YEARN SNX STAKING SMART CONTRACT AUDIT

May 24, 2021

MixBytes()

CONTENTS

1. INTRODUCTION

1.1 DISCLAIMER

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Yearn. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 PROJECT OVERVIEW

Part of Yearn Strategy Mix.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> Cross check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level
Provide the client with an interim report

05 Bug fixing & re-check.
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

The main purpose of the project is to give users to add additional ability to use the Synthetix protocol managed by strategy.

1.5 PROJECT DASHBOARD

Client	Yearn
Audit name	SNX Staking
Initial version	91b839df4a350d80cb583795bccafe0836fdb732
Final version	-
SLOC	529
Date	2021-05-04 - 2021-05-24
Auditors engaged	2 auditors

FILES LISTING

Strategy.sol	Strategy.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	1
Warning	5
Comment	4

CONCLUSION

Smart contract has been audited and several suspicious places were found. During audit no critical issues were identified. One issue was marked major, as it might lead to unintended behavior. Several issues were marked as warnings and comments. After working on audit report all issues were acknowledged by client or declared as no issue, according to client's commentary. Thus contracts assumed as secure to use according to our security criteria.

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	"Sandwich attack" on user withdrawal
File	Strategy.sol
Severity	Major
Status	Acknowledged

DESCRIPTION

In some rare conditions, the strategy is using AMM DEX to Strategy.sol#L348 inside of the user-handled transaction. This is vulnerable to the "sandwich attack".

RECOMMENDATION

Although vulnerability conditions are rare and hard to exploit, it is recommended to protect AMM DEX swap operations with slippage technique.

CLIENT'S COMMENTARY

1. **Sandwich attack on user withdrawal**:
   The strategy is subject to this attack only when withdrawing 100% of want from it (unlocking 100% of collateral and repaying 100% of debt). And only in the rare condition of losses.
   When winding down, the strategy needs to repay full amount of debt to unlock collateral. This means that if debt is higher than cash (i.e. the vault in which we invested incurred in losses OR debt increased faster for any reason), the strategy will need to sell want to be able to repay full debt and unlock 100% of collateral. This means that it will incur in losses. This ONLY happens when 100% of want is withdrawed from the strategy (either migration, debtRatio == 0, or the last user withdrawal causing a 100% withdrawal from vault).

The attack is only possible if

1. debt > cash

2. 100%-of-want withdrawal
3. someone is watching for that to happen and sandwich attack us

The preferred solution is to implement a slippage protection, even if this situation is rare. However slippage protection should not be implemented in Strategy level but in something like the ySwaps (being already built by Yearn) , and all the strategies should use it. Not only for withdrawal but also for harvesting. This technique would be using a price oracle and revert if DEX price is different than price oracle.

The agreed upon way to act is:

- don't redeploy current debt-taker strategies until a ySwaps with slippage protection is deployed. once it is available, redeploy with new ySwaps as the way to swap
- for new debt-taker strategies: only implement prepareMigration if the debt is transferrable (e.g. Maker), otherwise, strategies should be revoked and a new strategy added the regular way
- If affected strategies need to be 100% liquidated in the meanwhile, act with caution. There are ways to mitigate even in the event of an attacker ready and waiting for us to wind down an strategy (which should not be the case)

2.3 WARNING

WRN-1	The approval value obtained in the constructor may not be enough for the long term of the smart contract
File	Strategy.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At lines: Strategy.sol#L79-L85
the smart contract constructor call `safeApproveA()` functions for different tokens. But in the process of work, the obtained value will only decrease. If this value decreases to zero, then the tokens will remain locked in the contract forever.

RECOMMENDATION

It is recommended to add a function to increase the value of approvals.

CLIENT'S COMMENTARY

It is a super long term thing. Approvals are 2 ** 256 - 1 (10e77) and its use is triggered mainly by yearn.

WRN-2	Default max_loss on underlying vault
File	Strategy.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At line: Strategy.sol#L512 the `withdrawFromSUSDVault()` function is not specifying max_loss parameter. This can lead to unavailability of withdrawals.

RECOMMENDATION

To implement function to change max_loss parameter by strategist.

CLIENT'S COMMENTARY

In case yvSUSD is in losses, we will need to use migrateSusdVault to unlock invested sUSD.

WRN-3	Handling losses from underlying vault
File	Strategy.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

The underlying SUSD vault may suffer a permanent loss. This will lead to a loss of corresponding SNX. However, such loss is not fairly distributed across vault users. On the first withdrawals no loss will be reported but on a later withdrawal attempts the strategy will report major losses to any users.

RECOMMENDATION

To implement some mechanics to fairly redistribute a losses.

CLIENT'S COMMENTARY

If the underlying sUSD vault incurs in losses, they are compensated with profits and not accounted as losses but considered not realised. This means that if a user is withdrawing 100% of strategy assets, they may have losses.

WRN-4	Probably incorrect using of `safeApprove`
File	Strategy.sol
Severity	Warning
Status	**No issue**

DESCRIPTION

At line Strategy.sol#L129 we see the single `safeApprove` without setting to zero.

RECOMMENDATION

Set approvement to zero before new approving

```
IERC20(susd).safeApprove(address(newSusdVault), 0);
```

CLIENT'S COMMENTARY

SafeApprove requires starting from 0 allowance. As this method is only to migrate to new sUSD vaults, it should always be 0.

WRN-5	Protected tokens
File	Strategy.sol
Severity	Warning
Status	**No issue**

DESCRIPTION

At line: Strategy.sol#L470-L475 we can't see any protected tokens.

RECOMMENDATION

We recommended to add protected tokens in the array.

CLIENT'S COMMENTARY

This was intended. Since SNX rewards are staked for a year, we wanted to have options to move tokens if the strategy was decomissioned.

2.4 COMMENTS

CMT-1	Excessive Gas usage
File	Strategy.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Second method `_unlockedWant()` call at line Strategy.sol#L260 is redundant and cost extra Gas.
Also, every access to synthetix invokes `resolver()` to get Synthetix router. This value is static and doesn't require
dynamic call.

RECOMMENDATION

It is recommended to put second `_unlockedWant` call under preceding `if` block after `reduceLockedCollateral` L255.

It is recomended to replace method `resolver` with variable (see README.md).

```
    constructor(IAddressResolver _snxResolver) public {
      synthetixResolver = _snxResolver;
    }
```

CLIENT'S COMMENTARY

Regarding _unlockedWant, impact is minor as _amountNeeded is 99% of times higher than unlockedWant. Regarding resolver, to be solved in a future iteration as it would save one SLOAD. We consider these a nice to have and will be fixed before a future redeployment.

CMT-2	Require without message
File	Strategy.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In the following function if revert occurs then user doesn't receive any information:
Strategy.sol#L100

RECOMMENDATION

We recommend to add message to require.

CLIENT'S COMMENTARY

Function is reserved for yearn team. Not to be used by any user. Saving gas on deployment.

CMT-3	Possible gas saving
File	Strategy.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Function `estimatedProfit` used only here Strategy.sol#L148, contains conversion Strategy.sol#L566. Probably this conversion is redundant, it is possible to return `estimatedProfit` in `sUSD` and convert to want with `sUSD` balances at Strategy.sol#L149, in this case, we will save one call to `_exchangeRates`.

RECOMMENDATION

Rename `estimatedProfit` to `estimatedProfitInSusd` and return it in `sUSD` and move `estimatedProfit` into `sUSDToWant`.

```
balanceOfWant().add(
    sUSDToWant(
        balanceOfSusdInVault().add(balanceOfSusd()).add(estimatedProfitInSusd())
    )
);
```

CLIENT'S COMMENTARY

We considered these a nice to have and will be fixed before a future redeployment.

CMT-4	Unnecessary gas usage
File	Strategy.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At line: Strategy.sol#L252 we see the row `uint256 unlockedWant = _unlockedWant();` and the same at line Strategy.sol#L260. It is redundant.

RECOMMENDATION

Move refresh unlockedWand value into previous if() block.

CLIENT'S COMMENTARY

We considered these a nice to have and will be fixed before a future redeployment.

3.ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

BLOCKCHAINS

- Ethereum
- Cosmos
- EOS
- Substrate

TECH STACK

- Python
- Solidity
- Rust
- C++

CONTACTS

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://t.me/MixBytes

https://twitter.com/mixbytes