SYNTHETIX

Solidity Security Review

SupplySchedule Smart Contract

Version: 2.0

December, 2019

Contents

Introduction

Sigma Prime was commercially engaged to perform a time-boxed security review of the latest `SupplySchedule` smart contract, 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 (`SupplySchedule` smart contract 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 `SupplySchedule` contract.

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 `SupplySchedule` smart contract determines the amount of mintable SNX tokens over the course of 195 weeks, using an exponential decay inflation schedule. The mechanics details behind the inflation mechanics are described in two separate Synthetix Improvement Proposals (SIP): SIP #23 and SIP #24.

Security Review Summary

This review was initially conducted on commit c6d26c1, and targets exclusively the sole file `SupplySchedule.sol`. This contract implements the SIP 23, 24 specifications, which were accessed at commit e73479e. This smart contract inherits the following contracts:

- `Owned`: Smart contract allowing inheriting contracts to implement a 2-step ownership transfer process;
- `Math`: Smart contract defining a helper function which implements an *exponentiation by squaring* algorithm.

Retesting activities targeted commit 472d911.

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 four (4) issues during this assessment, of which:

- Two (2) are classified as medium risk,
- One (1) is classified as low risk,
- One (1) is classified as informational.

All these vulnerabilities have been acknowledged and/or resolved by the development team.

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 within Synthetix's `SupplySchedule` smart contract. 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 `SupplySchedule` contract, 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
SUP-01	Potential for Week Slippage Between Mint Events	Medium	Resolved
SUP-02	Annual Percentage Yield Dictates a Different Interest Rate	Medium	Resolved
SUP-03	Inconsistent Dates Across SIPs and Source Code	Low	Resolved
SUP-04	Miscellaneous General Comments	Informational	Resolved

SUP-01	Potential for Week Slippage Between Mint Events		
Asset	`SupplySchedule.sol`		
Status	**Resolved:** In commit 472d911		
Rating	Severity: Medium	Impact: Medium	Likelihood: Medium

Description

It is likely that the number of mints executed will fall behind the scheduled number of mints.

During a `recordMintEvent()` the `remainingWeeksToMint` is calculated based on $\frac{\texttt{now} - \texttt{lastMintEvent}}{\texttt{1 weeks}}$ and rounded down. If `lastMintEvent` is not set exactly to the start of the week then there will be some slippage from when the next `recordMintEvent()` may be called as it must be at least 1 week from the previous event.

As an example consider the following scenario:

- Week 40: `lastMintEvent` is Wednesday
- Week 41: `recordMintEvent()` called on the Friday. Thus, `lastMintEvent` is now on a Friday
- Week 42: `recordMintEvent()` will not succeeed until 1 week from `lastMintEvent` and thus will not succeed until Friday, falling behind the scheduled time of Wednesday.

Recommendations

We recommend setting `lastMintEvent` to the beginning of each week that it is called, thereby preventing any possible slippage.

Alternatively, have `weeksSinceLastIssuance()` calculate its result based on `weekCounter` (the week number that was last minted).

Resolution

The development team resolved this issue by implementing the first recommendation above.

SUP-02	Annual Percentage Yield Dictates a Different Interest Rate		
Asset	`SupplySchedule.sol`		
Status	**Resolved:** See Recommendations		
Rating	Severity: Medium	Impact: Medium	Likelihood: Medium

Description

The interest rate specified in SIP #23 and SIP #24 is 2.50%. However due to the weekly compounding nature of the rate, the real interest rate or APY (Annual Percentage Yield) is about 2.53%.

The APY can be calculated as follows:

- $rate = \frac{0.025}{52}$
- $numberofperiods = 52$
- $APY = (1 + rate)^{numberofperiods} - 1$
- $APY = (1 + \frac{0.025}{52})^{52} - 1$
- $APY \approx 2.53\%$

In addition, the Solidity code assumes a fixed 52 weeks per year, when there are 52.143 weeks per standard year (or 52.286 on a leap year).

This issue affects the amount minted and may have an unexpected impact on the economic incentives of stakers.

Recommendations

We recommend considering the following:

- Detail the annual percentage yield in addition to the weekly interest rate.
- Explicitly define terminal inflation rate in SIP #24 as a compounding inflation rate of x per week (where x compounding 52 times results in an APY close to 2.5%).
- Define an appropriate `TERMINAL_SUPPLY_RATE_WEEKLY` constant instead of `TERMINAL_SUPPLY_RATE_ANNUAL`, to avoid conversions between weekly and annual yield within the contract.

Recommendations

The SIP implementation has been updated: the effective rate of inflation compounded weekly over a year is 2.53% APY. (see SIP #24).

The development team provided the following comment:

*The team and the **SIP #24** author (deltatiger) agreed that the impact of the additional yield being negligible and no need to adjust for compounding.*

SUP-03	Inconsistent Dates Across SIPs and Source Code		
Asset	`SupplySchedule.sol`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

The following are a list of important dates that are inconsistent:

- **SIP #23**:
  - *"proposes an annual inflation rate of 2.5% **on week 235"***
- **SIP #24**:
  - *"starting on September 7, 2023, the 233rd week on the SNX inflation schedule"*
  - *"drops below 2.5% on September 6, 2023"*
  - *"Starting on September 7, 2023, the weekly issuance of SNX tokens will adjust to 2.5% on an annualized basis"* Note: this falls on a Thursday, so not an integer number of weeks from the inflation start date of Wednesday March 3, 2019.
- `SupplySchedule.sol` :
  - line [58]: *"// Supply Decay stops* after *Week 234 (195 weeks of inflation decay)"*

Recommendations

According to the code the start dates of each week, without slippage, are as follows:

- **Week 0:** Wednesday 2019-03-06 - start date
- **Week 1:** Wednesday 2019-03-13 - first mint
- **Week 40:** Wednesday 2019-12-11 - first week with decay
- **Week 233:** Wednesday 2023-08-23 - last week with decay
- **Week 234:** Wednesday 2023-08-30 - first week with terminal supply.

We recommend:

- Modifying all dates to match what is specified in the code or modifying the code to match the dates and ensuring all dates are consistent
- Defining the terminal inflation rate in SIP #24 in terms of weeks in order to avoid potential inaccuracies when converting from weekly to annual inflation rates
- Ensuring all defined dates explicitly denote a full timestamp with timezone (UTC).

Please note SUP-02 when considering when to define the start of terminal supply, as this affects when the annual supply would drop below 2.5%.

Resolution

All SIPs have been modified to match the Solidity code (The SIP author is in a different timezone to the development team).

SUP-04	Miscellaneous General Comments
Asset	`SupplySchedule.sol`
Status	**Resolved:** See inline comments
Rating	Informational

Description

This section details miscellaneous findings discovered by the testing team that do not have a direct security implication:

1. `mintPeriodDuration` and `initialWeeklySupply` are not modified and may be set as constants.
   ✓ Resolved in commit [472d911]
2. Provided item 1, `tokenDecaySupplyForWeek()` accesses no state and may be set to `pure`.
   ✓ Resolved in commit [472d911]
3. The function `setSynthetixProxy()` may allow accidental setting of `synthetixProxy` to the zero address.
   ✓ Resolved in commit [472d911]
4. It is not externally obvious when `setSynthetixProxy()` is called. We recommend emitting a relevant event.
   ✓ Resolved in commit [472d911]
5. The `LogInt` event is never emitted.
   ✓ Resolved in commit [472d911]
6. If `synthetixProxy` has `useDELEGATECALL` set to true then the `onlySynthetix` will not return true as `msg.sender` is `synthetixProxy`.
7. The function `mintableSupply()` can be set to `external`.
   ✓ Resolved in commit [472d911]
8. The function `setSynthetixProxy()` does not emit an event.
   ✓ Resolved in commit [472d911]
9. If `amount` in `setMinterReward()` is greater than `mintableSupply()` for a given week then `Synthetix.mint()` cannot be successfully called.
   ✓ Resolved in commit [472d911]
10. Rename `SUPPLY_DECAY_END` to `TERMINAL_SUPPLY_START`, to remain consistent with other period constants and reduce potential for confusion on whether the end is *on* or *after* `SUPPLY_DECAY_END`.
11. There is no way to change/upgrade the SupplySchedule. In `Synthetix.sol` this is set in constructor, but no other setters are available. If there is an issue, deployment of a new `Synthetix` is required.
12. `weekCounter` is somewhat ambiguous as to whether it refers to the current week or the week that was last minted. We recommend renaming to something more explicit like `lastMintWeek`.

Recommendations

Ensure that the comments are understood and acknowledged, and consider implementing the suggestions above.

Resolution

The development team implemented most of the recommendations listed above and provided the following comments for the remaining issues:

- ***6:*** *useDELETEGATECALL has been deprecated and should never be called. It was implemented as a fallback which is now tech debt.*
- ***10:*** *Recommendation not implemented.*
- ***11:*** `SupplySchedule.sol` *is not meant to change / be upgraded. So no setter is foreseen to be required also saving bytecode space for SNX functionality.*
- ***12:*** *Recommendation not implemented.*

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: deploy
  deployment
  ✓ can deploy a test rig (5055ms, 58765050 gas)

SIP specification
  ✓ dates specified in SIP 23, 24 should be internally consistent (1ms, 101314 gas)

Contract: SupplySchedule
  ✓ should deploy successfully with relevant values initialized (29ms)
  ✓ dates and constants should be as specified in SIP 23, 24 (48ms)
  ✓ should have a weekly inflation that decays 1.25% per week, during the decay period. (113ms
    )
  ✓ should have a terminal supply rate of ~2.53% pa as per SIP 24 (42ms)
  ✓ Confirm amount minted is as expected for duration (381ms, 46267 gas)
  isolated SupplySchedule tests of onlySynthetix functions
    ✓ should be mintable when the relevant week has passed. (487ms, 70363 gas)
    isolated SupplySchedule tests using dummy proxy contract
      ✓ should return False when isMintable is immediately called after a recordMintEvent (182
    ms, 111698 gas)
      ✓ terminal supply rate should start on week 235 (AFTER week 234) (4288ms, 214650 gas)

10 passing (56s)
```

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.

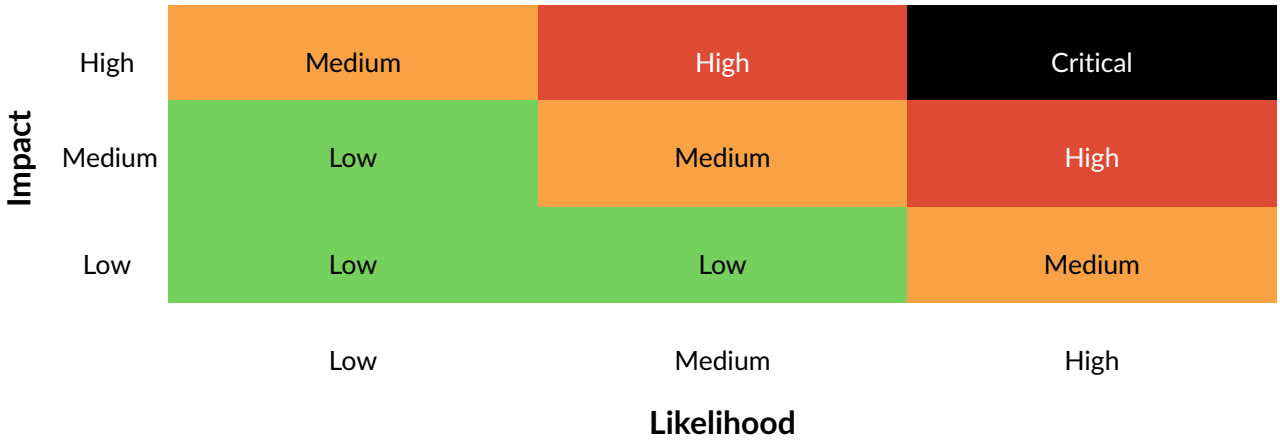

Table 1: Severity Matrix - How the severity of a vulnerability is given based on the *impact* and the *likelihood* of a vulnerability.

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].