CERTIK

# Security Assessment

# NIFTSY

Jul 14th, 2021

# Table of Contents

# Summary

This report has been prepared for NIFTSY to discover issues and vulnerabilities in the source code of the NIFTSY project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

# Overview

## Project Summary

| | |
|---|---|
| Project Name | NIFTSY |
| Platform | Ethereum |
| Language | Solidity |
| Codebase | https://github.com/niftsy/niftsysmarts |
| Commit | 83e7f4f9aa4d4cc2be83af5eee6895e4b12bd83a<br>0dc1b6dd78b6339c5b6f8c41ceadd60808dd6fd1<br>2a3bc7cc5378a02a0587e04db9a8c1ed9bbfabcb<br>9fb493f1efd458f337048fca29e308ea9373d054 |

## Audit Summary

| | |
|---|---|
| Delivery Date | Jul 14, 2021 |
| Audit Methodology | Static Analysis, Manual Review |
| Key Components | |

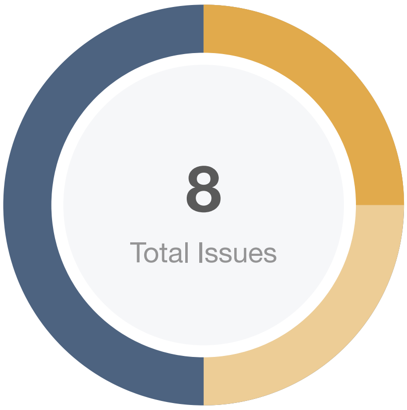
## Vulnerability Summary

| Vulnerability Level | Total | Pending | Partially Resolved | Resolved | Acknowledged | Declined |
|---|---|---|---|---|---|---|
| ● Critical | 0 | 0 | 0 | 0 | 0 | 0 |
| ● Major | 0 | 0 | 0 | 0 | 0 | 0 |
| ● Medium | 2 | 0 | 0 | 2 | 0 | 0 |
| ● Minor | 2 | 0 | 2 | 0 | 0 | 0 |
| ● Informational | 4 | 0 | 1 | 3 | 0 | 0 |
| ● Discussion | 0 | 0 | 0 | 0 | 0 | 0 |

## Audit Scope

| ID | file | SHA256 Checksum |
|---|---|---|
| MRY | MinterRole.sol | ee5c5ea5f76767658f865add25a3b1bdcb057a0e8660eca3b063d91c85c18b2b |
| NER | NiftsyERC20.sol | cf91861b1918589270a9d486edb034c942340f2b280d4451769225ec793f0b57 |
| WBY | WrapperBase.sol | 09969b1be63ba161ed916f86d5696704c531468511d4e3eae02ce690efdc0fbf |
| WWE | WrapperWithERC20Collateral.sol | e68a542323d9bfe067a8017e9b649472f900cac839df79a849f1e89b939d4ea8 |

# Findings



| | | |
|---|---|---|
| 🟥 | **Critical** | **0** (0.00%) |
| 🟧 | **Major** | **0** (0.00%) |
| 🟨 | **Medium** | **2** (25.00%) |
| 🟨 | **Minor** | **2** (25.00%) |
| 🟦 | **Informational** | **4** (50.00%) |
| 🟩 | **Discussion** | **0** (0.00%) |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| **NER-01** | Privileged ownership | **Centralization / Privilege** | ● **Medium** | ✓ **Resolved** |
| NER-02 | Comparison to boolean constant | Language Specific | ● Informational | ✓ Resolved |
| WBY-01 | Comparison to boolean constant | Language Specific | ● Informational | ✓ Resolved |
| WBY-02 | Lack of Input Validation | Volatile Code | ● Informational | ✓ Resolved |
| **WBY-03** | Privileged ownership | **Centralization / Privilege** | ● **Minor** | **Partially Resolved** |
| **WBY-04** | Fee Charging in the Future | **Centralization / Privilege** | ● **Informational** | **Partially Resolved** |
| **WWE-01** | Privileged ownership | **Centralization / Privilege** | ● **Minor** | **Partially Resolved** |
| WWE-02 | Potential Reentrant to Sensitive Functions | Logical Issue | ● Medium | ✓ Resolved |

# NER-01 | Privileged ownership

| Category | Severity | Location | Status |
|---|---|---|---|
| **Centralization / Privilege** | ● **Medium** | contracts/NiftsyERC20.sol: 22, 28, 46 | ✓ **Resolved** |

## Description

The minter of contract `NiftsyERC20` has the permission to:

1. mint tokens as most as 500000000 * e18,
2. burn tokens of any account,
3. transfer tokens between accounts.

without obtaining the consensus of the community.

## Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

## Alleviation

The development team heeded our advice and resolved it with commit 860c409c59bbb08038f4492ac6044b1623a34191.

# NER-02 | Comparison to boolean constant

| Category | Severity | Location | Status |
|---|---|---|---|
| Language Specific | ● Informational | contracts/NiftsyERC20.sol: 43 | ✓ Resolved |

## Description

There is a comparison to boolean constant.

## Recommendation

Consider removing the equality to the boolean constant.

## Alleviation

The development team heeded our advice and resolved it with commit 109dd61a58202c7aad419a42cf1b558eb40a110b.

# WBY-01 | Comparison to boolean constant

| Category | Severity | Location | Status |
|---|---|---|---|
| Language Specific | ● Informational | contracts/WrapperBase.sol: 280 | ✓ Resolved |

## Description

There is a comparison to boolean constant.

## Recommendation

Consider removing the equality to the boolean constant.

## Alleviation

The development team heeded our advice and resolved it with commit 109dd61a58202c7aad419a42cf1b558eb40a110b.

# WBY-02 | Lack of Input Validation

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | ● Informational | contracts/WrapperBase.sol: 67 | ✓ Resolved |

## Description

The assigned value to `projectToken` in the constructor of `WrapperBase` should be verified as a non-zero value to prevent error.

## Recommendation

Check that the passed-in values are non-zero values.

Example:

```
require(_erc20!= address(0), "projectToken is a zero value");
```

## Alleviation

The development team heeded our advice and resolved it with commit 2a3bc7cc5378a02a0587e04db9a8c1ed9bbfabcb.

# WBY-03 | Privileged ownership

| Category | Severity | Location | Status |
|---|---|---|---|
| Centralization / Privilege | ● Minor | contracts/WrapperBase.sol: 256 | Partially Resolved |

## Description

The owner of contract `WarpperBase` has the permission to set `protokolFee` and `chargeFeeAfter` without obtaining the consensus of the community.

## Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

## Alleviation

**[NIFTSY team]**: After some discussion with the team, we decided to leave this code unchanged. The control of these functions will be transferred to the multisig address,accordance with the auditor's recommendations.

# WBY-04 | Fee Charging in the Future

| Category | Severity | Location | Status |
|---|---|---|---|
| **Centralization / Privilege** | ● **Informational** | contracts/WrapperBase.sol: 124 | **Partially Resolved** |

## Description

There is a fee-charging in the future when wrapping an ERC720 by the function `warp721`.

## Alleviation

**[NIFTSY team]**: After some discussion with the team, we decided to leave this code unchanged. Community operators should be able to resolve the issue of protocol monetization, and the control of these functions will be transferred to the multisig address, accordance with the auditor's recommendations.

# WWE-01 | Privileged ownership

| Category | Severity | Location | Status |
|---|---|---|---|
| **Centralization / Privilege** | ● **Minor** | contracts/WrapperWithERC20Collateral.sol: 89, 99 | **Partially Resolved** |

## Description

The owner of contract `WarpperWithERC20Collateral` has the permission to set `setCollateralStatus` and `setMaxERC20CollateralCount` without obtaining the consensus of the community.

## Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

## Alleviation

**[NIFTSY]**: After some discussion with the team, we decided to leave this code unchanged. The control of these functions will be transferred to the multisig address,accordance with the auditor's recommendations.

# WWE-02 | Potential Reentrant to Sensitive Functions

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Medium | contracts/WrapperWithERC20Collateral.sol: 42 | ✓ Resolved |

## Description

As `_erc20` is a smart contract reference which implementation may be unknown to the user and potentially includes logic to reentrant the sensitive functions such as `addERC20Collateral()`.

## Recommendation

We advise developers to adopt `nonReentrant` modifier in openzeppelin to sensitive functions `addERC20Collateral()`.

## Alleviation

The development team heeded our advice and resolved this issue in commit 0dc1b6dd78b6339c5b6f8c41ceadd60808dd6fd1.

# Appendix

## Finding Categories

### Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

### Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

### Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

### Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

## Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

# Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

# About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK