SOLIDIFIED

Audit Report for Origin Protocol - December 17, 2020

Summary

Audit Report prepared by Solidified covering the Origin protocol (excluding staking and compensation contract).

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below in various rounds. The final debrief took place on December 16, 2020, and the results are presented here.

Audited Files

The audit covers the Origin protocol, excluding the staking compensation smart contracts, which is covered by a separate audit report. The following contracts are in scope of the audit:

```
contracts
├── governance
│   ├── Governable.sol
│   ├── Governor.sol
│   └── InitializableGovernable.sol
├── interfaces
│   ├── IBasicToken.sol
│   ├── IEthUsdOracle.sol
│   ├── IMinMaxOracle.sol
│   ├── IPriceOracle.sol
│   ├── IStrategy.sol
│   ├── ITimelock.sol
│   ├── IVault.sol
│   └── uniswap
│       ├── IUniswapV2Pair.sol
│       └── IUniswapV2Router02.sol
├── liquidity
│   └── LiquidityReward.sol
├── oracle
│   ├── AggregatorV3Interface.sol
│   ├── ChainlinkOracle.sol
│   ├── MixOracle.sol
│   └── UniswapLib.sol
├── proxies
│   ├── InitializeGovernedUpgradeabilityProxy.sol
```

```
│   └── Proxies.sol
├── staking
│   └── SingleAssetStaking.sol
├── strategies
│   ├── AaveStrategy.sol
│   ├── CompoundStrategy.sol
│   ├── IAave.sol
│   ├── ICRVMinter.sol
│   ├── ICompound.sol
│   ├── ICurveGauge.sol
│   ├── ICurvePool.sol
│   └── ThreePoolStrategy.sol
├── timelock
│   ├── MinuteTimelock.sol
│   └── Timelock.sol
├── token
│   └── OUSD.sol
├── utils
│   ├── Helpers.sol
│   ├── InitializableAbstractStrategy.sol
│   ├── InitializableERC20Detailed.sol
│   └── StableMath.sol
└── vault
    ├── Vault.sol
    ├── VaultAdmin.sol
    ├── VaultCore.sol
    ├── VaultInitializer.sol
    └── VaultStorage.sol
```

Supplied in the following source code repositories:

https://github.com/OriginProtocol/origin-dollar

commit number `0936691ee0d81f53be9f50a080a0a8f5ead2ed26`

Intended Behavior

The smart contract implements a stablecoin protocol with associated liquidity, vault, reward, staking, and oracle components.

SOLIDIFIED

Audit Report for Origin Protocol - December 17, 2020

Executive Summary

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	High	-

Issues Found

Solidified found that the Origin protocol contracts contain no critical issues, no major issues, and 3 minor issues, in addition to 2 informational notes.

We recommend all issues are amended, while the notes are up to the team's discretion, as it refers to best practices.

Issue #	Description	Severity	Status
1	LiquidityReward.sol: New campaigns can overwrite active campaigns	Minor	Non-issue
2	Duplicate event emission	Minor	Resolved
3	Ambiguous admin transaction hash	Minor	Resolved
4	Incorrect NatSpec comments in ThreePoolStrategy.sol and InitializableAbstractStrategy.sol	Note	Resolved
5	Unclear Usage of Oracles	Note	Acknowledged

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

1. LiquidityReward.sol: New campaigns can overwrite active campaigns

It is possible to call `startCampaign()` even when a campaign is already active, without having to call `stopCampaign()` first. This would have the effect of overwriting the active campaign's parameters. Whilst this action can only be performed by the governance role, it may be unintentional.

Recommendation
Consider using a lock to enforce a campaign to be explicitly stopped before being replaced by a new campaign.

Update
The team has clarified that this behavior is intentional, in order to allow for updating a campaigns parameter. **The issue has, therefore, been marked as "non-issue".**

2. Duplicate event emission

In `VaultCore.sol`, function `mintMultiple()` the statement emit `Mint(msg.sender, priceAdjustedTotal)` is executed twice, leading to duplicate events.

Recommendation
Remove duplicate.

Update
Fixed.

3. Ambiguous admin transaction hash

In `TimeLock.sol` and `MinuteTimelock.sol` the transaction hash is calculated as:

```
bytes32 txHash = keccak256(
      abi.encode(target, value, signature, data, eta)
);
```

Since both consecutive values signature and data could be of any size, there could be many variations of signature and data values which would evaluate to the same hash.
This fix might be more important for the `MinuteTimelock.sol` contract because queued transactions can be executed by anyone, not only admin.

Recommendation
Hash both or at least one of the signature and hash values before calculating the transaction hash:

```
bytes32 txHash = keccak256(
      abi.encode(target, value, signature, keccak256(bytes(data)), eta)
);
```

Notes

4. Incorrect NatSpec comments in ThreePoolStrategy.sol and InitializableAbstractStrategy.sol

The `deposit()` and `withdraw()` functions in `ThreePoolStrategy.sol` and `InitializableAbstractStrategy.sol` do not return the amount deposited or withdrawn as indicated in the NatSpec documentation.

Recommendation
Adapt behavior to documentation or change documentation.

Update

Fixed.

5. Unclear Usage of Oracles

It is not clear how price oracles are used in the protocol. The library `UniswapLib.sol` is not used for any of the oracles in the current version of the protocol, leaving a single Chainlink price feed. However, the `MixOracle.sol` contract seems to suggest that the lowest price from a number of feeds should be used. The current codebase makes it hard to measure resistance to Oracle manipulation of malfunctioning.

Recommendation
Consider documenting oracle usage and cleaning up the codebase.

Update
The team has clarified that in addition to the included Oracles, the existing Open Price Feed oracle will be used. This will be documented.

6. Staking contract may run out of funds

There is no way to control the amount `USER_STAKE_TYPE` staked.
The existing `SingleAssetStaking.sol` contract's token balance can be "consumed" by rewards for `USER_STAKE_TYPE` stakes.
It could interfere with the intended `preApprovedStake()` functionality because both operations (the user stake and pre-approved stake) use the same token's pool (the balance of the `SingleAssetStaking.sol` contract).

Recommendation
Consider adding a governor-configurable maximum total of staked amount of `USER_STAKE_TYPE` type stakes.

Audit Report for Origin Protocol - December 17, 2020

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Origin Protocol or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.