SYNTHETIX

Multicurrency Smart Contracts

Solidity Security Review

Version: 2.0

January, 2019

Contents

Introduction

Sigma Prime was commercially engaged to perform a time-boxed security assessment of an assortment of smart contracts associated with an upgrade to the Synthetix (formerly Havven) system that introduces tokens denominated by various fiat currencies and modifies the Synth (formerly Nomin) fee mechanism. This upgrade has been dubbed "*multicurrency*".

This assessment focuses solely on the security aspects of the Solidity implementation of the in-scope files. The more general economic structure of the platform and related economic game theoretic attacks on the platform are outside the scope of this assessment.

Specifically, this assessment does not provide any judgements upon the present or future price stability of the tokens in this system.

Disclaimer

Sigma Prime makes all effort but holds no responsibility for the findings of this security assessment. Sigma Prime does not provide any guarantees relating to the function of the smart contracts. Sigma Prime makes no judgements on, or provides any security assessment regarding, the underlying business model or the individuals involved in the project.

Document Structure

The first section introduces the project and provides an overview of the functionality of the contracts contained within the scope of the security assessment. A summary of the discovered vulnerabilities is then provided, followed by a detailed assessment in which each vulnerability is assigned (i) a severity rating (see Vulnerability Severity Classification), (ii) an open/closed/resolved status and (iii) a recommendation. Additional findings which do not have direct security implications but are of potential interest are marked as *informational*.

Outputs from automated tests developed during the assessment are also included for reference (see Test Suite). The appendix provides additional documentation, including the severity matrix used to classify vulnerabilities within the Synthetix contracts.

Project Overview

The Havven system has been assessed by Sigma Prime on multiple previous occasions. Since these reviews, the team has renamed the system to Synthetix and significantly revised the system. The most notable of these revisions is the addition of Synths (formerly Nomins) of various denominations and a restructure of the fee determination logic.

Synthetix have approached Sigma Prime again to request an assessment of some of the files implicated by these changes. This review targets those files, which are exhaustively listed in the "Review Summary" section.

Review Summary

The review was conducted in multiple stages. An original commit was initially reviewed and the contract authors were provided a report. The authors then updated the code and provided a new commit hash, targeted by the testing team for retesting activities. The commit hashes reviewed are listed below, in chronological order (the final commit hash represents the files upon which this review is based):

1. Initial commit: b3962e3d025b9087a0bef1814d9f83a69e4ff690
2. Final commit: 22ca4064ed1f295675d2d8d2c6e21c9e52825dab

Below is a list of *core files* that are the focus of this review:

- ExchangeRates.sol
- FeePool.sol
- Synthetix.sol
- Synth.sol
- ReentrancyPreventer.sol
- SafeDecimalMath.sol
- TokenFallbackCaller.sol

The following *dependency files* are also included in the review as they are inherited by the core files or contain libraries that the core files inherit:

- ExternStateToken.sol
- Proxyable.sol
- SafeMath.sol
- SelfDestructible.sol

Inherited files are not reviewed in their entirety, only so far as they are used by the core files.

Sigma Prime has previously reviewed the majority of the core and dependency files at prior commit hashes. In the case that a file had been previously reviewed by Sigma Prime, a review was conducted on the differential on each files between the commit of the original review and the commit of the present review. For each case this occured, we list the file and the prior commit hash upon which the differential was created in Table 1.

During this review, the "Havven" ecosystem was renamed to "Synthetix". Table 2 provides a list of major nomenclature changes. Instances of the old nomenclature remain in this report and the test suite.

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.

File	Prior Reviewed Commit Hash	Change Since Prior Review
ExchangeRates.sol	No prior review	—
ExternStateToken.sol	b2a3d48	Minor
FeePool.sol	No prior review	—
Synthetix.sol	b5a6cdb	Significant
Synth.sol	b2a3d48	Significant
Proxyable.sol	b2a3d48	Trivial
ReentrancyPreventer.sol	b2a3d48	Trivial
SafeDecimalMath.sol	b2a3d48	Significant
SafeMath.sol	No prior review	—
SelfDestructible.sol	b2a3d48	Significant
TokenFallbackCaller.sol	b2a3d48	Trivial

Table 1: Prior Reviews

Old Name	New Name
Havven	Synthetix
Nomin	Synth
HDR	XDR
HAV	SYN
nXXX	sXXX

Table 2: Name Changes

Vulnerability Summary

The testing team only identified informational issues during this review.

Of these informational issues, all are considered closed through modifications to the code or acknowledgments.

In prior reviews, Sigma Prime have raised vulnerabilities regarding centralization aspects (i.e., full administrative control from the `owner` account) and the possibility of Synth fee-avoidance using token-wrapping contracts. Synthetix have previously acknowledged and accepted these vulnerabilities and we therefore omit them from this review and direct the reader to our prior reviews for more information.

Breakdown Per Contract

ExchangeRates.sol

Several informational issues were noted.

FeePool.sol

Several informational issues were noted.

Synthetix.sol

Several informational issues were noted.

Synth.sol

Several informational issues were noted.

ReentrancyPreventer.sol

No vulnerabilites were identified.

SafeDecimalMath.sol

No vulnerabilites were identified.

TokenFallbackCaller.sol

No vulnerabilites were identified.

ExternStateToken.sol

No vulnerabilites were identified.

Proxyable.sol

No vulnerabilites were identified.

SafeMath.sol

No vulnerabilites were identified.

SelfDestructible.sol

No vulnerabilites were identified.

Detailed Findings

This section provides a detailed description of the vulnerabilities identified within the *Havven* smart contracts in scope for 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 gas optimisations, are also described in this section and are labelled as "informational".

Each issue 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
SYN-01	Rounding Error Prevents Burning All Synths	Informational	**Closed**
SYN-02	`ExchangeRates` Miscellaneous General Comments	Informational	**Closed**
SYN-03	`FeePool` Miscellaneous General Comments	Informational	**Closed**
SYN-04	`Synthetix` Miscellaneous General Comments	Informational	**Resolved**
SYN-05	`Synth` Miscellaneous General Comments	Informational	**Closed**

SYN-01	Rounding Error Prevents Burning All Synths
Asset	Synthetix.sol
Status	**Closed:** See Resolution.
Rating	Informational

Description

In certain scenarios, a rounding error can occur whereby issuance of the maximum number of Synths falls short by the smallest unit (1×10^{-18} Synths).

This rounding error was discovered using the tests shown in the `synths.js` test suite, provided alongside this report, and demonstrates the error in action.

The second line in the following code block should fail, but does not. This is because `issueMaxSynths` does not issue the maximum number of Synths.

```
rig.synthetix.issueMaxSynths(rig.currencyKeys[3], {from: HAVVEN_HOLDER_B})
rig.synthetix.issueSynths(rig.currencyKeys[i], toBN(1), {from: HAVVEN_HOLDER_B})
```

Listing 1: `test/synths.js` line [544]

Similarly, when burning nomins, a similar error occurs:

```
current_neur_debt_balance = await rig.synthetix.debtBalanceOf(HAVVEN_HOLDER_B, rig.
    currencyKeys[3])
rig.synthetix.burnSynths(rig.currencyKeys[3], current_neur_debt_balance, {from:
    HAVVEN_HOLDER_B})
```

Listing 2: `test/synths.js` line [703]

In this case, the user attempts to fetch their "debt balance", which is the number of Syths they are permitted to burn. When attempting to burn Synths, they must burn slightly less, and so inclusion of `.sub(toBN(1))` is necessary to burn all possible Nomins on line 2 of Listing 2.

An executable example of this can be found in the file: `tests/test/roundingError.js`, whereby if the subtraction (`.sub(toBN(1))`) is removed, the test will not pass.

While it is difficult to precicely identify the source of this rounding error, we suspect that the issue is the result of combining the `divideDecimalRound` function in `Synthetix::effectiveValue` (which has 18 decimals of precision), and the `divideDecimalRoundPrecice` function in `Synthetix::_addToDebtRegister` (which has 27 decimals of precision).

Recommendations

Be consistent with the amount of decimal precision used throughout the codebase. Excess precision should be truncated, not rounded.

Resolution

The development team acknowledges this issue, see below response:

"Fixed precision decimal calculations sometimes introduce unavoidable rounding errors. For example, in the scenario where 3 people are trying to divide $10 equally, they will each end up with $3.33, and a cent is lost due to decimal precision. In our case, we have 18 decimal places of precision, so the value lost is much smaller than $0.01, but some of our calculations are still subject to this effect. This issue is an example of this type of problem, and we have been unable to find a consistent mathematical way to combat this effect."

SYN-02	`ExchangeRates` Miscellaneous General Comments
Asset	ExchangeRates.sol
Status	**Closed:** See Resolution.
Rating	Informational

Description

This section details miscellaneous informational aspects found within the `ExchangeRates` smart contract. These items are understood to not pose immediate security risks.

1. **USD Synth (sUSD) price can be set to something other than 1:** this directly contradicts the comment on line [87].
2. **Misleading comments:**
   (a) The comment on `243` states that the `currencyKeys` parameter is a set when it is a list (i.e., it may contain non-distinct items).
   (b) The comment on line [271] incorrectly states that a "specific" time will be returned whilst multiple times will actually be returned (a list of times).

Resolution

(2b) has been acknowledged, whilst all other items have been resolved.

SYN-03	`FeePool` Miscellaneous General Comments
Asset	FeePool.sol
Status	**Closed:** See Resolution.
Rating	Informational

Description

This section details miscellaneous informational aspects found within the `FeePool` smart contract. These items are understood to not pose immediate security risks.

1. **Spelling error in comment:** line [518] "…*fee period does is not yet*…".
2. **Division by zero:** the division on line [509] fails with a "divide by zero" error if there are no nomins issued.
3. **Gas savings**:
   (a) line [227], line [281], line [436], line [459], line [520]: declaring the loop iterator, `i`, as a `uint` would produce gas savings when compared to using a `uint8`.
   (b) consider using `bytes32` instead of `bytes4` to save gas (160 gas per `SSTORE`)
   (c) line [108]: There is no need to initialise `startingDebtIndex` and `feesToDistribute` to `0`
   (d) The following functions can be declared as `external` to save gas:
      i. `setFeeAuthority`
      ii. `setFeePeriodDuration`
      iii. `setSynthetix`
      iv. `amountReceivedFromTransfer`
      v. `amountReceivedFromExchange`
4. **Missing developer comments:** The following functions are missing developer comments:
   (a) `_payFees`
   (b) `_recordFeePayment`
   (c) `claimFees`
5. **Warnings when compiling with `solc`:** excessive noise from the compiler can make it easy to miss new warnings which may have security implications. Ensure that the code does not unnecessarily trigger `solc` warnings. Warnings were observed using `solc` version `0.4.25+commit.59dbf8f1.Linux.g++`.

Resolution

(3b) and (5) have been acknowledged, whilst all other items have been resolved.

SYN-04	`Synthetix` Miscellaneous General Comments
Asset	Synth.sol
Status	**Resolved:** See Resolution.
Rating	Informational

Description

This section details miscellaneous informational aspects found within the `Nomin` smart contract. These items are understood to not pose immediate security risks.

1. **XDR Synth deletion:** Using the `removeSynth()` function on line [196], it is possible for the `owner` to remove the `XDR` Synth. This is only possible if the `totalSupply` of XDR is zero.
2. **Inconsistent Camel Case:** on line [261] a function is declared as `sethavvenState`. For consistency, use `setSynthetixState`.
3. **Duplicate checks:** the `require` statements on line [434] and line [435] are duplicates that are also conducted in the `_internalExchange` function.
4. **Anybody can issue synths:** the `issueSynths` function's comments specify that *Issuance is only allowed if the havven price isn't stale and the sender is an issuer* whereas in the function's implementation, anyone with HAV tokens can issue nomins
5. **Zero-value issues:** it is possible to issue `0` Synths, this creates unnecessary events.
6. **Unnecessary comment:** a comment spans line [645] and line [646] which is repeated on line [661].
7. **Unnecessary modifier:** the `onlyFeePool` modifier is defined on line [855] but is never used.
8. **Spelling mistake:** Comment on line [304] contains a typographical error.
9. **Missing developer comments:** The following functions are missing developer comments:
   (a) `synthInitiatedExchange`
   (b) `synthInitiatedFeePayment`
   (c) `_internalExchange`
   (d) `_addToDebtRegister`
   (e) `collateralisationRatio`
   (f) `debtBalanceOf`
10. **Gas savings**: Consider moving line [738] after line [741] to avoid an extra/unnecessary call to `debtBalanceOf` when `totalOwnedSynthetix` is equal to zero.
11. **Useless condition:** on line [495], the `result` boolean is always `true` (cf function `_internalExchange`).
12. **Warnings when compiling with `solc`:** excessive noise from the compiler can make it easy to miss new warnings which may have security implications. Ensure that the code does not unnecessarily trigger `solc` warnings. Warnings were observed using `solc` version `0.4.25+commit.59dbf8f1.Linux.g++`.

Resolution

All items have been resolved.

SYN-05	`Synth` Miscellaneous General Comments
Asset	Synth.sol
Status	**Closed:** See Resolution.
Rating	Informational

Description

This section details miscellaneous informational aspects found within the `Synth` smart contract. These items are understood to not pose immediate security risks.

- **Warnings when compiling with `solc`:** excessive noise from the compiler can make it easy to miss new warnings which may have security implications. Ensure that the code does not unnecessarily trigger `solc` warnings. Warnings were observed using `solc` version `0.4.25+commit.59dbf8f1.Linux.g++`.

Resolution

This item has been acknowledged.

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.

Please note, the script `run_tests.sh` is used to execute each file individually, since running all tests with `truffle test` does not complete due to an issue with `ganache-cli`

```
Contract: deploy
  deployment
    ✓ can deploy a test rig (2146ms)

1 passing (2s)

Contract: exchangerates
  constructor
    ✓ sets public variables (1936ms)
    ✓ initial updateRates are correct (2016ms)
  functions
    ✓ Testing [updateRates] (2830ms)
    ✓ Testing [isRateStale], [anyRateIsStale], [setRateStalePeriod], and [effectiveValue]
  (3527ms)
    ✓ Testing [deleteRate] (2568ms)
    ✓ Testing [setOracle] (2189ms)

6 passing (15s)

Contract: FeePool
  FeePool test
    ✓ should allow owner to transfer synthetixs (2045ms)
    ✓ should allow owner to update exchange fee rate (1870ms)
    ✓ should allow owner to update transfer fee rate (1932ms)
    ✓ should revert when anyone but the owner tries to update fee rates (1864ms)
    ✓ should revert when owner updates the exchange fee rate to a value >
  MAX_TRANSFER_FEE_RATE (1792ms)
    ✓ should revert when owner updates the transfer fee rate to a value >
  MAX_EXCHANGE_FEE_RATE (1862ms)
    ✓ should allow owner to update fee authority (1908ms)
    ✓ should revert when anyone but the owner tries to update the fee authority (1806ms)
    ✓ should revert when anyone but the owner tries to update the fee authority (1909ms)
    ✓ should allow owner to update the synthetix state variable (1855ms)
    ✓ should revert when anyone but the owner tries to update the synthetix state variable
  (1874ms)
    ✓ should allow owner to change feePeriodDuration (1870ms)
    ✓ should revert when anyone but the owner attempts to update feePeriodDuration (1844ms)
    ✓ should revert when owner attempts to update feePeriod with value <
  MIN_FEE_PERIOD_DURATION (1792ms)
    ✓ should revert when owner attempts to update feePeriod with value >
  MAX_FEE_PERIOD_DURATION (1782ms)
    ✓ should check the amountReceivedFromExchange is correct (1995ms)
    ✓ should check the amountReceivedFromTransfer is correct (1981ms)
    ✓ should have consistent exchange fee calculation methods (2026ms)
    ✓ should have consistent transfer fee calculation methods (1852ms)
    ✓ should return the correct exchange fee incurred (1801ms)
    ✓ should return the correct transfer fee incurred (1891ms)
    ✓ should return the correct penalty (2424ms)
    ✓ should return the correct exchange fee incurred (1922ms)
    ✓ should not give users fees if they are not entitled to them (3604ms)
    ✓ should roll over the unclaimed fees (4601ms)
    ✓ should allow users to claim their fees (3131ms)

26 passing (55s)

Contract: synth-rounding
  rounding error
```

```
    ✓ It's not possible to burn all Synths. (2880ms)
    ✓ Issuance of Synths suffers a rounding error (8901ms)

2 passing (12s)

Contract: synthetix
  constructor
    ✓ sets public variables (2051ms)
  add/remove synth contract
    ✓ can add a synth contract only once and then remove it only once (2392ms)
    ✓ can have all synths deleted in forward order (2124ms)
    ✓ can have all synths deleted in reverse order (1905ms)
    ✓ does not permit deleteing the XDR nomin (1937ms)

5 passing (10s)

Contract: all_synths
  constructor
    ✓ sets public variables on each Synth (2335ms)
  setters
    ✓ disallows reconfiguration, except by owner (2562ms)
    ✓ allows reconfiguration by owner (4133ms)
  mutative functions
    ✓ Everything should start at zero, synths can only be issued correctly, transferred, and
   burned. (13337ms)
    ✓ Synths can be issued, fee period progressed, synths burned, and hav's transferred,
  without generating fees. (30245ms)
    ✓ Synths can be issued and transferred, fees are produced. (11741ms)
    ✓ Synths can be exchanged into different flavours, and exchange fees are taken. (14258ms
  )


7 passing (1m)

Contract: synth
  token fallback
    ✓ calls token fallback (2391ms)
    ✓ does not allow reentrancy from token fallback (2200ms)

2 passing (5s)
```

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.

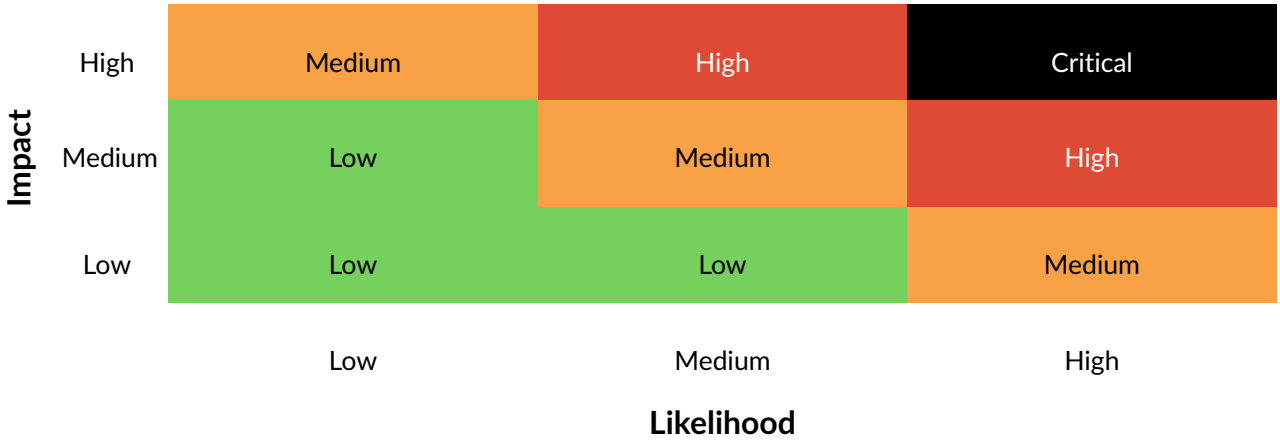

Table 3: Severity Matrix - How the severity of a vulnerability is given based on the *impact* and the *likelihood* of a vulnerability.