Guvenkaya® The Bedrock of Security

The Sweat Foundation Ltd

Tiered Jars & Boosters Smart Contract Security Review

Lead Security Engineer: Michal Bajor
Date of Engagement: 29th December 2025 - 12th January 2026
Visit: www.guvenkaya.co

Contents

About Us

Guvenkaya is a security research firm specializing in Rust security, Web3 security of Non-EVM protocols, and Web2 security. With our expertise, we provide both security auditing services and custom security solutions

About The Sweat Foundation

The Sweat Foundation is an organization behind Sweat Economy, an innovative project at the intersection of fitness and crypto. It motivates users to stay active by converting their steps into SWEAT Token. This approach promotes health and fitness and works as an entry point to crypto for many users.

Audit Results

Guvenkaya conducted a security assessment of the SWEAT Jar code changes introduced in PR #144 (Tiered Score Based Jars & Boosters). During this engagement, 4 findings were reported: 1 Low and 3 Informational severity issues. The Low finding highlights possible arithmetic overflow paths, while the Informational findings cover APY representation constraints, booster lifecycle edge-cases, and global APY clamping behavior.

Project Scope

SWEAT Jar (PR #144)

Files	Link
Tiered Score Based Jars & Boosters	https://github.com/Sweat-Foundation/sweat-jar/pull/144

Out of Scope

The audit included reviewing the code changes in scope for security vulnerabilities, coding practices, and architecture. The audit does not include a review of external dependencies, off-chain services, or oracle infrastructure beyond how they are consumed by the smart contracts.

Timeline

Start of the audit	Draft report	Final report
29th December 2025	12th January 2026	14th January 2026

Methodology

RESEARCH INTO PROJECT ARCHITECTURE

PREPARING ATTACK VECTORS

SETTING UP AN ENVIRONMENT

MANUAL CODE REVIEW OF THE CODE

ASSESSMENT OF RUST SECURITY ISSUES

ASSESSMENT OF NEAR SECURITY ISSUES

ASSESSMENT OF ARITHMETIC ISSUES

BUSINESS LOGIC VULNERABILITY ASSESSMENT

ONCHAIN TESTING USING NEAR WORKSPACES

BEST PRACTICES AND CODE QUALITY

CHECKING FOR CODE REFACTORING/SIMPLIFICATION POSSIBILITIES

ARCHITECTURE IMPROVEMENT SUGGESTIONS

PREPARING POCS AND/OR TESTS FOR EACH CRITICAL/HIGH/MEDIUM ISSUES

Severity Breakdown

01. Likelihood Ratings

Likely: The vulnerability is easily discoverable and not overly complex to exploit.
Possible: The vulnerability presents some challenges either in discovery or in the complexity of the attack.
Rare: The vulnerability is either very difcult to discover or complex to exploit, or both.
This matrix provides a nuanced view, taking into account both the ease of discovering a vulnerability and the complexity involved in exploiting it.

02. Impact

Severe: Exploitation could result in critical loss or compromise, such as full system control, substantial financial loss, or severe reputational damage.
Moderate: Exploitation may lead to limited data loss, partial compromise, moderate financial impact, or noticeable degradation of services.
Negligible: Exploitation has minimal impact, such as minor data exposure without significant consequences or slight inconvenience without substantial disruption.

03. Severity Ratings

Critical: Assigned to vulnerabilities with severe impact and a likely likelihood of exploitation.
High: For vulnerabilities with either severe impact but only a possible likelihood, or moderate impact with a likely likelihood.
Medium: Used for vulnerabilities with severe impact but a rare likelihood, moderate impact with a possible likelihood, or negligible impact with a likely likelihood.
Low: For vulnerabilities with moderate impact and rare likelihood, or negligible impact with a possible likelihood.
Informational: The lowest severity rating, typically for vulnerabilities with negligible impact and a rare likelihood of exploitation.

CRITICAL **HIGH** **MEDIUM** Low Informational

Likelihood Matrix:

Attack Complexity \ Discovery Ease	Obvious	Concealed	Hidden
Complex	Possible	Rare	Rare
Moderate	Likely	Possible	Rare
Straightforward	Likely	Possible	Possible

Likelihood/Impact Matrix:

Likelihood \ Impact	Severe	Moderate	Negligible
Likely	**CRITICAL**	**HIGH**	**MEDIUM**
Possible	**HIGH**	**MEDIUM**	Low
Rare	**MEDIUM**	Low	Informational

Findings Summary

01. Remediation Complexity: This measures how difcult it is to fx the vulnerability once it has been identifed.

Simple: Patches or fixes are readily available and easily implemented.

Moderate: Requires some time and resources to remediate, but well within the capabilities of most organizations.

Difficult: Remediation requires significant resources, specialized skills, or substantial changes to systems or architecture.

02. Status: This measures how difcult it is to fx the vulnerability once it has been identifed.

Not Fixed: Indicates that the vulnerability has been identifed but no remedial action has been taken yet. This status is crucial for newly discovered vulnerabilities or those awaiting prioritization.

Fixed: This status is applied when the vulnerability has been successfully remediated. It implies that appropriate measures (like patching, confguration changes, or architectural modifcations) have been implemented to resolve the issue.

Acknowledged: This status is used for vulnerabilities that have been recognized, but for various reasons (such as risk acceptance, cost, or other business decisions), have not been fxed. It indicates that the risk posed by the vulnerability is known and has been consciously accepted.

Scheduled: This status indicates that the vulnerability has been acknowledged and a plan is in place to fix it in the future. It signifies that while remediation hasn't yet occurred, the issue has been prioritized and is part of the planned development roadmap.

Finding	Impact	Likelihood	Severity	Remediation Complexity	Remediation Status
GUV-1: Possible overflows	Moderate	Rare	Low	Simple	Fixed
GUV-2: Possible misinterpretation of the APY value.	Negligible	Rare	Informational	Moderate	Fixed
GUV-3: Possible Booster lifecycle panic	Negligible	Rare	Informational	Simple	Fixed
GUV-4: Inconsistent global APY clamping	Negligible	Rare	Informational	Simple	Fixed

Findings Details

GUV-1 Possible overflows - Low

If the booster is active, the **DailyScore::to_capped_apy** calculates the **`self.value.min(cap) + self.booster`**, each individual component being a **`u16`** types. When added together, they might overflow, if their sum is bigger than **`u16::MAX`**. The contract enables overflow-checks so whenever it would be encounter, the contract will panic instead.

model/src/data/score/mod.rs:to_capped_apy

```
pub fn to_capped_apy(&self, cap: Score, include_booster: bool) -> UDecimal {
        (self.value.min(cap) + if include_booster { self.booster } else { 0 }).to_apy()
    }
```

Another case is of a possible overflow is in **get_interest** function. It calculates the interest as **`term_in_milliseconds * apy * principal`** using **`u128`** type. That type can contain large numbers, so an actual overflow in production environment is not likely, but still possible given the right combination of values.

Recommendation

It is recommended to use the the **`saturating_`** variants of mathematical operations to make sure overflow is not encountered.

Remediation - Fixed

The Sweat Foundation team has fixed the issue in this commit:
607f3f61ab38faed74fe23e16ee825a55c289f17

GUV-2 Possible misinterpretation of the APY value. - Informational

100% APY (as indicated in the PR docs) seems to not be possible due to type constraints

The APY is represented as **`u16`** which is then converted to **`UDecimal`** type. The "encoding" makes **1% = 1000**. Since **`u16`**'s max value is ~65k, the max APY that can be represented is about **65.5%**. This, however, might not be an issue, depending if the PR description is correct.

Recommendation

An altnerative type representing the Score can be considered, **`u32`** will be able to represent the whole range of per-cent values using the current encoding.

Remediation - Fixed

The Sweat Foundation team has fixed the issue in this commit:
607f3f61ab38faed74fe23e16ee825a55c289f17

GUV-3 Possible Booster lifecycle panic - Informational

First, the booster timestamp might case a panic within the contract if it's older than **`DAYS_STORED`** in the past. This panic requires an oracle to pass an older-than-expected value, which possibly is not expected to happen, however for resiliency, it would be advised to not panic, and instead simply ignore those values.

Recommendation

We recommend gracefully handling the errorouns cases here instead of panicing.

Remediation - Fixed

The Sweat Foundation team has fixed the issue in these commits:
3f60fa06e1530aa743e6d230cb028deede60614e and
8abef8c4b010de7a9aa8ba9be38da000adda98b5

GUV-4 Inconsistent global APY clamping - Informational

The current APY cap is implemented as capping the value and adding booster ontop of it, which might increase the total APY to be above the defined cap. This might be intended, however. Documentation suggests that it's not.

Recommendation

We recommend double checking the design to make sure which approach is the intended one and changing the implementation (if needed) accordingly - making the cap act on a sum of actual value with a booster.

Remediation - Fixed

The Sweat Foundation team has fixed the issue in this commit:
607f3f61ab38faed74fe23e16ee825a55c289f17