SOLIDIFIED

Audit Report for Reflexer Labs - January 26, 2021

Summary

Audit Report prepared by Solidified covering the Reflexer Finance incentives and rate calculation smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on January 22, 2021, and the results are presented here.

Fixes have been provided and are reflected in this audit report.

Audited Files

The contracts audited were supplied in five private source code repository and are limited to the following scope:

GEB Incentives (whole repo, commit: `761468a14c85170e8e6ec914c3aaa66d3dc3ed37`)
https://github.com/reflexer-labs/geb-incentives

GebProxyIncentivesActions -
https://github.com/reflexer-labs/geb-proxy-actions/blob/93f42e6b7f78730e3ebab3433884ddaa3a4f5ec5/src/GebProxyActions.sol#L1350
DSDelegateToken -
https://github.com/reflexer-labs/ds-token/blob/553626ee2785a3342e8263d1eace36d1256a5141/src/delegate.sol#L1
PIRawPerSecondCalculator -
https://github.com/reflexer-labs/geb-rrfm-calculators/blob/785b3aa33bdf43f1b35bb27e1ef28729db76be9f/src/calculator/PIRawPerSecondCalculator.sol#L1
RateSetter -
https://github.com/reflexer-labs/geb-rrfm-rate-setter/blob/304a599f794c547083c29a090960683d8a0aa70c/src/RateSetter.sol

Intended Behavior

The contracts audited implement the Reflexer incentive contract, an ERC20 token implementation with additional support for signature delegation, the Reflexer rate calculation, and rate setter components, and a UI proxy contract aimed at bundling operations into single transactions.

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	High	The code is relatively complex due to the nature of the project.
Code readability and clarity	Medium	Code readability is generally good but slightly hampered by using different coding style conventions, which is partially due to code reuse from different projects.
Level of Documentation	High	-
Test Coverage	High	The team submitted unit tests which provide good coverage for the audit. In addition, the mathematical algorithms are covered with fuzz testing.

SOLIDIFIED

Audit Report for Reflexer Labs - January 26, 2021

Issues Found

Solidified found that the Reflexer contracts contain no critical issue, no major issue, 2 minor issues, in addition to 4 informational notes.

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

Issue #	Description	Severity	Status
1	GebProxyActions.sol: Inconsistencies in leftover token/ETH transfers	Minor	Non-issue
2	RateSetterMath.sol: Edge Case not Covered by Overflow Protection	Minor	Resolved
3	delegate.sol: Malleable signatures accepted	Note	-
4	delegate.sol: Unnecessary return statement	Note	-
5	Possible Gas Savings	Note	-
6	StakingRewardsFactory.sol: Leftover Uniswap Comment	Note	Resolved

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

1. GebProxyActions.sol: Inconsistencies in leftover token/ETH transfers

Most functions in this contract that receive Ether and/or tokens return leftover change to the caller. However, the behavior in the case of tokens to be returned differs from the behavior of ETH being returned. Token transfer that fails causes the transaction to revert because of the underlying token implementation. However, the result of an Ether transfer is not checked.

Recommendation
Consider unifying the criterium for both cases.

Update
This is intended behavior.
Team reply: “The ETH transfers not being checked is intended.
We expect the bulk of users using an EOA, with a minority using multisigs to manage their Safes. With EOAs and the standard multisig these transfers should never fail.
There is no impact in the unlikely case the transfer fails (i.e.: user is using some sort of smart-contract wallet that does not accepts ETH transfers), the ETH balance will remain in the user owned proxy, and the user can either directly transfer it to somewhere else or transfer the proxy ownership to an EOA or standard multisig and then withdraw it.”

2. RateSetterMath.sol: Edge Case not Covered by Overflow Protection

The function function `multiply(int x, int y)` does not handle a special overflow case when `x=-1` and `y= -2**255 (MIN_INT)`.

Recommendation
Protect against this edge case.

Update
Resolved by removing unused function.

Notes

3. delegate.sol: Malleable signatures accepted

The `delegateBySig()` function uses the built-in `ecrecover()`. This function still allows malleable signatures for backward compatibility reasons. Signatures that have an `s` value larger than `0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0` are usually rejected for Ethereum address post EIP-2.

Recommendation
Consider rejecting signatures with `s` values in the upper ranges, even though it may not be a security issue in this case.

4. delegate.sol: Unnecessary return statement

The functions `delegate()` and `delegateBySig()` do not return anything according to the signature. Also, the internal function `_delegate()` does not return anything as well.

```
function delegate(address delegatee) public {
            return _delegate(msg.sender, delegatee);
}
```

Recommendation

Remove the return keyword or set a proper return value.

5. Possible Gas Savings

There are some gas optimizations that can be performed:

1. Use the proxy pattern to lower deployment cost of `stakingRewards`
2. Keeping error messages under 32 bytes also saves some gas
3. The `domainSeparator` used in `delegateBySig()` could be calculated in the constructor to avoid repeated invocations

Recommendation
If gas usage is a concern, consider implementing some of these recommendations.

7. StakingRewardsFactory.sol: Leftover Uniswap Comment

The comment above `notifyRewardAmount()` is a leftover from Uniswap and not applicable to this project.

Recommendation
Remove or correct the comment.

Update
Resolved

Disclaimer

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