Synthetix

Solidity Security Review

SIP #9, #10, #14

Version: 2.0

Contents

Introduction

Sigma Prime was commercially engaged to perform a time-boxed security review of specific changes made to the `DelegateApprovals` , `Exchanger` , `Issuer` , `FeePool` and `Synthetix` smart contracts, part of the Synthetix platform. This review focused solely on the security aspects of the Solidity implementation of the contract, but also includes general recommendations and informational comments. The more general economic structure of the system and related economic game theoretic attacks on the platform are outside the scope of this assessment.

Disclaimer

Sigma Prime makes all effort but holds no responsibility for the findings of this security review. Sigma Prime does not provide any guarantees relating to the function of the smart contract. Sigma Prime makes no judgements on, or provides any security review regarding, the underlying business model or the individuals involved in the project. This document is based upon a time-boxed analysis of the underlying smart contracts. Statements are not guaranteed to be accurate and do not exclude the possibility of undiscovered vulnerabilities.

Document Structure

The first section provides an overview of the functionality of the contracts contained within the scope of the security review. A summary followed by a detailed review of the discovered vulnerabilities is then given, which assigns each vulnerability a severity rating (see Vulnerability Severity Classification), an *open/closed/resolved* status and a recommendation. Additionally, findings which do not have direct security implications (but are potentially of interest) are marked as *informational*. Outputs of automated testing that were developed during this assessment are also included for reference (in the Appendix: Test Suite).

The appendix provides additional documentation, including the severity matrix used to classify vulnerabilities found within the contracts (`DelegateApprovals`).

Overview

The Synthetix platform aims to produce collateralized stable coins. It uses two types of tokens:

1. **Synths:** Tokens which aim to be stable compared to fiat or crypto currencies (sUSD, sAUD, sEUR, sBTC, etc.);
2. **Synthetix:** Fixed-supply token which receives fees from the exchanging of Synths via synthetix.exchange.

The Synthetix platform has been assessed by Sigma Prime on multiple previous occasions. Since these reviews, the team has considerably revised the system and introduced significant changes to the underlying contracts.

The `DelegateApprovals` smart contract allows trusted wallet addresses, known as Delegates, to mint, burn and exchange tokens on behalf of users. The Delegate also has the ability to claim weekly rewards on the user's behalf. This enables hardware wallets to partake in the Synthetix ecosystem without exposing the private keys of the wallet.

The details behind the `DelegateApprovals` contract are described in a dedicated Synthetix Improvement Proposal (SIP): SIP #10 and SIP #14.

Additionally, in the `FeePool` contract, the fee claim window has been reduced to one fee period. Details and rationale behind this change are described in SIP #9.

Security Review Summary

This review was initially conducted on commit 7eb85d4, and targets exclusively the changes introduced by the following Synthetix Improvement Proposals:

- **SIP #10** & **SIP #14**: PR #447
- **SIP #9**: PR #464

Retesting activities targeted commit 1ae5459.

The following smart contracts are updated by these pull requests:

- `DelegateApprovals`
- `Exchanger`
- `Issuer`
- `FeePool`
- `Synthetix`

Their inheritance structure is described below:

- `DelegateApprovals` :
  - `Owned` : Smart contract allowing inheriting contracts to implement a 2-step ownership transfer process;
- `Exchanger` :
  - `MixinResolver` : A mixin to give the inheritor access to the `AddressResolver` instance;
- `Issuer` :
  - `MixinResolver` : A mixin to give the inheritor access to the `AddressResolver` instance;
- `FeePool` :
  - `Proxyable` : An abstract base contract designed to work with the Synthetix proxy;
  - `SelfDestructible` : A contract that can be self destructed by its owner after a delay;
  - `LimitedSetup` : A contract which can disable functions a set time after deployment;
  - `MixinResolver` : A mixin to give the inheritor access to the `AddressResolver` instance;
- `Synthetix` :
  - `ExternStateToken` : A partial ERC20 token contact with an external state, which all tokens in Synthetix are built upon;
  - `MixinResolver` : A mixin to give the inheritor access to the `AddressResolver` instance;

The manual code-review section of the reports are focused on identifying any and all issues/vulnerabilities associated with the business logic implementation of the contracts. Specifically, their internal interactions, intended functionality and correct implementation with respect to the underlying functionality of the Ethereum Virtual Machine (for example, verifying correct storage/memory layout). Additionally, the manual review process focuses on all known Solidity anti-patterns and attack vectors. These include, but are not limited to, the following vectors: re-entrancy, front-running, integer overflow/underflow and correct visibility specifiers. For a more thorough, but non-exhaustive list of examined vectors, see [1, 2].

In prior reviews, Sigma Prime have raised vulnerabilities regarding centralization aspects (i.e. administrative control from the owner account). Synthetix have previously acknowledged and accepted these vulnerabilities, so we omit them from this review and direct the reader to our prior reviews for more information.

The testing team identified a total of five (5) issues during this assessment, all of which are classified as informational.

To support this review, the testing team used the following automated testing tools:

- Rattle: `https://github.com/trailofbits/rattle`
- Mythril: `https://github.com/ConsenSys/mythril`
- Slither: `https://github.com/trailofbits/slither`
- Surya: `https://github.com/ConsenSys/surya`

Output for these automated tools is available upon request.

Detailed Findings

This section provides a detailed description of the vulnerabilities identified during this review. Each vulnerability has a severity classification which is determined from the likelihood and impact of each issue by the matrix given in the Appendix: Vulnerability Severity Classification.

A number of additional properties of the contracts, including comments not directly related to the security posture of the smart contracts, are also described in this section and are labelled as *"informational"*.

Each vulnerability is also assigned a **status**:

- ***Open:*** the issue has not been addressed by the project team;
- ***Resolved:*** the issue was acknowledged by the project team and updates to the affected contract(s) have been made to mitigate the related risk;
- ***Closed:*** the issue was acknowledged by the project team but no further actions have been taken.

Summary of Findings

ID	Description	Severity	Status
DEL-01	Misleading Function Name (`removeAllDelegatePowers`)	Informational	Resolved
DEL-02	`WithdrawApproval` Event Emitted Regardless of Account Approval	Informational	Resolved
DEL-03	Insufficient Address Validation for `EternalStorage`	Informational	Resolved
DEL-04	Interface Variable Name Mismatch	Informational	Resolved
DEL-05	Imprecise Naming of `closeCurrentFeePeriod()` Variables	Informational	Open

DEL-01	Misleading Function Name (`removeAllDelegatePowers`)
Asset	`DelegateApprovals.sol`
Status	**Resolved:** See Resolution
Rating	Informational

Description

The `DelegateApprovals` smart contract defines the `APPROVE_ALL` action, which is leveraged to allow users to grant a delegate all powers/privileges defined in the contract by calling the `approveAllDelegatePowers()` function. When the `removeAllDelegatePowers` function is called, it removes all these powers from the delegate.

However, calling `removeAllDelegatePowers()` will not actually remove a power if it had been added manually. i.e. If a user calls `approveBurnOnBehalf()` then `removeAllDelegatePowers()` , the *"burn"* power will not be removed. This appears to be somewhat by design but could easily confuse users.

Recommendations

Ensure this behaviour is understood and documented clearly.

Resolution

The development team updated the `DelegateApprovals` smart contract to allow the `removeAllDelegatePowers` function to effectively remove all the delegated powers, including the ones that have been set individually. See commit 5e95ee7 for further details.

DEL-02	`WithdrawApproval` Event Emitted Regardless of Account Approval
Asset	`DelegateApprovals.sol`
Status	**Resolved:** See Resolution
Rating	Informational

Description

The `_withdrawApproval()` function, called within the `removeX` functions , always emits the `WithdrawApproval` event, regardless of the status of the delegate to be removed.

For example, this event will be emitted by the `DelegateApprovals` contract even when `removeIssueOnBehalf` with a `delegate` that hasn't been previously authorised.

Recommendations

Ensure this behaviour is understood and consider only emitting the `WithdrawApproval` event when a delegate is actually removed (i.e. the delegate was previously approved by the user for a particular action). This would require a change to the `EternalStorage` contract to get a return value in the `setBooleanValue()` function.

Resolution

The testing team modified the `_withdrawApproval()` internal function to only change the state of the `eternalStorage` contract and emit the `WithdrawApproval` event if the delegation privilege exists. See commit 5e95ee7 for further details.

DEL-03	Insufficient Address Validation for `EternalStorage`
Asset	`DelegateApprovals.sol`
Status	**Resolved:** See Resolution
Rating	Informational

Description

The `constructor` and `setEternalStorage()` functions allow the `owner` of the `DelegateApprovals` contract to set the `eternalStorage` address to the zero address. Note that this issue is raised as informational only as the `owner` can always update this value by calling `setEternalStorage()` .

Recommendations

Consider adding a check for the zero address inside the `setEternalStorage()` function and the constructor using a require statement.

Resolution

The development team updated the `setEternalStorage()` function to include a check against the zero address (i.e. `require(_eternalStorage != address(0)`). See commit 9d192f6 for further details.

DEL-04	Interface Variable Name Mismatch
Asset	`IDelegateApprovals.sol`
Status	**Resolved:** See Resolution
Rating	Informational

Description

The interface `IDelegateApprovals` defines the 4 following functions:

- `canIssueFor` ;
- `canBurnFor` ;
- `canClaimFor` ;
- `canExchangeFor` .

These functions take an address input parameter named `owner` . In the actual `DelegateApprovals` smart contract, this name parameter is declared with the name `authoriser` .

Recommendations

Consider changing the `owner` variable name to `authoriser` in the `IDelegateApprovals` interface for consistency purposes.

Resolution

The interface `IDelegateApprovals` has been updated by the development team to match the implementation (`DelegateApprovals`).

DEL-05	Imprecise Naming of `closeCurrentFeePeriod()` Variables
Asset	`FeePool.sol`
Status	**Open**
Rating	Informational

Description

The `closeCurrentFeePeriod` function uses two variables, `secondLastFeePeriod` and `lastFeePeriod`. The naming of these variables are not specific enough and can lead to confusion, especially for when `FEE_PERIOD_LENGTH` is 2.

Recommendations

Consider changing `lastFeePeriod` to `oldestFeePeriod` and `secondLastFeePeriod` to `secondOldestFeePeriod`.

Add the following comment above the variable declarations:

`// Note:  when FEE_PERIOD_LENGTH = 2, secondOldestFeePeriod is the current`.

Appendix A Test Suite

A non-exhaustive list of tests were constructed to aid this security review and are given along with this document. The `truffle` framework was used to perform these tests and the output is given below.

```
Contract: DelegateApprovals
  Testing DelegateApprovals Contract
    ✓ Should deploy successfully with relevant contract initialized (45ms)
    ✓ Should not allow anyone non-owners to change the eternal storage contract (78ms, 23183
    gas)
    ✓ Should allow the owner to change the eternal storage contract (90ms, 29987 gas)
    ✓ Allows the owner to set the eternal storage to the zero address (75ms, 14874 gas)
    ✓ Should allow an arbitrary user to act as a delegate for multiple users (251ms, 98900 gas
    )
    ✓ Should allow arbitrary users to approve arbitrary addresses to issue synths on their
    behalf (130ms, 49450 gas)
    ✓ Should allow arbitrary users to approve arbitrary addresses to burn synths on their
    behalf (120ms, 49406 gas)
    ✓ Should allow arbitrary users to approve arbitrary addresses to claim fees on their
    behalf (155ms, 49164 gas)
    ✓ Should allow arbitrary users to approve arbitrary addresses to exchange on their behalf
    (109ms, 49208 gas)
    ✓ Should allow arbitrary users to approve arbitrary addresses for all operations (149ms,
    49516 gas)
    ✓ Should not allow arbitrary users that were previously approved and then removed to
    approve arbitrary addresses for all operations (174ms, 68641 gas)
    ✓ Should allow users to approve a single delegate power and have remove all delegate
    powers work as expected (377ms, 128605 gas)
    ✓ Should allow arbitrary users to remove approval to issue (119ms, 68465 gas)
    ✓ Should allow arbitrary users to remove approval to burn (136ms, 68487 gas)
    ✓ Should allow arbitrary users to remove approval to claim (129ms, 68267 gas)
    ✓ Allows users to remove approvals to non-existing delegates (112ms, 79331 gas)
  Testing Exchanger Contract
    ✓ Allows delegates to exchange on behalf of an arbitrary user through the synthetix
    exchange (316ms, 556449 gas)
    ✓ Should not allow delegated exchanges if account has not been approved by user (66ms,
    50519 gas)
  Testing Issuer and Synthetix Contracts
    ✓ Allows delegates to issue synths on behalf of an arbitrary user through the synthetix
    exchange (270ms, 434120 gas)
    ✓ Should not allow delegated issuance if account has not been approved by user (68ms,
    49638 gas)
    ✓ Allows delegates to issue the maximum number of synths on behalf of an arbitrary user
    through the synthetix exchange (279ms, 433242 gas)
    ✓ Should not allow delegated issuance if account has not been approved by user (130ms,
    49827 gas)
    ✓ Allows delegates to burn synth on behalf of an arbitrary user through the synthetix
    exchange (494ms, 714209 gas)
    ✓ Allows delegate to issue synths on behalf of user and another delegate to burn synths on
     behalf of the same user (519ms, 612456 gas)
     ✓ Should not allow a non-approved delegate to burn synths on behalf of a user (1157ms,
    1772320 gas)
     ✓ Allows delegates to burn synth on behalf of a user to the target c-ratio through the
    synthetix exchange (528ms, 954340 gas)
    ✓ Allows user to burn synth to the target c-ratio through the synthetix exchange (461ms,
    886561 gas)
  Testing FeePool Contract
    ✓ Allows only an approved delegate to claim fees on behalf of an arbitrary user through
    the fee pool (550ms, 1037284 gas)
    ✓ Allows 3 accounts who have issued the same amount to receive the same reward (377ms,
    472273 gas)
    ✓ Should allow rewards to rollover after mint (766ms, 687230 gas)
    ✓ Should allow rewards to rollover before mint (513ms, 702230 gas)
31 passing (1m)
```

Appendix B Vulnerability Severity Classification

This security review classifies vulnerabilities based on their potential impact and likelihood of occurance. The total severity of a vulnerability is derived from these two metrics based on the following matrix.

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
		Likelihood		

Table 1: Severity Matrix - How the severity of a vulnerability is given based on the *impact* and the *likelihood* of a vulnerability.

References

[1] Sigma Prime. Solidity Security. Blog, 2018, Available: https://blog.sigmaprime.io/solidity-security.html. [Accessed 2018].

[2] NCC Group. DASP - Top 10. Website, 2018, Available: http://www.dasp.co/. [Accessed 2018].