EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Starfish OS Protocol
Website:	https://www.sfos.io/
Platform:	Binance Smart Chain
Language:	Solidity
Date:	June 16th, 2022

Table of contents

Introduction

EtherAuthority was contracted by the Starfish OS team to perform the Security audit of the smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 16th, 2022.

The purpose of this audit was to address the following:

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

Project Background

- Starfish organizational consensus collaboration system. Starfish OS is a value community that aggregates users, KOLs, media and organizations.
- Starfish OS is a Web3 ecosystem that integrates linked games, NFT, DAO, and DeFi.
- Starfish OS is a lightweight nurturance GameFi Starfish needs to be fed to grow! Value loop The consumption and output of starfish SFO keep a dynamic balance.Consensus lock Lock the consensus of ecological long-term value with the formation mechanism.
- Starfish OS is an NFT smart contract having functions like daoEdit, daoList, daoRemove, daoCouncliList, proposalEdit, app, nftTranfer, etc.

Audit scope

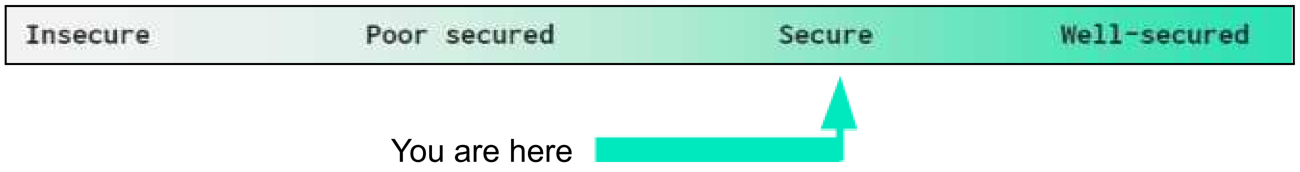
Name	Code Review and Security Analysis Report for Starfish OS Protocol Smart Contracts
Platform	**BSC / Solidity**
File 1	KOL_ProtectV2.sol
File 1 MD5 Hash	6391A739854F05E2B6DFEC2CE4447275
File 2	SFO_DAO.sol
File 2 MD5 Hash	3E0D292F690E175E028E2C8EC3BE0C7C
File 3	SKT.sol
File 3 MD5 Hash	102284E2A576A13D0E1F9B7FBA153F7F
File 4	SPT.sol
File 4 MD5 Hash	4376E12509D30DE853E622CBEEE65C6B
File 5	SMT.sol
File 5 MD5 Hash	AC37733EF845A17C85F984A51AC81B60
File 6	SUT.sol
File 6 MD5 Hash	2AFB01A5D47B1AEF76FFF8A37CF9D81C
Audit Date	June 16th, 2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 KOL_ProtectV2.sol ● KOL_ProtectV2 has functions like: app, setProtect, etc.	YES, This is valid.
File 2 SFO_DAO.sol ● SFO_DAO has functions like: daoAdd, daoEdit, daoExist, daoRemove, etc.	YES, This is valid.
File 3 SKT.sol ● Name: StarFish-KOL-NFT ● Symbol: SKT	YES, This is valid.
File 4 SPT.sol ● Name: StarFish-Pro-NFT ● Symbol: SPT.	YES, This is valid.
File 5 SMT.sol ● Name: StarFish-KOL-NFT ● Symbol: SMT	YES, This is valid.
File 6 SUT.sol ● Name: StarFish-KOL-NFT ● Symbol: SUT	YES, This is valid.

Audit Summary

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



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

We found 0 critical, 0 high, 0 medium and 1 low and some very low level issues.

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

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	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 6 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 Starfish OS 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 Starfish OS Protocol.

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

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

Documentation

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

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

Another source of information was its official website https://www.sfos.io/ 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

KOL_ProtectV2.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	onERC721Received	write	Passed	No Issue
8	app	external	Passed	No Issue
9	setProtect	external	access only Owner	No Issue
10	tokenTranfer	external	access only Owner	No Issue
11	nftTranfer	external	access only Owner	No Issue

SFO_DAO.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	nonReentrant	modifier	Passed	No Issue
8	setTokenAllow	external	access only Owner	No Issue
9	daoAdd	external	Passed	No Issue
10	daoEdit	external	Passed	No Issue
11	daoList	external	Passed	No Issue
12	daoExist	read	Passed	No Issue
13	daoRemove	external	access only Owner	No Issue
14	daoCouncliList	external	Passed	No Issue
15	councliAssign	external	Infinite loop, Critical operation lacks event log	Refer Audit Findings
16	councliApply	external	Passed	No Issue
17	councliQuit	external	Passed	No Issue
18	councliAt	read	Passed	No Issue
19	lpToTokenPrice	read	Passed	No Issue
20	proposalAdd	external	Passed	No Issue
21	proposalEdit	external	Passed	No Issue
22	proposalFinsh	external	Passed	No Issue

23	proposalRemove	external	Critical operation lacks event log	Refer Audit Findings
24	prosalList	external	Passed	No Issue
25	daoJoin	external	Passed	No Issue
26	userDaoList	external	Passed	No Issue
27	daoQuit	external	Passed	No Issue
28	vote	external	Infinite loop	Refer Audit Findings
29	voteFinsh	external	Infinite loop	Refer Audit Findings
30	voteRecord	external	Passed	No Issue
31	voteRecordList	external	Passed	No Issue
32	userCouncli	read	Passed	No Issue
33	userVoteRecordQuery	external	Passed	No Issue
34	voteRecordConcat	write	Passed	No Issue
35	voteRecordResolve	external	Passed	No Issue

SKT.sol

Functions

Sl.	**Functions**	**Type**	**Observation**	**Conclusion**
1	constructor	write	Passed	No Issue
2	_baseURI	internal	Passed	No Issue
3	mint	write	Passed	No Issue
4	pause	write	Passed	No Issue
5	unpause	write	Passed	No Issue
6	_baseURI	internal	Passed	No Issue
7	setBaseTokenURI	write	Passed	No Issue

SPT.sol

Functions

Sl.	**Functions**	**Type**	**Observation**	**Conclusion**
1	constructor	write	Passed	No Issue
2	_baseURI	internal	Passed	No Issue
3	mint	write	Passed	No Issue
4	pause	write	Passed	No Issue
5	unpause	write	Passed	No Issue
6	_baseURI	internal	Passed	No Issue
7	setBaseTokenURI	write	Passed	No Issue
8	_beforeTokenTransfer	internal	Passed	No Issue

SMT.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_baseURI	internal	Passed	No Issue
3	mint	write	Passed	No Issue
4	pause	write	Passed	No Issue
5	unpause	write	Passed	No Issue
6	_baseURI	internal	Passed	No Issue
7	setBaseTokenURI	write	Passed	No Issue
8	_beforeTokenTransfer	internal	Passed	No Issue

SUT.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_baseURI	internal	Passed	No Issue
3	mint	write	Passed	No Issue
4	pause	write	Passed	No Issue
5	unpause	write	Passed	No Issue
6	_baseURI	internal	Passed	No Issue
7	setBaseTokenURI	write	Passed	No Issue

Severity Definitions

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

Audit Findings

Critical Severity

No critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Critical operation lacks event log: **SFO_DAO.sol**

Missing event log for:

- counciAssign
- proposalRemove

Resolution: Write an event log for listed events.

Very Low / Informational / Best practices:

(1) Infinite loop: **SFO_DAO.sol**

In below functions ,for loops do not have upper length limit , which costs more gas:

- voteFinsh
- vote
- counciAssign

Resolution: Upper bound should have a certain limit in for loops.

Centralization

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

- nftTranfer: KOL_ProtectV2 owner can transfer NFT.
- tokenTranfer: KOL_ProtectV2 owner can transfer Token.
- setProtect: KOL_ProtectV2 owner can set protected status.
- setTokenAllow: SFO_DAO owner can set token allow address.
- daoRemove: SFO_DAO owner can remove dao.
- councliAssign: SFO_DAO owner can council assign address.
- proposalRemove: SFO_DAO owner can remove proposal.

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

Conclusion

We were given a contract code in the form of Github weblink. And we have used all possible tests based on given objects as files. We have not observed any major issues in the smart contracts. **So, the smart contracts are ready for the mainnet deployment.**

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

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

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

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

Our Methodology

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

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

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

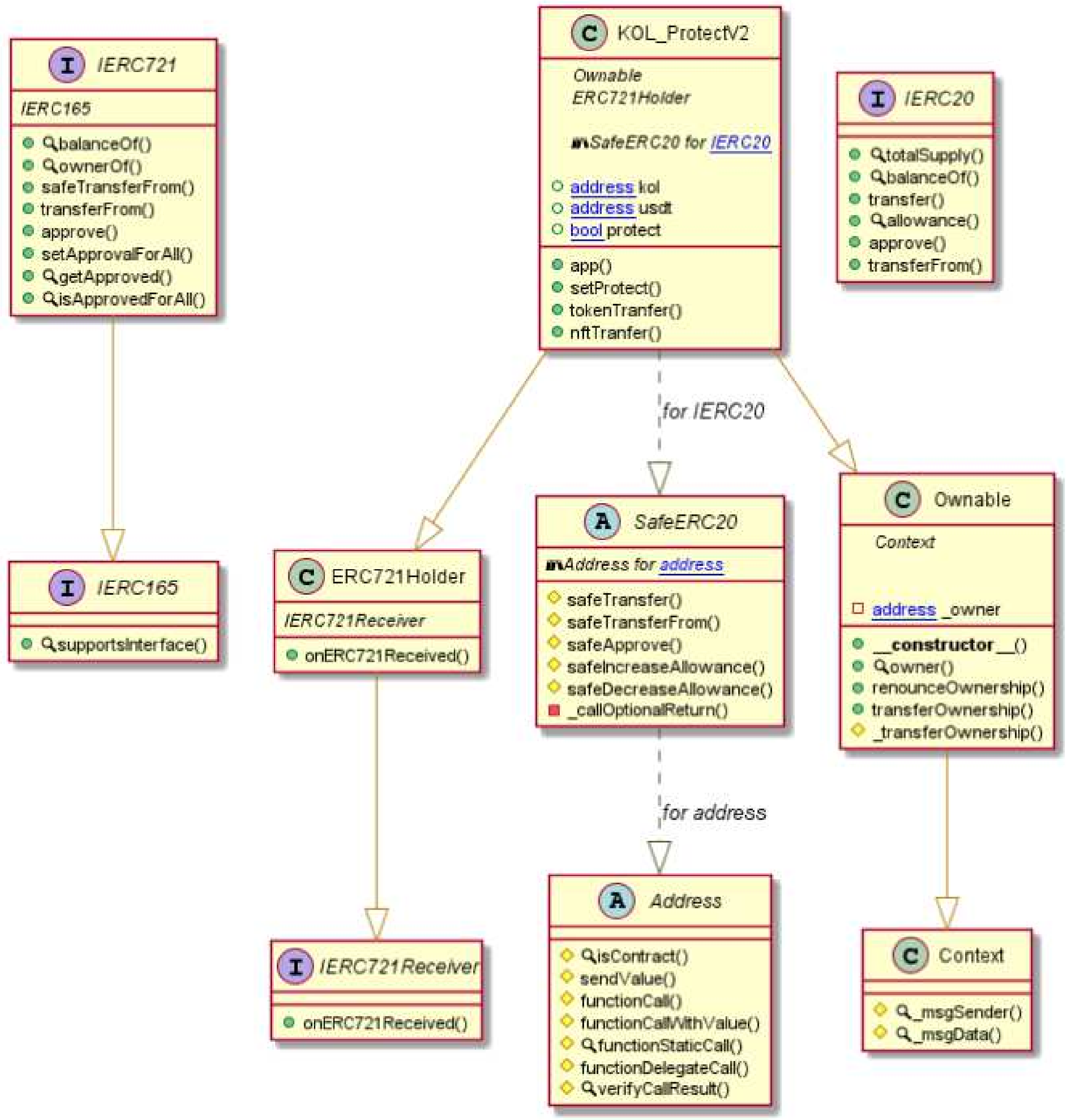
Technical Disclaimer

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

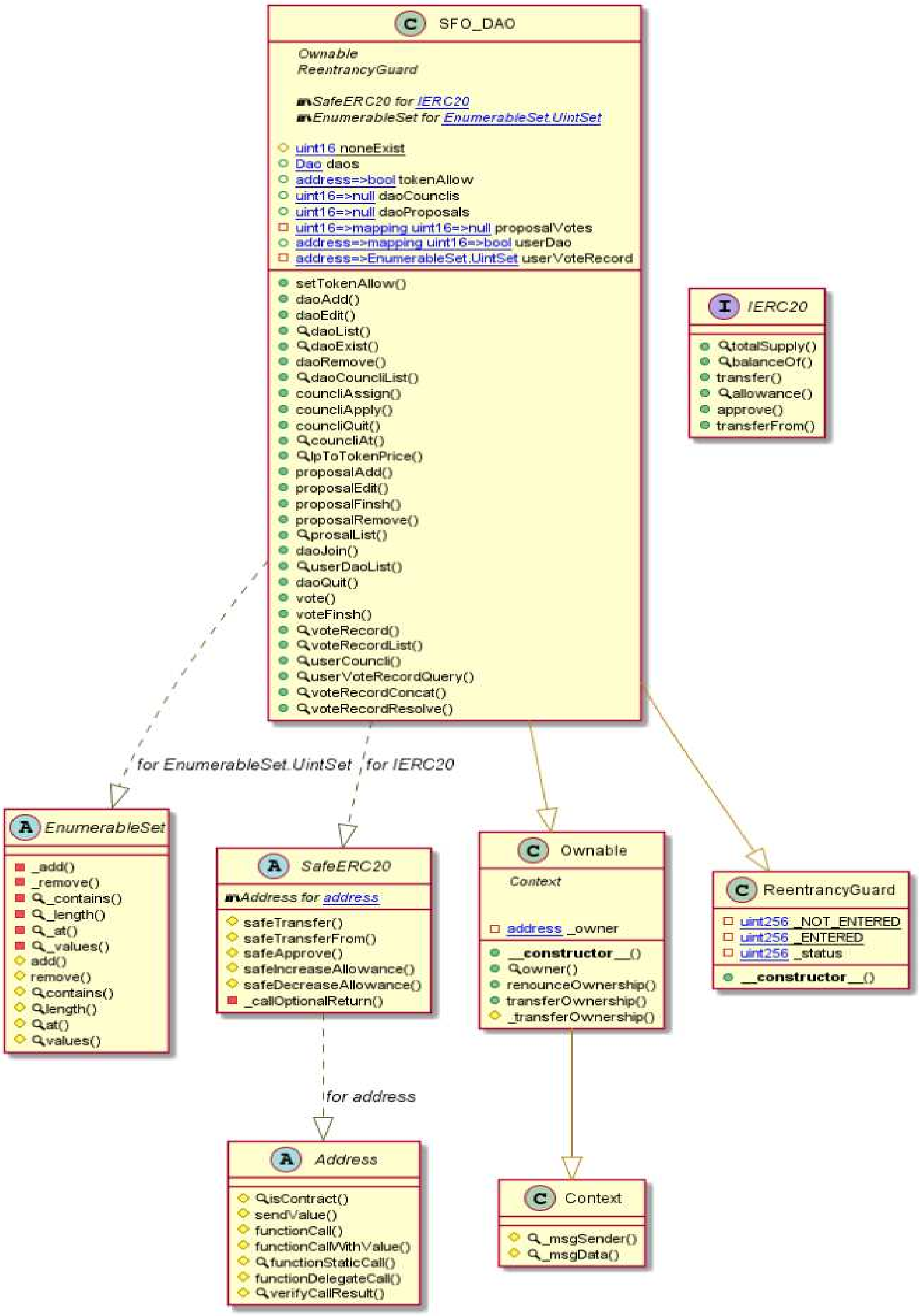
Appendix

Code Flow Diagram - Starfish OS Protocol

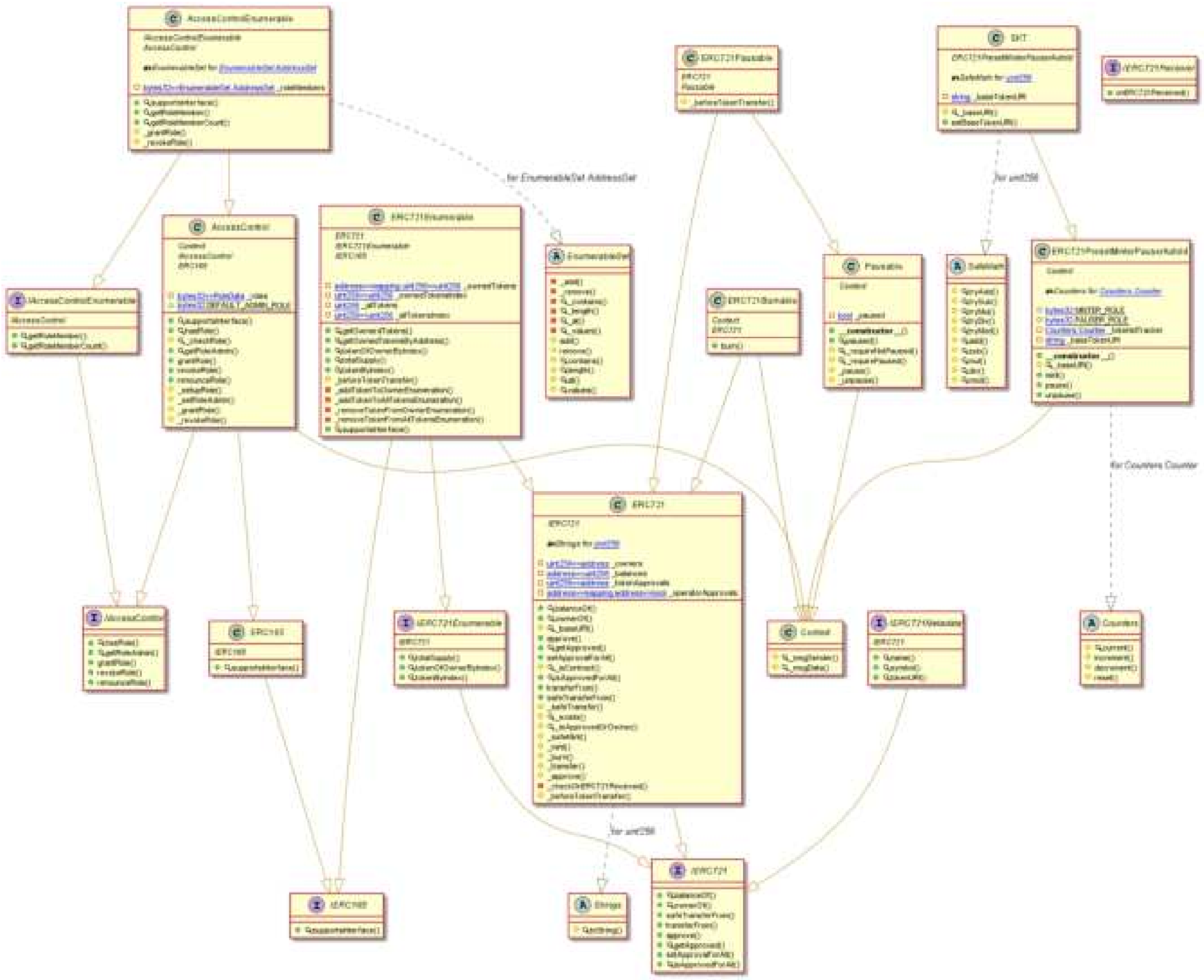
KOL_ProtectV2 Diagram



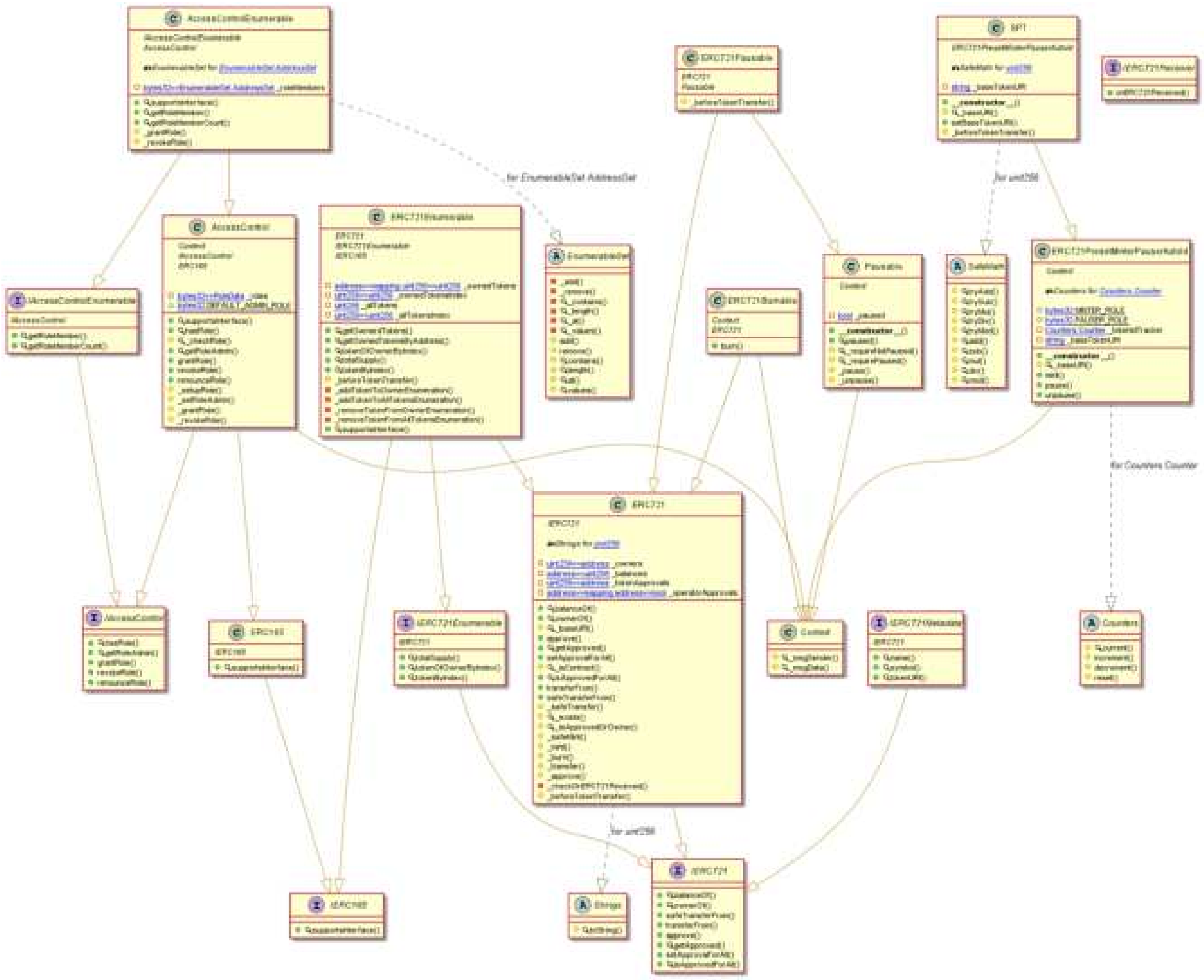
SFO_DAO Diagram



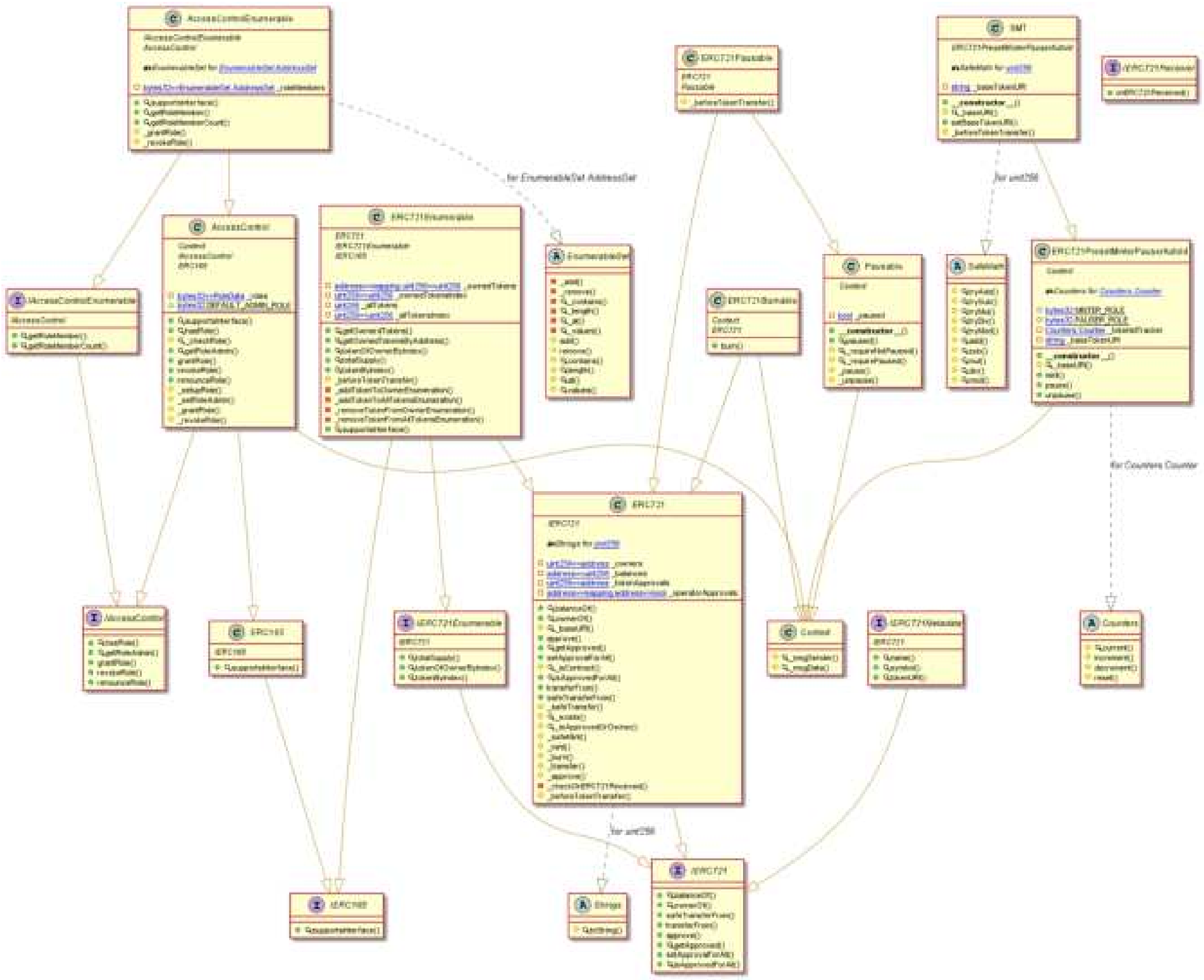
SKT Diagram


AccessControlEnumerable
ERC721Pausable
SKT
IERC721Receiver
for EnumerableSet.AddressSet
for uint256
AccessControl
ERC721Enumerable
EnumerableSet
Pausable
SafeMath
ERC721PresetMinterPauserAutoId
IAccessControlEnumerable
ERC721Burnable
for Counters.Counter
ERC721
IAccessControl
ERC165
IERC721Enumerable
Context
IERC721Metadata
Counters
for uint256
IERC165
Strings
IERC721

SPT Diagram

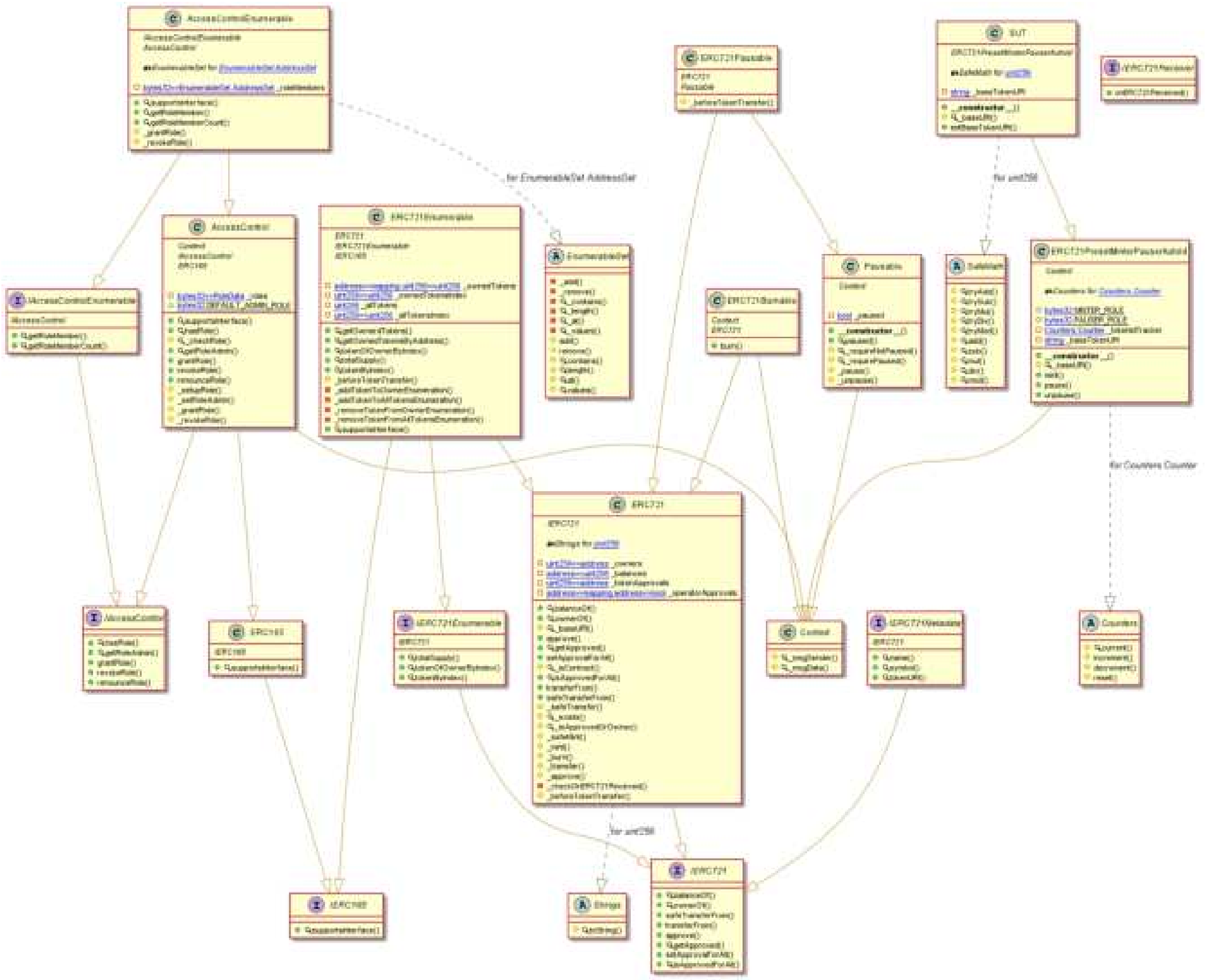

AccessControlEnumerable
ERC721Pausable
SPT
IERC721Receiver
for EnumerableSet.AddressSet
for uint256
AccessControl
ERC721Enumerable
EnumerableSet
Pausable
SafeMath
ERC721PresetMinterPauserAutoId
IAccessControlEnumerable
ERC721Burnable
for Counters.Counter
ERC721
IAccessControl
ERC165
IERC721Enumerable
Context
IERC721Metadata
Counters
for uint256
IERC165
Strings
IERC721

SMT Diagram



AccessControlEnumerable
ERC721Pausable
SMT
IERC721Receiver
for EnumerableSet.AddressSet
for uint256
AccessControl
ERC721Enumerable
EnumerableSet
Pausable
SafeMath
ERC721PresetMinterPauserAutoId
IAccessControlEnumerable
ERC721Burnable
for Counters.Counter
ERC721
IAccessControl
ERC165
IERC721Enumerable
Context
IERC721Metadata
Counters
for uint256
IERC165
Strings
IERC721

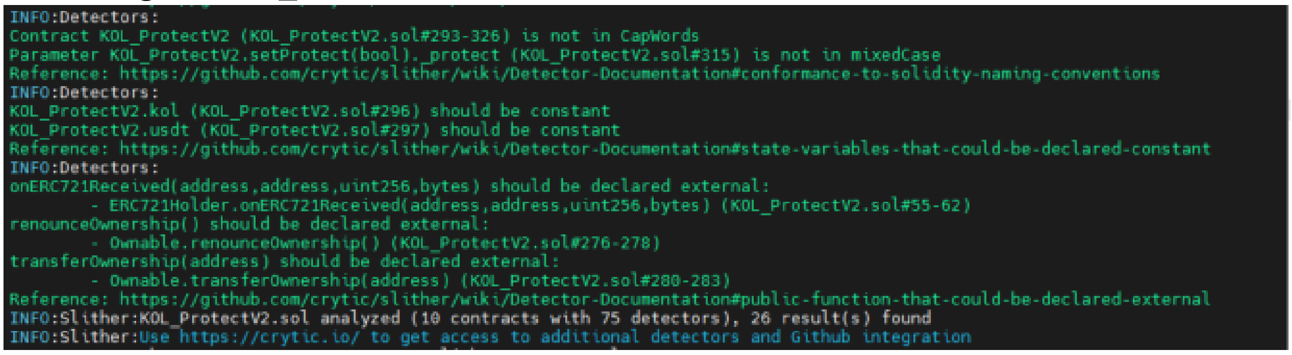
SUT Diagram



AccessControlEnumerable
ERC721Pausable
SUT
IERC721Receiver
for EnumerableSet.AddressSet
for uint256
IAccessControlEnumerable
AccessControl
ERC721Enumerable
EnumerableSet
ERC721Burnable
Pausable
SafeMath
ERC721PresetMinterPauserAutoId
for Counters.Counter
ERC721
IAccessControl
ERC165
IERC721Enumerable
Context
IERC721Metadata
Counters
for uint256
IERC165
Strings
IERC721

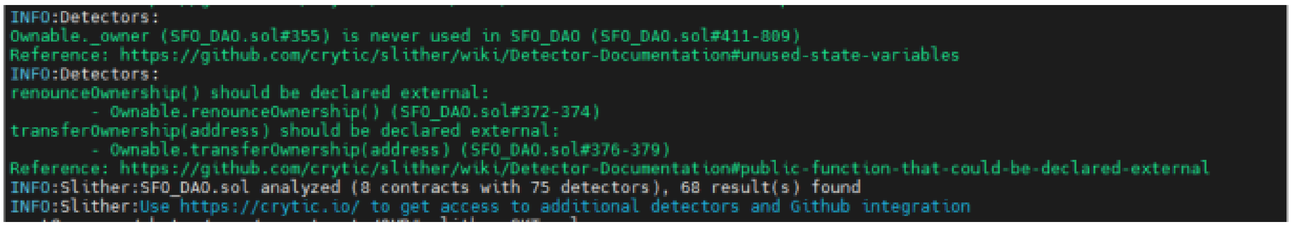
Slither Results Log

Slither log >> KOL_ProtectV2.sol



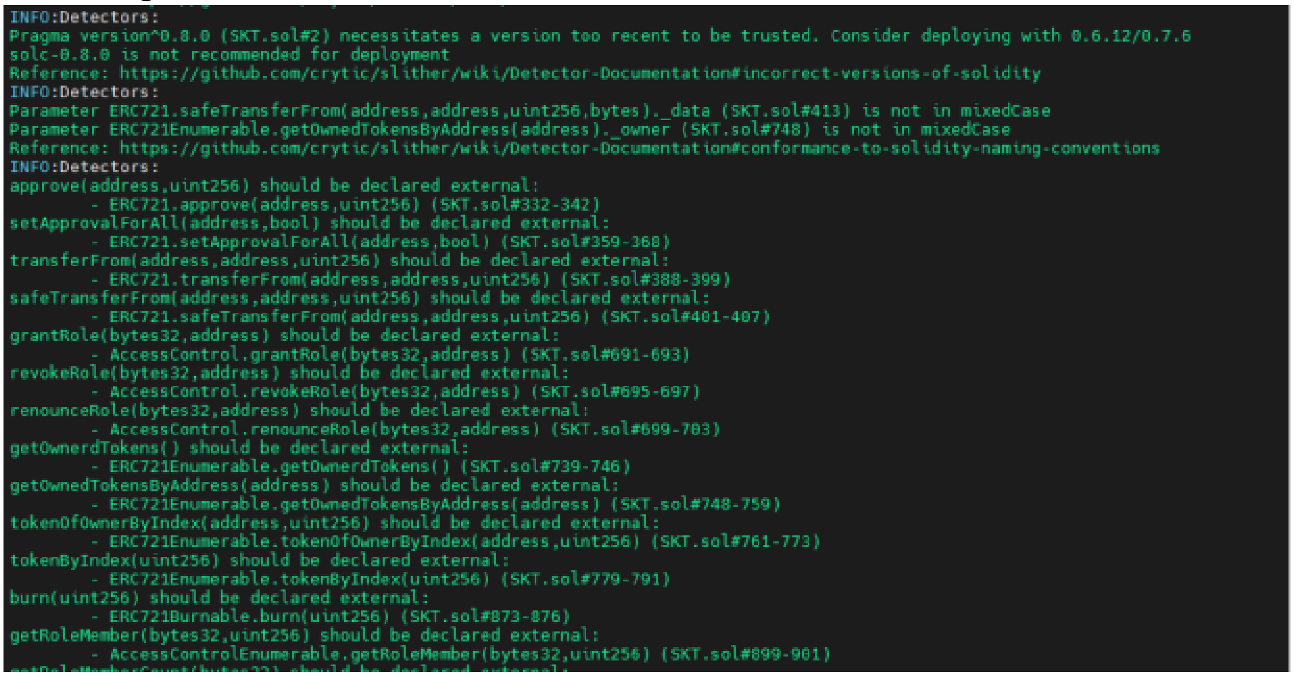
```
INFO:Detectors:
Contract KOL_ProtectV2 (KOL_ProtectV2.sol#293-326) is not in CapWords
Parameter KOL_ProtectV2.setProtect(bool)._protect (KOL_ProtectV2.sol#315) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
KOL_ProtectV2.kol (KOL_ProtectV2.sol#296) should be constant
KOL_ProtectV2.usdt (KOL_ProtectV2.sol#297) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
onERC721Received(address,address,uint256,bytes) should be declared external:
        - ERC721Holder.onERC721Received(address,address,uint256,bytes) (KOL_ProtectV2.sol#55-62)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (KOL_ProtectV2.sol#276-278)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (KOL_ProtectV2.sol#280-283)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:KOL_ProtectV2.sol analyzed (10 contracts with 75 detectors), 26 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> SFO_DAO.sol

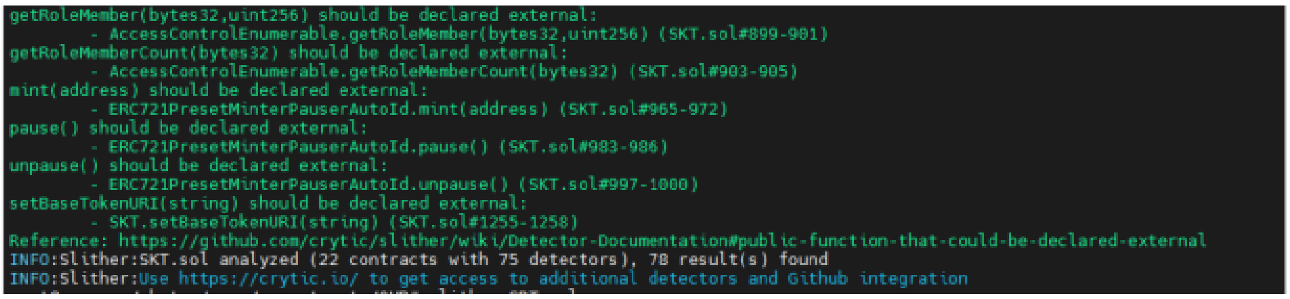


```
INFO:Detectors:
Ownable._owner (SFO_DAO.sol#355) is never used in SFO_DAO (SFO_DAO.sol#411-809)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SFO_DAO.sol#372-374)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (SFO_DAO.sol#376-379)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SFO_DAO.sol analyzed (8 contracts with 75 detectors), 68 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> SKT.sol

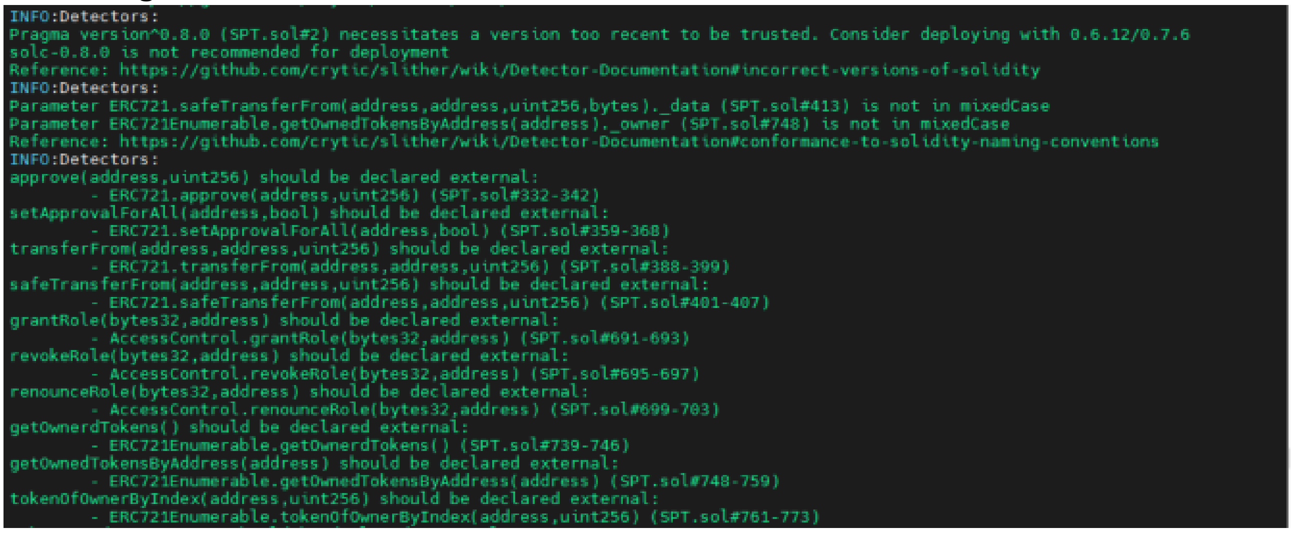


```
INFO:Detectors:
Pragma version^0.8.0 (SKT.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (SKT.sol#413) is not in mixedCase
Parameter ERC721Enumerable.getOwnedTokensByAddress(address)._owner (SKT.sol#748) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (SKT.sol#332-342)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (SKT.sol#359-368)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (SKT.sol#388-399)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (SKT.sol#401-407)
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (SKT.sol#691-693)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (SKT.sol#695-697)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (SKT.sol#699-703)
getOwnerdTokens() should be declared external:
        - ERC721Enumerable.getOwnerdTokens() (SKT.sol#739-746)
getOwnedTokensByAddress(address) should be declared external:
        - ERC721Enumerable.getOwnedTokensByAddress(address) (SKT.sol#748-759)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (SKT.sol#761-773)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (SKT.sol#779-791)
burn(uint256) should be declared external:
        - ERC721Burnable.burn(uint256) (SKT.sol#873-876)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerable.getRoleMember(bytes32,uint256) (SKT.sol#899-901)
```

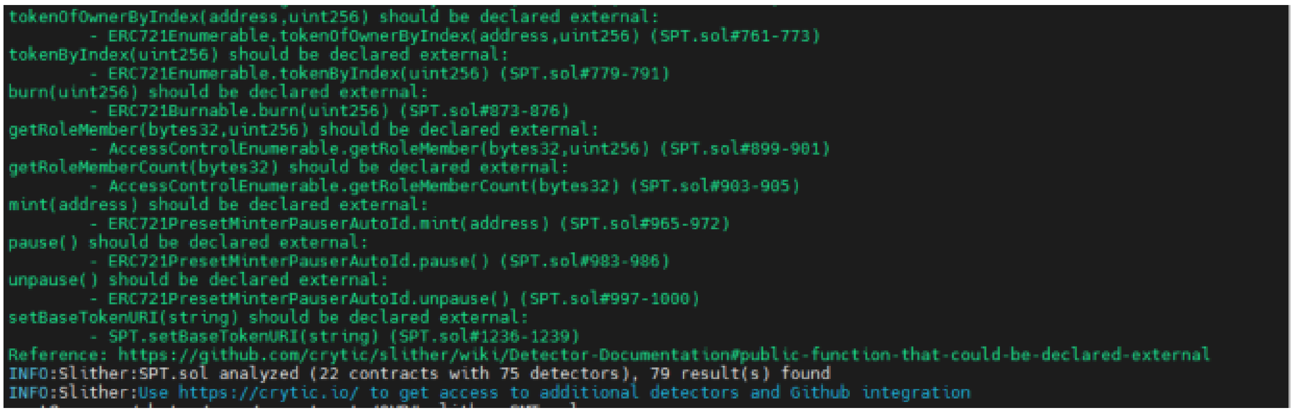


```
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerable.getRoleMember(bytes32,uint256) (SKT.sol#899-901)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControlEnumerable.getRoleMemberCount(bytes32) (SKT.sol#903-905)
mint(address) should be declared external:
        - ERC721PresetMinterPauserAutoId.mint(address) (SKT.sol#965-972)
pause() should be declared external:
        - ERC721PresetMinterPauserAutoId.pause() (SKT.sol#983-986)
unpause() should be declared external:
        - ERC721PresetMinterPauserAutoId.unpause() (SKT.sol#997-1000)
setBaseTokenURI(string) should be declared external:
        - SKT.setBaseTokenURI(string) (SKT.sol#1255-1258)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SKT.sol analyzed (22 contracts with 75 detectors), 78 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> SPT.sol

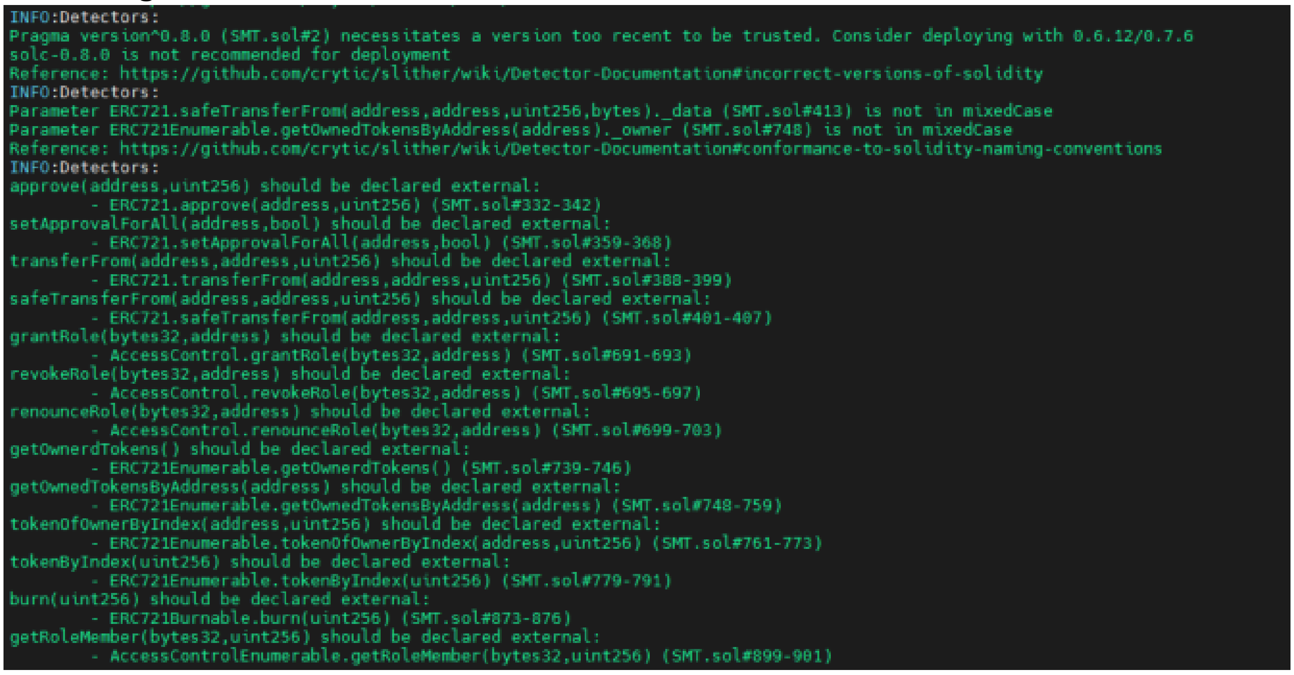


```
INFO:Detectors:
Pragma version^0.8.0 (SPT.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (SPT.sol#413) is not in mixedCase
Parameter ERC721Enumerable.getOwnedTokensByAddress(address)._owner (SPT.sol#748) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (SPT.sol#332-342)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (SPT.sol#359-368)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (SPT.sol#388-399)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (SPT.sol#401-407)
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (SPT.sol#691-693)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (SPT.sol#695-697)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (SPT.sol#699-703)
getOwnerdTokens() should be declared external:
        - ERC721Enumerable.getOwnerdTokens() (SPT.sol#739-746)
getOwnedTokensByAddress(address) should be declared external:
        - ERC721Enumerable.getOwnedTokensByAddress(address) (SPT.sol#748-759)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (SPT.sol#761-773)
```

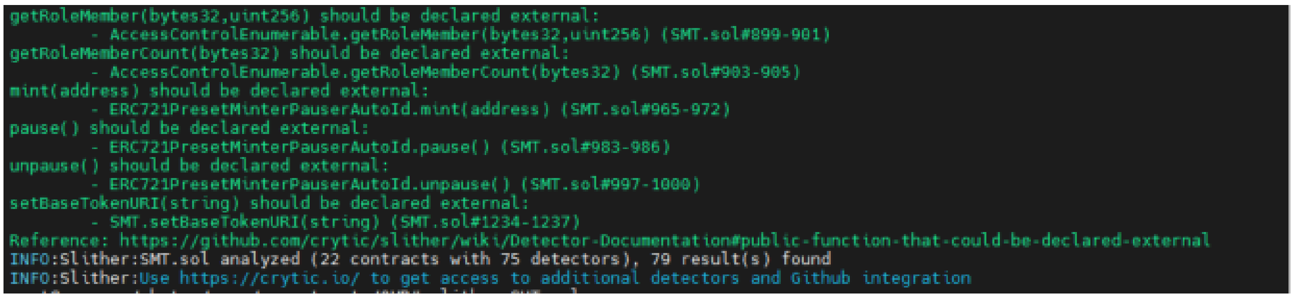


```
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (SPT.sol#761-773)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (SPT.sol#779-791)
burn(uint256) should be declared external:
        - ERC721Burnable.burn(uint256) (SPT.sol#873-876)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerable.getRoleMember(bytes32,uint256) (SPT.sol#899-901)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControlEnumerable.getRoleMemberCount(bytes32) (SPT.sol#903-905)
mint(address) should be declared external:
        - ERC721PresetMinterPauserAutoId.mint(address) (SPT.sol#965-972)
pause() should be declared external:
        - ERC721PresetMinterPauserAutoId.pause() (SPT.sol#983-986)
unpause() should be declared external:
        - ERC721PresetMinterPauserAutoId.unpause() (SPT.sol#997-1000)
setBaseTokenURI(string) should be declared external:
        - SPT.setBaseTokenURI(string) (SPT.sol#1236-1239)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SPT.sol analyzed (22 contracts with 75 detectors), 79 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> SMT.sol

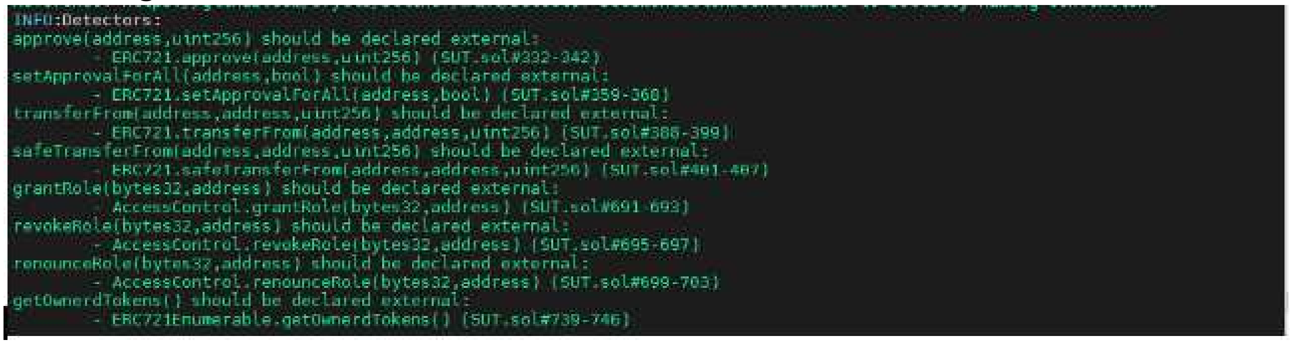


```
INFO:Detectors:
Pragma version^0.8.0 (SMT.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (SMT.sol#413) is not in mixedCase
Parameter ERC721Enumerable.getOwnedTokensByAddress(address)._owner (SMT.sol#748) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (SMT.sol#332-342)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (SMT.sol#359-368)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (SMT.sol#388-399)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (SMT.sol#401-407)
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (SMT.sol#691-693)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (SMT.sol#695-697)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (SMT.sol#699-703)
getOwnerdTokens() should be declared external:
        - ERC721Enumerable.getOwnerdTokens() (SMT.sol#739-746)
getOwnedTokensByAddress(address) should be declared external:
        - ERC721Enumerable.getOwnedTokensByAddress(address) (SMT.sol#748-759)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (SMT.sol#761-773)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (SMT.sol#779-791)
burn(uint256) should be declared external:
        - ERC721Burnable.burn(uint256) (SMT.sol#873-876)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerable.getRoleMember(bytes32,uint256) (SMT.sol#899-901)
```

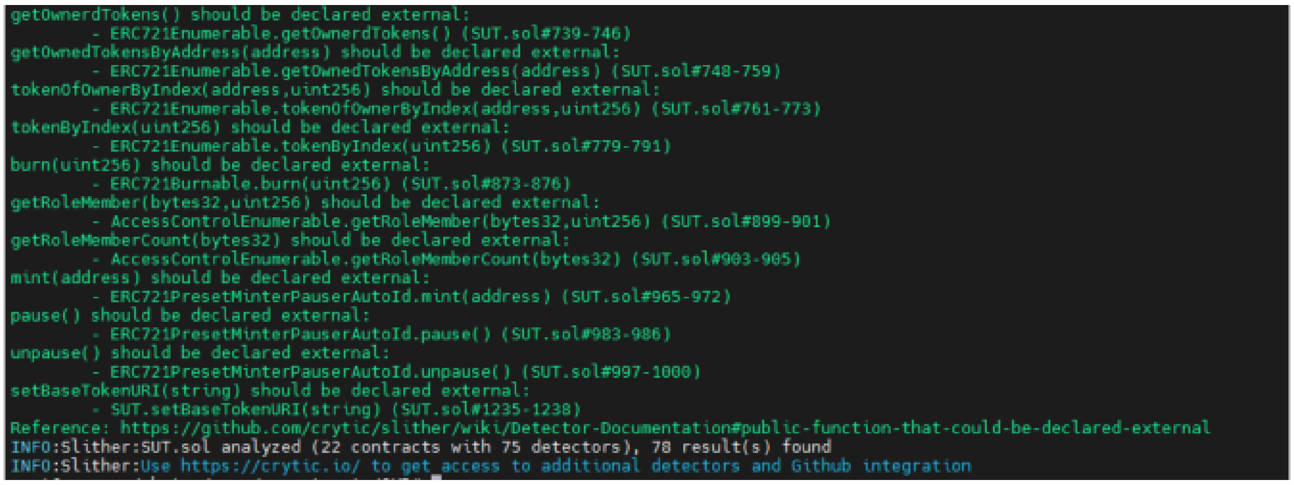


```
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerable.getRoleMember(bytes32,uint256) (SMT.sol#899-901)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControlEnumerable.getRoleMemberCount(bytes32) (SMT.sol#903-905)
mint(address) should be declared external:
        - ERC721PresetMinterPauserAutoId.mint(address) (SMT.sol#965-972)
pause() should be declared external:
        - ERC721PresetMinterPauserAutoId.pause() (SMT.sol#983-986)
unpause() should be declared external:
        - ERC721PresetMinterPauserAutoId.unpause() (SMT.sol#997-1000)
setBaseTokenURI(string) should be declared external:
        - SMT.setBaseTokenURI(string) (SMT.sol#1234-1237)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SMT.sol analyzed (22 contracts with 75 detectors), 79 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> SUT.sol



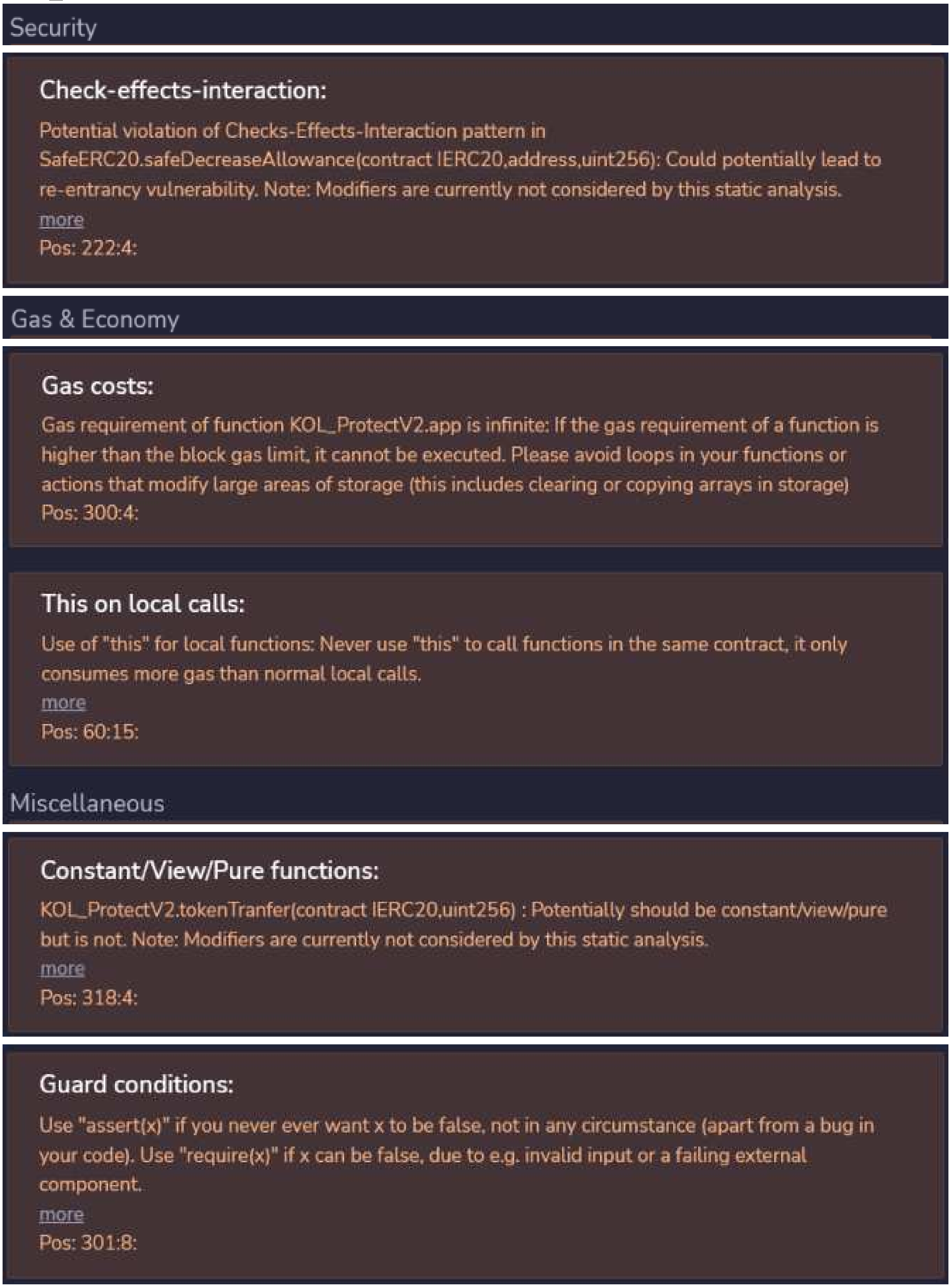
```
INFO:Detectors:
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (SUT.sol#332-342)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (SUT.sol#359-360)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (SUT.sol#386-399)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (SUT.sol#401-407)
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (SUT.sol#691-693)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (SUT.sol#695-697)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (SUT.sol#699-703)
getOwnerdTokens() should be declared external:
        - ERC721Enumerable.getOwnerdTokens() (SUT.sol#739-746)
```



```
getOwnerdTokens() should be declared external:
        - ERC721Enumerable.getOwnerdTokens() (SUT.sol#739-746)
getOwnedTokensByAddress(address) should be declared external:
        - ERC721Enumerable.getOwnedTokensByAddress(address) (SUT.sol#748-759)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (SUT.sol#761-773)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (SUT.sol#779-791)
burn(uint256) should be declared external:
        - ERC721Burnable.burn(uint256) (SUT.sol#873-876)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerable.getRoleMember(bytes32,uint256) (SUT.sol#899-901)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControlEnumerable.getRoleMemberCount(bytes32) (SUT.sol#903-905)
mint(address) should be declared external:
        - ERC721PresetMinterPauserAutoId.mint(address) (SUT.sol#965-972)
pause() should be declared external:
        - ERC721PresetMinterPauserAutoId.pause() (SUT.sol#983-986)
unpause() should be declared external:
        - ERC721PresetMinterPauserAutoId.unpause() (SUT.sol#997-1000)
setBaseTokenURI(string) should be declared external:
        - SUT.setBaseTokenURI(string) (SUT.sol#1235-1238)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SUT.sol analyzed (22 contracts with 75 detectors), 78 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

KOL_ProtectV2.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeDecreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 222:4:

Gas & Economy

Gas costs:

Gas requirement of function KOL_ProtectV2.app 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: 300:4:

This on local calls:

Use of "this" for local functions: Never use "this" to call functions in the same contract, it only consumes more gas than normal local calls.
more
Pos: 60:15:

Miscellaneous

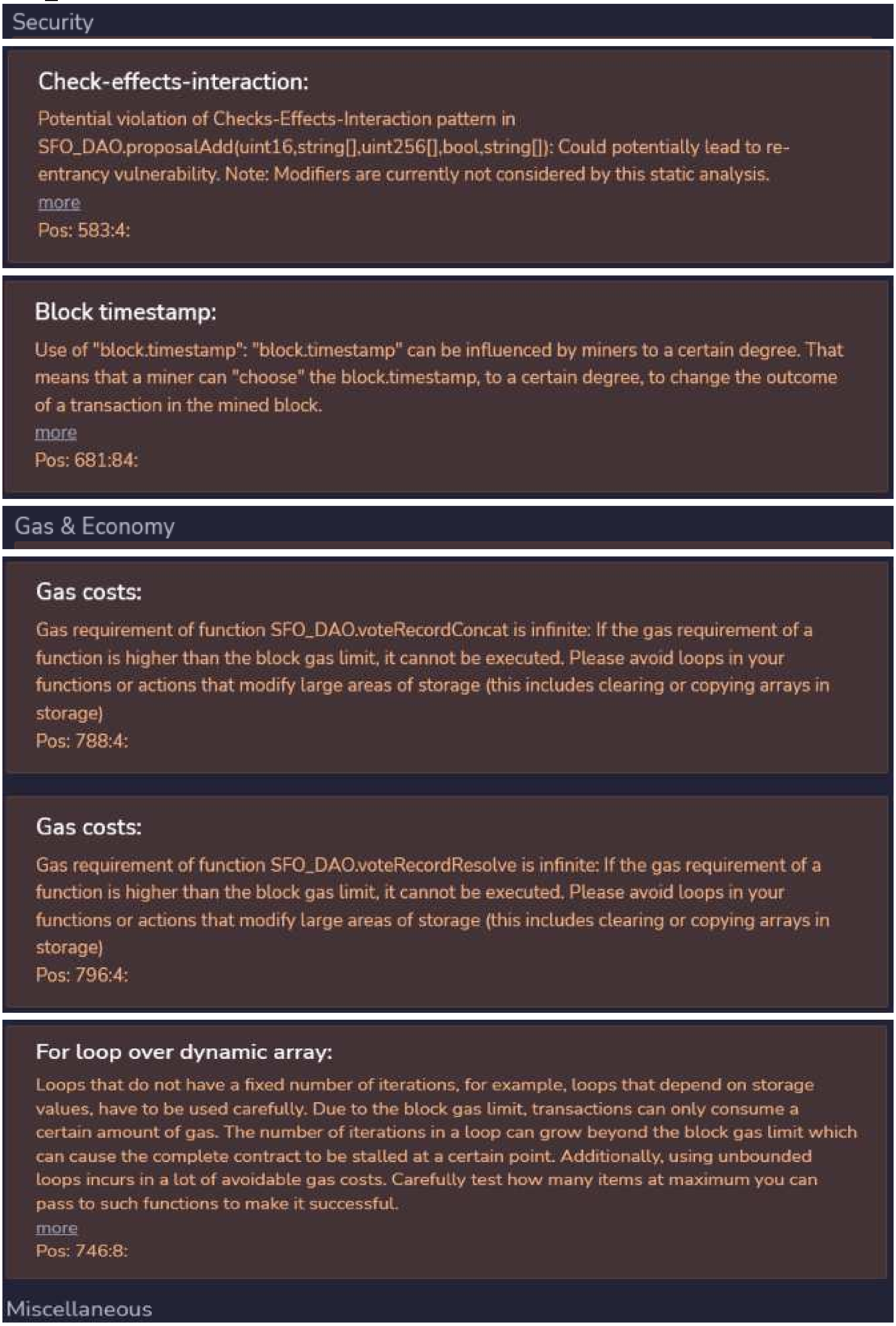
Constant/View/Pure functions:

KOL_ProtectV2.tokenTranfer(contract IERC20,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 318:4:

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: 301:8:

SFO_DAO.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SFO_DAO.proposalAdd(uint16,string[],uint256[],bool,string[]): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 583:4:

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: 681:84:

Gas & Economy

Gas costs:

Gas requirement of function SFO_DAO.voteRecordConcat 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: 788:4:

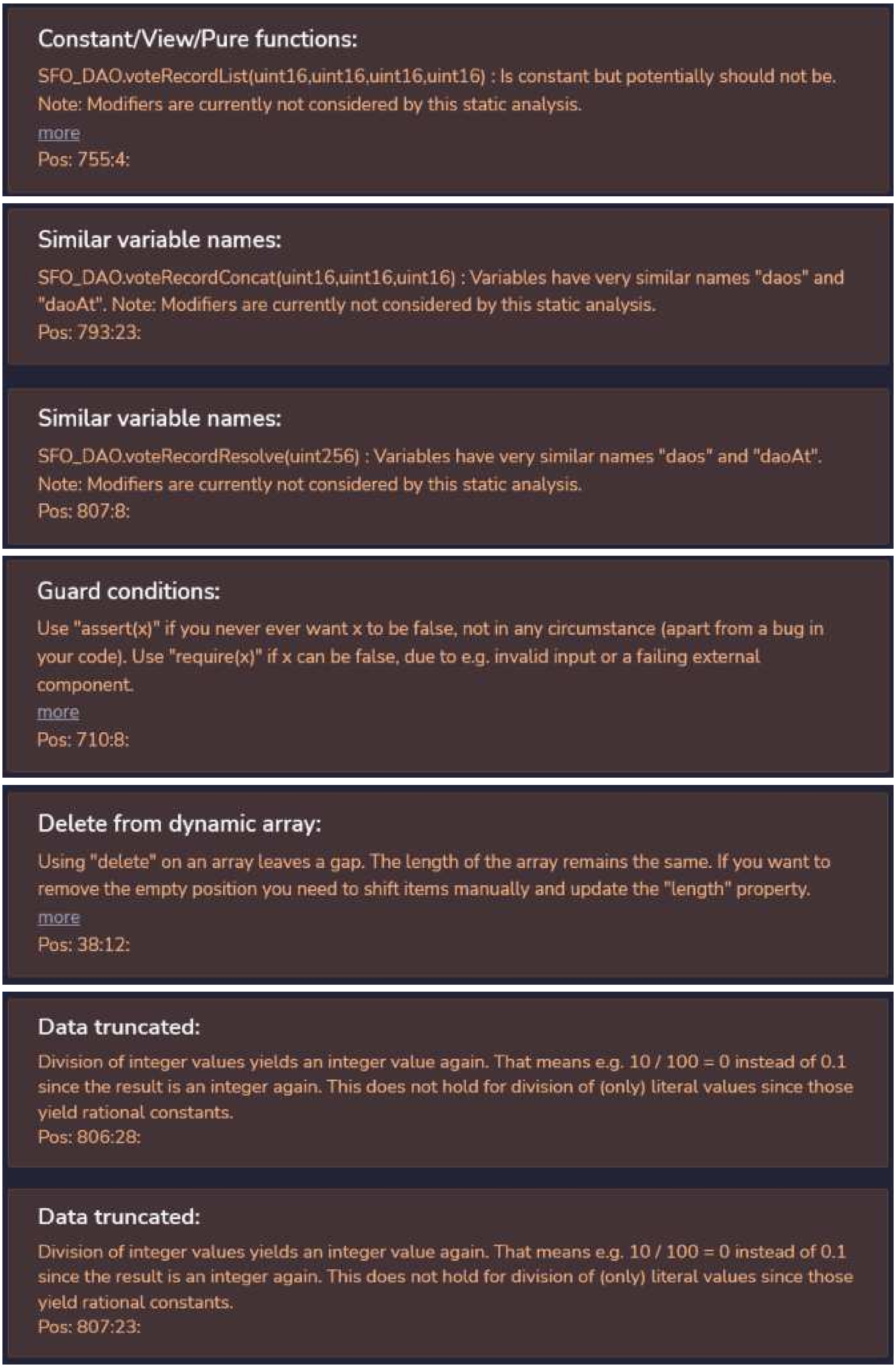
Gas costs:

Gas requirement of function SFO_DAO.voteRecordResolve 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: 796:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 746:8:

Miscellaneous



Constant/View/Pure functions:

SFO_DAO.voteRecordList(uint16,uint16,uint16,uint16) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 755:4:

Similar variable names:

SFO_DAO.voteRecordConcat(uint16,uint16,uint16) : Variables have very similar names "daos" and "daoAt". Note: Modifiers are currently not considered by this static analysis.
Pos: 793:23:

Similar variable names:

SFO_DAO.voteRecordResolve(uint256) : Variables have very similar names "daos" and "daoAt". Note: Modifiers are currently not considered by this static analysis.
Pos: 807:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 710:8:

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: 38:12:

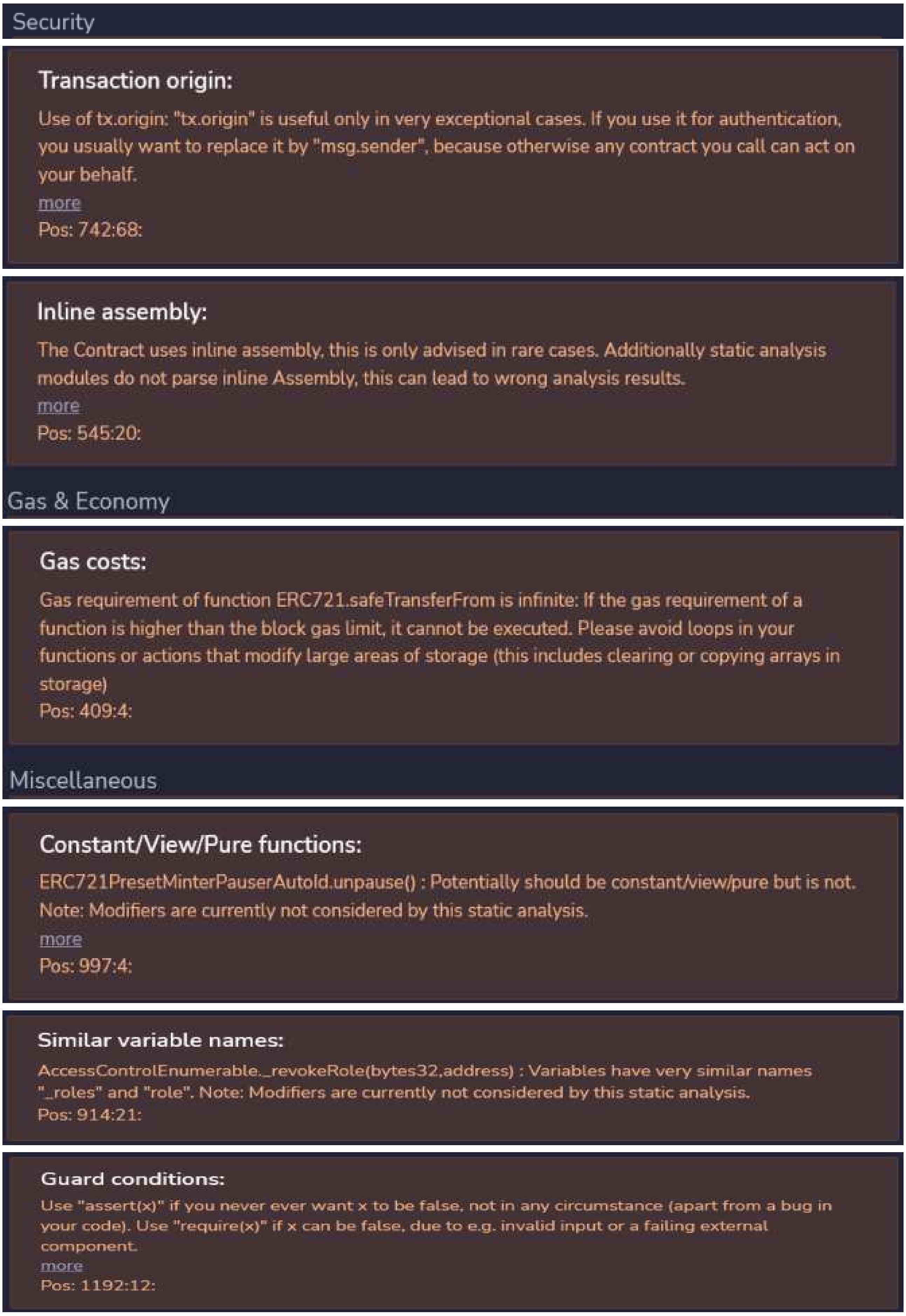
Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 806:28:

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: 807:23:

SKT.sol



Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 742:68:

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: 545:20:

Gas & Economy

Gas costs:

Gas requirement of function ERC721.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: 409:4:

Miscellaneous

Constant/View/Pure functions:

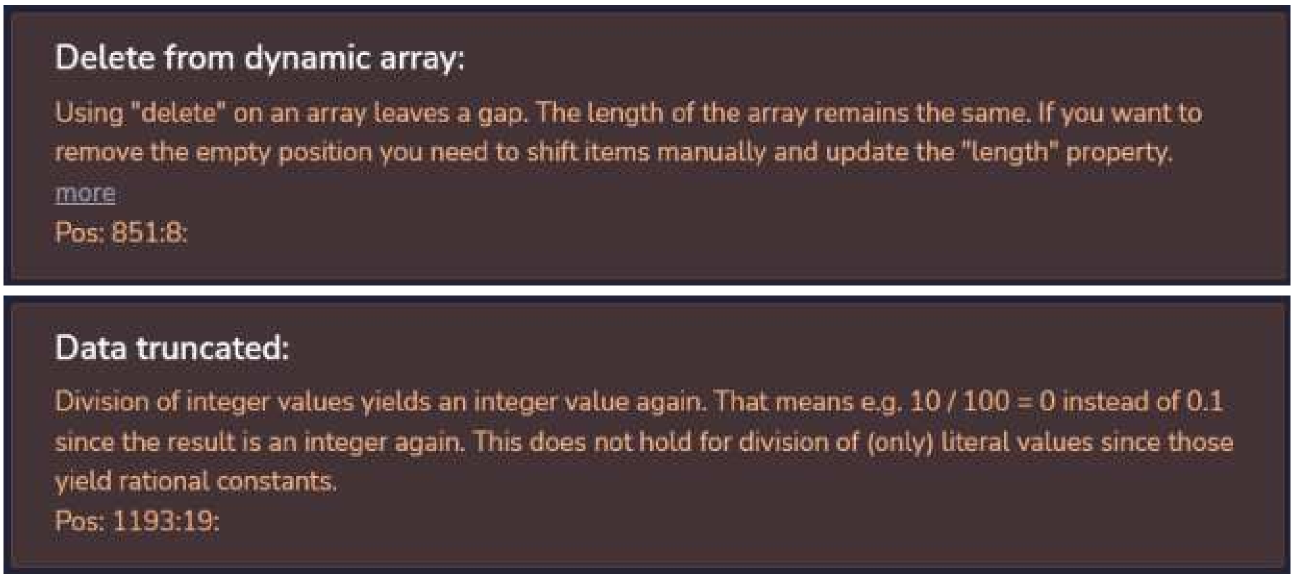
ERC721PresetMinterPauserAutoId.unpause() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 997:4:

Similar variable names:

AccessControlEnumerable._revokeRole(bytes32,address) : Variables have very similar names "_roles" and "role". Note: Modifiers are currently not considered by this static analysis.
Pos: 914:21:

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: 1192:12:



Delete from dynamic array:

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

more

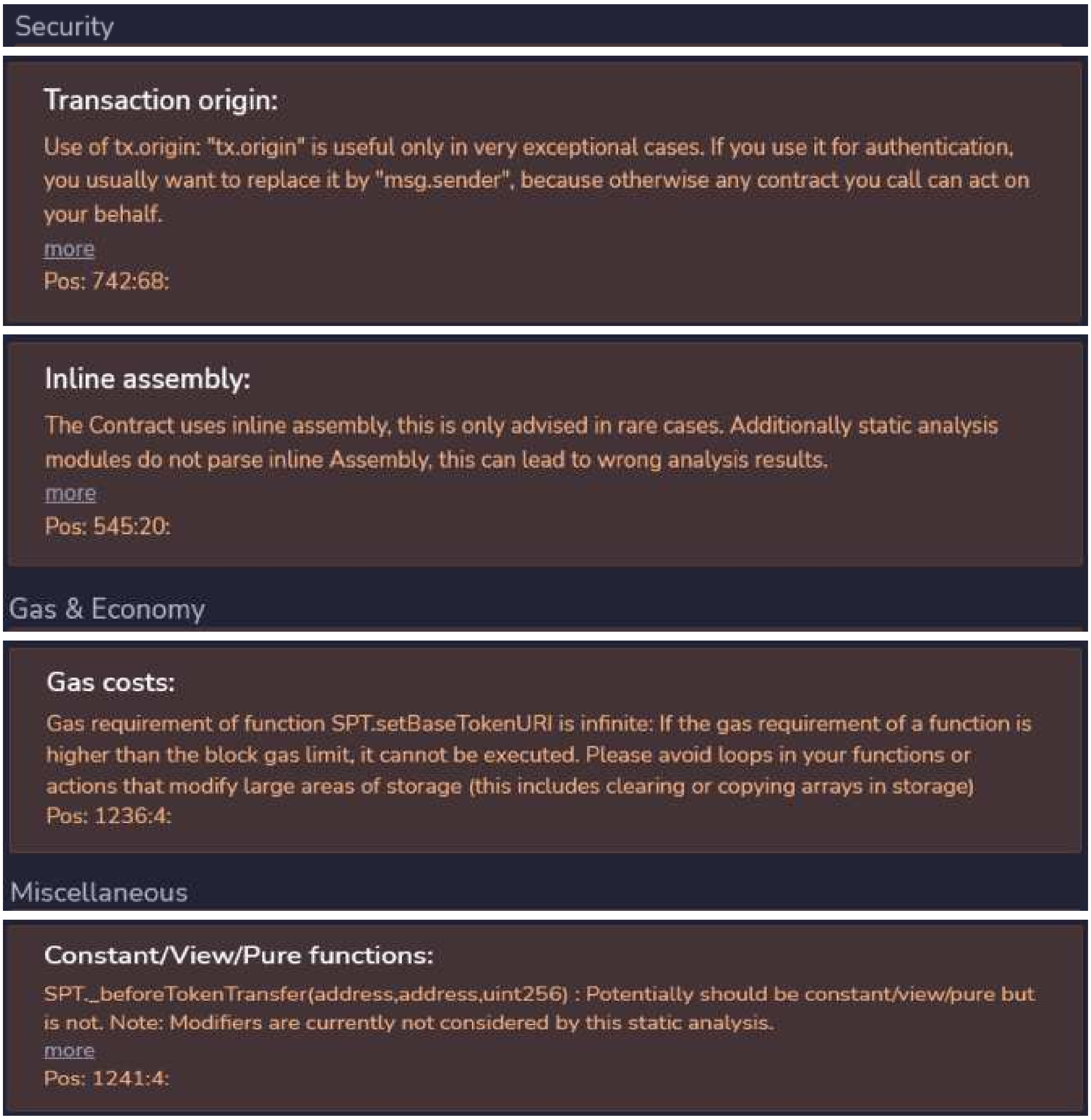
Pos: 851:8:

Data truncated:

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

Pos: 1193:19:

SPT.sol



Security

Transaction origin:

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

more

Pos: 742:68:

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: 545:20:

Gas & Economy

Gas costs:

Gas requirement of function SPT.setBaseTokenURI 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: 1236:4:

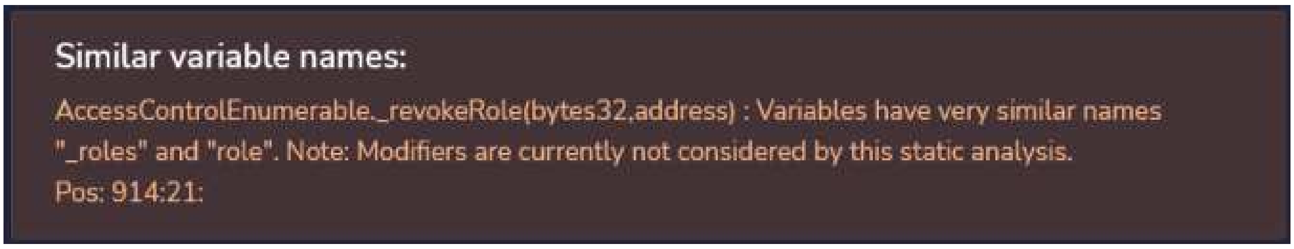
Miscellaneous

Constant/View/Pure functions:

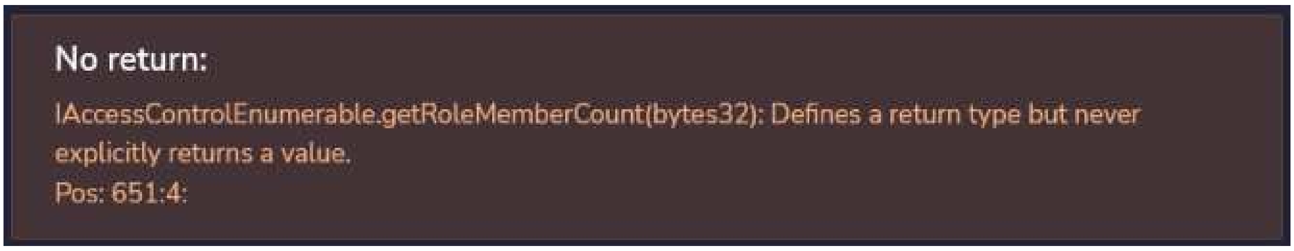
SPT._beforeTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

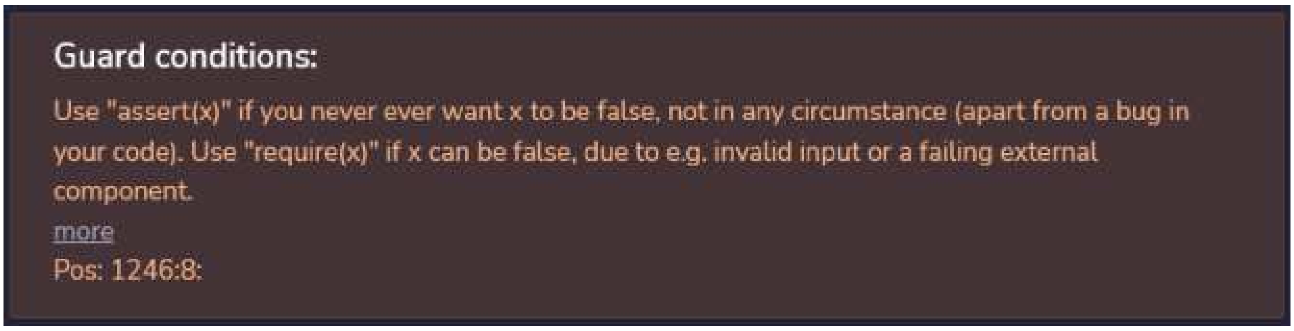
Pos: 1241:4:


Similar variable names:

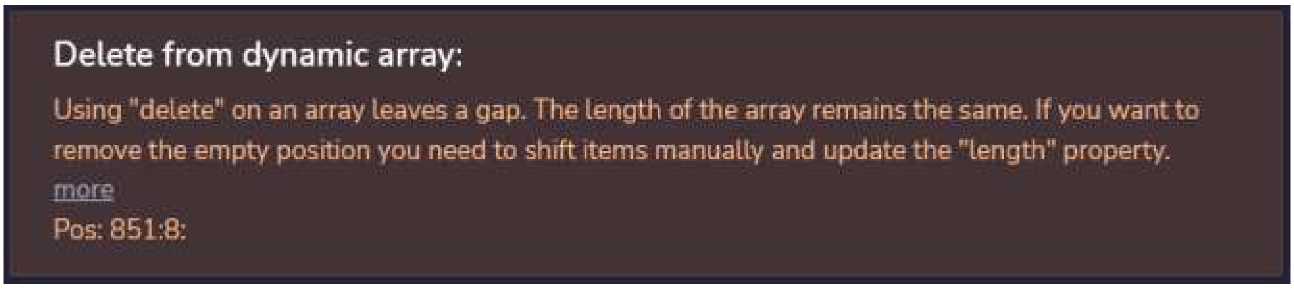
AccessControlEnumerable._revokeRole(bytes32,address) : Variables have very similar names "_roles" and "role". Note: Modifiers are currently not considered by this static analysis.
Pos: 914:21:


No return:

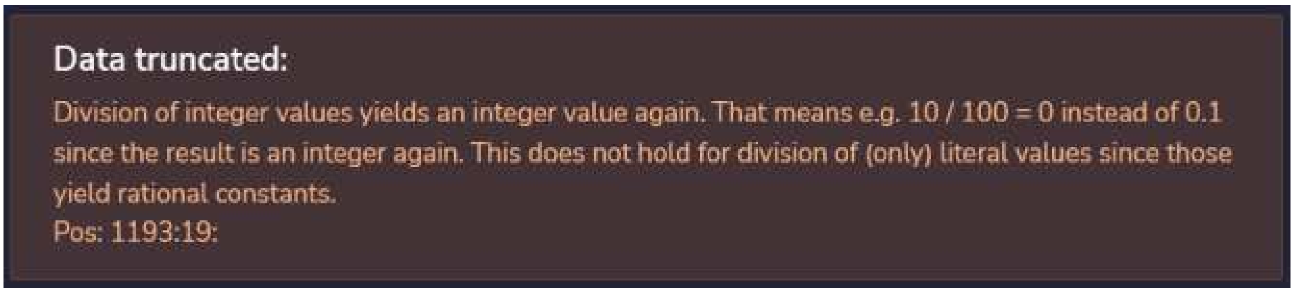
IAccessControlEnumerable.getRoleMemberCount(bytes32): Defines a return type but never explicitly returns a value.
Pos: 651:4:


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: 1246:8:

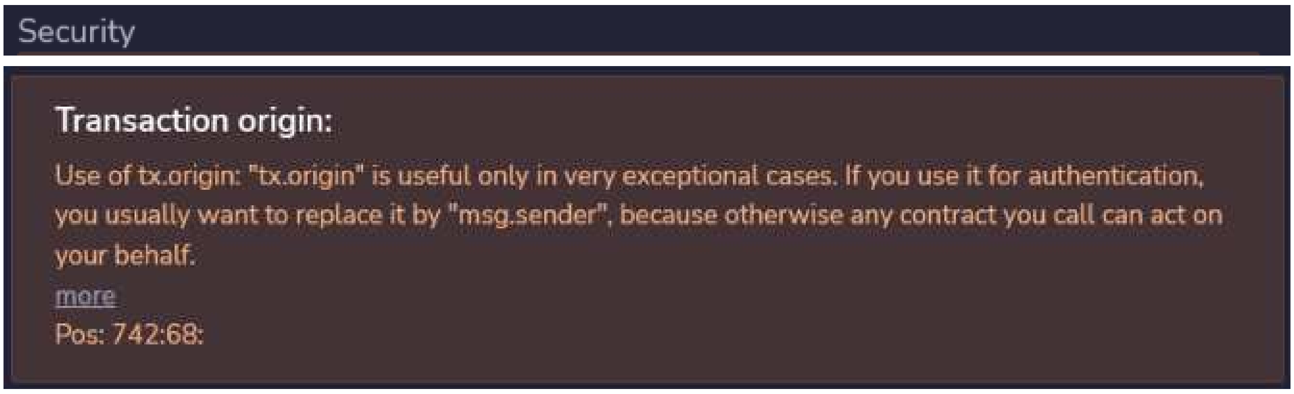

Delete from dynamic array:

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: 851:8:


Data truncated:

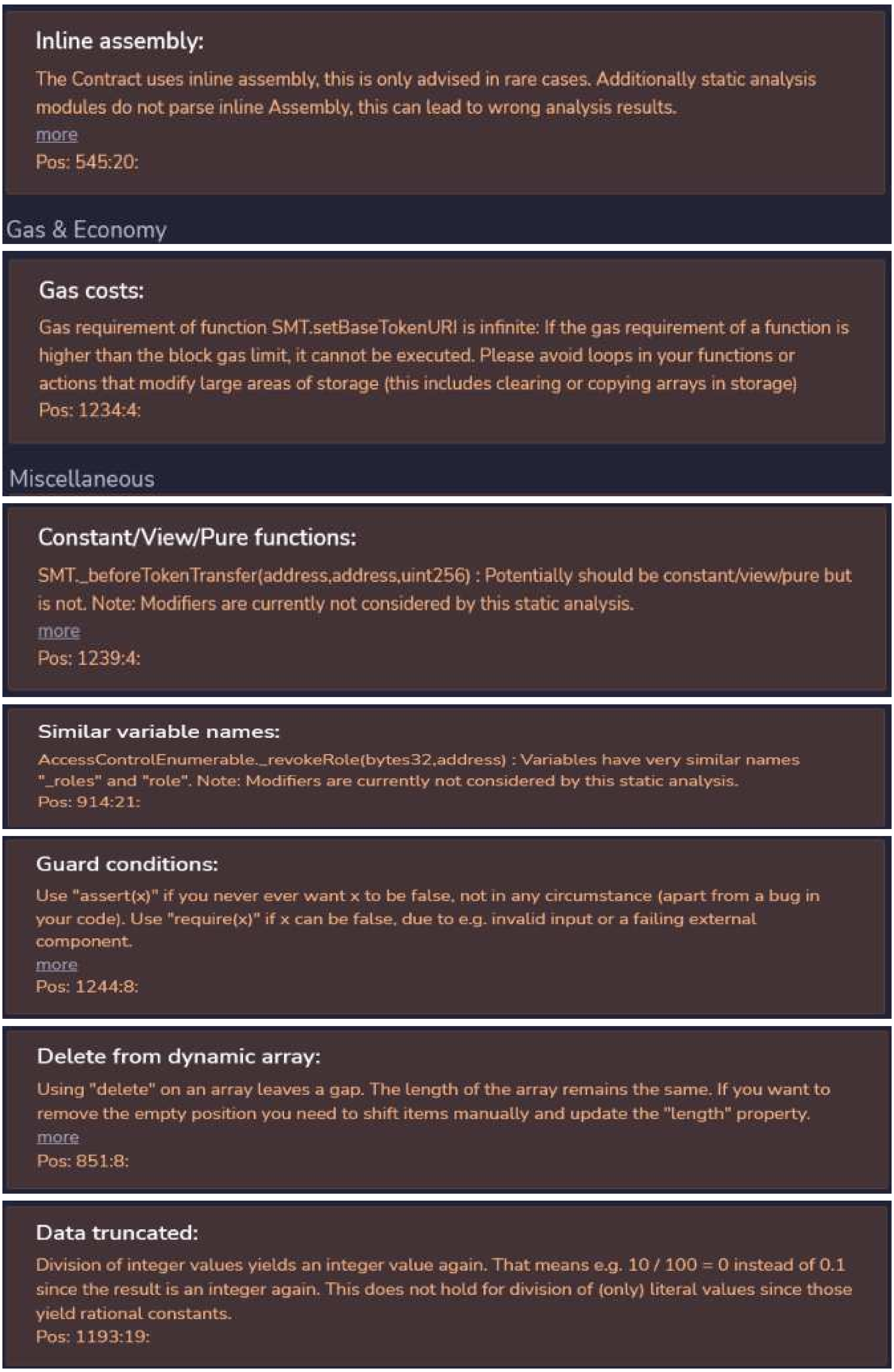
Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1193:19:

SMT.sol


Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 742:68:



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: 545:20:

Gas & Economy

Gas costs:

Gas requirement of function SMT.setBaseTokenURI 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: 1234:4:

Miscellaneous

Constant/View/Pure functions:

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

more

Pos: 1239:4:

Similar variable names:

AccessControlEnumerable._revokeRole(bytes32,address) : Variables have very similar names "_roles" and "role". Note: Modifiers are currently not considered by this static analysis.

Pos: 914:21:

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: 1244:8:

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

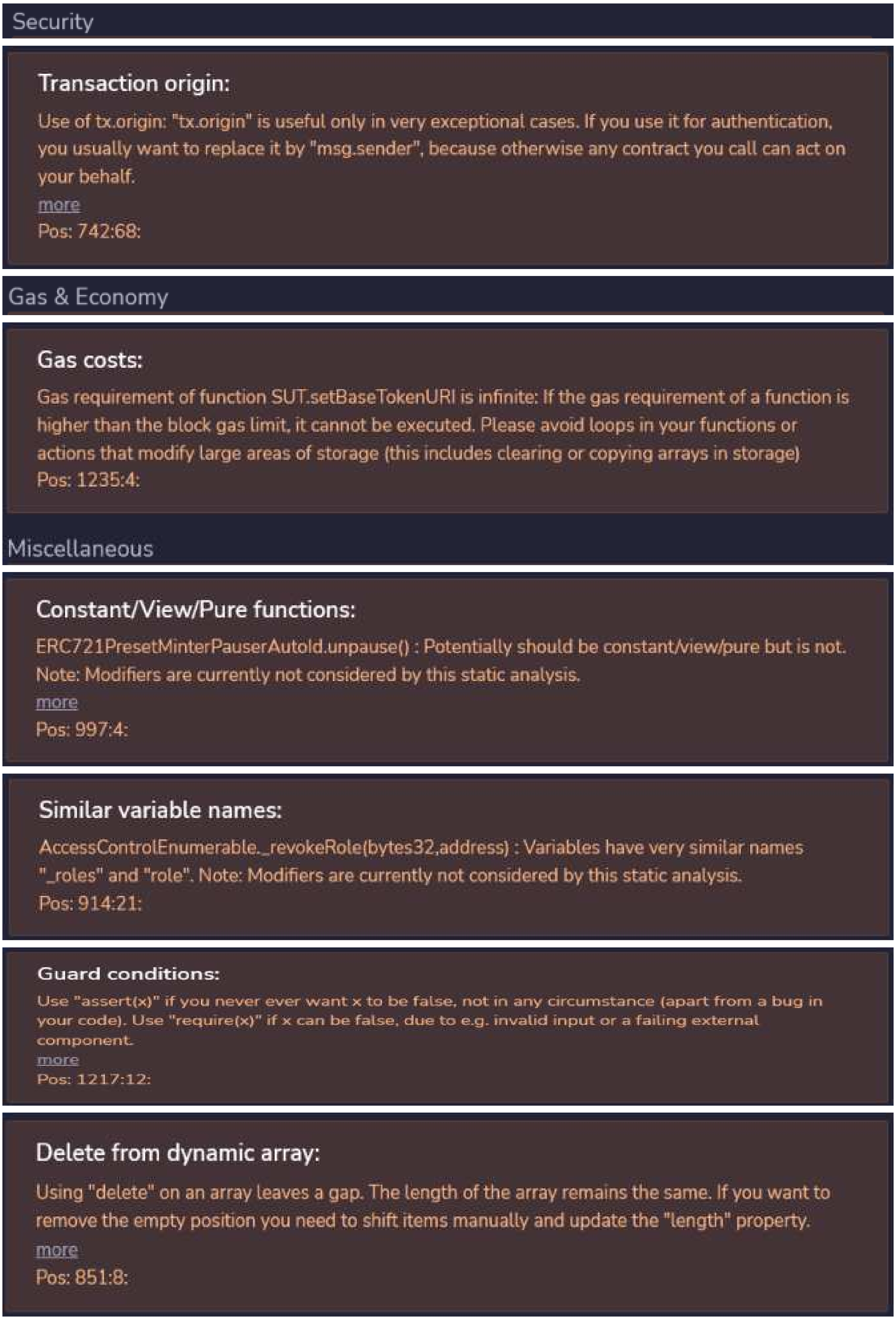
Pos: 851:8:

Data truncated:

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

Pos: 1193:19:

SUT.sol



Security

Transaction origin:

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

more

Pos: 742:68:

Gas & Economy

Gas costs:

Gas requirement of function SUT.setBaseTokenURI 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: 1235:4:

Miscellaneous

Constant/View/Pure functions:

ERC721PresetMinterPauserAutoId.unpause() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 997:4:

Similar variable names:

AccessControlEnumerable._revokeRole(bytes32,address) : Variables have very similar names "_roles" and "role". Note: Modifiers are currently not considered by this static analysis.

Pos: 914:21:

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: 1217:12:

Delete from dynamic array:

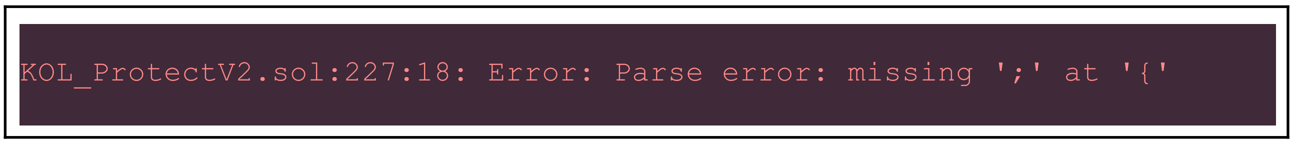
Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.

more

Pos: 851:8:

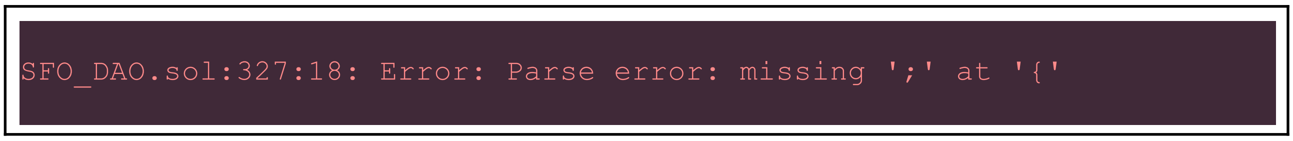
Solhint Linter

KOL_ProtectV2.sol



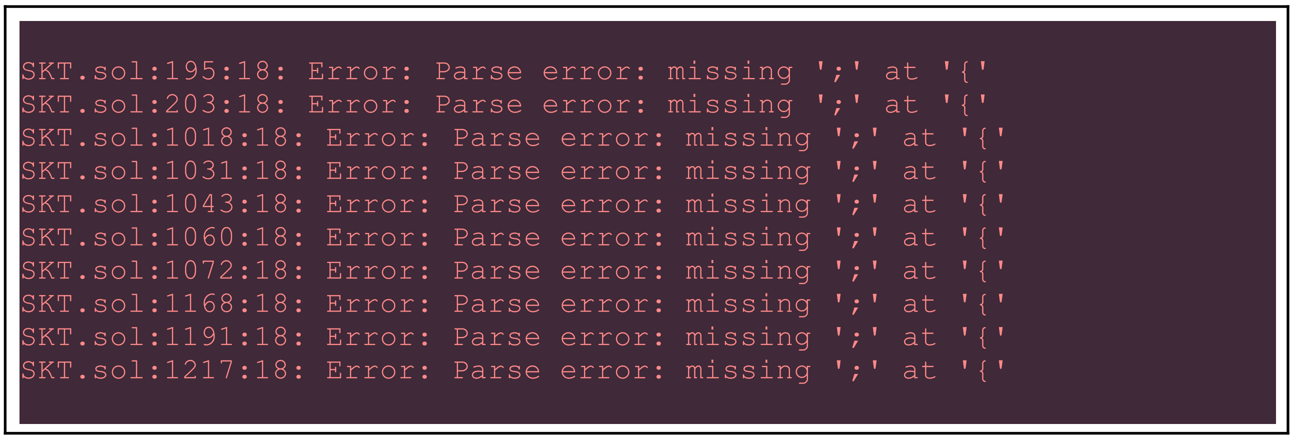
```
KOL_ProtectV2.sol:227:18: Error: Parse error: missing ';' at '{'
```

SFO_DAO.sol



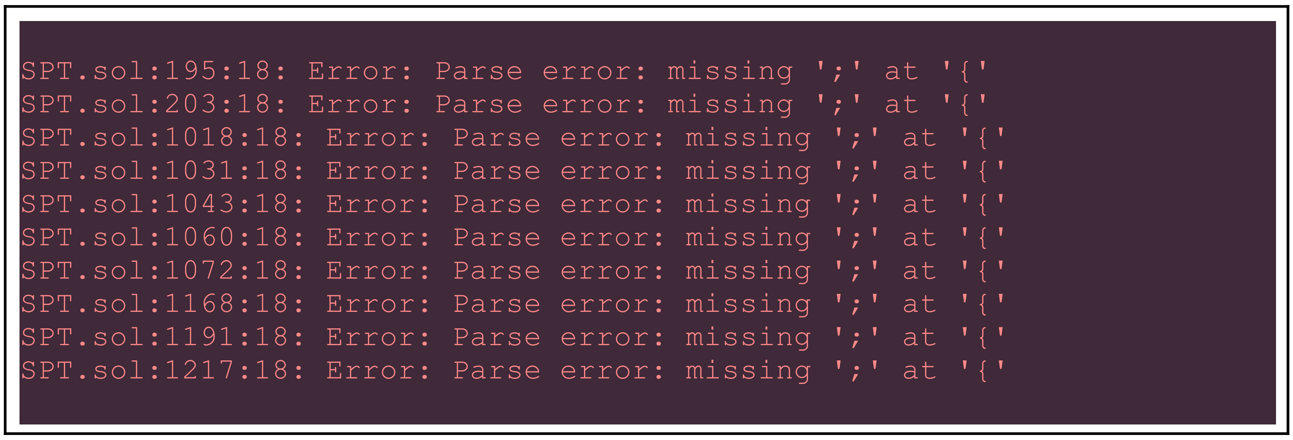
```
SFO_DAO.sol:327:18: Error: Parse error: missing ';' at '{'
```

SKT.sol



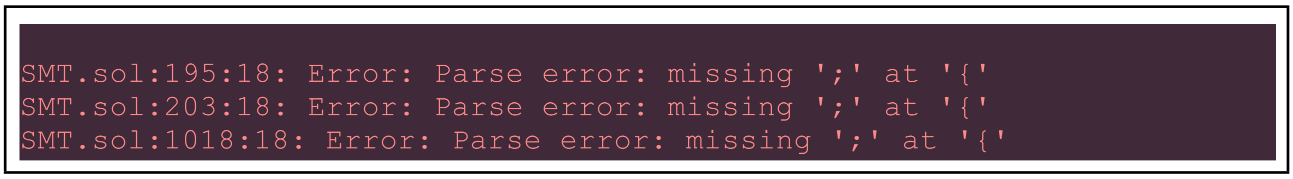
```
SKT.sol:195:18: Error: Parse error: missing ';' at '{'
SKT.sol:203:18: Error: Parse error: missing ';' at '{'
SKT.sol:1018:18: Error: Parse error: missing ';' at '{'
SKT.sol:1031:18: Error: Parse error: missing ';' at '{'
SKT.sol:1043:18: Error: Parse error: missing ';' at '{'
SKT.sol:1060:18: Error: Parse error: missing ';' at '{'
SKT.sol:1072:18: Error: Parse error: missing ';' at '{'
SKT.sol:1168:18: Error: Parse error: missing ';' at '{'
SKT.sol:1191:18: Error: Parse error: missing ';' at '{'
SKT.sol:1217:18: Error: Parse error: missing ';' at '{'
```

SPT.sol

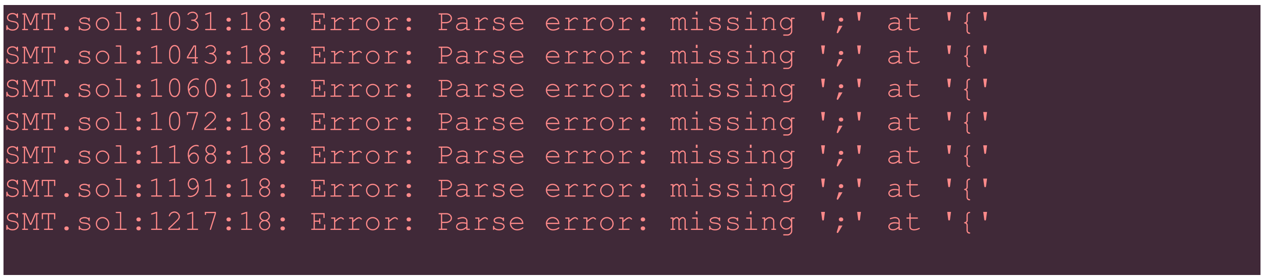


```
SPT.sol:195:18: Error: Parse error: missing ';' at '{'
SPT.sol:203:18: Error: Parse error: missing ';' at '{'
SPT.sol:1018:18: Error: Parse error: missing ';' at '{'
SPT.sol:1031:18: Error: Parse error: missing ';' at '{'
SPT.sol:1043:18: Error: Parse error: missing ';' at '{'
SPT.sol:1060:18: Error: Parse error: missing ';' at '{'
SPT.sol:1072:18: Error: Parse error: missing ';' at '{'
SPT.sol:1168:18: Error: Parse error: missing ';' at '{'
SPT.sol:1191:18: Error: Parse error: missing ';' at '{'
SPT.sol:1217:18: Error: Parse error: missing ';' at '{'
```

SMT.sol

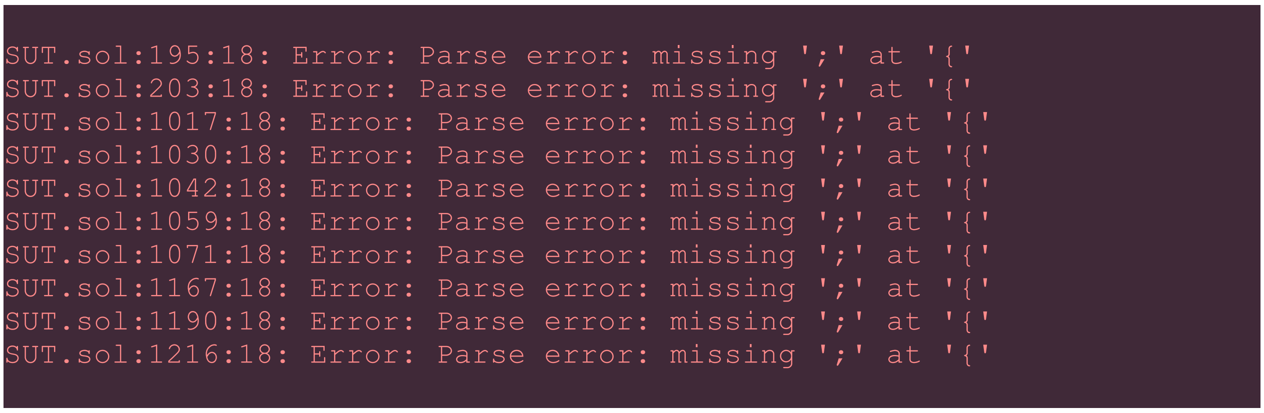


```
SMT.sol:195:18: Error: Parse error: missing ';' at '{'
SMT.sol:203:18: Error: Parse error: missing ';' at '{'
SMT.sol:1018:18: Error: Parse error: missing ';' at '{'
```



```
SMT.sol:1031:18: Error: Parse error: missing ';' at '{'
SMT.sol:1043:18: Error: Parse error: missing ';' at '{'
SMT.sol:1060:18: Error: Parse error: missing ';' at '{'
SMT.sol:1072:18: Error: Parse error: missing ';' at '{'
SMT.sol:1168:18: Error: Parse error: missing ';' at '{'
SMT.sol:1191:18: Error: Parse error: missing ';' at '{'
SMT.sol:1217:18: Error: Parse error: missing ';' at '{'
```

SUT.sol



```
SUT.sol:195:18: Error: Parse error: missing ';' at '{'
SUT.sol:203:18: Error: Parse error: missing ';' at '{'
SUT.sol:1017:18: Error: Parse error: missing ';' at '{'
SUT.sol:1030:18: Error: Parse error: missing ';' at '{'
SUT.sol:1042:18: Error: Parse error: missing ';' at '{'
SUT.sol:1059:18: Error: Parse error: missing ';' at '{'
SUT.sol:1071:18: Error: Parse error: missing ';' at '{'
SUT.sol:1167:18: Error: Parse error: missing ';' at '{'
SUT.sol:1190:18: Error: Parse error: missing ';' at '{'
SUT.sol:1216:18: Error: Parse error: missing ';' at '{'
```

Software analysis result:

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

EtherAuthority