Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Selfient
Platform: Polygon
Language: Solidity
Date: August 26th, 2023

Table of contents

Introduction 4
Project Background 4
Audit Scope 5
Claimed Smart Contract Features 6
Audit Summary 8
Technical Quick Stats 9
Code Quality 10
Documentation 10
Use of Dependencies 10
AS-IS overview 11
Severity Definitions 15
Audit Findings 16
Conclusion 19
Our Methodology 20
Disclaimers 22
Appendix

- Code Flow Diagram 23
- Slither Results Log 28
- Solidity static analysis 34
- Solhint Linter 40

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

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the Selfient team to perform the Security audit of the Selfient smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on August 26th, 2023.

The purpose of this audit was to address the following:

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

Project Background

- Selfient is a contract that can be divided into multiples, each with unique functionalities:
    - **LinearAgreement:** The Smart Employment Agreement is being implemented for linear agreements.
    - **MilestoneAgreement:** The Smart Employment Agreement is being implemented for milestone agreements.
    - **MockUSDC:** The Smart contract is used for minting tokens.
    - **SelfientAdmin:** The Smart contract performs administrative functions for manager contract functionality, including allowing ERC20 payment tokens listing, registering SEA contracts, and distributing fees.
    - **SelfientManager:** The Smart contract is utilized for managing agreement creation, funding, withdrawals, and funds distribution.
- The smart contracts have functions like deposit and withdraw funds, update agreements, mint, burn, etc.

Audit scope

Name	Code Review and Security Analysis Report for Selfient Smart Contracts
Platform	**Polygon / Solidity**
File 1	LinearAgreement.sol
File 1 MD5 Hash	11AF972143B0036584DC0D3B412E03D6
File 2	MilestoneAgreement.sol
File 2 MD5 Hash	2CB691864059F695092A30F506C3CA1D
File 3	MockUSDC.sol
File 3 MD5 Hash	B80F9FBFB1CD9C95A35B9294DDDAB6E3
File 4	SelfientAdmin.sol
File 4 MD5 Hash	3064DE859D2CA46C3A37B1069329535A
File 5	SelfientManager.sol
File 5 MD5 Hash	3CBDACC49A4C50C32E5A05A35C2612F2
Audit Date	August 26th, 2023

Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 LinearAgreement.sol **The Selfient Manager has control over the following functions:** • Create a new and terminated agreement. • Deposit funds. • Withdraw funds. **Other Specifications:** • Implementation of the Smart Employment Agreement for linear agreements. • Initial deposits are made by the hirer and can be withdrawn by the talent in linear increments based on the time passed since the beginning of the agreement.	**YES, This is valid.**
File 2 MilestoneAgreement.sol **The Selfient Manager has control over the following functions:** • Create a new and terminated agreement. • Deposit funds. • Withdraw funds. **Other Specifications:** • Implementation of the Smart Employment Agreement for milestone agreements. • Milestones are initially defined by the hirer, with funds deposited per milestone,and can be withdrawn by the talent on a per-milestone basis.	**YES, This is valid.**
File 3 MockUSDC.sol • Name: MockUSDC • Symbol: MUSDC	**YES, This is valid.**

• Decimals: 6 • Total Supply: 40 Quadrillion	
File 4 SelfientAdmin.sol **<u>The Selfient Manager has control over the following functions:</u>** • Set the token address. • Set the revoke token address. • Set the agreement fees. • Set the fee wallet address. • Set the Hirer Agreement Fee address. **<u>Other Specifications:</u>** • Performs administrative functions to facilitate the manager contract functionality including allowlisting ERC20 payment tokens, registering SEA contracts, and distributing fees	**YES, This is valid.**
File 5 SelfientManager.sol **<u>The Selfient Manager has control over the following functions:</u>** • Set the Grants `role` to `account`. • Revokes `role` from `account`. **<u>Other Specifications:</u>** • The Smart contract is utilized for managing agreement creation, funding, withdrawals, and funds distribution.	**YES, This is valid.**

Audit Summary

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

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

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

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

Technical Quick Stats

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

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 5 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in Selfient 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 Selfient Protocol.

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

Code parts are well commented on smart contracts.

Documentation

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

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

Use of Dependencies

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

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

AS-IS overview

LinearAgreement.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	createAgreement	external	access only Role	No Issue
3	terminateAgreement	external	access only Role	No Issue
4	depositFunds	external	access only Role	No Issue
5	withdrawFunds	external	access only Role	No Issue
6	agreementStatus	external	Passed	No Issue
7	claimableValue	read	Passed	No Issue
8	getTrimmedAgreementFields	external	Passed	No Issue
9	earlyWithdrawFunds	external	access only Role	No Issue
10	transferFunds	internal	Passed	No Issue
11	withdrawFundsInternal	internal	Passed	No Issue
12	onlyRole	modifier	Passed	No Issue
13	supportsInterface	read	Passed	No Issue
14	hasRole	read	Passed	No Issue
15	_checkRole	internal	Passed	No Issue
16	_checkRole	internal	Passed	No Issue
17	getRoleAdmin	read	Passed	No Issue
18	grantRole	write	access only Role	No Issue
19	revokeRole	write	access only Role	No Issue
20	renounceRole	write	Passed	No Issue
21	_setRoleAdmin	internal	Passed	No Issue
22	_grantRole	internal	Passed	No Issue
23	_revokeRole	internal	Passed	No Issue

MilestoneAgreement.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	createAgreement	external	access only Role	No Issue
3	depositFunds	external	access only Role	No Issue
4	claimableValue	read	Passed	No Issue
5	withdrawFunds	write	access only Role	No Issue
6	withdrawFundsInternal	internal	Passed	No Issue
7	terminateAgreement	write	Passed	No Issue
8	agreementStatus	external	Passed	No Issue
9	earlyWithdrawFunds	external	access only Role	No Issue
10	getTrimmedAgreementFields	external	Passed	No Issue
11	transferFunds	internal	Passed	No Issue
12	recreateMessage	internal	Passed	No Issue

13	verifySignature	internal	Passed	No Issue
14	isAgreementTerminated	internal	Passed	No Issue
15	computeMerkleRoot	internal	Passed	No Issue
16	hashMilestone	internal	Passed	No Issue
17	_hashPair	write	Passed	No Issue
18	_efficientHash	write	Passed	No Issue
19	onlyRole	write	Passed	No Issue
20	supportsInterface	read	Passed	No Issue
21	hasRole	read	Passed	No Issue
22	_checkRole	internal	Passed	No Issue
23	_checkRole	internal	Passed	No Issue
24	getRoleAdmin	read	Passed	No Issue
25	grantRole	write	access only Role	No Issue
26	revokeRole	write	access only Role	No Issue
27	renounceRole	write	Passed	No Issue
28	_setRoleAdmin	internal	Passed	No Issue
29	_grantRole	internal	Passed	No Issue
30	_revokeRole	internal	Passed	No Issue

MockUSDC.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	mint	write	Passed	No Issue
3	name	read	Passed	No Issue
4	symbol	read	Passed	No Issue
5	decimals	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	transfer	write	Passed	No Issue
9	allowance	read	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	_transfer	internal	Passed	No Issue
15	_update	internal	Passed	No Issue
16	_mint	internal	Passed	No Issue
17	_burn	internal	Passed	No Issue
18	_approve	internal	Passed	No Issue
19	_approve	internal	Passed	No Issue
20	_spendAllowance	internal	Passed	No Issue
21	burn	write	Passed	No Issue
22	burnFrom	write	Passed	No Issue
23	onlyOwner	modifier	Passed	No Issue
24	owner	read	Passed	No Issue

25	_checkOwner	internal	Passed	No Issue
26	renounceOwnership	write	access only Owner	No Issue
27	transferOwnership	write	access only Owner	No Issue
28	_transferOwnership	internal	Passed	No Issue

SelfientAdmin.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	allowToken	external	access only Role	No Issue
3	revokeToken	external	access only Role	No Issue
4	validateToken	read	Passed	No Issue
5	registerSEA	external	access only Role	No Issue
6	setAgreementFee	external	access only Role	No Issue
7	setFeewallet	external	access only Role	No Issue
8	setHirerAgreementFee	external	access only Role	No Issue
9	distributeAgreementFee	internal	Passed	No Issue
10	onlyRole	write	Passed	No Issue
11	supportsInterface	read	Passed	No Issue
12	hasRole	read	Passed	No Issue
13	_checkRole	internal	Passed	No Issue
14	_checkRole	internal	Passed	No Issue
15	getRoleAdmin	read	Passed	No Issue
16	grantRole	write	access only Role	No Issue
17	revokeRole	write	access only Role	No Issue
18	renounceRole	write	Passed	No Issue
19	_setRoleAdmin	internal	Passed	No Issue
20	_grantRole	internal	Passed	No Issue
21	_revokeRole	internal	Passed	No Issue

SelfientManager.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getAgreement	external	Passed	No Issue
3	createAgreement	external	Passed	No Issue
4	depositFunds	external	Passed	No Issue
5	earlyWithdrawFunds	external	Passed	No Issue
6	withdrawFunds	external	Passed	No Issue
7	terminateAgreement	external	Passed	No Issue
8	transferFunds	internal	Passed	No Issue
9	verifySignature	internal	Passed	No Issue

10	recreateMessage	internal	Passed	No Issue
11	_incrementAgreementCounter	internal	Passed	No Issue
12	_getAgreementContract	internal	Passed	No Issue
13	_retrieveAgreement	internal	Passed	No Issue
14	allowToken	external	access only Role	No Issue
15	revokeToken	external	access only Role	No Issue
16	validateToken	read	Passed	No Issue
17	registerSEA	external	access only Role	No Issue
18	setAgreementFee	external	access only Role	No Issue
19	setFeewallet	external	access only Role	No Issue
20	setHirerAgreementFee	external	access only Role	No Issue
21	distributeAgreementFee	internal	Passed	No Issue
22	onlyRole	write	Passed	No Issue
23	supportsInterface	read	Passed	No Issue
24	hasRole	read	Passed	No Issue
25	_checkRole	internal	Passed	No Issue
26	_checkRole	internal	Passed	No Issue
27	getRoleAdmin	read	Passed	No Issue
28	grantRole	write	access only Role	No Issue
29	revokeRole	write	access only Role	No Issue
30	renounceRole	write	Passed	No Issue
31	_setRoleAdmin	internal	Passed	No Issue
32	_grantRole	internal	Passed	No Issue
33	_revokeRole	internal	Passed	No Issue

Severity Definitions

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

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No critical severity vulnerabilities were found in the contract code.

High Severity

No high severity vulnerabilities were found in the contract code.

Medium

No medium severity vulnerabilities were found in the contract code.

Low

No low severity vulnerabilities were found in the contract code.

Very Low / Informational / Best practices:

(1) Unused constant and import **SelfientManager.sol**

Constant variable:

```
32  contract SelfientManager is AccessControl, ISelfientManager, SelfientAdmin {
33    using SafeERC20 for IERC20;
34    using ECDSA for bytes32;
35
36    bytes32 public constant SELFIENT_MANAGER = keccak256("SELFIENT_MANAGER");
37    uint256 public agreementCounter = 0;
38
39    // Agreement Id => Agreement Type
40    mapping(uint256 => uint8) public agreementTypeById;
41
```

The SELFIENT_MANAGER constant is defined but never used.

import:

```
13
14  import "@openzeppelin/contracts/access/AccessControl.sol";
15  import "@openzeppelin/contracts/utils/Address.sol";
16  import "@openzeppelin/contracts/utils/cryptography/ECDSA.sol";
17
18  import "../interfaces/ISmartEmploymentAgreement.sol";
19  import "../interfaces/ISelfientManager.sol";
20  import "../interfaces/ISelfientAdmin.sol";
21
22  import "./SelfientAdmin.sol";
23  import "./LinearAgreement.sol";
24
25  /**
26   * @title Selfient Manager
27   * @author Developed by Labrys on behalf of Selfient
28   * @custom:contributor Arjun Menon (arjunmenon.eth)
29   * @custom:contributor mfbevan (mfbevan.eth)
30   * @notice Manages agreement creation, funding, withdrawals and handles funds distribution
31   */
32  contract SelfientManager is AccessControl, ISelfientManager, SelfientAdmin {
33    using SafeERC20 for IERC20;
```

The ISelfientAdmin interface is imported but never used.

Resolution: We suggest If not needed, please remove it.

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:

LinearAgreement.sol

- createAgreement: Selfient Manager can create a new agreement.
- terminateAgreement: Selfient Manager can terminate agreement.
- depositFunds: Selfient Managers can deposit funds.
- withdrawFunds: Selfient Managers can withdraw funds.
- earlyWithdrawFunds: Selfient Managers can early withdraw funds.

MilestoneAgreement.sol

- createAgreement: Selfient Manager can create a new agreement.

- depositFunds: Selfient Managers can deposit funds.
- withdrawFunds: Selfient Managers can withdraw funds.
- terminateAgreement: Selfient Manager can terminate agreement.
- earlyWithdrawFunds: Selfient Managers can early withdraw funds.

SelfientAdmin.sol

- allowToken: Selfient admin can set token address.
- revokeToken: Selfient admin can set revoke token address.
- registerSEA: Selfient admin can register SEA address.
- setAgreementFee: Agreement fee can be set by the Selfient admin.
- setFeewallet: Fee wallet address can be set by the Selfient admin.
- setHirerAgreementFee: Hirer Agreement Fee address can be set by the Selfient admin.

AccessControl.sol

- grantRole: Grants `role` to `account` can be set by the owner.
- revokeRole: Revokes `role` from `account` by the owner.

Ownable.sol

- renounceOwnership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transferOwnership: Current owner can transfer ownership of the contract to a new account.

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

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We had observed 1 Informational severity issue in the smart contracts. but that is not a critical one. **So, the smart contracts are ready for the mainnet deployment.**

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

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

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

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

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

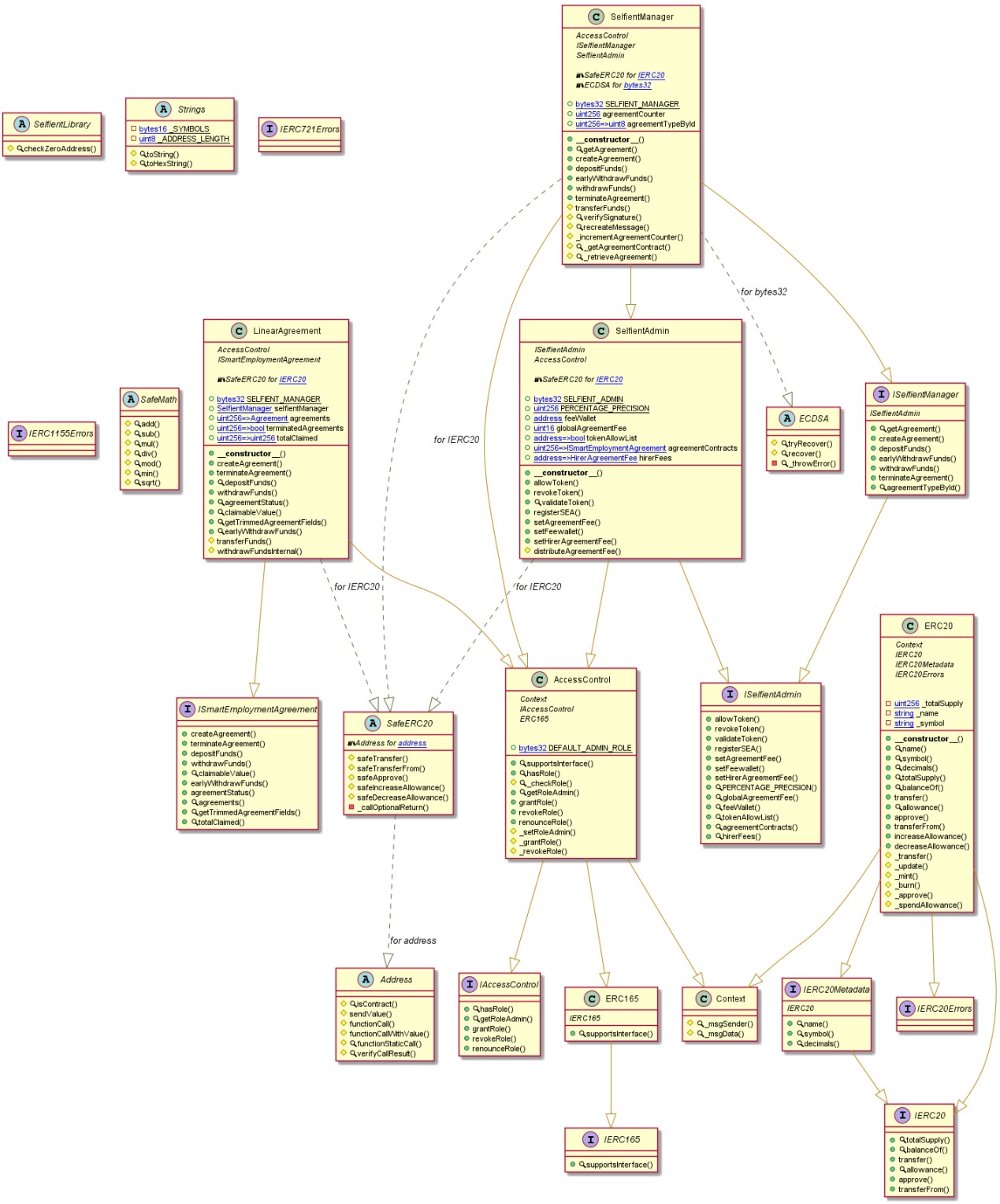
Technical Disclaimer

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

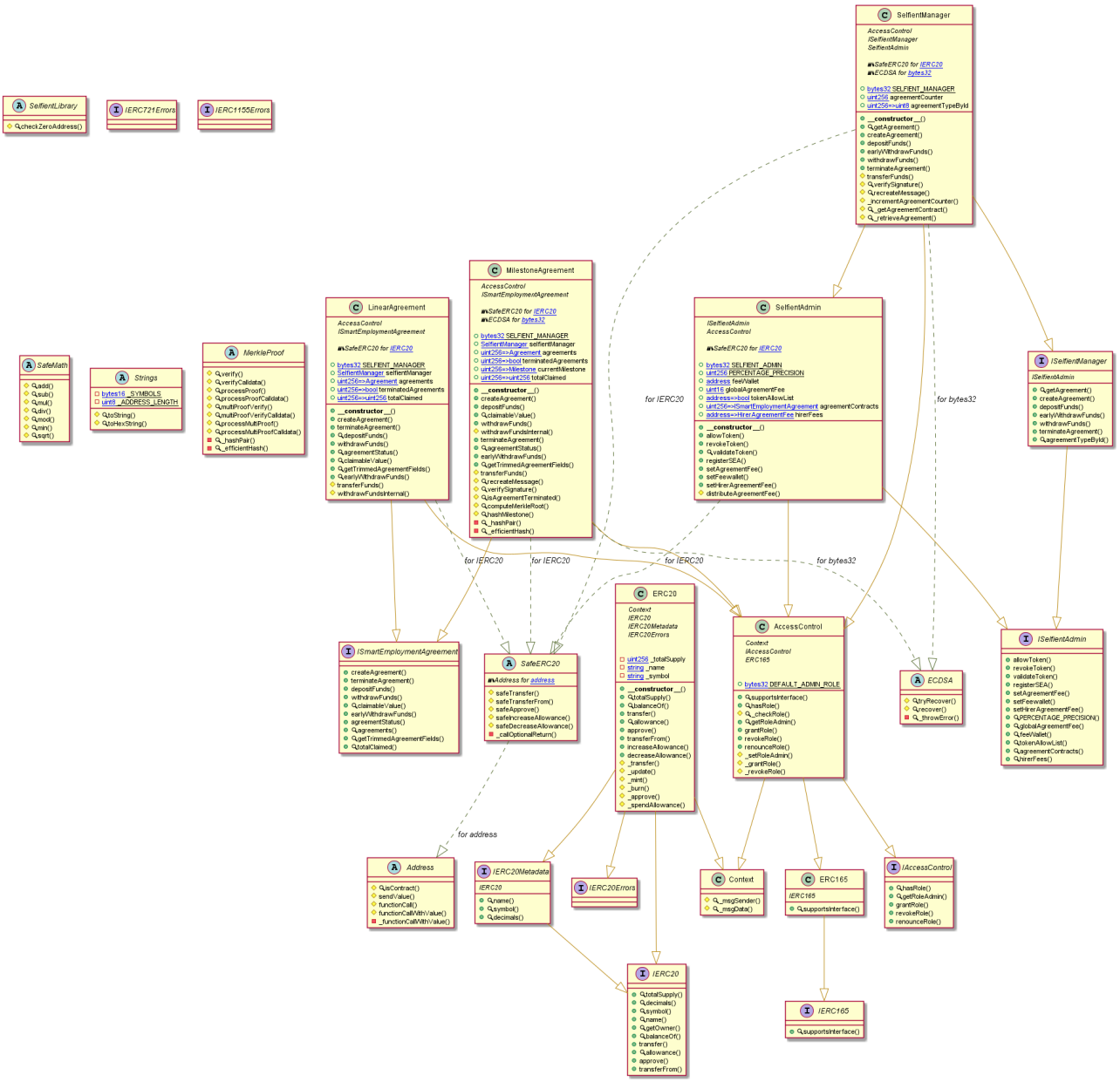
Appendix

Code Flow Diagram - Selfient Protocol

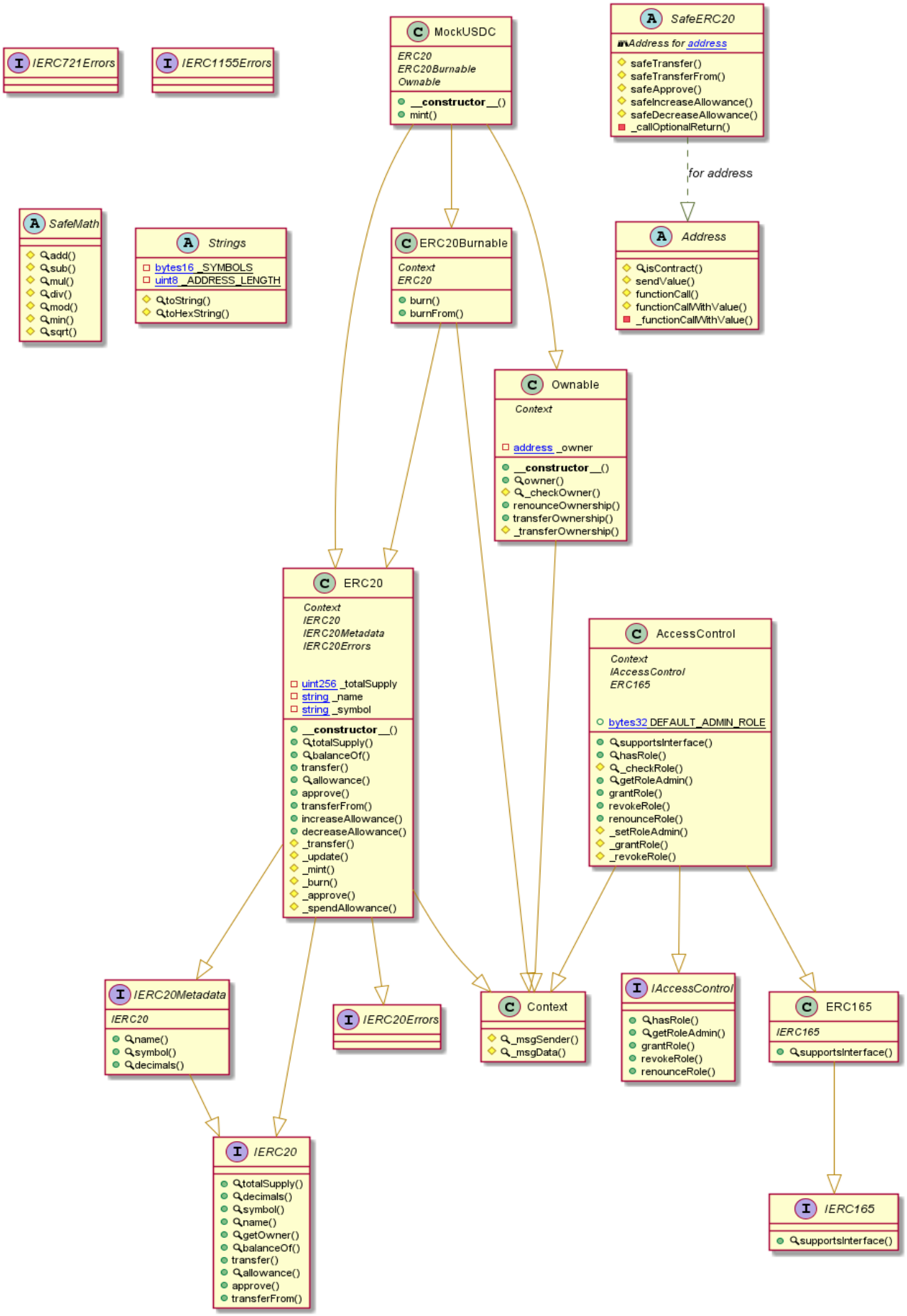
LinearAgreement Diagram



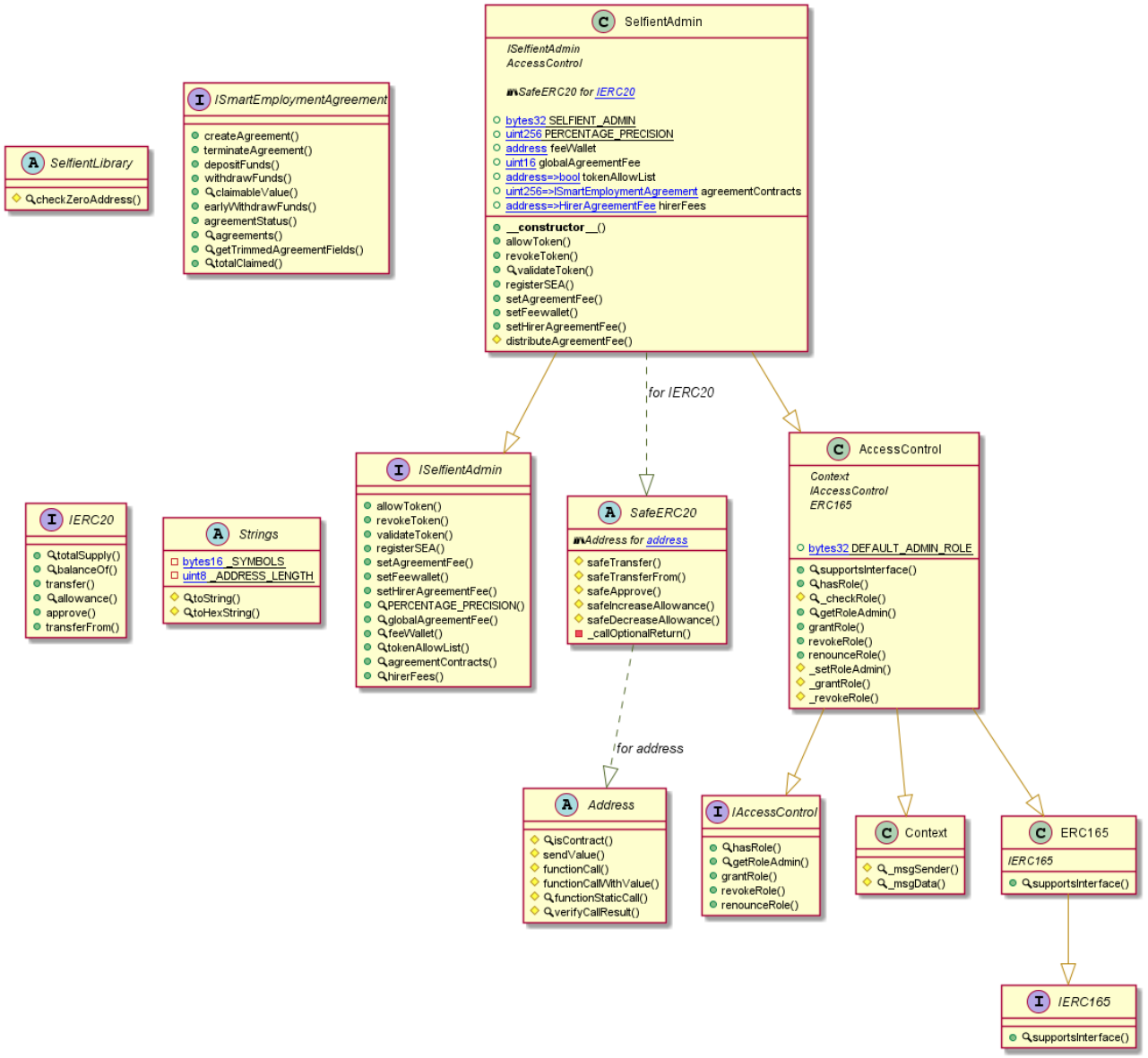
MilestoneAgreement Diagram



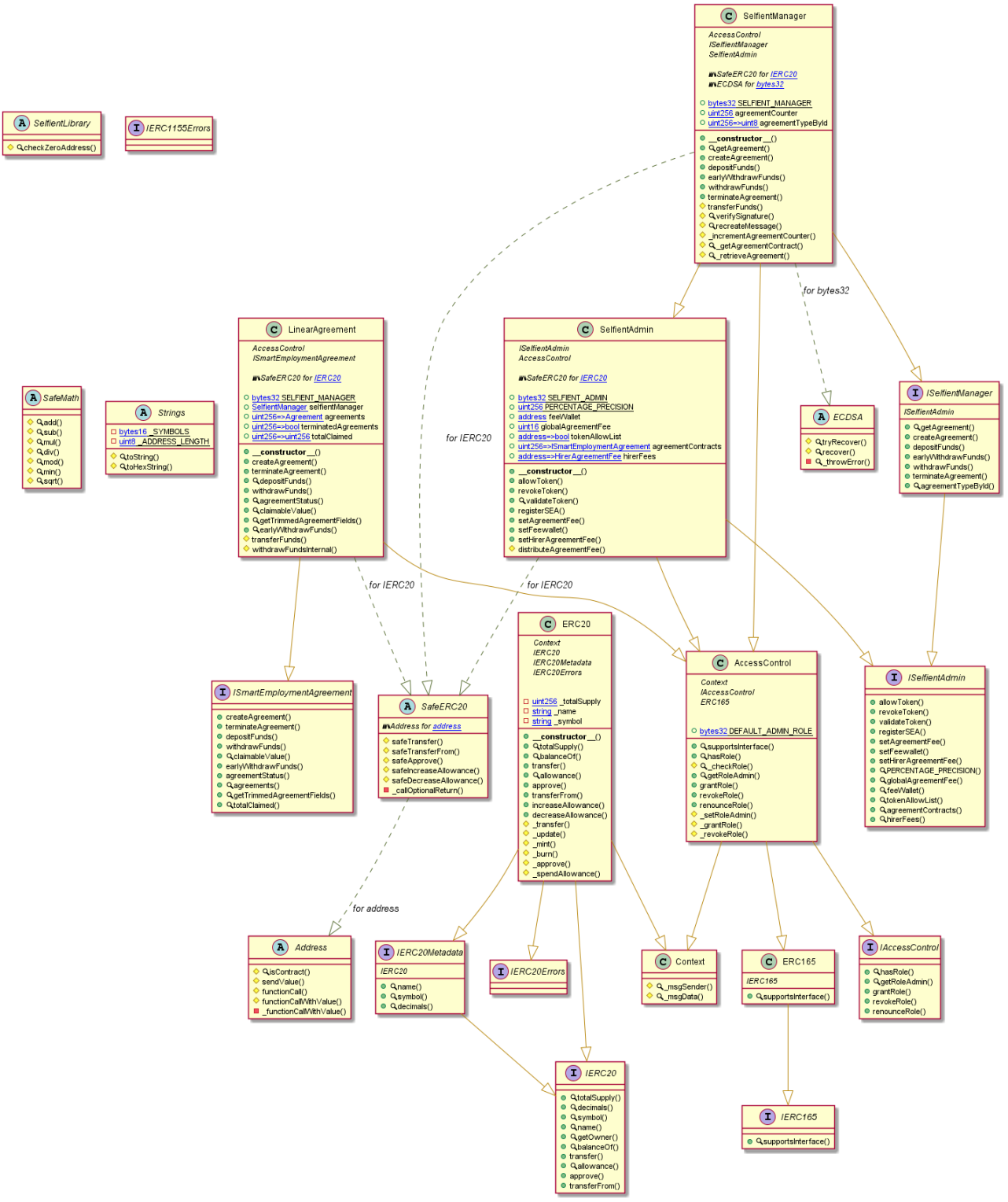
MockUSDC Diagram



SelfientAdmin Diagram



SelfientManager Diagram



Slither Results Log

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

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

Slither log >> LinearAgreement.sol

```
SelfientAdmin.setFeewallet(address)._address (LinearAgreement.sol#592) lacks a zero-check on :
                - feeWallet = _address (LinearAgreement.sol#597)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SelfientManager.createAgreement(ISmartEmploymentAgreement.NewAgreement,bytes) (LinearAgreement.sol#1044-1115):
        External calls:
        - depositAmount = agreementContract.createAgreement(newAgreementId,_agreement,_data) (LinearAgreement.sol#1097-1101)
        - distributeAgreementFee(_agreement.hirer,depositAmount,_agreement.currency) (LinearAgreement.sol#1103-1107)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (LinearAgreement.sol#361)
                - (success,returndata) = target.call{value: value}(data) (LinearAgreement.sol#261)
                - _token.safeTransferFrom(_hirer,feeWallet,fee) (LinearAgreement.sol#641)
        External calls sending eth:
        - distributeAgreementFee(_agreement.hirer,depositAmount,_agreement.currency) (LinearAgreement.sol#1103-1107)
                - (success,returndata) = target.call{value: value}(data) (LinearAgreement.sol#261)
        Event emitted after the call(s):
        - FeeDistributed(fee) (LinearAgreement.sol#640)
                - distributeAgreementFee(_agreement.hirer,depositAmount,_agreement.currency) (LinearAgreement.sol#1103-1107)
Reentrancy in SelfientManager.depositFunds(uint256,ISmartEmploymentAgreement.Milestone,bytes) (LinearAgreement.sol#1117-1148):
        External calls:
        - agreementContract.depositFunds(_agreementId,_milestone,_data) (LinearAgreement.sol#1134)
        - distributeAgreementFee(agreement.hirer,_milestone.amount,agreement.currency) (LinearAgreement.sol#1136-1140)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (LinearAgreement.sol#361)
                - (success,returndata) = target.call{value: value}(data) (LinearAgreement.sol#261)
                - _token.safeTransferFrom(_hirer,feeWallet,fee) (LinearAgreement.sol#641)
        External calls sending eth:
        - distributeAgreementFee(agreement.hirer,_milestone.amount,agreement.currency) (LinearAgreement.sol#1136-1140)
                - (success,returndata) = target.call{value: value}(data) (LinearAgreement.sol#261)
        Event emitted after the call(s):
        - FeeDistributed(fee) (LinearAgreement.sol#640)
                - distributeAgreementFee(agreement.hirer,_milestone.amount,agreement.currency) (LinearAgreement.sol#1136-1140)
        External calls sending eth:
        - withdrawFundsInternal(_agreementId,false) (LinearAgreement.sol#1363)
                - (success,returndata) = target.call{value: value}(data) (LinearAgreement.sol#261)
        Event emitted after the call(s):
        - FundsWithdrawn(_agreementId,hirerFunds) (LinearAgreement.sol#1371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

LinearAgreement.terminateAgreement(uint256) (LinearAgreement.sol#1351-1378) uses timestamp for comparisons
        Dangerous comparisons:
        - hirerFunds > 0 (LinearAgreement.sol#1370)
LinearAgreement.agreementStatus(uint256) (LinearAgreement.sol#1394-1415) uses timestamp for comparisons
        Dangerous comparisons:
        - agreements[_agreementId].agreementId == 0 (LinearAgreement.sol#1397)
        - block.timestamp >= startDate && block.timestamp < agreementEndDate (LinearAgreement.sol#1410)
LinearAgreement.claimableValue(uint256) (LinearAgreement.sol#1417-1451) uses timestamp for comparisons
        Dangerous comparisons:
        - agreements[_agreementId].agreementId == 0 (LinearAgreement.sol#1418)
        - terminatedAgreements[_agreementId] || lastClaim >= agreementEndDate || block.timestamp <= lastClaim || block.timesta
mp <= startDate (LinearAgreement.sol#1429-1432)
        - lastClaim == 0 (LinearAgreement.sol#1439)
        - block.timestamp >= agreementEndDate (LinearAgreement.sol#1440)
        - block.timestamp >= agreementEndDate (LinearAgreement.sol#1446)
LinearAgreement.withdrawFundsInternal(uint256,bool) (LinearAgreement.sol#1482-1504) uses timestamp for comparisons
        Dangerous comparisons:
        - withdrawalAmount == 0 && _revertOnEmptyWithdrawalAmount (LinearAgreement.sol#1488)
        - withdrawalAmount != 0 (LinearAgreement.sol#1497)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.verifyCallResult(bool,bytes,string) (LinearAgreement.sol#283-301) uses assembly
        - INLINE ASM (LinearAgreement.sol#293-296)
Strings.toString(uint256) (LinearAgreement.sol#373-387) uses assembly
        - INLINE ASM (LinearAgreement.sol#376-377)
        - INLINE ASM (LinearAgreement.sol#380-382)
ECDSA.tryRecover(bytes32,bytes) (LinearAgreement.sol#654-668) uses assembly
        - INLINE ASM (LinearAgreement.sol#659-663)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version0.8.17 (LinearAgreement.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0
.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (LinearAgreement.sol#225-230):
        - (success) = recipient.call{value: amount}() (LinearAgreement.sol#228)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (LinearAgreement.sol#252-263):
        - (success,returndata) = target.call{value: value}(data) (LinearAgreement.sol#261)
Low level call in Address.functionStaticCall(address,bytes,string) (LinearAgreement.sol#269-278):
        - (success,returndata) = target.staticcall(data) (LinearAgreement.sol#276)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter SelfientLibrary.checkZeroAddress(address,string)._address (LinearAgreement.sol#10) is not in mixedCase
Function ISelfientAdmin.PERCENTAGE_PRECISION() (LinearAgreement.sol#178) is not in mixedCase
Parameter SelfientAdmin.allowToken(address)._tokenContract (LinearAgreement.sol#544) is not in mixedCase
Parameter SelfientAdmin.revokeToken(address)._tokenContract (LinearAgreement.sol#554) is not in mixedCase
Parameter SelfientAdmin.validateToken(address)._address (LinearAgreement.sol#563) is not in mixedCase
Parameter SelfientAdmin.registerSEA(address,uint8)._contractAddress (LinearAgreement.sol#570) is not in mixedCase
Parameter SelfientAdmin.registerSEA(address,uint8)._contractId (LinearAgreement.sol#571) is not in mixedCase
Parameter SelfientAdmin.setAgreementFee(uint16)._fee (LinearAgreement.sol#583) is not in mixedCase
Parameter SelfientAdmin.setFeewallet(address)._address (LinearAgreement.sol#592) is not in mixedCase
Parameter SelfientAdmin.setHirerAgreementFee(address,uint16,bool)._address (LinearAgreement.sol#601) is not in mixedCase
Parameter SelfientAdmin.setHirerAgreementFee(address,uint16,bool)._fee (LinearAgreement.sol#602) is not in mixedCase
Parameter SelfientAdmin.setHirerAgreementFee(address,uint16,bool)._disabled (LinearAgreement.sol#603) is not in mixedCase
Parameter SelfientAdmin.distributeAgreementFee(address,uint256,address)._hirer (LinearAgreement.sol#618) is not in mixedCase
Parameter SelfientAdmin.distributeAgreementFee(address,uint256,address)._agreementValue (LinearAgreement.sol#619) is not in mi
xedCase
Parameter SelfientAdmin.distributeAgreementFee(address,uint256,address)._tokenAddress (LinearAgreement.sol#620) is not in mixe
dCase
Parameter SelfientManager.getAgreement(uint256)._agreementId (LinearAgreement.sol#1031) is not in mixedCase
Parameter SelfientManager.createAgreement(ISmartEmploymentAgreement.NewAgreement,bytes)._agreement (LinearAgreement.sol#1045)
is not in mixedCase
Parameter SelfientManager.createAgreement(ISmartEmploymentAgreement.NewAgreement,bytes)._data (LinearAgreement.sol#1046) is no
t in mixedCase
```

```
Parameter SelfientManager.withdrawFunds(uint256)._agreementId (LinearAgreement.sol#1173) is not in mixedCase
Parameter SelfientManager.terminateAgreement(uint256)._agreementId (LinearAgreement.sol#1188) is not in mixedCase
Parameter SelfientManager.transferFunds(address,address,uint256,address)._from (LinearAgreement.sol#1204) is not in mixedCase
Parameter SelfientManager.transferFunds(address,address,uint256,address)._to (LinearAgreement.sol#1205) is not in mixedCase
Parameter SelfientManager.transferFunds(address,address,uint256,address)._amount (LinearAgreement.sol#1206) is not in mixedCas
e
Parameter SelfientManager.transferFunds(address,address,uint256,address)._tokenAddress (LinearAgreement.sol#1207) is not in mi
xedCase
Parameter SelfientManager.verifySignature(bytes32,bytes,address,string)._message (LinearAgreement.sol#1215) is not in mixedCas
e
Parameter SelfientManager.verifySignature(bytes32,bytes,address,string)._signature (LinearAgreement.sol#1216) is not in mixedC
ase
Parameter SelfientManager.verifySignature(bytes32,bytes,address,string)._signer (LinearAgreement.sol#1217) is not in mixedCase
Parameter LinearAgreement.createAgreement(uint256,ISmartEmploymentAgreement.NewAgreement,bytes)._agreementId (LinearAgreement.
sol#1320) is not in mixedCase
Parameter LinearAgreement.createAgreement(uint256,ISmartEmploymentAgreement.NewAgreement,bytes)._agreement (LinearAgreement.so
l#1321) is not in mixedCase
Parameter LinearAgreement.terminateAgreement(uint256)._agreementId (LinearAgreement.sol#1352) is not in mixedCase
Parameter LinearAgreement.withdrawFunds(uint256)._agreementId (LinearAgreement.sol#1389) is not in mixedCase
Parameter LinearAgreement.agreementStatus(uint256)._agreementId (LinearAgreement.sol#1395) is not in mixedCase
Parameter LinearAgreement.claimableValue(uint256)._agreementId (LinearAgreement.sol#1417) is not in mixedCase
Parameter LinearAgreement.getTrimmedAgreementFields(uint256)._agreementId (LinearAgreement.sol#1454) is not in mixedCase
Parameter LinearAgreement.transferFunds(address,uint256,address)._to (LinearAgreement.sol#1473) is not in mixedCase
Parameter LinearAgreement.transferFunds(address,uint256,address)._amount (LinearAgreement.sol#1474) is not in mixedCase
Parameter LinearAgreement.transferFunds(address,uint256,address)._tokenAddress (LinearAgreement.sol#1475) is not in mixedCase
Parameter LinearAgreement.withdrawFundsInternal(uint256,bool)._agreementId (LinearAgreement.sol#1483) is not in mixedCase
Parameter LinearAgreement.withdrawFundsInternal(uint256,bool)._revertOnEmptyWithdrawalAmount (LinearAgreement.sol#1484) is not
 in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
Variable SelfientManager._getAgreementContract(uint256)._agreementContract (LinearAgreement.sol#1249) is too similar to Selfie
ntAdmin.agreementContracts (LinearAgreement.sol#528-530)
Variable SelfientManager._retrieveAgreement(uint256)._agreementContract (LinearAgreement.sol#1264) is too similar to SelfientA
dmin.agreementContracts (LinearAgreement.sol#528-530)
Variable LinearAgreement.SELFIENT_MANAGER (LinearAgreement.sol#1301) is too similar to LinearAgreement.constructor(address,add
ress)._selfientManager (LinearAgreement.sol#1311)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

ERC20._name (LinearAgreement.sol#831) should be immutable
ERC20._symbol (LinearAgreement.sol#832) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
LinearAgreement.sol analyzed (23 contracts with 84 detectors), 102 result(s) found
```

Slither log >> MilestoneAgreement.sol

```
SelfientAdmin.setFeewallet(address)._address (MilestoneAgreement.sol#1021) lacks a zero-check on :
                - feeWallet = _address (MilestoneAgreement.sol#1026)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SelfientManager.createAgreement(ISmartEmploymentAgreement.NewAgreement,bytes) (MilestoneAgreement.sol#1222-1291)
:
        External calls:
        - depositAmount = agreementContract.createAgreement(newAgreementId,_agreement,_data) (MilestoneAgreement.sol#1273-1277
)
        - distributeAgreementFee(_agreement.hirer,depositAmount,_agreement.currency) (MilestoneAgreement.sol#1279-1283)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (MilestoneAgreement.sol#525)
                - (success,returndata) = target.call{value: weiValue}(data) (MilestoneAgreement.sol#278)
                - _token.safeTransferFrom(_hirer,feeWallet,fee) (MilestoneAgreement.sol#1070)
        External calls sending eth:
        - distributeAgreementFee(_agreement.hirer,depositAmount,_agreement.currency) (MilestoneAgreement.sol#1279-1283)
                - (success,returndata) = target.call{value: weiValue}(data) (MilestoneAgreement.sol#278)
        Event emitted after the call(s):
        - FeeDistributed(fee) (MilestoneAgreement.sol#1069)
                - distributeAgreementFee(_agreement.hirer,depositAmount,_agreement.currency) (MilestoneAgreement.sol#1279-1283
)
```

```
Reentrancy in SelfientManager.depositFunds(uint256,ISmartEmploymentAgreement.Milestone,bytes) (MilestoneAgreement.sol#1293-132
3):
        External calls:
        - agreementContract.depositFunds(_agreementId,_milestone,_data) (MilestoneAgreement.sol#1309)
        - distributeAgreementFee(agreement.hirer,_milestone.amount,agreement.currency) (MilestoneAgreement.sol#1311-1315)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (MilestoneAgreement.sol#525)
                - (success,returndata) = target.call{value: weiValue}(data) (MilestoneAgreement.sol#278)
                - _token.safeTransferFrom(_hirer,feeWallet,fee) (MilestoneAgreement.sol#1070)
        External calls sending eth:
        - distributeAgreementFee(agreement.hirer,_milestone.amount,agreement.currency) (MilestoneAgreement.sol#1311-1315)
                - (success,returndata) = target.call{value: weiValue}(data) (MilestoneAgreement.sol#278)
        Event emitted after the call(s):
        - FeeDistributed(fee) (MilestoneAgreement.sol#1069)
                - distributeAgreementFee(agreement.hirer,_milestone.amount,agreement.currency) (MilestoneAgreement.sol#1311-13
15)
Reentrancy in MilestoneAgreement.depositFunds(uint256,ISmartEmploymentAgreement.Milestone,bytes) (MilestoneAgreement.sol#1689-
1731):
        External calls:
        - withdrawFundsInternal(_agreementId,currentMilestone[_agreementId].amount) (MilestoneAgreement.sol#1716-1719)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (MilestoneAgreement.sol#525)
                - _token.safeTransfer(_to,_amount) (MilestoneAgreement.sol#1876)
                - (success,returndata) = target.call{value: weiValue}(data) (MilestoneAgreement.sol#278)
        External calls sending eth:
        - withdrawFundsInternal(_agreementId,currentMilestone[_agreementId].amount) (MilestoneAgreement.sol#1716-1719)
                - (success,returndata) = target.call{value: weiValue}(data) (MilestoneAgreement.sol#278)
        Event emitted after the call(s):
        - FundsDeposited(_agreementId,_milestone.amount) (MilestoneAgreement.sol#1730)
Reentrancy in LinearAgreement.terminateAgreement(uint256) (MilestoneAgreement.sol#731-756):
        External calls:
        - withdrawFundsInternal(_agreementId,false) (MilestoneAgreement.sol#741)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (MilestoneAgreement.sol#525)
                - _token.safeTransfer(_to,_amount) (MilestoneAgreement.sol#853)
                - (success,returndata) = target.call{value: weiValue}(data) (MilestoneAgreement.sol#278)
        External calls sending eth:
        - withdrawFundsInternal(_agreementId,false) (MilestoneAgreement.sol#741)
                - (success,returndata) = target.call{value: weiValue}(data) (MilestoneAgreement.sol#278)
```

```
        Event emitted after the call(s):
        - FundsWithdrawn(_agreementId,hirerFunds) (MilestoneAgreement.sol#749)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

LinearAgreement.terminateAgreement(uint256) (MilestoneAgreement.sol#731-756) uses timestamp for comparisons
        Dangerous comparisons:
        - hirerFunds > 0 (MilestoneAgreement.sol#748)
LinearAgreement.agreementStatus(uint256) (MilestoneAgreement.sol#771-791) uses timestamp for comparisons
        Dangerous comparisons:
        - agreements[_agreementId].agreementId == 0 (MilestoneAgreement.sol#774)
        - block.timestamp >= startDate && block.timestamp < agreementEndDate (MilestoneAgreement.sol#786)
LinearAgreement.claimableValue(uint256) (MilestoneAgreement.sol#793-826) uses timestamp for comparisons
        Dangerous comparisons:
        - agreements[_agreementId].agreementId == 0 (MilestoneAgreement.sol#794)
        - terminatedAgreements[_agreementId] || lastClaim >= agreementEndDate || block.timestamp <= lastClaim || block.timesta
mp <= startDate (MilestoneAgreement.sol#804-807)
        - lastClaim == 0 (MilestoneAgreement.sol#814)
        - block.timestamp >= agreementEndDate (MilestoneAgreement.sol#815)
        - block.timestamp >= agreementEndDate (MilestoneAgreement.sol#821)
LinearAgreement.withdrawFundsInternal(uint256,bool) (MilestoneAgreement.sol#856-877) uses timestamp for comparisons
        Dangerous comparisons:
        - withdrawalAmount == 0 && _revertOnEmptyWithdrawalAmount (MilestoneAgreement.sol#862)
        - withdrawalAmount != 0 (MilestoneAgreement.sol#870)
MilestoneAgreement.claimableValue(uint256) (MilestoneAgreement.sol#1733-1748) uses timestamp for comparisons
        Dangerous comparisons:
        - agreements[_agreementId].agreementId == 0 (MilestoneAgreement.sol#1737)
        - block.timestamp < milestoneEndDate (MilestoneAgreement.sol#1743)
MilestoneAgreement.terminateAgreement(uint256) (MilestoneAgreement.sol#1779-1797) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= startDate && block.timestamp < milestoneEndDate (MilestoneAgreement.sol#1792)
MilestoneAgreement.agreementStatus(uint256) (MilestoneAgreement.sol#1799-1826) uses timestamp for comparisons
        Dangerous comparisons:
        - agreements[_agreementId].agreementId == 0 (MilestoneAgreement.sol#1802)
        - totalClaimed[_agreementId] == agreements[_agreementId].amount (MilestoneAgreement.sol#1809)
        - block.timestamp >= startDate && block.timestamp < milestoneEndDate (MilestoneAgreement.sol#1817)
        - block.timestamp - milestoneEndDate <= 864000 (MilestoneAgreement.sol#1821)
MilestoneAgreement.isAgreementTerminated(uint256) (MilestoneAgreement.sol#1900-1918) uses timestamp for comparisons
```

```
MilestoneAgreement.isAgreementTerminated(uint256) (MilestoneAgreement.sol#1900-1918) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > milestoneEndDate && block.timestamp - milestoneEndDate > 864000 (MilestoneAgreement.sol#1911-1912)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (MilestoneAgreement.sol#224-231) uses assembly
        - INLINE ASM (MilestoneAgreement.sol#227-229)
Address._functionCallWithValue(address,bytes,uint256,string) (MilestoneAgreement.sol#270-292) uses assembly
        - INLINE ASM (MilestoneAgreement.sol#284-287)
Strings.toString(uint256) (MilestoneAgreement.sol#537-551) uses assembly
        - INLINE ASM (MilestoneAgreement.sol#540-541)
        - INLINE ASM (MilestoneAgreement.sol#544-546)
ECDSA.tryRecover(bytes32,bytes) (MilestoneAgreement.sol#1083-1097) uses assembly
        - INLINE ASM (MilestoneAgreement.sol#1088-1092)
MerkleProof._efficientHash(bytes32,bytes32) (MilestoneAgreement.sol#1590-1596) uses assembly
        - INLINE ASM (MilestoneAgreement.sol#1591-1595)
MilestoneAgreement._efficientHash(bytes32,bytes32) (MilestoneAgreement.sol#1963-1972) uses assembly
        - INLINE ASM (MilestoneAgreement.sol#1967-1971)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
MerkleProof._efficientHash(bytes32,bytes32) (MilestoneAgreement.sol#1590-1596) uses assembly
        - INLINE ASM (MilestoneAgreement.sol#1591-1595)
MilestoneAgreement._efficientHash(bytes32,bytes32) (MilestoneAgreement.sol#1963-1972) uses assembly
        - INLINE ASM (MilestoneAgreement.sol#1967-1971)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version0.8.17 (MilestoneAgreement.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (MilestoneAgreement.sol#233-238):
        - (success) = recipient.call{value: amount}() (MilestoneAgreement.sol#236)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (MilestoneAgreement.sol#270-292):
        - (success,returndata) = target.call{value: weiValue}(data) (MilestoneAgreement.sol#278)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Parameter MilestoneAgreement.createAgreement(uint256,ISmartEmploymentAgreement.NewAgreement,bytes)._agreementId (MilestoneAgreement.sol#1625) is not in mixedCase
Parameter MilestoneAgreement.createAgreement(uint256,ISmartEmploymentAgreement.NewAgreement,bytes)._agreement (MilestoneAgreement.sol#1626) is not in mixedCase
Parameter MilestoneAgreement.createAgreement(uint256,ISmartEmploymentAgreement.NewAgreement,bytes)._data (MilestoneAgreement.sol#1627) is not in mixedCase
Parameter MilestoneAgreement.depositFunds(uint256,ISmartEmploymentAgreement.Milestone,bytes)._agreementId (MilestoneAgreement.sol#1690) is not in mixedCase
Parameter MilestoneAgreement.depositFunds(uint256,ISmartEmploymentAgreement.Milestone,bytes)._milestone (MilestoneAgreement.sol#1691) is not in mixedCase
Parameter MilestoneAgreement.depositFunds(uint256,ISmartEmploymentAgreement.Milestone,bytes)._data (MilestoneAgreement.sol#1692) is not in mixedCase
Parameter MilestoneAgreement.claimableValue(uint256)._agreementId (MilestoneAgreement.sol#1733) is not in mixedCase
Parameter MilestoneAgreement.withdrawFunds(uint256)._agreementId (MilestoneAgreement.sol#1751) is not in mixedCase
Parameter MilestoneAgreement.withdrawFundsInternal(uint256,uint256)._agreementId (MilestoneAgreement.sol#1761) is not in mixedCase
Parameter MilestoneAgreement.withdrawFundsInternal(uint256,uint256)._amount (MilestoneAgreement.sol#1762) is not in mixedCase
Parameter MilestoneAgreement.terminateAgreement(uint256)._agreementId (MilestoneAgreement.sol#1780) is not in mixedCase
Parameter MilestoneAgreement.agreementStatus(uint256)._agreementId (MilestoneAgreement.sol#1800) is not in mixedCase
Parameter MilestoneAgreement.earlyWithdrawFunds(uint256,bytes,bytes)._agreementId (MilestoneAgreement.sol#1829) is not in mixedCase
Parameter MilestoneAgreement.earlyWithdrawFunds(uint256,bytes,bytes)._hirerSignature (MilestoneAgreement.sol#1830) is not in mixedCase
Parameter MilestoneAgreement.earlyWithdrawFunds(uint256,bytes,bytes)._talentSignature (MilestoneAgreement.sol#1831) is not in mixedCase
Parameter MilestoneAgreement.getTrimmedAgreementFields(uint256)._agreementId (MilestoneAgreement.sol#1859) is not in mixedCase
Parameter MilestoneAgreement.transferFunds(address,uint256,address)._to (MilestoneAgreement.sol#1870) is not in mixedCase
Parameter MilestoneAgreement.transferFunds(address,uint256,address)._amount (MilestoneAgreement.sol#1871) is not in mixedCase
Parameter MilestoneAgreement.transferFunds(address,uint256,address)._tokenAddress (MilestoneAgreement.sol#1872) is not in mixedCase
Parameter MilestoneAgreement.verifySignature(bytes32,bytes,address,string)._message (MilestoneAgreement.sol#1891) is not in mixedCase
Parameter MilestoneAgreement.verifySignature(bytes32,bytes,address,string)._signature (MilestoneAgreement.sol#1892) is not in mixedCase
```

```
Parameter MilestoneAgreement.computeMerkleRoot(ISmartEmploymentAgreement.Milestone[])._milestones (MilestoneAgreement.sol#1921) is not in mixedCase
Parameter MilestoneAgreement.hashMilestone(ISmartEmploymentAgreement.Milestone)._milestone (MilestoneAgreement.sol#1944) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (MilestoneAgreement.sol#329)" inContext (MilestoneAgreement.sol#324-332)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable LinearAgreement.SELFIENT_MANAGER (MilestoneAgreement.sol#676) is too similar to LinearAgreement.constructor(address,address)._selfientManager (MilestoneAgreement.sol#686)
Variable SelfientManager._getAgreementContract(uint256)._agreementContract (MilestoneAgreement.sol#1420) is too similar to SelfientAdmin.agreementContracts (MilestoneAgreement.sol#957-959)
Variable SelfientManager._retrieveAgreement(uint256)._agreementContract (MilestoneAgreement.sol#1434) is too similar to SelfientAdmin.agreementContracts (MilestoneAgreement.sol#957-959)
Variable MilestoneAgreement.SELFIENT_MANAGER (MilestoneAgreement.sol#1604) is too similar to MilestoneAgreement.constructor(address,address)._selfientManager (MilestoneAgreement.sol#1616)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

ERC20 (MilestoneAgreement.sol#342-466) does not implement functions:
        - IERC20Metadata.decimals() (MilestoneAgreement.sol#339)
        - IERC20.getOwner() (MilestoneAgreement.sol#303)
        - IERC20Metadata.name() (MilestoneAgreement.sol#335)
        - IERC20Metadata.symbol() (MilestoneAgreement.sol#337)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

ERC20._name (MilestoneAgreement.sol#347) should be immutable
ERC20._symbol (MilestoneAgreement.sol#348) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
MilestoneAgreement.sol analyzed (25 contracts with 84 detectors), 147 result(s) found
```

Slither log >> MockUSDC.sol

```
Address.isContract(address) (MockUSDC.sol#95-102) uses assembly
        - INLINE ASM (MockUSDC.sol#98-100)
Address._functionCallWithValue(address,bytes,uint256,string) (MockUSDC.sol#141-163) uses assembly
        - INLINE ASM (MockUSDC.sol#155-158)
Strings.toString(uint256) (MockUSDC.sol#463-481) uses assembly
        - INLINE ASM (MockUSDC.sol#468-470)
        - INLINE ASM (MockUSDC.sol#473-475)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version0.8.17 (MockUSDC.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (MockUSDC.sol#104-109):
        - (success) = recipient.call{value: amount}() (MockUSDC.sol#107)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (MockUSDC.sol#141-163):
        - (success,returndata) = target.call{value: weiValue}(data) (MockUSDC.sol#149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Redundant expression "this (MockUSDC.sol#201)" inContext (MockUSDC.sol#195-204)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

```
MockUSDC (MockUSDC.sol#614-624) does not implement functions:
        - IERC20Metadata.decimals() (MockUSDC.sol#212)
        - IERC20.getOwner() (MockUSDC.sol#174)
        - IERC20Metadata.name() (MockUSDC.sol#208)
        - IERC20Metadata.symbol() (MockUSDC.sol#210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
MockUSDC.sol analyzed (18 contracts with 84 detectors), 38 result(s) found
```

Slither log >> SelfientAdmin.sol

```
SelfientAdmin.setFeewallet(address)._address (SelfientAdmin.sol#628) lacks a zero-check on :
                - feeWallet = _address (SelfientAdmin.sol#633)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Address.verifyCallResult(bool,bytes,string) (SelfientAdmin.sol#284-302) uses assembly
        - INLINE ASM (SelfientAdmin.sol#294-297)
Strings.toString(uint256) (SelfientAdmin.sol#374-392) uses assembly
        - INLINE ASM (SelfientAdmin.sol#379-381)
        - INLINE ASM (SelfientAdmin.sol#384-386)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version0.8.17 (SelfientAdmin.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (SelfientAdmin.sol#226-231):
        - (success) = recipient.call{value: amount}() (SelfientAdmin.sol#229)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (SelfientAdmin.sol#253-264):
        - (success,returndata) = target.call{value: value}(data) (SelfientAdmin.sol#262)
Low level call in Address.functionStaticCall(address,bytes,string) (SelfientAdmin.sol#270-279):
        - (success,returndata) = target.staticcall(data) (SelfientAdmin.sol#277)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter SelfientLibrary.checkZeroAddress(address,string)._address (SelfientAdmin.sol#10) is not in mixedCase
Function ISelfientAdmin.PERCENTAGE_PRECISION() (SelfientAdmin.sol#179) is not in mixedCase
Parameter SelfientAdmin.allowToken(address)._tokenContract (SelfientAdmin.sol#575) is not in mixedCase
Parameter SelfientAdmin.revokeToken(address)._tokenContract (SelfientAdmin.sol#586) is not in mixedCase
Parameter SelfientAdmin.validateToken(address)._address (SelfientAdmin.sol#596) is not in mixedCase
Parameter SelfientAdmin.registerSEA(address,uint8)._contractAddress (SelfientAdmin.sol#604) is not in mixedCase
Parameter SelfientAdmin.registerSEA(address,uint8)._contractId (SelfientAdmin.sol#605) is not in mixedCase
Parameter SelfientAdmin.setAgreementFee(uint16)._fee (SelfientAdmin.sol#618) is not in mixedCase
Parameter SelfientAdmin.setFeewallet(address)._address (SelfientAdmin.sol#628) is not in mixedCase
Parameter SelfientAdmin.setHirerAgreementFee(address,uint16,bool)._address (SelfientAdmin.sol#637) is not in mixedCase
Parameter SelfientAdmin.setHirerAgreementFee(address,uint16,bool)._fee (SelfientAdmin.sol#638) is not in mixedCase
Parameter SelfientAdmin.setHirerAgreementFee(address,uint16,bool)._disabled (SelfientAdmin.sol#639) is not in mixedCase
Parameter SelfientAdmin.distributeAgreementFee(address,uint256,address)._hirer (SelfientAdmin.sol#660) is not in mixedCase
Parameter SelfientAdmin.distributeAgreementFee(address,uint256,address)._agreementValue (SelfientAdmin.sol#661) is not in mixe
dCase
Parameter SelfientAdmin.distributeAgreementFee(address,uint256,address)._tokenAddress (SelfientAdmin.sol#662) is not in mixedC
ase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
SelfientAdmin.sol analyzed (13 contracts with 84 detectors), 46 result(s) found
```

Slither log >> SelfientManager.sol

```
SelfientAdmin.setFeewallet(address)._address (SelfientManager.sol#1056) lacks a zero-check on :
                - feeWallet = _address (SelfientManager.sol#1061)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SelfientManager.createAgreement(ISmartEmploymentAgreement.NewAgreement,bytes) (SelfientManager.sol#1262-1333):
        External calls:
        - depositAmount = agreementContract.createAgreement(newAgreementId,_agreement,_data) (SelfientManager.sol#1315-1319)
        - distributeAgreementFee(_agreement.hirer,depositAmount,_agreement.currency) (SelfientManager.sol#1321-1325)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (SelfientManager.sol#533)
                - (success,returndata) = target.call{value: weiValue}(data) (SelfientManager.sol#273)
                - _token.safeTransferFrom(_hirer,feeWallet,fee) (SelfientManager.sol#1105)
        External calls sending eth:
        - distributeAgreementFee(_agreement.hirer,depositAmount,_agreement.currency) (SelfientManager.sol#1321-1325)
                - (success,returndata) = target.call{value: weiValue}(data) (SelfientManager.sol#273)
        Event emitted after the call(s):
        - FeeDistributed(fee) (SelfientManager.sol#1104)
                - distributeAgreementFee(_agreement.hirer,depositAmount,_agreement.currency) (SelfientManager.sol#1321-1325)
Reentrancy in SelfientManager.depositFunds(uint256,ISmartEmploymentAgreement.Milestone,bytes) (SelfientManager.sol#1335-1366):
        External calls:
        - agreementContract.depositFunds(_agreementId,_milestone,_data) (SelfientManager.sol#1352)
        - distributeAgreementFee(agreement.hirer,_milestone.amount,agreement.currency) (SelfientManager.sol#1354-1358)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (SelfientManager.sol#533)
                - (success,returndata) = target.call{value: weiValue}(data) (SelfientManager.sol#273)
                - _token.safeTransferFrom(_hirer,feeWallet,fee) (SelfientManager.sol#1105)
        External calls sending eth:
        - distributeAgreementFee(agreement.hirer,_milestone.amount,agreement.currency) (SelfientManager.sol#1354-1358)
                - (success,returndata) = target.call{value: weiValue}(data) (SelfientManager.sol#273)
        Event emitted after the call(s):
        - FeeDistributed(fee) (SelfientManager.sol#1104)
                - distributeAgreementFee(agreement.hirer,_milestone.amount,agreement.currency) (SelfientManager.sol#1354-1358)
```

```
Reentrancy in LinearAgreement.terminateAgreement(uint256) (SelfientManager.sol#756-783):
        External calls:
        - withdrawFundsInternal(_agreementId,false) (SelfientManager.sol#768)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (SelfientManager.sol#533)
                - _token.safeTransfer(_to,_amount) (SelfientManager.sol#884)
                - (success,returndata) = target.call{value: weiValue}(data) (SelfientManager.sol#273)
        External calls sending eth:
        - withdrawFundsInternal(_agreementId,false) (SelfientManager.sol#768)
                - (success,returndata) = target.call{value: weiValue}(data) (SelfientManager.sol#273)
        Event emitted after the call(s):
        - FundsWithdrawn(_agreementId,hirerFunds) (SelfientManager.sol#776)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
LinearAgreement.terminateAgreement(uint256) (SelfientManager.sol#756-783) uses timestamp for comparisons
        Dangerous comparisons:
        - hirerFunds > 0 (SelfientManager.sol#775)
LinearAgreement.agreementStatus(uint256) (SelfientManager.sol#799-820) uses timestamp for comparisons
        Dangerous comparisons:
        - agreements[_agreementId].agreementId == 0 (SelfientManager.sol#802)
        - block.timestamp >= startDate && block.timestamp < agreementEndDate (SelfientManager.sol#815)
LinearAgreement.claimableValue(uint256) (SelfientManager.sol#822-856) uses timestamp for comparisons
        Dangerous comparisons:
        - agreements[_agreementId].agreementId == 0 (SelfientManager.sol#823)
        - terminatedAgreements[_agreementId] || lastClaim >= agreementEndDate || block.timestamp <= lastClaim || block.timesta
mp <= startDate (SelfientManager.sol#834-837)
        - lastClaim == 0 (SelfientManager.sol#844)
        - block.timestamp >= agreementEndDate (SelfientManager.sol#845)
        - block.timestamp >= agreementEndDate (SelfientManager.sol#851)
LinearAgreement.withdrawFundsInternal(uint256,bool) (SelfientManager.sol#887-909) uses timestamp for comparisons
        Dangerous comparisons:
        - withdrawalAmount == 0 && _revertOnEmptyWithdrawalAmount (SelfientManager.sol#893)
        - withdrawalAmount != 0 (SelfientManager.sol#902)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (SelfientManager.sol#219-226) uses assembly
        - INLINE ASM (SelfientManager.sol#222-224)
Address._functionCallWithValue(address,bytes,uint256,string) (SelfientManager.sol#265-287) uses assembly
        - INLINE ASM (SelfientManager.sol#279-282)
Strings.toString(uint256) (SelfientManager.sol#545-563) uses assembly
        - INLINE ASM (SelfientManager.sol#550-552)
        - INLINE ASM (SelfientManager.sol#555-557)
ECDSA.tryRecover(bytes32,bytes) (SelfientManager.sol#1119-1133) uses assembly
        - INLINE ASM (SelfientManager.sol#1124-1128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version0.8.17 (SelfientManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0
.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (SelfientManager.sol#228-233):
        - (success) = recipient.call{value: amount}() (SelfientManager.sol#231)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (SelfientManager.sol#265-287):
        - (success,returndata) = target.call{value: weiValue}(data) (SelfientManager.sol#273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter SelfientLibrary.checkZeroAddress(address,string)._address (SelfientManager.sol#10) is not in mixedCase
Parameter LinearAgreement.createAgreement(uint256,ISmartEmploymentAgreement.NewAgreement,bytes)._agreementId (SelfientManager.
sol#717) is not in mixedCase
Parameter LinearAgreement.createAgreement(uint256,ISmartEmploymentAgreement.NewAgreement,bytes)._agreement (SelfientManager.so
l#718) is not in mixedCase
Parameter LinearAgreement.terminateAgreement(uint256)._agreementId (SelfientManager.sol#757) is not in mixedCase
Parameter LinearAgreement.withdrawFunds(uint256)._agreementId (SelfientManager.sol#794) is not in mixedCase
Parameter LinearAgreement.agreementStatus(uint256)._agreementId (SelfientManager.sol#800) is not in mixedCase
Parameter LinearAgreement.claimableValue(uint256)._agreementId (SelfientManager.sol#822) is not in mixedCase
Parameter LinearAgreement.getTrimmedAgreementFields(uint256)._agreementId (SelfientManager.sol#859) is not in mixedCase
Parameter LinearAgreement.transferFunds(address,uint256,address)._to (SelfientManager.sol#878) is not in mixedCase
Parameter LinearAgreement.transferFunds(address,uint256,address)._amount (SelfientManager.sol#879) is not in mixedCase
Parameter LinearAgreement.transferFunds(address,uint256,address)._tokenAddress (SelfientManager.sol#880) is not in mixedCase
Parameter LinearAgreement.withdrawFundsInternal(uint256,bool)._agreementId (SelfientManager.sol#888) is not in mixedCase
Parameter LinearAgreement.withdrawFundsInternal(uint256,bool)._revertOnEmptyWithdrawalAmount (SelfientManager.sol#889) is not
in mixedCase
Function ISelfientAdmin.PERCENTAGE_PRECISION() (SelfientManager.sol#962) is not in mixedCase
Parameter SelfientAdmin.allowToken(address)._tokenContract (SelfientManager.sol#1008) is not in mixedCase
Parameter SelfientAdmin.revokeToken(address)._tokenContract (SelfientManager.sol#1018) is not in mixedCase
Parameter SelfientAdmin.validateToken(address)._address (SelfientManager.sol#1027) is not in mixedCase
```

```
Parameter SelfientManager.transferFunds(address,address,uint256,address)._from (SelfientManager.sol#1422) is not in mixedCase
Parameter SelfientManager.transferFunds(address,address,uint256,address)._to (SelfientManager.sol#1423) is not in mixedCase
Parameter SelfientManager.transferFunds(address,address,uint256,address)._amount (SelfientManager.sol#1424) is not in mixedCas
e
Parameter SelfientManager.transferFunds(address,address,uint256,address)._tokenAddress (SelfientManager.sol#1425) is not in mi
xedCase
Parameter SelfientManager.verifySignature(bytes32,bytes,address,string)._message (SelfientManager.sol#1433) is not in mixedCas
e
Parameter SelfientManager.verifySignature(bytes32,bytes,address,string)._signature (SelfientManager.sol#1434) is not in mixedC
ase
Parameter SelfientManager.verifySignature(bytes32,bytes,address,string)._signer (SelfientManager.sol#1435) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (SelfientManager.sol#325)" inContext (SelfientManager.sol#319-328)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable LinearAgreement.SELFIENT_MANAGER (SelfientManager.sol#698) is too similar to LinearAgreement.constructor(address,addr
ess)._selfientManager (SelfientManager.sol#708)
Variable SelfientManager._getAgreementContract(uint256)._agreementContract (SelfientManager.sol#1466) is too similar to Selfie
ntAdmin.agreementContracts (SelfientManager.sol#992-994)
Variable SelfientManager._retrieveAgreement(uint256)._agreementContract (SelfientManager.sol#1481) is too similar to SelfientA
dmin.agreementContracts (SelfientManager.sol#992-994)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

ERC20 (SelfientManager.sol#338-474) does not implement functions:
        - IERC20Metadata.decimals() (SelfientManager.sol#335)
        - IERC20.getOwner() (SelfientManager.sol#298)
        - IERC20Metadata.name() (SelfientManager.sol#331)
        - IERC20Metadata.symbol() (SelfientManager.sol#333)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

ERC20._name (SelfientManager.sol#345) should be immutable
ERC20._symbol (SelfientManager.sol#346) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
SelfientManager.sol analyzed (22 contracts with 84 detectors), 103 result(s) found
```

Solidity Static Analysis

LinearAgreement.sol

Check-effects-interaction:

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

more

Pos: 70:2:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 262:41:

Gas costs:

Gas requirement of function LinearAgreement.createAgreement 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: 61:2:

Constant/View/Pure functions:

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

more

Pos: 148:2:

Similar variable names:

LinearAgreement.createAgreement(uint256,struct ISmartEmploymentAgreement.NewAgreement,bytes) : Variables have very similar names "agreements" and "_agreement". Note: Modifiers are currently not considered by this static analysis.

Pos: 69:8:

Email: audit@EtherAuthority.io

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: 197:13:

MilestoneAgreement.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SelfientManager.terminateAgreement(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 214:2:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 494:4:

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: 107:24:

Gas costs:

Gas requirement of function MilestoneAgreement.createAgreement 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: 61:2:

For loop over dynamic array:

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

more

Pos: 87:4:

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: 435:4:

Constant/View/Pure functions:

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

more

Pos: 187:2:

Constant/View/Pure functions:

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

more

Pos: 187:2:

No return:

MilestoneAgreement._efficientHash(bytes32,bytes32): Defines a return type but never explicitly returns a value.

Pos: 490:2:

Similar variable names:

MilestoneAgreement.createAgreement(uint256,struct ISmartEmploymentAgreement.NewAgreement,bytes) : Variables have very similar names "agreements" and "_agreement". Note: Modifiers are currently not considered by this static analysis.
Pos: 112:6:

MockUSDC.sol

Gas costs:

Gas requirement of function MockUSDC.mint is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 26:2:

SelfientAdmin.sol

Constant/View/Pure functions:

ISelfientAdmin.setAgreementFee(uint16) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 147:2:

Constant/View/Pure functions:

SelfientAdmin.validateToken(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 86:2:

No return:

ISmartEmploymentAgreement.totalClaimed(uint256): Defines a return type but never explicitly returns a value.
Pos: 369:2:

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: 169:18:

SelfientManager.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SelfientManager.createAgreement(struct ISmartEmploymentAgreement.NewAgreement,bytes): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 70:2:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SelfientManager.terminateAgreement(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 214:2:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 262:41:

Gas costs:

Gas requirement of function SelfientManager.terminateAgreement 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: 101:2:

Gas costs:

Gas requirement of function SelfientManager.registerSEA 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: 93:2:

Constant/View/Pure functions:

SelfientManager.verifySignature(bytes32,bytes,address,string) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 256:2:

Similar variable names:

SelfientManager.createAgreement(struct ISmartEmploymentAgreement.NewAgreement,bytes) : Variables have very similar names "agreementContract" and "agreementContracts". Note: Modifiers are currently not considered by this static analysis.
Pos: 86:4:

Similar variable names:

SelfientManager.withdrawFunds(uint256) : Variables have very similar names "agreementContract" and "agreementContracts". Note: Modifiers are currently not considered by this static analysis.
Pos: 206:6:

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: 169:18:

Solhint Linter

LinearAgreement.sol

```
Compiler version 0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path
@openzeppelin/contracts/access/AccessControl.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:13
global import of path
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
Pos: 1:14
global import of path @openzeppelin/contracts/token/ERC20/IERC20.sol
is not allowed. Specify names to import individually or bind all
exports of the module into a name (import "path" as Name)
Pos: 1:15
global import of path @openzeppelin/contracts/token/ERC20/ERC20.sol
is not allowed. Specify names to import individually or bind all
exports of the module into a name (import "path" as Name)
Pos: 1:16
global import of path ISmartEmploymentAgreement.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:18
global import of path SelfientLibrary.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:19
global import of path SelfientManager.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:21
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 3:52
Avoid making time-based decisions in your business logic
Pos: 7:82
Avoid making time-based decisions in your business logic
Pos: 9:163
Avoid making time-based decisions in your business logic
Pos: 41:163
Avoid making time-based decisions in your business logic
Pos: 7:184
Avoid making time-based decisions in your business logic
Pos: 7:185
Avoid making time-based decisions in your business logic
Pos: 11:193
Avoid making time-based decisions in your business logic
```

```
Pos: 16:196
Avoid making time-based decisions in your business logic
Pos: 9:199
Avoid making time-based decisions in your business logic
Pos: 14:203
Avoid making time-based decisions in your business logic
Pos: 42:261
```

MilestoneAgreement.sol

```
Compiler version 0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path
@openzeppelin/contracts/access/AccessControl.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:13
global import of path
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
Pos: 1:14
global import of path @openzeppelin/contracts/utils/Address.sol is
not allowed. Specify names to import individually or bind all exports
of the module into a name (import "path" as Name)
Pos: 1:15
global import of path @openzeppelin/contracts/token/ERC20/ERC20.sol
is not allowed. Specify names to import individually or bind all
exports of the module into a name (import "path" as Name)
Pos: 1:16
global import of path
@openzeppelin/contracts/utils/cryptography/ECDSA.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:17
global import of path
@openzeppelin/contracts/utils/cryptography/MerkleProof.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
Pos: 1:18
global import of path ISmartEmploymentAgreement.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:20
global import of path SelfientLibrary.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:21
global import of path SelfientManager.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:22
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
```

```
Pos: 3:57
Avoid making time-based decisions in your business logic
Pos: 25:106
Avoid making time-based decisions in your business logic
Pos: 42:182
Avoid making time-based decisions in your business logic
Pos: 9:198
Avoid making time-based decisions in your business logic
Pos: 9:254
Avoid making time-based decisions in your business logic
Pos: 41:254
Avoid making time-based decisions in your business logic
Pos: 9:281
Avoid making time-based decisions in your business logic
Pos: 41:281
Avoid making time-based decisions in your business logic
Pos: 9:285
Avoid making time-based decisions in your business logic
Pos: 7:415
Avoid making time-based decisions in your business logic
Pos: 7:416
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 5:493
```

MockUSDC.sol

```
Compiler version 0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path
@openzeppelin/contracts/access/AccessControl.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:13
global import of path
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
Pos: 1:14
global import of path @openzeppelin/contracts/token/ERC20/IERC20.sol
is not allowed. Specify names to import individually or bind all
exports of the module into a name (import "path" as Name)
Pos: 1:15
global import of path @openzeppelin/contracts/token/ERC20/ERC20.sol
is not allowed. Specify names to import individually or bind all
exports of the module into a name (import "path" as Name)
Pos: 1:16
global import of path
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol is
not allowed. Specify names to import individually or bind all exports
of the module into a name (import "path" as Name)
Pos: 1:17
global import of path @openzeppelin/contracts/access/Ownable.sol is
not allowed. Specify names to import individually or bind all exports
of the module into a name (import "path" as Name)
```

```
Pos: 1:18
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 3:21
```

SelfientAdmin.sol

```
Compiler version 0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path
@openzeppelin/contracts/access/AccessControl.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:13
global import of path @openzeppelin/contracts/token/ERC20/IERC20.sol
is not allowed. Specify names to import individually or bind all
exports of the module into a name (import "path" as Name)
Pos: 1:14
global import of path
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
Pos: 1:15
global import of path @openzeppelin/contracts/utils/Address.sol is
not allowed. Specify names to import individually or bind all exports
of the module into a name (import "path" as Name)
Pos: 1:16
global import of path ISelfientAdmin.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:18
global import of path ISmartEmploymentAgreement.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:19
global import of path SelfientLibrary.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:20
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 3:54
```

SelfientManager.sol

```
Compiler version 0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path
@openzeppelin/contracts/access/AccessControl.sol is not allowed.
```

```
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:13
global import of path @openzeppelin/contracts/utils/Address.sol is
not allowed. Specify names to import individually or bind all exports
of the module into a name (import "path" as Name)
Pos: 1:14
global import of path
@openzeppelin/contracts/utils/cryptography/ECDSA.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:15
global import of path ISmartEmploymentAgreement.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:17
global import of path ISelfientManager.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:18
global import of path ISelfientAdmin.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:19
global import of path SelfientAdmin.sol is not allowed. Specify names
to import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:21
global import of path LinearAgreement.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:22
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 3:46
```

Software analysis result:

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

Email: audit@EtherAuthority.io