# SubGame Stake Audit Report

Version 1.0.0

Serial No. 2021072900022027

Presented by Fairyproof

June 29, 2021

FAIRYPROOF

# 01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the SubGame Stake module, at the request of the SubGame team.

**Audited Code's Github Repository:**

https://github.com/SubGame-Network/subgame-network/tree/develop

**Audited Code's Github Commit Number:**

f56d17b8ee2e5375d6c4ae42213c552edc4b076d

**Audited Source Files:**

The calculated SHA-256 values for the audited files when the audit was done are as follows:

```
default_weight.rs: 0xac172deb8a6065484c81dd84f82cc83f6969e9da6826913173648ad23eda771d
lib.rs           : 0x20f526829ad7c7187bef7e11d53eba3f8d8414fa5dec8e0bfc58d79a33835ebf
mock.rs          : 0x21637f7f9c469cadf7503942a1a255e979e96cb166047e1253e003ee4288e694
tests.rs         : 0x935bb300a11e28a0e7b15372672b2b70651a02c5dbcb1948e4b62a7bbf019d02
chain_spec.rs    : 0x0f7b5aa310a17f5f8993a4acdaacf7d1da990e14762cc8e8511e4e744ccb3741
constants.rs     : 0x05486a5a2b8256d21a325bbfa4ff39cddd01c6fcd345fd7e924f83566c1edcb8
lib.rs           : 0x5814a8eb2fc3ff5d698f691cbb0431ba1e2b4b1788e3d887744ed8078445c1ee
```

The source files audited include all the files with the extension ".rs" as follows:

```
develop/
├── pallets/stake/src
|   ├── default_weight.rs
|   ├── lib.rs
|   ├── mock.rs
|   └── tests.rs
├── node/src
|   └── chain_spec.rs
└── runtime/src
    ├── constants.rs
    └── lib.rs
```

The goal of this audit is to review SubGame Stake's Rust implementation for its registry, staking and withdrawal functions, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

# — Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding system security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from offchain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

# — Methodology

The above files' code was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The codebase was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Fairyproof auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Fairyproof to make sure we understand the size, scope, and functionality of the project's source code.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the

specifications, sources, and instructions provided to Fairyproof describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run the test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the source code to improve maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

## — Structure of the document

This report contains a list of issues and comments on all the above source files. Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

## — Documentation

For this audit, we used the following sources of truth about how the SubGame Stake should work:

https://www.subgame.org/

These were considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the SubGame team or reported an issue.

## — Comments from Auditor

No vulnerabilities with critical, high, medium or low-severity were found in the above source code.

# 02. About Fairyproof

Fairyproof is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Fairyproof has developed industry security standards for designing and deploying blockchain applications.

# 03. Introduction to SubGame

SubGame is a public chain based on the Polkadot Parachain technology. It hopes to build a public blockchain with cross-chain interoperability. In addition to creating game applications, it can also build various types of application scenarios to create a common cross-chain industry. The blockchain infrastructure provides unlimited possibilities for distributed mobile applications.

# 04. Major functions of audited code

The audited code implements SubGame's staking module. This module provides services to external users or applications by allowing them to register, stake and withdraw. The code implementation validates access control, account length, account balances, global states etc.

# 05. Key points in audit

During the audit, we worked closely with the SubGame team, did static analysis and formal verification, and executed unit tests in the following areas:

- arithmic operations
- DoS attacks
- setting of transaction fees
- random number generation
- exception handling
- consumption of resources
- design and implementation of tokenomics (such as transaction fees, governance etc)

# 06. Coverage of issues

The issues that the Fairyproof team covered when conducting the audit include but are not limited to the following ones:

- Memory Leak
- Unreleased Resources
- Overflow/Underflow
- Integer Truncation
- Goroutine Unsafe Exiting
- Closing Channel Twice
- Missing Check for Unsafe External Input
- Unsafe Encryption
- Unsafe Random Number Generation
- Data Security
- DOS Attack
- Security in Consensus Algorithm
- Handling of Block
- Handling of Transaction
- Database Security
- Security of Third-party Libraries
- Logic Vulnerability

# 07. Severity level reference

Every issue in this report was assigned a severity level from the following:

**Critical** severity issues need to be fixed as soon as possible.

**High** severity issues will probably bring problems and should be fixed.

**Medium** severity issues could potentially bring problems and should eventually be fixed.

**Low** severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

# 08. Major areas that need attention

Based on the provided files the Fairyproof team focused on the possible issues and risks related to the following functions or areas.

## - Arithmic Operations

We checked all the code sections, which have arithmetic operations and might introduce overflow or underflow.

We didn't find issues or risks in these functions or areas at the time of writing.

## - DoS Attacks

We checked whether or not the audited code might suffer DoS attacks.

We didn't find issues or risks in these functions or areas at the time of writing.

## - Random Number Generation

We checked whether or not there were issues in random number generation.

We didn't find issues or risks in these functions or areas at the time of writing.

## - Exception Handling

We checked whether or not there were proper exception handlings.

We didn't find issues or risks in these functions or areas at the time of writing.

## - Miscellaneous

We didn't find issues or risks in other functions or areas at the time of writing.

## 09. List of issues by severity

### A. Critical

- N/A

### B. High

- N/A

### C. Medium

- N/A

### D. Low

- N/A

## 10. List of issues by source file

- N/A

## 11. Issue descriptions

- N/A

# 12. Recommendations to enhance the overall security

We list some recommendations in this section. They are not mandatory but will enhance the overall security of the system if they are adopted.

## - N/A