SOLIDIFIED

Audit Report for Origin - May 5, 2022

Summary

Audit Report prepared by Solidified covering a number of Origin components added to the core protocol, peripherals and NFT launcher.

The OGV token is a standard ERC20 token based on Open Zeppelin's implementation, with no customization.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on April 25, 2022, and the results are presented here.

Audited Files

The source code has been supplied in several source code repositories:

OUSD governance
Repo: https://github.com/OriginProtocol/ousd-governance
Commit hash: `501414bdebdcce024091c5cf5f63aa2b5bdb9838`
Files:
- contracts/*.sol

Story NFT generative
Repo: https://github.com/OriginProtocol/nft-launchpad
Commit hash: `3b54b2baa333373ffb4cc83119a1d720d06cd806`
Files:
- contracts/contracts/nft/OriginERC721a_v1.sol

Wrapped OUSD
Repo: https://github.com/OriginProtocol/origin-dollar
Commit hash: `30564e25d640749af447fb9e5d7c4fb00c37779f`
Files:
- contracts/contracts/token/WrappedOusd.sol
- contracts/lib/openzeppelin/contracts/token/ERC20/extensions/ERC4626.sol

OUSD harvester
Repo: https://github.com/OriginProtocol/origin-dollar

Commit hash: `30564e25d640749af447fb9e5d7c4fb00c37779f`
Files:
- contracts/contracts/harvest/Harvester.sol

OUSD dripper
Repo: https://github.com/OriginProtocol/origin-dollar
Commit hash: `30564e25d640749af447fb9e5d7c4fb00c37779f`
Files:
- contracts/contracts/harvest/Dripper.sol

Intended Behavior

The audited codebase implements a number of additions to two core Origin products, the Origin rebasing stablecoin protocol and the NFT launcher, including a governance solution.

Findings

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	Low	-
Code readability and clarity	High	-
Level of Documentation	Medium	-
Test Coverage	High	-

Issues Found

Solidified found that the Origin contracts contain no critical issues, no major issues, 3 minor issues, and 8 informational notes.

We recommend issues are amended, while informational notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	NFTLaunchpad: OriginERC721a_v1.sol: Excess ETH for minting is not refunded	Minor	Acknowledged
2	Harvester: Harvester.sol: Swap and harvest might fail if the number of strategies and reward addresses grow very large	Minor	Acknowledged
3	NFTLaunchpad: OriginERC721a_v1.sol: Minter signatures can be replayed several times by user	Minor	Acknowledged
4	NFTLaunchpad: OriginERC721a_v1.sol: Consider using ERC-712 for signatures	Note	-
5	Consider restricting version pragma to later compiler versions	Note	-
6	Governance: VoteLockerCurve.sol: Users can lock up 0 tokens	Note	-
7	Harvester: Harvester.sol: Unused Variable and unnecessary external call	Note	-
8	Harvester: Harvester.sol: Comments for harvesting do not match implementation	Note	-
9	Dripper: Dripper.sol: Unnecessary cast	Note	-
10	Dripper: Dripper.sol: Unnecessary external call	Note	-

Critical Issues

No critical issues have been found

Major Issues

No major issues have been found

Minor Issues

1. NFTLaunchpad: OriginERC721a_v1.sol: Excess ETH for minting is not refunded

The function `mint()` accepts payment in ETH for the minting of an NFT. However, the users may overpay and no refund/change is given. This may lead to the user overpaying accidentally.

Recommendation

Consider sending excess ETH back to the caller.

2. Harvester: Harvester.sol: Swap and harvest might fail if the number of strategies and reward addresses grow very large

The functions `harvest()` and `swap()` cause nested iterations over dynamic data structures. These data structures might grow too large if too many strategy and reward accounts are added, causing the iterations to hit the block gas limit and revert. Whilst this is unlikely to occur, the impact could be locked rewards.

Recommendation

Consider adding paging support or ensure that the number of strategies and reward accounts is limited.

3. NFTLaunchpad: `OriginERC721a_v1.sol`: Minter signatures can be replayed several times by user

---

The function `mint()` requires a signature by a minter admin to be submitted. However, there is no nonce to make messages unique, meaning that users can re-submit signatures several times. Whilst the number of tokens per user that can be minted is limited by the `mintLimit` parameter, a signature that authorized a smaller mint can still be re-used, allowing users to mint tokens that should not yet be issued.

Recommendation
Consider using a nonce for signature uniqueness or ensuring that the contract is only ever used to mint all tokens in one go.

Team Reply
The team states that allowing minting with a single signature until `mintLimit` is reached is intended functionality.

Informational Notes

4. NFTLaunchpad: `OriginERC721a_v1.sol`: Consider using ERC-712 for signatures

The function `mint()` requires a signature by the minter to be submitted and resolved. Using ERC-712 would allow for human readable signature messages and simplify the signature validation code due to existing reference implementations.

Recommendation
Consider using the ERC-712 implementation provided by OpenZeppellin.

5. Consider restricting version pragma to later compiler versions

The contracts for some of the components allow solidity compiler versions `0.8.0` or higher, whereas some require compiler version `0.8.4`. However, the recent Solidity releases include several bug fixes which are missing from the earlier versions allowed.

Recommendation
Consider updating the compiler version to `0.8.4` or greater.

6. Governance: VoteLockerCurve.sol: Users can lock up 0 tokens

The contract allows users to lock up 0 tokens. This has no impact on security but may create undesired data entries and event emissions.

Recommendation
Consider disallowing the staking of 0 tokens.

7. Harvester: Harvester.sol: Unused Variable and unnecessary external call

The function `_harvest()` makes an unnecessary call to `strategy.getRewardTokenAddresses()` to initialize an unused variable `rewardTokenAddresses`.

Recommendation
Consider removing the unnecessary call and unused variable.

8. Harvester: Harvester.sol: Comments for harvesting do not match implementation

The comment for function `harvest()` states that this function should be callable by the vault. However, the implementation uses the `onlyGovernor` modifier. The `onlyVaultOrGovenor` modifier might be intended for this.

Recommendation
Consider reviewing the implementation and alter if necessary.

9. Dripper: Dripper.sol: Unnecessary cast

The cast to `uint192` in line 91 is unnecessary as `dripDuration` is of type `uint256`.

Recommendation
Consider removing the cast.

10. Dripper: Dripper.sol: Unnecessary external call

The call on line 92 for an external call to `collect()` could be replaced by a call to the internal `_collect()`.

Recommendation
Consider removing the external call.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Origin 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.

Oak Security GmbH