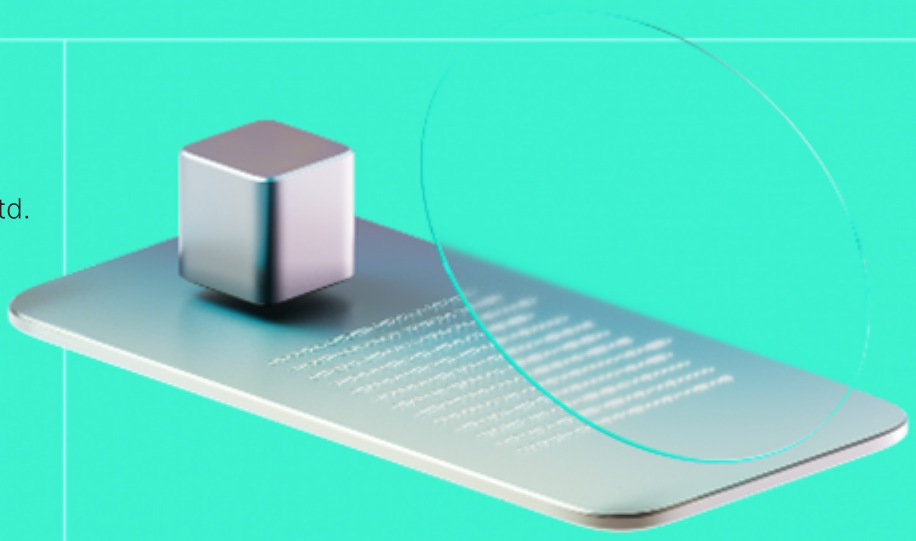
HACKEN

Smart Contract Code Review And Security Analysis Report

Customer: The Sweat Foundation Ltd.

Date: 19/01/2024



We thank Sweatco for allowing us to conduct a Smart Contract Security Assessment. This document outlines our methodology, limitations, and results of the security assessment.

Sweat Economy is a system that allows users to earn $SWEAT tokens by walking. The Sweatco Claim feature is an extension of the Sweat Wallet application, designed to provide users with a secure and controlled environment for their earned $SWEAT tokens.

Platform: NEAR Protocol

Language: Rust

Tags: Claims

Timeline: 20.12.2023 - 18.01.2024

Methodology: https://hackenio.cc/sc_methodology

Last Review Scope

Repository	https://github.com/sweatco/sweat-claim
Commit	ffa3791f9e722b98736efdd2ca773c444eccd7cd

Audit Summary

10/10	9/10	88.79%	9/10
Security Score	Code quality score	Test coverage	Documentation quality score

Total 9.2/10

The system users should acknowledge all the risks summed up in the risks section of the report

3	3	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	2
Low	0

Vulnerability	Status
F-2023-0249 - Outdated borsh library version in Cargo.toml	Fixed
F-2023-0250 - Compilation Issues Due to Conflicting near-sdk Versions in Cargo.toml	Fixed
F-2023-0295 - Inconsistent Claim Availability Logic	Fixed

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for The Sweat Foundation Ltd.
Audited By	Noah Jelich
Approved By	Noah Jelich
Website	sweatco.in
Changelog	29/12/2023. - Preliminary Report 18/01/2024. - 2nd Review

Table to Contents

System Overview

The claim feature is an extension of the Sweat Wallet application and aims to safely store the $SWEAT minted for a given users based on their steps provided by the Sweatcoin Oracle and converted to $SWEAT as per the token's minting curve.

Prior to this "claim" feature, $SWEAT accrued from steps was calculated several times per day as determined by the Sweatcoin Oracle and $SWEAT was minted accordingly by the `token.sweat` contract and transferred to the given user's wallet address. The goal of the "claim" feature is to given the user more control over their $SWEAT earned from walking. This is accomplished by diverting minted $SWEAT to a new contract where it will accrue until a user claims it.

The contract furthermore caters for edge cases in user behaviour which current places the Sweat economy at risk. E.g. If a user churns and disbands the project then there should be a mechanism to recover $SWEAT that was minted to a user's address but abandoned by the user. Currently this is impossible as Sweat Wallet is a self-custody thereby rendering complete control of funds to the user. Having a contract where minted $SWEAT accrues provides a degree of separation in terms of ownerships rights /control of minted $SWEAT. Sweat Wallet may therefore impose a condition that $SWEAT which is not claimed after a set amount of time may be burned from the claim contract. This will not only create a healthier economy (supply vs demand) but furthermore provide a method for maintaining an efficient contract size.

Privileged roles

- It Oracle role can perform token burns, a full reset of the system, set the burn and claim period, as well as record batches for holding.

Executive Summary

This report presents an in-depth analysis and scoring of the customer's smart contract project. Detailed scoring criteria can be referenced in the scoring methodology.

Documentation quality

The total Documentation Quality score is **9** out of **10**.

- Functional requirements are provided.
- Technical description is mostly provided.
  - Lower primitives are documented, but there is lacking documentation of top-level api functions.

Code quality

The total Code Quality score is **9** out of **10**.

- The code is well written and architected.
- The development environment is configured.

Test coverage

Code coverage of the project is **88.79%** (line coverage).

- Coverage can be measured for the integration tests using [wasmcov] (https://hknio.github.io/wasmcov/docs/NEAR).

Security score

Upon auditing, the code was found to contain **0** critical, **0** high, **2** medium, and **0** low severity issues, leading to a security score of **10** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's smart contract yields an overall score of **9.2.** This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

Risks

- The projest is built on a custom fork of near-sdk standard libraries, which are out of scope. While the changes are minimal, this represents a known unknown in potential behaviour.
  - near-sdk = { git = "https://github.com/sweatco/near-sdk-rs", rev = "8c48b26cc48d969c1e5f3162141fe9c824fccecd" }
  - near-contract-standards = { git = "https://github.com/sweatco/near-sdk-rs", rev = "8c48b26cc48d969c1e5f3162141fe9c824fccecd" }
- The project has large out-of-scope external dependencies that comprise a significant portion of functionality:
  - integration-trait = { git = "https://github.com/sweatco/integration-trait.git", rev = "83a69f7ba4acab9405bf935f7dfc70f2f5279c40" }
  - integration-utils = { git = "https://github.com/sweatco/integration-utils.git", rev = "0b3c2faf0db31cdb3481be4d35dbb48b62a98618" }
  - sweat-model = { git = "https://github.com/sweatco/sweat-near", rev = "82b5acf1c743d38191b996dc8eff3ac289e467d6" }
  - sweat-integration = { git = "https://github.com/sweatco/sweat-near", rev = "82b5acf1c743d38191b996dc8eff3ac289e467d6" }
- The Oracle role has high unregulated privileges in the system, including a full system reset.

Findings

Vulnerability Details

F-2023-0250 - Compilation Issues Due to Conflicting near-sdk Versions in Cargo.toml - Medium

Description:

The project is encountering compilation issues due to conflicting versions of the `near-sdk` crate specified in the `Cargo.toml` file. The issue arises from mixing `near-sdk` versions, where one version (4.1.1) is specified as a dependency in `near-workspaces` 0.9.0, and another version (4.1.1) is used throughout the rest of the project.

Except from the `Cargo.toml` file illustrating the conflicting versions:

```
[[package]]
name = "near-workspaces"
version = "0.9.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum = "3a14e772e49ba9644c06dad20f635b6463f74d378fa19822bfc35fef479c72e5"
dependencies = [
 ...
 "near-sdk 4.1.1 (registry+https://github.com/rust-lang/crates.io-index)",
 ...
]

[[package]]
name = "near-contract-standards"
version = "4.1.1"
source = "git+https://github.com/sweatco/near-sdk-rs?rev=8c48b26cc48d969c1e5f3162141fe9c8
dependencies = [
 "near-sdk 4.1.1 (git+https://github.com/sweatco/near-sdk-rs?rev=8c48b26cc48d969c1e5f3162
 ...
]
```

This conflict occurs because of the usage of forks for all the `near` contract packages. The presence of multiple definitions of the `alt_bn128_pairing_check` symbol leads to compilation issues.

Status: Fixed

Classification

Severity: Medium

Recommendations

Recommendation:

To resolve this issue, it is recommended to patch the dependency in the `Cargo.toml` file. Ensure that the `near-workspaces` crate uses the same version of `near-sdk` as the rest of the project. This will eliminate the conflict and prevent the multiple definitions of the `alt_bn128_pairing_check` symbol.

To patch the dependency, add the following entry to the `Cargo.toml` file:

```
[patch.crates-io]
near-sdk = { git = "https://github.com/sweatco/near-sdk-rs", rev = "8c48b26cc48d969c1e5f3
```

This patch entry instructs Cargo to use the specified version of `near-sdk` from the provided Git repository and revision.

After adding the patch entry, rebuild the project by running `cargo build` or `cargo run` to apply the patch and resolve the compilation issues.

Note: patching dependencies should be considered a temporary solution. It is recommended to collaborate with the maintainers of the packages to resolve any compatibility issues and avoid the need for patching in the long term.

F-2023-0295 - Inconsistent Claim Availability Logic - Medium

Description:

The current implementation of the claim availability logic at line 42 in `contract/src/claim/api.rs` allows users to claim instantly if they haven't claimed before, regardless of the specified claim period. However, the technical specifications indicate that users should wait for the claim period to elapse, including before making their first claim. This inconsistency could lead to a deviation from the intended behavior and user expectations.

```
let Some(last_claim_at) = account_data.last_claim_at else {
    return ClaimAvailabilityView::Available;
};
```

Assets:

- contract/src/claim/api.rs [https://github.com/sweatco/sweat-claim/tree/main]

Status: Fixed

Classification

Severity: Medium

Recommendations

Recommendation:

To align with the technical specifications and ensure consistent behavior, it is recommended to modify the claim availability logic. Instead of allowing instant claiming for users who haven't claimed before, the code should enforce the claim period, even for the first claim.

This can be achieved by adjusting the conditional statement to check if the current time is greater than or equal to the sum of the last claim time and the claim period. If the condition is not met, the function should return a view indicating that the claim is not yet available.

F-2023-0249 - Outdated borsh library version in Cargo.toml - Info

Description: The `Cargo.toml` file in the project contains an outdated version of the `borsh` library. The current version is 0.10.3, while the recommended version is 1.3.0. It is important to keep dependencies up to date to benefit from bug fixes, new features, and performance improvements.

Status: Fixed

Classification

Severity: Info

Recommendations

Recommendation: To resolve this issue, update the `borsh` library to the recommended version 1.3.0 in the `Cargo.toml` file. After making the necessary changes, save the file and run the appropriate command to update the dependencies in the project. For example, if you are using `cargo` as your package manager, you can run `cargo update` in the terminal to update the `borsh` library to the latest version.

Please note that when updating a library, it is important to check for any breaking changes or additional steps required for the migration. Refer to the library's documentation or release notes for more information on how to update to the new version.

Observation Details

F-2023-0319 - Tests Can Be Excluded From Coverage Report - Info

Description: The current definition of the `make cov` command includes test files (`tests.rs`) in the coverage report. That skews the final result.

Status: Fixed

Recommendations

Recommendation: Test files can be excluded from the coverage measurement using the `--ignore-filename-regex` flag:

```
cov: ##@Testing Run unit tests with coverage.
    cargo llvm-cov --hide-instantiations --open --ignore-filename-regex tests.rs
```

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Severity Definitions

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution, do not affect security score but can affect code quality score.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details (Initial Review)

Repository	https://github.com/sweatco/sweat-claim
Commit	ffa3791f9e722b98736efdd2ca773c444eccd7cd
Whitepaper	https://drive.google.com/file/d/1IPkIRcEQvgJkCaeYvGh43yjWI-Dj5_6i/view
Requirements	
Technical Requirements	

Contracts in Scope (Initial Review)

contract/src/lib.rs;3d128446bace506c36148e0e0eacb210cf2fc924db3b56b8de99a5425f9d6088

contract/src/auth/api.rs;bb473cd72a84baf4515a6542ffe0d8eda56eed23dd4ec64184bb55dbb7d81fc6

contract/src/auth/mod.rs;274b1983864b577d0f0ed71506623fa7bc45b34c3a2529d882c9e8c504a880ad

contract/src/burn/api.rs;c4f944d04011d9b3ba8e47e8c48cb4555b982680de97869d89a99aa8124294df

contract/src/burn/mod.rs;274b1983864b577d0f0ed71506623fa7bc45b34c3a2529d882c9e8c504a880ad

contract/src/claim/api.rs;f0b6bff0b06bcb91ad8760d0a5e31510ea99a1470f93bd46b419bfabd840b990

contract/src/claim/mod.rs;274b1983864b577d0f0ed71506623fa7bc45b34c3a2529d882c9e8c504a880ad

contract/src/clean/api.rs;c2345b3f8377f8cf2444f0105c28b77372a730b90a472cd42057eec17bd1868c

contract/src/clean/mod.rs;274b1983864b577d0f0ed71506623fa7bc45b34c3a2529d882c9e8c504a880ad

contract/src/common/asserts.rs;c0bc54ca31e9779c16eda45a87f43746ff1a92fa722ecc910901f6768315a817

contract/src/common/mod.rs;ff029b2e627d245a3cffe5f3bfee68b8f9b661844ef77018b01d3da5945f8db3

contract/src/config/api.rs;1f04756c6bf9c86a60237764e3a0df9ab1cdef4620e1884a3af22cbc75a89807

contract/src/config/mod.rs;274b1983864b577d0f0ed71506623fa7bc45b34c3a2529d882c9e8c504a880ad

contract/src/record/api.rs;247dbe9648c7a787efc4cf3dccfa1a64f468114c964ced455bc326b391ec4be4

contract/src/record/mod.rs;274b1983864b577d0f0ed71506623fa7bc45b34c3a2529d882c9e8c504a880ad

model/src/account_record.rs;1f5c5d93a6572f7b14b5032e753459581cadceb7b147af5ac1c0fdfca2acb501

model/src/api.rs;5128d613a50c2f2f4bfce6fec386cf30e5c9aad472452cfcc7af344205a11395

model/src/event.rs;75b6a384457bed841ee9d08e6c522a2cb4df4e68babdcc397bc5c102a3ce0b7c

Contracts in Scope (Initial Review)

model/src/lib.rs;ee6123d7b1f9628961cecf2231ccbbd928c3b4bbf195485d2304971af323968e

Scope Details (2nd Review)

Repository	https://github.com/sweatco/sweat-claim
Commit	bf93665d86bb36dfab898e3788ebd5a1a09e52e6
Whitepaper	https://drive.google.com/file/d/1lPklRcEQvgJkCaeYvGh43yjWl-Dj5_6i/view
Requirements	
Technical Requirements	

Contracts in Scope (2nd Review)

contract/src/lib.rs;80cccb87d14e180ca675e6eeeb38fa3d617179b267dac1b6914c3c4edb656aae

contract/src/auth/api.rs;bb473cd72a84baf4515a6542ffe0d8eda56eed23dd4ec64184bb55dbb7d81fc6

contract/src/auth/mod.rs;274b1983864b577d0f0ed71506623fa7bc45b34c3a2529d882c9e8c504a880ad

contract/src/burn/api.rs;c4f944d04011d9b3ba8e47e8c48cb4555b982680de97869d89a99aa8124294df

contract/src/burn/mod.rs;274b1983864b577d0f0ed71506623fa7bc45b34c3a2529d882c9e8c504a880ad

contract/src/claim/api.rs;40995b92ac6f4c2689832fa07bfcea50aa9efe3b145979641685c9afc1bc7a72

contract/src/claim/mod.rs;274b1983864b577d0f0ed71506623fa7bc45b34c3a2529d882c9e8c504a880ad

contract/src/clean/api.rs;c2345b3f8377f8cf2444f0105c28b77372a730b90a472cd42057eec17bd1868c

contract/src/clean/mod.rs;274b1983864b577d0f0ed71506623fa7bc45b34c3a2529d882c9e8c504a880ad

contract/src/common/asserts.rs;01273e5a47f5912b71c29f719ac5445b93dec124ce80a3c471bc885825cd4d31

contract/src/common/mod.rs;56817bfae267249473eeb436089003b781293df5d5ae3a5cbde9a8ca36bf868f

contract/src/config/api.rs;1f04756c6bf9c86a60237764e3a0df9ab1cdef4620e1884a3af22cbc75a89807

contract/src/config/mod.rs;274b1983864b577d0f0ed71506623fa7bc45b34c3a2529d882c9e8c504a880ad

contract/src/record/api.rs;c2af3c6ab4d20230351b8bf367be3d2923091f417bc402f26d6846e9d766a0a5

contract/src/record/mod.rs;274b1983864b577d0f0ed71506623fa7bc45b34c3a2529d882c9e8c504a880ad

model/src/account_record.rs;9ece8a9c8edfe96f0f1f9ec48ca8bd0eb437f28b9d31191b30ea103c50deea3b

model/src/api.rs;5128d613a50c2f2f4bfce6fec386cf30e5c9aad472452cfcc7af344205a11395

model/src/event.rs;75b6a384457bed841ee9d08e6c522a2cb4df4e68babdcc397bc5c102a3ce0b7c

Contracts in Scope (2nd Review)

model/src/lib.rs;5f54333c181704215e5212f16526abf7fb98e5cf3a6c10bf0d33fcbac82f939d