OpenZeppelin | security

Origin Sonic Staking Audit

February 14, 2025

Table of Contents

Summary

Type	DeFi
Timeline	From 2025-2-10 To 2025-2-13
Languages	Solidity

Total Issues	8 (0 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	1 (0 resolved)
Low Severity Issues	3 (0 resolved)
Notes & Additional Information	4 (0 resolved)

Scope

We audited the OriginProtocol/origin-dollar repository, reviewing the code changes introduced in pull request #2335 at commit 097f3f3.

In scope were the following files:

```
contracts
└── contracts
    └── strategies
        └── sonic
            ├── SonicStakingStrategy.sol
            └── SonicValidatorDelegator.sol
```

System Overview

Origin is introducing a staking strategy tailored for the efficient staking and delegation of Sonic's native Sonic (S) token. This approach leverages the Special Fee Contract (SFC), a component that orchestrates a network of validators and their respective delegations, ensuring a seamless and secure staking process. At the heart of this strategy is the ability for users to engage with a dedicated vault by depositing the wrapped version of the native S token (wS). Once the wS tokens are allocated to the strategy, they undergo an unwrapping back into the native S token. The strategy deposits these S tokens into the SFC, from where it strategically delegates them to a carefully selected group of validators. This delegation is not arbitrary, as they are chosen by the Origin governance, which ensures that only the most reliable and efficient validators are chosen to maximize the staking benefits.

The strategy begins with the undelegation of the S tokens from a chosen Sonic validator. Following the SFC two-week waiting period, the strategy facilitates the withdrawal of the S tokens, while also offering the flexibility to either restake rewards for compounded returns or collect them back into the vault for other potential uses. A notable feature of this strategy is the empowerment of specific roles, namely the Registrator or Strategist, with the authority to delegate tokens to validators. This ensures that the delegation process remains streamlined and aligned with the strategic objectives of the Origin governance.

In essence, Origin's new staking strategy for Sonic's S token represents a comprehensive and user-friendly approach to staking. It not only simplifies the process of staking and delegation through its integration with the SFC, but also offers users flexibility and control over their staking rewards, all while ensuring that the selection of validators is governed by a transparent and democratic process.

The primary objective of the audit was to evaluate the integration and functionality of the staking strategy with both the Origin Sonic Vault and the SFC mechanisms. This comprehensive examination was aimed at verifying the critical aspects of the staking process: the delegation, undelegation, reward restaking, reward collection, and withdrawal of the S tokens, ensuring that each step adheres to the designed protocols and security standards.

Trust Assumptions

In the context of evaluating the staking strategy, some trust assumptions have been made to streamline the focus of the analysis and ensure that the review is conducted within a defined scope. These assumptions pertain to the operational integrity and functionality of the SFC and the Origin vault. By assuming that both of these components function properly, the audit can concentrate on the integration and performance of the staking strategy itself, instead of the underlying mechanisms of these systems.

The SFC contract is assumed to accurately record and execute delegation and undelegation requests without errors, ensuring that staked tokens are correctly allocated to validators and can be retrieved following the established protocols. Secondly, it is assumed that it reliably calculates and distributes staking rewards to participants based on the tokens staked and the duration of staking, adhering to the predefined reward mechanisms.

The Origin vault is integral to the staking strategy, serving as the initial point of entry for users' assets and facilitating their participation in the staking process. Thus, it is assumed the vault correctly handles the allocation of the wrapped S tokens ensuring that users' assets are accurately processed and accounted for throughout their lifecycle within the staking strategy.

Privileged Roles

There are several functions that require their caller to have a privileged role or can only be called by the vault. Within the staking strategy, there are three key roles: Registrator, Strategist, and Governance.

Registrator and Strategist

The Registrator and Strategist roles are designated by the governance of the Origin protocol and are able to do the following:

- Undelegation of S tokens from a specific Sonic validator via the `undelegate` function, thereby starting the withdrawal process.
- Finalize the withdrawal process with the `withdrawFromSFC` function.
- Collection of rewards using the `collectRewards` function.

- Lastly, setting the default validator ID that would be delegated to upon deposit via the `setDefaultValidatorId` function.

It is important to note that it is assumed that the holders of the Registrator and Strategist roles will act in the best interest of the protocol, while also invoking the above functions in a timely manner.

Governance

The Strategy contract inherits from the Origin `Governable` contract allowing for the use of specific functions to be controlled by governance. This allows users to vote for changes to be made in the Strategy contract, such as:

- The setting of the Registrator role via the `setRegistrator` function.
- The setting of the platform token address via the `setPTokenAddress` function and removal of the platform token address with the `removePToken` function.
- The initialization of the Strategy contract with the `initialize` function
- The transfer of all Wrapped Sonic deposits back to the vault using the `withdrawAll` function. Note that this does not withdraw from delegated validators, only from the Strategy contract itself.
- Lastly, the ability to support or unsupport validators with the `supportValidator` and `unsupportValidator` functions, respectively.

It is assumed that Governance is properly conducted in a transparent manner, and that decisions, once made, are implemented effectively. This assumption encompasses several critical facets of governance, each contributing to the system's overall health, responsiveness, and trustworthiness.

Vault

Assumptions regarding the vault have been stated earlier in the report. The vault is not a particularly privileged role but has the authorization to:

- Deposit partial or full balance of the wrapped Sonic tokens from the vault to the strategy in order to be delegated to a validator. This is done with the `deposit` and `depositAll` functions.
- Withdraw partial or full balance of the Wrapped Sonic tokens from the strategy to the vault (only done if there are some wS tokens lingering in the contract). The withdrawal is performed using the `withdraw` and `withdrawAll` functions.

Medium Severity

M-01 Inconsistent Total Value with Fully Slashed Validators

The `withdrawFromSFC` function will invoke the `withdraw` function of Sonic's Special Fee Contract (SFC) and decrease `pendingWithdrawals` by the undelegated amount specified in the withdrawal request. Consequently, the return value of the `checkBalance` function will decrease by the same undelegated amount.

However, if the delegated validator is fully slashed (the penalty exceeds the undelegated amount), the SFC will not permit the withdrawal and will revert if the `withdraw` function is called. In this scenario, the `withdrawFromSFC` function cannot be executed, and the `pendingWithdrawals` will still hold the undelegated amount, despite the strategy being unable to reclaim these funds. Moreover, the rebasing and minting functions within the vault will be impacted, as they rely on the `checkBalance` function to calculate the total value.

Consider implementing a function to ensure that withdrawal requests are completed when a validator is fully slashed.

Low Severity

L-01 `collectRewards` Can Be Front-Run With `restakeRewards` and Lead to Temporarily Locked Rewards

The registrator or strategist can call the `collectRewards` function to claim any pending validator rewards. Meanwhile, anyone can call the `restakeRewards` function to restake any pending validator rewards. This could lead to scenarios where the registrator or strategist sends transactions to collect rewards, but malicious users can repeatedly front-run these transactions with invocations of `restakeRewards`, thus causing the transactions to fail because the `rewardsAmount` will be zero. Even if the registrator or strategist calls the

`undelegate` function to reclaim the rewards, they must wait for a period before unlocking the withdrawal amount.

Consider making the `restakeRewards` function only callable by specific users.

L-02 Incomplete Docstrings

Within `SonicValidatorDelegator.sol`, multiple instances of incomplete docstrings were identified:

- In the `undelegate` function, the return value is not documented.
- In the `isWithdrawnFromSFC` function, the `_withdrawId` parameter is not documented.
- In the `setRegistrator` function, the `_address` parameter is not documented.
- In the `setDefaultValidatorId` function, the `_validatorId` parameter is not documented.
- In the `supportValidator` function, the `_validatorId` parameter is not documented.
- In the `unsupportValidator` function, the `_validatorId` parameter is not documented.
- In the `isSupportedValidator` function, the `_validatorId` parameter is not documented.

Consider thoroughly documenting all functions/events (and their parameters or return values) that are part of a contract's public API. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

L-03 Missing Docstrings

Within `SonicValidatorDelegator.sol`, multiple instances of missing docstrings were identified:

- The `Delegated` event
- The `Undelegated` event
- The `Withdrawn` event
- The `RegistratorChanged` event
- The `SupportedValidator` event
- The `UnsupportedValidator` event
- The `DefaultValidatorIdChanged` event

- The `initialize` function

Consider thoroughly documenting all functions (and their parameters) that are part of any contract's public API. Functions implementing sensitive functionality, even if not public, should be clearly documented as well. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Notes & Additional Information

N-01 Lack of Indexed Event Parameters

Within `SonicValidatorDelegator.sol`, multiple instances of missing indexed event parameters were identified:

- The `Undelegated` event
- The `Withdrawn` event

To improve the ability of off-chain services to search and filter for specific events, consider indexing event parameters.

N-02 Missing Named Parameters in Mapping

Since Solidity 0.8.18, developers can utilize named parameters in mappings. This means that mappings can take the form of `mapping(KeyType KeyName? => ValueType ValueName?)`. This updated syntax provides a more transparent representation of a mapping's purpose. In the `SonicValidatorDelegator` contract, the `withdrawals` mapping does not have any named parameters.

Consider adding named parameters to mappings in order to improve the readability and maintainability of the codebase.

N-03 Unused Named Return Variable

Named return variables are a way to declare variables that are meant to be used within a function's body for the purpose of being returned as the function's output. They are an alternative to explicit in-line `return` statements. In `SonicValidatorDelegator.sol` contract, the `withdrawId` return variable for the `undelegate` function is unused.

Consider either using or removing the `withdrawId` named return variable.

N-04 Misleading Inline Documentation

Throughout the codebase, multiple instances of inaccurate or misleading documentation were identified:

- The documentation of the `collectRewards` function states, "Claim any pending validator rewards for all supported validators". However, the current implementation does not check if the validator is supported. In addition, according to the documentation of the `unsupportValidator` function, unsupported validators can still collect rewards.
- The documentation of the `withdrawAll` function states, "Any native S in this strategy will not be withdrawn". However, the implementation does withdraw any native S token in the strategy. This is done by first wrapping the S token balance of the strategy and then withdrawing the wS token balance from the strategy to the vault.
- The documentation of the `_deposit` function states, "Deposit Wrapped Sonic (wS) to this strategy so it can later be delegated to a validator". The misleading part here is the "so it can later be delegated to a validator", as the function will unwrap the wS token and delegate the S token right away.

Clear inline documentation is fundamental to outlining the intentions of the code. Mismatches between the documentation and the implementation can lead to serious misconceptions about how the system is expected to behave. Therefore, consider fixing any discrepancies to avoid confusion for developers, users, and auditors.

Conclusion

We audited Origin's Sonic Staking contracts which enable staking and delegation of Sonic's native S token in the SFC contract. One medium-severity issue was identified, while several recommendations were made that were aimed at improving the readability and clarity of the codebase, and facilitating future audits, integrations, and development. The contracts were found to be very well-thought-out, well-documented, and of a high quality. We thank the Origin team for making themselves available and responsive throughout the audit period. We look forward to this staking strategy's success both on-chain and in DeFi generally.