CERTIK

Security Assessment

Sister in Law Protocol

Apr 6th, 2021

Summary

This report has been prepared for Sister in Law Protocol smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in 29 findings that ranged from major to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Sister in Law Protocol
Description	a decentralized automatic investment platform
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/sil-finance/sister-in-law/tree/main/contracts
Commits	b21251f678494356b3f4787ad1ed2b437a6f1e8c

Audit Summary

Delivery Date	Apr 06, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	25
● Critical	0
● Major	2
● Minor	9
● Informational	14
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
DCR	DelegateCaller.sol	9ee26266e705c919c589c68eb9a9ee1b3aa0937a438e843203a03a3f5e09c620
MPD	MatchPairDelegateStable.sol	289b8d5f6545b9ddf1e1d47c060f6cb6761a4d332dc25d871e65efe12e6e423d
MPV	MatchPairDelegateV2.sol	f08b42366cf05df8b32433a93076bdb480617981df081ad0afd47e39bd74cb86
MAT	MatchPairDelegator.sol	33febd50d9b759753e2bcc19db41e67f6900bb089aea18eab09b2988716591b7
MPN	MatchPairNormalV2.sol	2ff06e33ad7c78a0f0c0731db54b29020c8a771df6fdff32b3dbc115bc5f0bad
MPS	MatchPairStable.sol	c066ee48c8a02ee9c807c8bf5adfcc47b778ca6b8bc197617316d6fb7ee07825
MSS	MatchPairStorageStable.sol	81dcce592ed59248227f364b31dc8b7522e35250aa808efe407bec183171753c
MSV	MatchPairStorageV2.sol	1ac73a8b584b98c5e28f794d264a3044dbe84ac339d4e0e2762b7c783ed5a5a9
PPL	PausePool.sol	0dc4b8e408094a54be3ac0c291afb19383f13fd886cff8ac06264ab80928378e
PCL	PriceChainLinkChecker.sol	0cbdbcea03fd38f1144e15d25090f2c57c10c8d7bc8ca4ad1d3f325bc58c0ad9
PSS	ProfitStrategySushiDelegate.sol	a8a3bc5c5165679f44111abba06ef785e7eddbe83691ad831683c1d391353b0f
PSP	ProfitStrategySushiProxy.sol	8cee312a3f87d28856f4378e3e442e5b7b6af0fa5ac4a113d65ec6d00ccce561
SMR	SilMaster.sol	ab9228fe27e98d7ccf1c57fdfbaccd88a7915a0690f65f9301463760d5ed50fc
STN	SilToken.sol	e8bb81f6d2bce4a668065bfa7d0d6a60599e3e91b8f22f74bd1f35035872b99a
SGD	StakeGatlingDelegate.sol	6aecfc605ad4fc337c7b84696095cbe05b6979a2d8db3fa663bf83571331407c
SGP	StakeGatlingProxy.sol	6556c54dd6afe9daeac91dd96ef7ef7c00f63daaa9d3d65f1d4d6814b20d20ee

ID	file	SHA256 Checksum
TLT	TrustList.sol	f1e04fce017ad2f2cc925e7414758e9ba23f6fb9d8055962723d19d9b9c82b69
IMP	interfaces/IMatchPair.sol	23bd990138f89d37f6516873af4f5cde7a2e27f50cf04ed423b917c82821b5a5
IMG	interfaces/IMigrateGatling.sol	1b03801d2316c5f2fd9b6b5a6910edd998ba05f001502a25d6a172a8e524a04c
IMR	interfaces/IMintRegulator.sol	88bb9bc2cbee2ce9e07a692ab611ef5caa525cd555491e660bc16cab81304dc8
IPS	interfaces/IPriceSafeChecker.sol	57102b9809de5bc74bc3da3603138cd9f926336f920af3ec9de1d6cfbb231ef3
IPR	interfaces/IProfitStrategy.sol	8c8385245607e282cc19aa0d895c701abd6c0a3601e9929b098cbbd539d726a8
IPO	interfaces/IProxyRegistry.sol	1102ee959106ed00bddc2354ecfe34cb121158a84b53bc72af57e1c98ebe6d67
ISG	interfaces/IStakeGatling.sol	575735d80f5b9315e1eb6cca13f3cb5be587ad3a92b925b98e30616aed99e5db
ISR	interfaces/IStakingRewards.sol	1b374ae0d4211ee41d31fc9701095393ae8072bcdc28757af5a811188b131a20
IUV	interfaces/IUniswapV2Router01.sol	89d0ecc779df20238c8c15b8ccc5155217002b6b5d1fdf1755019b85453a5395
IWE	interfaces/IWETH.sol	cd98a9720ebe90034290ffac4c21e4cda87e436fc2beb1b3062b6de768a1f3f0
MSD	stable_exp/MatchPairStableDelegateV2.sol	a657020737e013933ed814300d8886f70e90589beb32a07136d2b34331a5a6e8
GSE	storage/GatlingStorage.sol	43bade753943a5e7fecace2cb702b21281440367936ce8541788660f92502f57
SSS	storage/StrategySushiStorage.sol	0601a09f6bc9cd33d34ee1772f660337c75178346a287537f08595a0d7276376
ERC	utils/ERC1967Proxy.sol	b2e66037d9f7e8ee39af464aab5ef222d030f9a6283535153a7e9aa28d43f3f6
MCS	utils/MasterCaller.sol	8263815ff4c75317282a8a1f9f78dcdeae27495600f2d19ff97a68ad7935a9bb
PRO	utils/Proxy.sol	35004ef05576f2749a1fd7ae3dacbcd01e093c93433507b5f7a99fb28298db45

ID	file	SHA256 Checksum
QSS	utils/QueueStableStakesFuns.sol	9f1156104d8f0fe921e4d68aee78f5c3d1400c99bc002450c8c371da3328d90a

Centralized Risks

To initial setup project correctly, improve overall project quality, preserve the upgradability, the following functions, which are gov role, are adopted in the codebase:

- _upgradeTo() to update address of `_delegate` in smart contract ProfitStrategySushiProxy.sol.
- _upgradeTo() to update address of `_delegate` in smart contract StakeGatlingProxy.sol.
- implementation() to update delegate implementation in smart contract MatchPairNormalV2.sol.
- matchPairRegister() to update delegate implementation in smart contract SilMaster.sol.

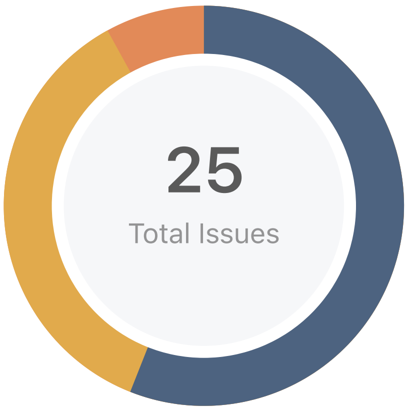
The advantage of the above functions in the codebase is that the client reserves the ability to adjust the project according to the runtime require to best serve the community. It is also worthy of note the potential drawbacks of these functions, which should be clearly stated through client's action/plan on how to prevent abusage of the these functionalities

To improve the trustworthiness of the project, any dynamic runtime updates in the project should be notified to the community. Any plan to invoke abovementioned functions should be also considered to move to the execution queue of Timelock contract.

Additionally, this protocol has external dependencies. All user deposits are immediately transferred to a third-party service (including SushiSwap). These third-party service contracts are not in the scope of this audit. The system should only be used if the service is appropriately trusted.

Also, all contracts under `sil-finance/sister-in-law/tree/main/contracts/uniswapv2` are not in the scope of this audit.

Findings



🟥	**Critical**	**0** (0.00%)
🟧	**Major**	**2** (8.00%)
🟨	**Minor**	**9** (36.00%)
🟦	**Informational**	**14** (56.00%)
🟩	**Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
MPN-1	Missing Important Checking in Function `setStakeGatling()`	Logical Issue	● Informational	ⓘ Acknowledged
MPV-1	Incorrect Naming Convention Utilization	Coding Style	● Informational	✓ Resolved
MPV-2	Local Variable Shadowing	Coding Style	● Minor	ⓘ Acknowledged
MPV-3	Code Redundancy	Coding Style	● Informational	✓ Resolved
MPV-4	Logic Issue of `index`	Logical Issue	● Major	✓ Resolved
MPV-5	Logic Issue of `updatePool`	Logical Issue	● Informational	✓ Resolved
MSD-1	Logic Issue of `index`	Logical Issue	● Major	✓ Resolved
MSD-2	Local Variable shadows function name	Gas Optimization	● Informational	ⓘ Acknowledged
MSD-3	Function state mutability can be restricted to `view`	Gas Optimization	● Informational	ⓘ Acknowledged
MSD-4	Redundant Codes of `MatchPairStableDelegateV2`	Coding Style	● Informational	ⓘ Acknowledged
MSD-5	Coding Style of Function `_burnLp()`	Coding Style	● Informational	ⓘ Acknowledged
MSV-1	State Variables That could be Declared Constant	Gas Optimization	● Informational	ⓘ Acknowledged
PCL-1	Code Redundancy	Coding Style	● Informational	✓ Resolved

ID	Title	Category	Severity	Status
SGD-1	Missing Emit Events	Gas Optimization	● Minor	Resolved
SGD-2	Need Validations on `setRouterPath()`	Logical Issue	● Minor	Resolved
SGP-1	Missing Emit Events	Gas Optimization	● Minor	Resolved
SMR-1	Incorrect Naming Convention Utilization	Coding Style	● Informational	Partially Resolved
SMR-2	Proper Usage of "public" and "external" Type	Gas Optimization	● Informational	Resolved
SMR-3	`Checks-effects-interactions` Pattern Not Used	Logical Issue	● Minor	Acknowledged
SMR-4	Transfer Not Safe	Data Flow	● Minor	Resolved
SMR-5	Missing Some Important Checks	Logical Issue	● Minor	Declined
SMR-6	Misleading Function Name	Logical Issue	● Informational	Resolved
SMR-7	Missing Important Checks in Function `add`	Logical Issue	● Minor	Acknowledged
SMR-8	Missing Important Checks in Function `depositETH`	Logical Issue	● Minor	Resolved
SSS-1	State Variables That could be Declared Constant	Gas Optimization	● Informational	Resolved

MPN-1 | Missing Important Checking in Function `setStakeGatling()`

Category	Severity	Location	Status
Logical Issue	● Informational	MatchPairNormalV2.sol: 22~25(MatchPairNormalV2)	ⓘ Acknowledged

Description

Match pair and its stakeGatling should set the same lptoken(pair). Better to check if `lpToken == _gatlinAddress.stakeLpPair()` when setting gatlingAddress.

Recommendation

Consider change the codes like:

```
    function setStakeGatling(address _gatlinAddress) public onlyOwner() {
        stakeGatling = IStakeGatling(_gatlinAddress);
        require(stakeGatling.stakeLpPair() == lpToken, "Lp pair not same");
    }
```

Alleviation

The recommendation was not taken into account, with the SIL team stating "They will ensure the data valid when setting gatlingAddress."

MPV-1 | Incorrect Naming Convention Utilization

Category	Severity	Location	Status
Coding Style	● Informational	MatchPairDelegateV2.sol: 42, 112	✓ Resolved

Description

Solidity defines a naming convention that should be followed. Refer to:
https://solidity.readthedocs.io/en/v0.7.1/style-guide.html#naming-conventions

Variables like `lpTokenAmoun1`, `totalTokenAmoun` contains typo in the variable name. In case the variable names shadow some function names, better to rename these getter functions starting with `getXXX`.

Recommendation

The recommendations outlined here are intended to improve the readability, and thus they are not rules, but rather guidelines to try and help convey the most information through the names of things.

Alleviation

The team heeded our advice and resolved this issue in commit a10f3e76fa5e3cafe1c1b3f692f879ee06aacb3b.

MPV-2 | Local Variable Shadowing

Category	Severity	Location	Status
Coding Style	● Minor	MatchPairDelegateV2.sol: 235	ⓘ Acknowledged

Description

Local variable `userPoint` shadows function `userPoint()`.

Recommendation

Consider to rename the local variable that shadows another component.

MPV-3 | Code Redundancy

Category	Severity	Location	Status
Coding Style	● Informational	MatchPairDelegateV2.sol: 86	Resolved

Description

Function `getPairAmount()` has two parameters that are not used.

Parameters `tokenA` and `tokenB` are not used.

Recommendation

Consider removing these two parameters.

Alleviation

The team heeded our advice and resolved this issue in commit
487cf869f468fcd5e2d8231a4d8f3654d8494634.

MPV-4 | Logic Issue of `index`

Category	Severity	Location	Status
Logical Issue	● Major	MatchPairDelegateV2.sol: 44, 219	✓ Resolved

Description

The `_index` value is used for control flow judgement in this contract. The aforementioned codes are assuming the value cannot be other than 0 or 1.

But this judgement logic is existing in many public functions in this contract. Callers can pass any input values.

Recommendation

Consider uniforming the judgement logic as below example:

```
 index == 0 /  index != 0
or
index == 1 /  index != 1
```

Alleviation

The team heeded our advice and resolved this issue in commit ab863fa219a17439f3c5370184984cd3b6f42e6a.

MPV-5 | Logic Issue of `updatePool`

Category	Severity	Location	Status
Logical Issue	● Informational	MatchPairDelegateV2.sol: 70	✓ Resolved

Description

It's meaningless to compare lower limits `minMintToken` and `minMinToken1` with pendingTokens. Actually minted amount of token is `amountA` and `amountB`.

Recommendation

Consider changing the codes like below:

```
function updatePool() private {


    (uint amountA, uint amountB) = getPairAmount( lpToken.token0(), lpToken.token1(),
pendingToken0, pendingToken1 );

    if( amountA > minMintToken0 && amountB > minMintToken1 ) {
        TransferHelper.safeTransfer(lpToken.token0(), address(lpToken), amountA);
        TransferHelper.safeTransfer(lpToken.token1(), address(lpToken), amountB);
        pendingToken0 = pendingToken0.sub(amountA);
        pendingToken1 = pendingToken1.sub(amountB);
        //mint LP
        uint liquidity = lpToken.mint(stakeGatling.lpStakeDst());
        //send Token to UniPair
        stakeGatling.stake(liquidity);
    }
}
```

Alleviation

The team heeded our advice and resolved this issue in commit 487cf869f468fcd5e2d8231a4d8f3654d8494634.

MSD-1 | Logic Issue of `index`

Category	Severity	Location	Status
Logical Issue	● Major	stable_exp/MatchPairStableDelegateV2.sol: 40~42, 204~206, 185~187	✓ Resolved

Description

The `_index` value is used for control flow judgement in this contract. The aforementioned codes are assuming the value cannot be other than 0 or 1.

But this judgement logic is existing in many public functions in this contract. Callers can pass any input values.

Recommendation

Consider uniforming the judgement logic as below example:

```
 index == 0 /  index != 0
or
index == 1 /  index != 1
```

Alleviation

The team heeded our advice and resolved this issue in commit ab863fa219a17439f3c5370184984cd3b6f42e6a.

MSD-2 | Local Variable shadows function name

Category	Severity	Location	Status
Gas Optimization	● Informational	stable_exp/MatchPairStableDelegateV2.sol: 386	ⓘ Acknowledged

Description

This declaration shadows an existing declaration.

```
 function userPoint(uint256 _index, address _user) public view returns (uint256) {
...
uint256 userPoint;
```

Recommendation

Consider to rename the local variable that shadows another component.

MSD-3 | Function state mutability can be restricted to `view`

Category	Severity	Location	Status
Gas Optimization	● Informational	stable_exp/MatchPairStableDelegateV2.sol: 91, 224, 305, 447	ⓘ Acknowledged

Description

Functions on the aforementioned lines can be restricted to `view`.

Recommendation

Consider to restrict the functions to `view`.

MSD-4 | Redundant Codes of `MatchPairStableDelegateV2`

Category	Severity	Location	Status
Coding Style	● Informational	stable_exp/MatchPairStableDelegateV2.sol: 24~29(MatchPairStableDelegateV2)	ⓘ Acknowledged

Description

This contract is used as delegator of `MatchPairStableV2`, so state variables could be managed only in the `MatchPairStableV2` contract.

Recommendation

Consider removing constructor of `MatchPairStableV2` contract.

MSD-5 | Coding Style of Function `_burnLp()`

Category	Severity	Location	Status
Coding Style	● Informational	stable_exp/MatchPairStableDelegateV2.sol: 202~207(MatchPairStableDelegateV2)	ⓘ Acknowledged

Description

In function `_burnLp()`, codes that checks if `_lpAmount > sentinelAmount` are annotated. Actually checkings in outer function can not replace checkings inside. Checkings outside this function are introduced to save gas but checkings inside keep the logic strong.

Recommendation

Consider unannotating the codes to keep codes lossely coupled.

MSV-1 | State Variables That could be Declared Constant

Category	Severity	Location	Status
Gas Optimization	● Informational	MatchPairStorageV2.sol: 28	ⓘ Acknowledged

Description

Constant state variables should be declared constant to save gas.

```
uint256 public sentinelAmount = 500;
```

Recommendation

Add constant attributes to state variables that never change. We recommend to change the codes like below examples:

```
uint256 public constant sentinelAmount = 500;
```

PCL-1 | Code Redundancy

Category	Severity	Location	Status
Coding Style	● Informational	PriceChainLinkChecker.sol: 39	✓ Resolved

Description

Unused local variable: roundID, startedAt, timeStamp, answeredInRound

```
function getLatestPrice() public view returns (uint256) {
        (
            uint80 roundID,
            int price,
            uint startedAt,
            uint timeStamp,
            uint80 answeredInRound
        ) = priceFeed.latestRoundData();
        return uint256(price);
    }
```

Recommendation

Consider to remove the redundant codes.

```
    function getLatestPrice() public view returns (uint256) {
        (
             ,
            int price,
             ,
             ,

        ) = priceFeed.latestRoundData();
        return uint256(price);
    }
```

Alleviation

The team heeded our advice and resolved this issue in commit a10f3e76fa5e3cafe1c1b3f692f879ee06aacb3b.

SGD-1 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	● Minor	StakeGatlingDelegate.sol: 29, 38, 166	✓ Resolved

Description

Several sensitive actions are defined without event declarations.

Examples:

Functions like : `constructor()` in the contract `StakeGatlingProxy`.

`setMatchPair()`,`setUpdatesRule()`, `setRouter()` in the contract `StakeGatlingDelegate`.

Recommendation

Consider adding events for sensitive actions, and emit them in the functions, for example:

```
    event Deployment(address indexed _pair,address indexed _delegate);
        constructor (address _pair, address _delegate) ERC1967Proxy(_delegate, '')
public {
        stakeLpPair = _pair;
        createAt = now;
        emit Deployment(_pair,_delegate);
    }
```

Alleviation

The team heeded our advice and resolved this issue in commit 487cf869f468fcd5e2d8231a4d8f3654d8494634.

SGD-2 | Need Validations on `setRouterPath()`

Category	Severity	Location	Status
Logical Issue	● Minor	StakeGatlingDelegate.sol: 64	✓ Resolved

Description

There are no validations in function `setRouterPath()`. In `UniswapRouter`, path is a list of erc20 token address that would be swapped in sequence. Therefore it is necessary to check if the first address and last address are the inputs and outputs expected.

Besides, there is no need to approve the allowance to `UniswapRouter` for all of the paths. Except `path[0]`, all tokens are swapped inside `UniswapPair`.

Recommendation

Consider to add a validation function to verify the paths set in routerPath0 and routerPath1 are in pairs.

Alleviation

The team heeded our advice and resolved this issue in commit c7930fef66555dc5fdffa4081ea246efba472983.

SGP-1 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	● Minor	StakeGatlingProxy.sol: 10	✓ Resolved

Description

Several sensitive actions are defined without event declarations.

Examples:

Functions like : `constructor()` in the contract `StakeGatlingProxy`.

`setMatchPair()`,`setUpdatesRule()`, `setRouter()` in the contract `StakeGatlingDelegate`.

Recommendation

Consider adding events for sensitive actions, and emit them in the functions, for example:

```
    event Deployment(address indexed _pair,address indexed _delegate);
        constructor (address _pair, address _delegate) ERC1967Proxy(_delegate, '')
public {
        stakeLpPair = _pair;
        createAt = now;
        emit Deployment(_pair,_delegate);
    }
```

Alleviation

The team heeded our advice and resolved this issue in commit 487cf869f468fcd5e2d8231a4d8f3654d8494634.

SMR-1 | Incorrect Naming Convention Utilization

Category	Severity	Location	Status
Coding Style	● Informational	SilMaster.sol: 58, 75, 104, 290	Partially Resolved

Description

Solidity defines a naming convention that should be followed. Refer to: https://solidity.readthedocs.io/en/v0.7.1/style-guide.html#naming-conventions

Parameter, variable shoud use mixedCase.

Examples:

Variables like: `devaddr`, `repurchaseaddr`, `ecosysaddraddr`,`bonus_multiplier` on the aforementioned lines.

Parameters like: `_devaddr`, `_repurchaseaddr`, `_ecosysaddraddr` , `_bonus_multiplier` on the aforementioned lines.

Better to keep the coding style consistent. Check the below example in contract `SilMaster`. On L294, `pool.totalDeposit1` can be replaced by `lpSupply1`, to keep consistent with L290.

```
        uint256 lpSupply0 = pool.totalDeposit0;
        uint256 lpSupply1 = pool.totalDeposit1;
        ...
        if(lpSupply0 > 0) {
L290        pool.accSilPerShare0 =
pool.accSilPerShare0.add(silReward.mul(1e12).div(lpSupply0).div(2));
        }
        // token1 side
        if(lpSupply1 > 0) {
L294        pool.accSilPerShare1 =
pool.accSilPerShare1.add(silReward.mul(1e12).div(pool.totalDeposit1).div(2));
        }
```

Recommendation

The recommendations outlined here are intended to improve the readability, and thus they are not rules, but rather guidelines to try and help convey the most information through the names of things.

Alleviation

The team heeded our advice and partially resolved this issue in commit 487cf869f468fcd5e2d8231a4d8f3654d8494634.

SMR-2 | Proper Usage of "public" and "external" Type

Category	Severity	Location	Status
Gas Optimization	● Informational	SilMaster.sol: 116, 160, 190, 208, 212, 367, 432, 490, 560, 570	✓ Resolved

Description

"public" functions that are never called by the contract could be declared "external" . When the inputs are arrays "external" functions are more efficient than "public" functions.

Examples:

Functions `initSetting()`, `setMintRegulator()`, `setMintRegulator()`, `updateSilPerBlock()`, `add()`, `holdWhaleSpear()`, `set()`, `depositEth()`, `withdrawToken()`, `withdrawSil()`, `dev()`, `repurchase()` on the aforementioned lines.

Recommendation

Consider using the "external" attribute for functions never called from the contract.

Alleviation

The team heeded our advice and resolved this issue in commit 487cf869f468fcd5e2d8231a4d8f3654d8494634.

SMR-3 | `Checks-effects-interactions` Pattern Not Used

Category	Severity	Location	Status
Logical Issue	● Minor	SilMaster.sol: 367	ⓘ Acknowledged

Description

During `depositEth()` function call state variables for balance are changed after transfers are done. This might lead to reentrancy issue. The order of external call/transfer and storage manipulation must follow checks-effects-interactions pattern.

Recommendation

It is recommended to follow checks-effects-interactions pattern for cases like this. It shields public functions from re-entrancy attacks. It's always a good practice to follow this pattern. `checks-effects-interactions` pattern also applies to ERC20 tokens as they can inform the recipient of a transfer in certain implementations.

Reference: https://docs.soliditylang.org/en/develop/security-considerations.html?highlight=check-effects%23use-the-checks-effects-interactions-pattern

SMR-4 | Transfer Not Safe

Category	Severity	Location	Status
Data Flow	● Minor	SilMaster.sol: 515	✓ Resolved

Description

During `safeSilTransfer()` function call , return value by sil.transfer() is not validated. Hence this function is not doing a safe transferring.

Recommendation

Consider to change the code as below example:

```
function safeSilTransfer(address _to, uint256 _amount) internal {
    uint256 silBal = sil.balanceOf(address(this));
    if (_amount > silBal) {
        require(sil.transfer(_to, silBal),'transfer failed');
    } else {
        require(sil.transfer(_to, _amount),'transfer failed');
    }
}
```

Alleviation

The team heeded our advice and resolved this issue in commit 487cf869f468fcd5e2d8231a4d8f3654d8494634.

SMR-5 | Missing Some Important Checks

Category	Severity	Location	Status
Logical Issue	● Minor	SilMaster.sol: 558, 563, 568	ⓧ Declined

Description

Functions `dev()`, `ecosys()`, `repurchase()` on the aforementioned lines are missing parameter validations.

If these functions were called by mistake, there is no way to recover.

Recommendation

We recommend to change the codes as below, so owner can recover mistakes.

```
function dev(address _devaddr) public {
    require(msg.sender == devaddr || msg.sender == owner, "dev: wut?");
    devaddr = _devaddr;
}
```

Alleviation

The recommendation was not taken into account, with the SIL team stating "This change will lead to excessive admin capability."

SMR-6 | Misleading Function Name

Category	Severity	Location	Status
Logical Issue	● Informational	SilMaster.sol: 151	✓ Resolved

Description

Function `setMintRegulator(uint, uint)` may use an improper function name that shadows existing function `setMintRegulator(address)` in the same contract.

The functionality is to control max acceptable amount when performing a `deposit` action incase `whaleSpear` is true. So it's nothing to do with `mint`.

Recommendation

Consider using a proper function name.

Alleviation

The team heeded our advice and resolved this issue in commit 487cf869f468fcd5e2d8231a4d8f3654d8494634.

SMR-7 | Missing Important Checks in Function `add`

Category	Severity	Location	Status
Logical Issue	● Minor	SilMaster.sol: 190	ⓘ Acknowledged

Description

Rewards will be messed up if same LP token is added more than once, it is necessary to check if same `matchPair` is already existing.

In another way, new pool could be set to 'paused', and resume it after you checked.

Recommendation

Consider adding redundancy checkings for match pair.

SMR-8 | Missing Important Checks in Function `depositETH`

Category	Severity	Location	Status
Logical Issue	● Minor	SilMaster.sol: 367	✓ Resolved

Description

Users have to proactively transfer a certain amount of ETH to call this function.

Parameters of this function depositEth() should be validated, especially _index.

When the token indexed by index in the pool is not ETH and also for some reason the ERC20 token is successfully transferred to deposit which means this call succeed, the eth user paid is no where to get back.

Of course the front-end service will restrict the inputs, but this function is public and can be called by ABI code. Plus it is better to do complete error handling in the contract.

Recommendation

Consider adding checking that if current token is ETH.

Alleviation

The team heeded our advice and resolved this issue in commit 487cf869f468fcd5e2d8231a4d8f3654d8494634.

SSS-1 | State Variables That could be Declared Constant

Category	Severity	Location	Status
Gas Optimization	● Informational	storage/StrategySushiStorage.sol: 10~14, 14	Resolved

Description

Constant state variables should be declared constant to save gas.

```
   address public stakeRewards = 0x8184b47518Fef40ad5E03EbDE2f6d6bde2FA1B33 ;//=
0xc2EdaD668740f1aA35E4D8f227fB8E17dcA888Cd;
   address public earnTokenAddr = 0x1A63bBB6E16f7Fc7D34817496985757CD550c2c0  ;//=
0x6B3595068778DD592e39A122f4f5a5cF09C90fE2;
```

Recommendation

Add constant attributes to state variables that never change. We recommend to change the codes like below examples:

```
   address public constant stakeRewards = 0x8184b47518Fef40ad5E03EbDE2f6d6bde2FA1B33
;//= 0xc2EdaD668740f1aA35E4D8f227fB8E17dcA888Cd;
   address public constant earnTokenAddr = 0x1A63bBB6E16f7Fc7D34817496985757CD550c2c0
;//= 0x6B3595068778DD592e39A122f4f5a5cF09C90fE2;
```

Alleviation

The team heeded our advice and resolved this issue in commit 487cf869f468fcd5e2d8231a4d8f3654d8494634.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete .

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Disclaimer

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we’re able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.