SNX FlashBurn Smart Contract Audit

1. Introduction

iosiro was commissioned by the Synthetix Grants DAO to conduct a smart contract audit of their FlashBurn smart contracts. The audit was performed by one auditor on 17 and 18 August 2021, consuming a total of 2 resource days.

This report is organized into the following sections.

- **Section 2 - Executive summary:** A high-level description of the findings of the audit.
- **Section 3 - Audit details:** A description of the scope and methodology of the audit.
- **Section 4 - Design specification:** An outline of the intended functionality of the smart contracts.
- **Section 5 - Detailed findings:** Detailed descriptions of the findings of the audit.

The information in this report should be used to understand the risk exposure of the smart contracts, and as a guide to improving the security posture of the smart contracts by remediating the issues that were identified. The results of this audit are only a reflection of the source code reviewed at the time of the audit and of the source code that was determined to be in-scope.

The purpose of this audit was to achieve the following:

- Ensure that the smart contracts functioned as intended.
- Identify potential security flaws.

Assessing the market effect, economics, game theory, or underlying business model of the platform were strictly beyond the scope of this audit.

Due to the unregulated nature and ease of transfer of cryptocurrencies, operations that store or interact with these assets are considered very high risk with regards to cyber attacks. As such, the highest level of security should be observed when interacting with these assets. This requires a forward-thinking approach, which takes into account the new and experimental nature of blockchain technologies. There are a number of techniques that can help to achieve this, some of which are described below.

- Security should be integrated into the development lifecycle.
- Defensive programming should be employed to account for unforeseen circumstances.
- Current best practices should be followed when possible.

2. Executive summary

This report presents the findings of the audit performed by iosiro of the smart contract implementation of the FlashBurn.

The purpose of FlashBurn is to provide functionality to burn sUSD debt with staked SNX. One medium risk issue was identified and remediated during the audit. Two informational issues were raised and remain open. Overall, the implementation accorded with the specification provided.

3. Audit details

3.1 Scope

The source code considered in-scope for the assessment is described below. Code from all other files was considered to be out-of-scope. Out-of-scope code that interacted with in-scope code was assumed to function as intended and introduce no functional or security vulnerabilities for the purposes of this audit.

3.1.1 Synthetix FlashBurn smart contracts

Project name: flashburn
Commit: 3aa02a3
File: SNXFlashLoanTool.sol

3.2 Methodology

A variety of techniques were used in order to perform the audit. These techniques are briefly described below.

3.2.1 Code review

The source code was manually inspected to identify potential security flaws. Code review is a useful approach for detecting security flaws, discrepancies between the specification and implementation, design improvements, and high risk areas of the system.

3.2.2 Dynamic analysis

The contracts were compiled, deployed, and tested in a test environment, both manually and through the test suite provided. Manual analysis was used to confirm that the code operated at a functional level, and to verify the exploitability of any potential security issues identified.

3.2.3 Automated analysis

Tools were used to automatically detect the presence of several types of security vulnerabilities, including reentrancy, timestamp dependency bugs, and transaction-ordering dependency bugs. The static analysis results were manually analyzed to remove false-positive results. True positive results would be indicated in this report. Static analysis tools commonly used include Slither, Securify, and MythX. Tools such as the Remix IDE, compilation output, and linters are also used to identify potential issues.

3.3 Risk ratings

Each issue identified during the audit has been assigned a risk rating. The rating is determined based on the criteria outlined below.

- **High risk** - The issue could result in a loss of funds for the contract owner or system users.
- **Medium risk** - The issue resulted in the code specification being implemented incorrectly.
- **Low risk** - A best practice or design issue that could affect the security of the contract.
- **Informational** - A lapse in best practice or a suboptimal design pattern that has a minimal risk of affecting the security of the contract.
- **Closed** - The issue was identified during the audit and has since been addressed to a satisfactory level to remove the risk that it posed.

4. Design specification

The following section outlines the intended functionality of the system at a high level.

4.1 Synthetix FlashBurn

FlashBurn allows users to burn sUSD debt using their staked SNX. Users low on liquidity or unable to acquire sUSD can use FlashBurn to sell a portion of their SNX and pay off their sUSD debt in one transaction.

The smart contract flash loans sUSD from Aave V2 to burn a specified amount of debt. By burning debt, user SNX is unstaked and becomes transferrable. The SNX is then sold on a DEX, such as 1inch for sUSD, to pay back the flash loan.

Users were expected to execute certain transactions prior to the burn, namely approving FlashBurn to burn sUSD on the user's behalf through the Synthetix `DelegateApprovals` contract and approve an SNX token allowance. Users were also expected to set a reasonable slippage tolerance for the SNX to sUSD exchange. This will be facilitated by the 1inch API.

5. Detailed findings

The following section includes in-depth descriptions of the findings of the audit.

5.1 High risk

No high-risk issues were present at the conclusion of the audit.

5.2 Medium risk

No medium-risk issues were present at the conclusion of the audit.

5.3 Low risk

No low-risk issues were present at the conclusion of the audit.

5.4 Informational

5.4.1 Aave `LendingPool` address not fetched dynamically

SNXFlashLoanTool.sol#L30

Description

The contract stored the Aave `LendingPool` address as an immutable state variable, however, the Aave documentation recommends that contracts always fetch the `LendingPool` address from the `LendingPoolAddressesProvider` contract. It was deemed unlikely that the `LendingPool` contract address would change as the contract made use of the upgradeable proxy pattern, so this issue has been raised as informational.

Recommendation

The `LendingPool` address should be fetched from the `LendingPoolAddressesProvider` contract dynamically instead of being stored as an immutable state variable.

5.4.2 Deprecated `safeApprove()` function used

SNXFlashLoanTool.sol#L54, *SNXFlashLoanTool.sol#L117*

Description

The contract made use of the deprecated `SafeERC20.safeApprove()` function. As noted in the `SafeERC20` contract, the `safeApprove()` function was vulnerable to the same transaction reordering issue as the standard ERC20 `approve()` function, as detailed here.

However, as the sUSD allowance was increased and spent immediately through the flash loan repayment, and the SNX was approved the maximum allowance, this issue was deemed to have minimal to no risk. As such, it has been raised as informational.

Recommendation

The `SafeERC20.safeIncreaseAllowance()` function should be used in place of `SafeERC20.safeApprove()` function.

5.5 Closed

5.5.1 Arbitrary call leads to possible token theft and reentrancy (medium risk)

SNXFlashLoanTool.sol#L140

Description

When exchanging SNX for sUSD, the `swap()` function performed an arbitrary call to the user-specified `exchange` address parameter. Validation on this parameter ensured it could not be the Aave `LENDING_POOL`, the Synthetix address or the SNX ERC20 address, but it could be any other address. This would include other tokens that may have balances on the contract. As such, any tokens could have been siphoned by a user by calling `approve()` on behalf of the contract.

Moreover, the `exchange` parameter could be used to call a user-controlled contract, which could be used to reenter the contract through the `burn()` function.

The risk of this issue was mitigated as the contract was not intended to store tokens.

Recommendation

A controlled list of exchange addresses should be used when exchanging SNX for sUSD, preventing arbitrary contract calls.

Update

All exchanges are now performed through the 1inch DEX, with the address stored as an immutable state variable.

ABOUT
CONTACT US
SMART CONTRACT AUDITING
PENETRATION TESTING
AUDIT REPORTS
PRIVACY POLICY
TERMS OF SERVICE