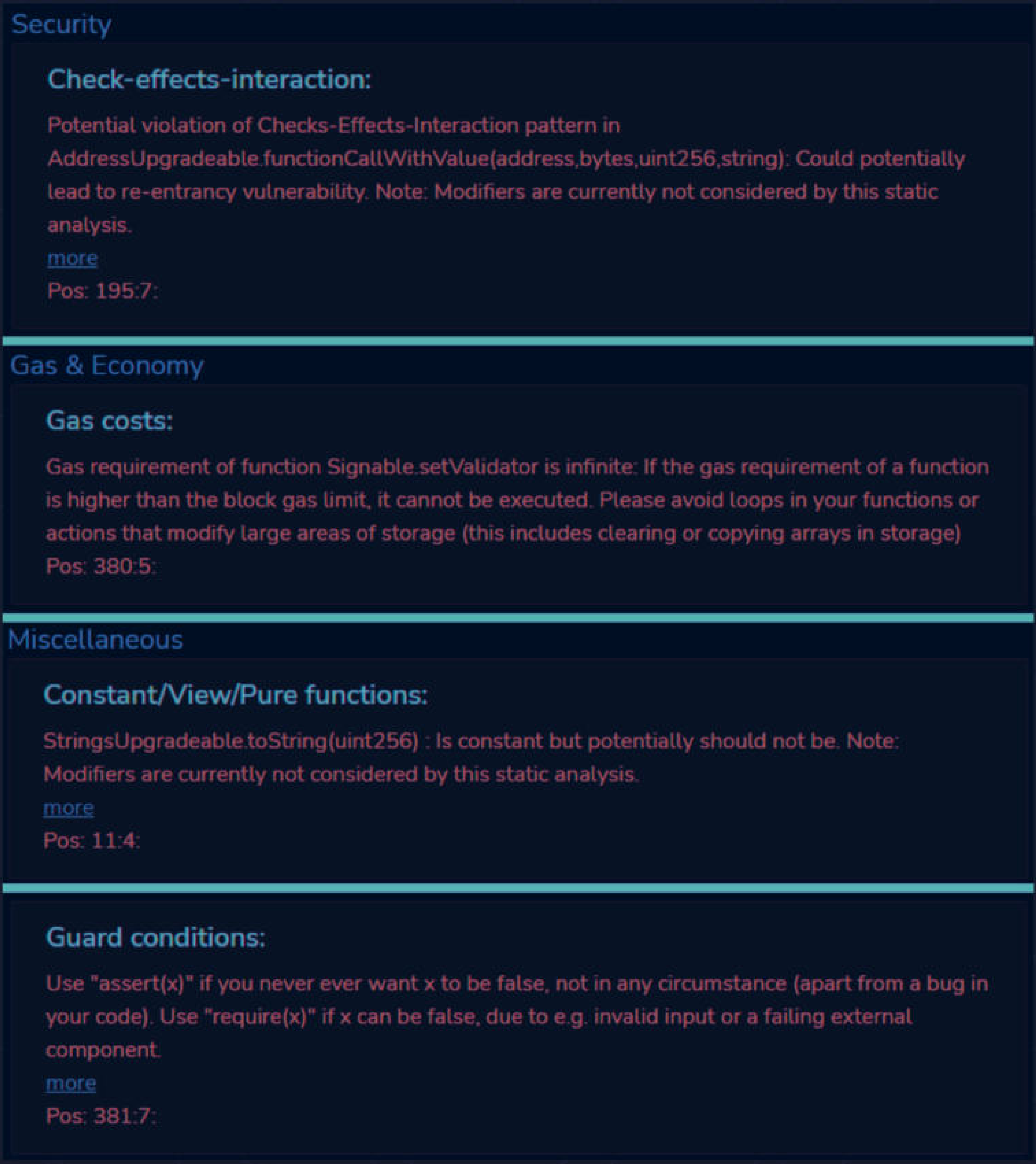
Pos: 1131:7:

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: 102:19:

Signable.sol



Security

Check-effects-interaction:

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

more

Pos: 195:7:

Gas & Economy

Gas costs:

Gas requirement of function Signable.setValidator 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: 380:5:

Miscellaneous

Constant/View/Pure functions:

StringsUpgradeable.toString(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 11:4:

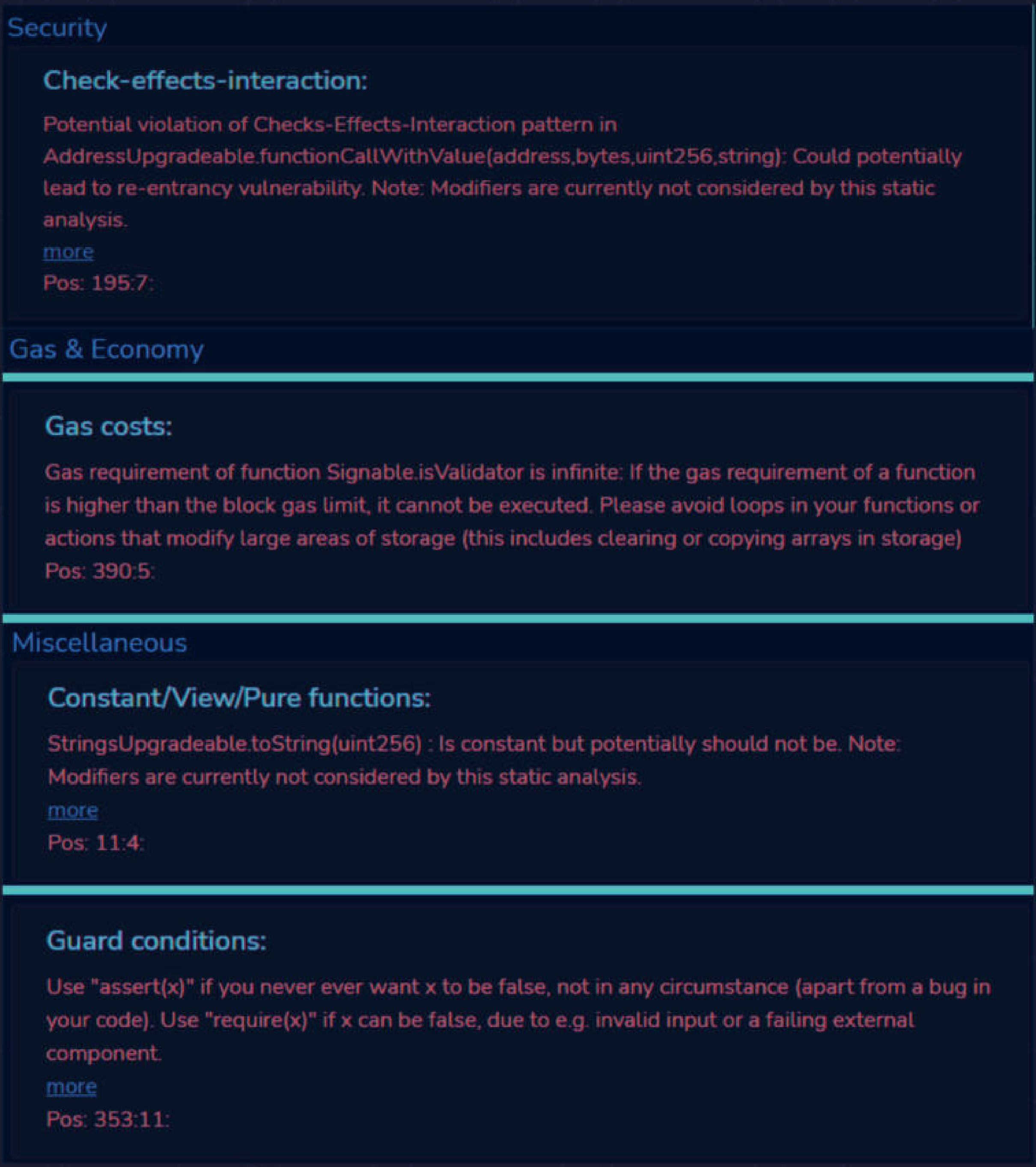
Guard conditions:

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: 381:7:

SignableStakes.sol



Security

Check-effects-interaction:

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

more

Pos: 195:7:

Gas & Economy

Gas costs:

Gas requirement of function Signable.isValidator 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: 390:5:

Miscellaneous

Constant/View/Pure functions:

StringsUpgradeable.toString(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 11:4:

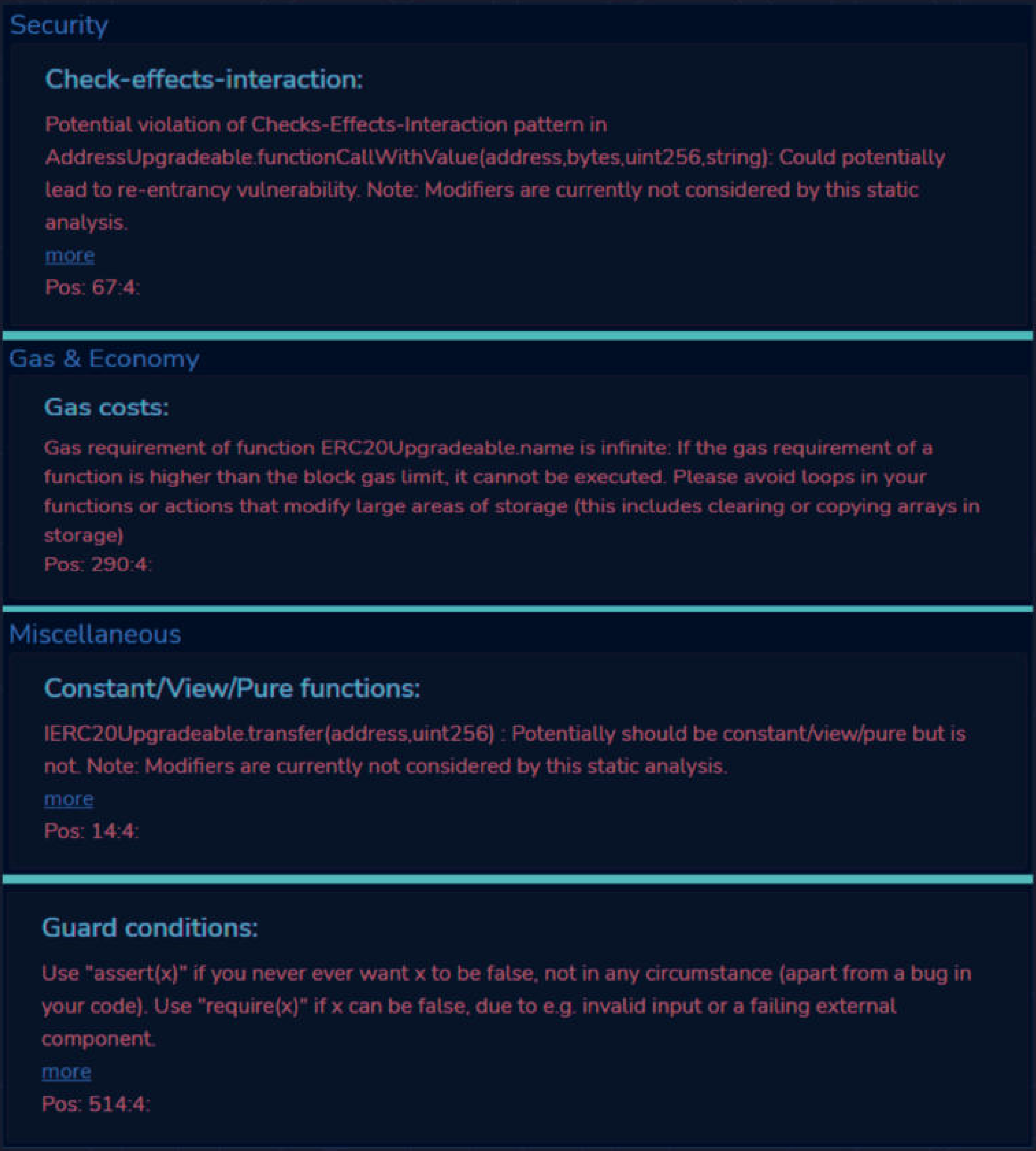
Guard conditions:

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: 353:11:

SideToken.sol



Security

Check-effects-interaction:

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

more

Pos: 67:4:

Gas & Economy

Gas costs:

Gas requirement of function ERC20Upgradeable.name 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: 290:4:

Miscellaneous

Constant/View/Pure functions:

IERC20Upgradeable.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 14:4:

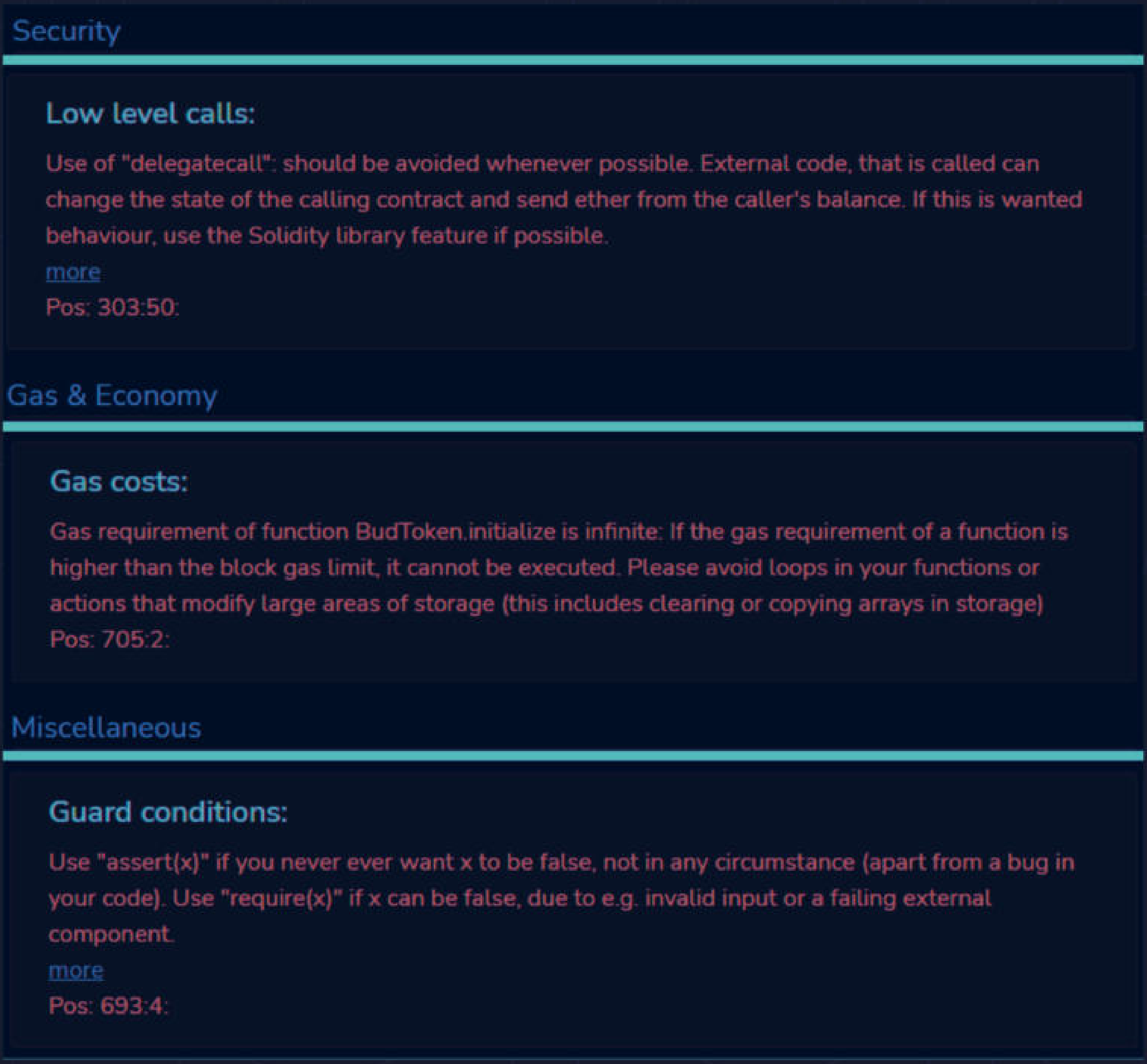
Guard conditions:

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: 514:4:

BudToken.sol



Security

Low level calls:

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

more

Pos: 303:50:

Gas & Economy

Gas costs:

Gas requirement of function BudToken.initialize 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: 705:2:

Miscellaneous

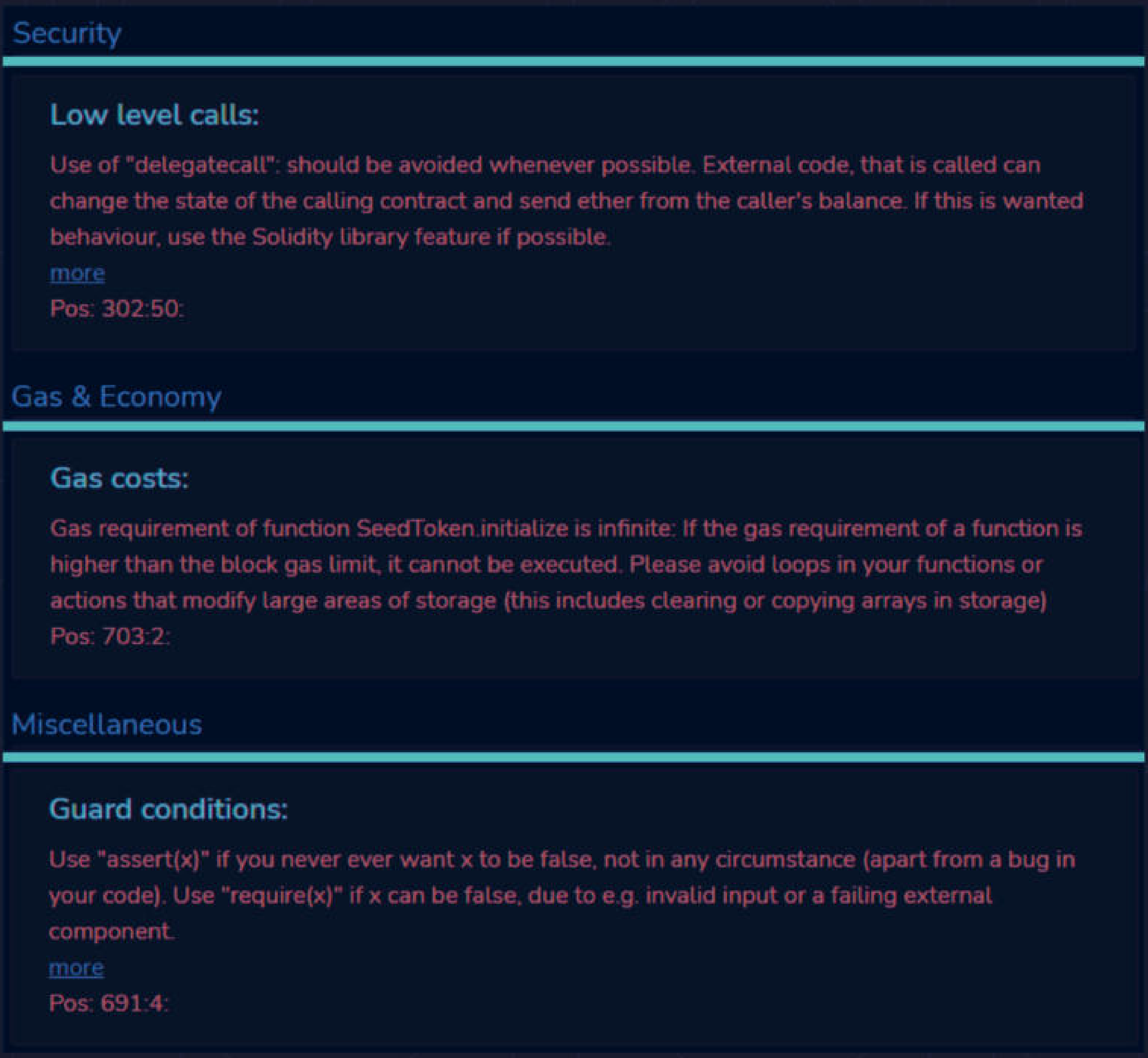
Guard conditions:

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: 693:4:

SeedToken.sol



Security

Low level calls:

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

more

Pos: 302:50:

Gas & Economy

Gas costs:

Gas requirement of function SeedToken.initialize 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: 703:2:

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: 691:4:

COMPLIANCE ANALYSIS

NftFactory.sol

```
NftFactory.sol:3565:64: Error: Parse error: mismatched input '(' expecting {';', '='}
NftFactory.sol:3566:60: Error: Parse error: mismatched input '(' expecting {';', '='}
NftFactory.sol:3567:79: Error: Parse error: mismatched input '(' expecting {';', '='}
NftFactory.sol:3568:99: Error: Parse error: mismatched input '(' expecting {';', '='}
NftFactory.sol:3573:99: Error: Parse error: mismatched input '(' expecting {';', '='}
NftFactory.sol:3576:67: Error: Parse error: mismatched input '(' expecting {';', '='}
NftFactory.sol:3581:106: Error: Parse error: mismatched input '(' expecting {';', '='}
NftFactory.sol:3596:73: Error: Parse error: mismatched input '(' expecting {';', '='}
```

SuperpowerNFT.sol

```
SuperpowerNFT.sol:2048:26: Error: Parse error: mismatched input '(' expecting {';', '='}
SuperpowerNFT.sol:2064:25: Error: Parse error: mismatched input '(' expecting {';', '='}
SuperpowerNFT.sol:2068:49: Error: Parse error: mismatched input '(' expecting {';', '='}
SuperpowerNFT.sol:2073:49: Error: Parse error: mismatched input '(' expecting {';', '='}
SuperpowerNFT.sol:2081:55: Error: Parse error: mismatched input '(' expecting {';', '='}
SuperpowerNFT.sol:2086:51: Error: Parse error: mismatched input '('
```

SuperpowerNFTBase.sol

```
SuperpowerNFTBase.sol:1887:48: Error: Parse error: mismatched input '(' expecting {';', '='}
SuperpowerNFTBase.sol:1942:24: Error: Parse error: mismatched input '(' expecting {';', '='}
SuperpowerNFTBase.sol:1950:27: Error: Parse error: mismatched input '(' expecting {';', '='}
SuperpowerNFTBase.sol:1953:34: Error: Parse error: mismatched input '(' expecting {';', '='}
SuperpowerNFTBase.sol:1963:24: Error: Parse error: mismatched input '(' expecting {';', '='}
SuperpowerNFTBase.sol:1977:34: Error: Parse error: mismatched input '(' expecting {';', '='}
SuperpowerNFTBase.sol:2000:43: Error: Parse error: mismatched input '(' expecting {';', '='}
```

WhitelistSlot.sol

```
WhitelistSlot.sol:13:20: Error: Parse error: mismatched input '(' expecting {';', '='}
WhitelistSlot.sol:14:20: Error: Parse error: mismatched input '(' expecting {';', '='}
WhitelistSlot.sol:15:26: Error: Parse error: mismatched input '(' expecting {';', '='}
WhitelistSlot.sol:30:25: Error: Parse error: mismatched input '(' expecting {';', '='}
WhitelistSlot.sol:48:90: Error: Parse error: mismatched input '(' expecting {';', '='}
WhitelistSlot.sol:60:25: Error: Parse error: mismatched input '(' expecting {';', '='}
```

Farm.sol

```
Farm.sol:2:1: Error: Compiler version 0.8.17 does not satisfy the r semver requirement
```

Farm.sol

```
FarmBridged.sol:2:1: Error: Compiler version 0.8.17 does not satisfy the r semver
requirement
```

Turf.sol

```
Turf.sol:2:1: Error: Compiler version 0.8.17 does not satisfy the r semver requirement
```

TurfBridged.sol

```
TurfBridged.sol:2:1: Err
```

Wormhole721.sol

```
Wormhole721.sol:1010:18: Error: Parse error: missing ';' at '{'
Wormhole721.sol:1030:18: Error: Parse error: missing ';' at '{'
Wormhole721.sol:1054:18: Error: Parse error: missing ';' at '{'
```

WormholeCommon.sol

```
WormholeCommon.sol:272:5: Error: Avoid using inline assembly. It is
acceptable only in rare cases
WormholeCommon.sol:282:5: Error: Avoid using inline assembly. It is
acceptable only in rare cases
WormholeCommon.sol:315:5: Error: Avoid using inline assembly. It is
acceptable only in rare cases
WormholeCommon.sol:340:17: Error: Code contains empty blocks
WormholeCommon.sol:461:3: Error: Explicitly mark visibility of state
```

WormholeTunnel.sol

```
WormholeTunnel.sol:340:17: Error: Code contains empty blocks
WormholeTunnel.sol:477:3: Error: Explicitly mark visibility of state
WormholeTunnel.sol:587:75: Error: Code contains empty blocks
WormholeTunnel.sol:602:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0) WormholeTunnel.sol:641:5: Error:
Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity
>=0.7.0)
```

GamePool.sol

```
GamePool.sol:1605:51: Error: Parse error: mismatched input '(' expecting {';', '='}
GamePool.sol:1646:36: Error: Parse error: mismatched input '(' expecting {';', '='}
GamePool.sol:1648:38: Error: Parse error: mismatched input '(' expecting {';', '='}
GamePool.sol:1690:36: Error: Parse error: mismatched input '(' expecting {';', '='}
GamePool.sol:1692:38: Error: Parse error: mismatched input '(' expecting {';', '='}
```

Signable.sol

```
Signable.sol:2:1: Error: Compiler version 0.8.17 does not satisfy the r semver
requirement
Signable.sol:90:13: Error: Avoid using inline
assembly. It is
acceptable only in rare cases
Signable.sol:327:5: Error: Function name must be in mixedCase
Signable.sol:331:5: Error: Function name must be in mixedCase
```

SignableStakes.sol

```
SignableStakes.sol:310:5: Error: Function name must be in mixedCase
SignableStakes.sol:310:67: Error: Code contains empty blocks
SignableStakes.sol:327:5: Error: Function name must be in mixedCase
SignableStakes.sol:331:5: Error: Function name must be in mixedCase
```

SideToken.sol

```
SideToken.sol:347:18: Error: Parse error: missing ';' at '{'
SideToken.sol:366:18: Error: Parse error: missing ';' at '{'
SideToken.sol:382:18: Error: Parse error: missing ';' at '{'
SideToken.sol:397:18: Error: Parse error: missing ';' at '{'
SideToken.sol:427:22: Error: Parse error: missing ';' at '{'
```

BudToken.sol

```
BudToken.sol:526:18: Error: Parse error: missing ';' at '{'
BudToken.sol:545:18: Error: Parse error: missing ';' at '{'
BudToken.sol:561:18: Error: Parse error: missing ';' at '{'
BudToken.sol:576:18: Error: Parse error: missing ';' at '{'
BudToken.sol:606:22: Error: Parse error: missing ';' at '{'
```

SeedToken.sol

```
SeedToken.sol:524:18: Error: Parse error: missing ';' at '{'
SeedToken.sol:543:18: Error: Parse error: missing ';' at '{'
SeedToken.sol:559:18: Error: Parse error: missing ';' at '{'
SeedToken.sol:574:18: Error: Parse error: missing ';' at '{'
SeedToken.sol:604:22: Error: Parse error: missing ';' at '{'
```

Software analysis result:

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

INSPECTOR
LOVELY

INSPECTOR LOVELY

INFO

Website: Inspector.lovely.finance

Telegram community: t.me/inspectorlovely

Twitter: twitter.com/InspectorLovely

inspector.lovely.finance