// HALBORN

Composable Finance - Staking rewards & fNFT Pallets

Substrate Pallet Security Audit

Prepared by: **Halborn**

Date of Engagement: **October 10th, 2022 - November 7th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/10/2022	Thiago Mathias
0.2	Draft Review	11/08/2022	Timur Guvenkaya
0.3	Draft Review	11/09/2022	Gabi Urrutia
1.0	Remediation Plan	11/24/2022	Michal Bajor
1.1	Remediation Plan Review	11/24/2022	Timur Guvenkaya
1.2	Remediation Plan Review	11/24/2022	Piotr Cielas
1.3	Remediation Plan Review	11/24/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Piotr Cielas	Halborn	Piotr.Cielas@halborn.com
Timur Guvenkaya	Halborn	Timur.Guvenkaya@halborn.com
Thiago Mathias	Halborn	Thiago.Mathias@halborn.com
Michal Bajor	Halborn	Michal.Bajor@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Composable Finance engaged Halborn to conduct a security audit on their Staking rewards and fNFT pallets beginning on October 10th, 2022 and ending on November 7th, 2022. The security assessment was scoped to the pallets provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided four weeks for the engagement and assigned two full-time security engineers to audit the security of the staking rewards & fNFT pallets. Security engineers are blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the pallets

In summary, Halborn identified some improvements to reduce the likelihood and impact of risks, which were acknowledged by the Composable Finance team. The main ones are the following:

- Remove deprecated macros
- Remove unused variable.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy regarding the scope of the Composable staking rewards & fNFT pallets. While manual

testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into the architecture, purpose, and use of the platform.
- Smart contract manual code review and walkthrough to identify any logic issue.
- Mapping out possible attack vectors
- Thorough assessment of safety and usage of critical Rust variables and functions in scope that could lead to arithmetic vulnerabilities.
- On chain testing of core functions(polkadot.js).
- Scanning dependencies for known vulnerabilities (cargo audit).

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

Code repositories:

1. Substrate Composable pallets
   (a) Repository: ComposableFi/composable
   (b) Commit ID: c28bf28e956f21cf3afd78de91750acc584ecb56
   (c) Pallets in scope:
      - Staking rewards (composable/code/parachain/frame/staking-rewards)
      - fNFT (composable/code/parachain/frame/fnft/)

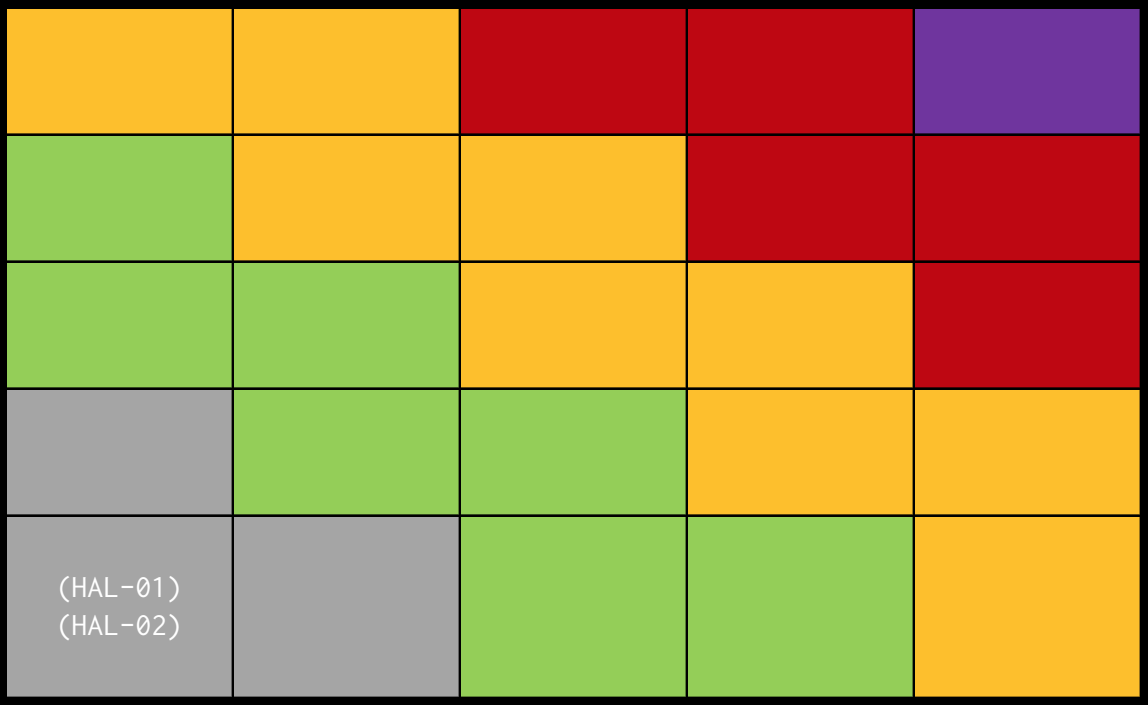
Out-of-scope: External libraries and financial related attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	0	2

LIKELIHOOD

IMPACT



EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
UNUSED VARIABLE	Informational	ACKNOWLEDGED
USAGE OF DEPRECATED MACRO	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) UNUSED VARIABLE - INFORMATIONAL

Description:

In the fnft pallet, the **burn** function has an entry named _maybe_check_owner that is not used in the code.
Unnecessary code snippets should be avoided to maintain a clean code.

Code Location:

Listing 1: fnft/src/lib.rs (Line 365)

```
362 fn burn(
363     collection: &Self::CollectionId,
364     instance: &Self::ItemId,
365     _maybe_check_owner: Option<&AccountIdOf<T>>,
366 ) -> DispatchResult {
367     Instance::<T>::try_mutate_exists(collection, instance, |entry|
 ↳  -> DispatchResult {
368         match entry {
369             Some((owner, _)) => {
370                 OwnerInstances::<T>::mutate(owner, |x| match x {
371                     Some(instances) => {
372                         instances.remove(&(*collection, *instance)
 ↳ );
373                     },
374                     None => {
375                         debug_assert!(false, "unreachable")
376                     },
377                 });
378                 *entry = None;
379                 Ok(())
380             },
381             None => Err(Error::<T>::InstanceNotFound.into()),
382         }
383     })?;
384
385     // TODO (vim): Remove account proxy ??
386     Self::deposit_event(Event::FinancialNftBurned {
387         collection_id: *collection,
```

```
388             instance_id: *instance,
389         });
390
391         Ok(())
392     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Remove the unused input variable.

Remediation Plan:

ACKNOWLEDGED: The Composable Finance team acknowledged this issue.

3.2 (HAL-02) USAGE OF DEPRECATED MACRO - INFORMATIONAL

Description:

The staking-rewards pallet is using #[transactional] macro to add another storage layer to the functions' execution, causing them not to alter the storage if an error is encountered. However, the staking-rewards pallet is using the Polkadot modules in version 0.9.27 which implements this behavior by default for all functions.

causing them not to alter storage if an error is encountered.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to delete the unnecessary macro.

Remediation Plan:

ACKNOWLEDGED: The Composable Finance team acknowledged this issue.

THANK YOU FOR CHOOSING

// HALBORN