# SPYWOLF

## Security Audit Report

Audit prepared for

**Utherverse: Token Claim**

Completed on

**December 1, 2024**

# OVERVIEW

This goal of this report is to review the main aspects of the project to help investors make an informative decision during their research process.

You will find a a summarized review of the following key points:

- ✔ Program’s source code
- ✔ Team transparency and goals
- ✔ Website's age, code, security and UX
- ✔ Whitepaper and roadmap
- ✔ Social media & online presence

*The results of this audit are purely based on the team’s evaluation and does not guarantee nor reflect the projects outcome and goal*

- SPYWOLF Team -

# TABLE OF CONTENTS

# UTHERVERSE (Token/Staking)

## PROJECT DESCRIPTION

Utherverse is not just another player in the metaverse space – we are the pioneers, with over 15 years of experience in building successful virtual economies and communities. Our track record speaks for itself, but our vision for the future is what truly sets us apart. By harnessing the immense potential of web3, blockchain, and AI, we are creating a metaverse that is unmatched in its immersion, adaptability, and profitability.

**Release Date:** Launches in November, 2024

**Category:** Token Claim

# CLAIM PROGRAM

Token Name
Claim Program

Symbol

Program Address
GJdNPUnhyjz4x47fibedUxmLY7Xx42E6SrQGfEJUzj9S

Network
Solana

Language
Rust

Deployment Date
Nov 18, 2024

Program Type
Claim

Total Supply

Status
mainnet

# TAXES

Buy Tax
0

Sell Tax
0

*This type of program does not have taxes

# Our Program Review Process

The contract review process pays special attention to the following:

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

**Blockchain security tools used:**

- Solana Program Library (SPL)
- Manual Auditing / Sec3 / Neodyme
- Rust Compiler
- Anchor Framework

# CODE REVIEW

| Vulnerability | |
|---|---|
| Insecure Access Control for Beneficiary Updates | Passed |
| State Update vs. Transfer Atomicity | Passed |
| Escrow Wallet Authority Hijack | Passed |
| Infinite Token Minting via Malicious Initialization | Passed |
| No Recovery Mechanism for Unclaimed Tokens | Passed |
| Program Initialization and Transaction Validation | Passed |

# VULNERABILITY ANALYSIS

## ERRORS FOUND

### 1. Insecure Access Control for Beneficiary Updates (Fixed)

Passed

The update_bulk_user_status function allows bulk updates to beneficiary statuses without verifying the caller's identity. This allows unauthorized actors to maliciously update all beneficiaries to is_claimed = true, effectively locking funds.

**Real Impact**

- Malicious actors can disrupt the claim process, locking all escrowed tokens permanently.
- Potential loss of user trust and system functionality.

**Fix Details:**

- Strong access control using Anchor constraints ensures that only the initializer wallet can update beneficiary statuses.
- The admin panel workflow adds an additional layer of verification and security.

**Assessment:**

- The combination of programmatic access control and a secure, manually verified admin workflow eliminates the risk of unauthorized updates. Given the client's trustworthiness, this can be classified as: **PASSED**

# VULNERABILITY ANALYSIS

## ERRORS FOUND

### 2. State Update vs. Transfer Atomicity (Fixed)

Passed

The claim function updates the beneficiary's state (is_claimed and in_process) before completing the token transfer. If the transfer fails (e.g., due to insufficient escrow balance or network issues), the state becomes inconsistent.

**Real Impact**

- Claims may be marked as completed even though tokens are not transferred.
- Loss of claim rights for the beneficiary due to inconsistent state.

**Fix Details:**

- Atomicity between token transfers and state updates ensures consistency.
- If the transfer fails, no state updates occur, maintaining data integrity.

**Assessment:**

- This fix fully resolves the inconsistency issue. The mechanism is robust and unlikely to fail under normal circumstances. This can be classified as: **PASSED**

# VULNERABILITY ANALYSIS

## ERRORS FOUND

### 3. Escrow Wallet Authority Hijack (Fixed)

Passed

The escrow_wallet authority is tied to a mutable account rather than a Program Derived Address (PDA). If the private key of the escrow authority is compromised, an attacker can drain all escrowed funds.

**Real Impact**

- Full loss of funds held in escrow.
- Severe damage to the program's integrity.

**Fix Details:**

- The escrow wallet authority is securely tied to a PDA, eliminating the risk of external hijacking.

**Assessment:**

- This implementation ensures that only the program can manage escrow funds, and the PDA approach is a standard best practice. It fully resolves the risk, so this can also be classified as: **PASSED**

# VULNERABILITY ANALYSIS

## ERRORS FOUND

### 4. Infinite Token Minting via Malicious Initialization (Fixed)

Passed

The initialize function does not enforce limits on the allocation amount or validate the beneficiaries. This allows malicious actors to:

1. Over-allocate tokens, devaluing the token supply.
2. Create fake beneficiaries, disrupting legitimate use.

**Real Impact**

- Economic devaluation due to excessive token allocation.
- Spam attacks, making the program unusable for genuine participants.

**Fix Details:**

- Backend input validations and manual verification by the admin ensure no excessive or malicious minting occurs.
- Data preprocessing and admin-controlled initialization remove the possibility of exploitation.

**Assessment:**

- The combination of backend validation, testing, and admin-controlled initialization provides a robust solution. This issue can be considered: **PASSED**

# VULNERABILITY ANALYSIS

## ERRORS FOUND

### 5. No Recovery Mechanism for Unclaimed Tokens (Fixed)

Passed

Tokens unclaimed by beneficiaries remain locked indefinitely, reducing the total token supply. This can occur due to:

- Misconfigured accounts.
- Forgotten or abandoned claims.

**Real Impact**

- Permanent loss of unclaimed tokens.
- Reduction in available supply for legitimate use.

**Fix Details:**

- The **withdraw_from_escrow()** function allows the admin to recover unclaimed tokens, ensuring no funds are permanently locked.

**Assessment:**

- The addition of a recovery mechanism resolves the issue entirely. Since the admin is the sole trusted entity, this can be classified as: **PASSED**

# VULNERABILITY ANALYSIS

## ERRORS FOUND

### Program Initialization: PASSED

Passed

- Initialization parameters are correctly enforced in terms of setting deadlines and token allocation amounts.
- Account derivations are secure, with proper seeds and bump validations.

### Transaction Validation: PASSED

Passed

- Critical functions validate inputs, avoiding overflows or underflows.
- Function reentrancy is effectively prevented using temporary state markers (e.g., in_process).

# STAKE PROGRAM

Token Name
Stake Program

Symbol

Program Address

Network
Solana

Language
Rust

Deployment Date
Nov 18, 2024

Program Type
Stake

Total Supply

Status
mainnet

## TAXES

Buy Tax
0

Sell Tax
0

*This type of program does not have taxes

# Our Program Review Process

The contract review process pays special attention to the following:

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

**Blockchain security tools used:**

- Solana Program Library (SPL)
- Manual Auditing / Sec3 / Neodyme
- Rust Compiler
- Anchor Framework

# CODE REVIEW

| Vulnerability | |
|---|---|
| Integer Overflow/Underflow | Passed |
| Lack of Input Validation for admin_withdraw | Passed |
| Unauthorized Access in update_pool_info | Low Risk |
| No Handling for Division by Zero | Passed |
| Stake Seed Conflict Check | Passed |
| Token Transfer Validation and Stake Initialization | Passed |
| Stake Initialization and Withdrawal | Passed |

# VULNERABILITY ANALYSIS

## ERRORS FOUND

### 1. Integer Overflow/Underflow (Fixed)

Passed

Unchecked arithmetic operations in calculations, such as rewards or staking amounts, may lead to overflow or underflow errors. While checked_mul is used in some instances, other areas, such as reward rate calculations, lack similar safeguards.

**Attack Scenario**

A malicious user could manipulate apy, apy_denominator, or other parameters to trigger overflow or underflow. This could result in:

- Excessive rewards credited to themselves.
- Locking up all tokens due to miscalculated reward rates.

**Fix Details:**

- The program now uses **checked_*** arithmetic methods (e.g., **checked_mul, checked_div**) throughout critical calculations like rewards and staking amounts.
- Rust's ? operator ensures safe handling of results, returning early on arithmetic errors.

**Assessment:**

- The use of safe arithmetic eliminates the risk of overflow or underflow in calculations. There are no remaining technical vulnerabilities here.
- Changed risk to **PASSED**

# VULNERABILITY ANALYSIS

## ERRORS FOUND

### 2. Lack of Input Validation for admin_withdraw Function (Fixed)

Passed

The admin_withdraw function does not validate the withdrawal amount, allowing requests for amounts exceeding the vault's balance.

**Real Impact**

- A failed transaction due to insufficient funds may freeze the contract.
- Malicious or careless admins could destabilize the program.

**Fix Details:**

- Added validation to ensure the vault has sufficient funds before processing withdrawal requests.
- Any withdrawal exceeding the vault balance now results in an error.

**Assessment:**

- The input validation ensures that withdrawals cannot exceed available funds, fully addressing the issue.
- There are no residual risks, assuming the admin account is secure.
- Changed risk to **PASSED**

# VULNERABILITY ANALYSIS

## ERRORS FOUND

### 3. Unauthorized Access in update_pool_info (Fixed)

Low Risk

The update_pool_info function relies on the admin's public key passed as a parameter. This lacks robust verification and could allow an attacker to spoof the admin address.

**Explanation:**

- An attacker could hijack pool control by manipulating the admin key.
- Lack of strict verification increases the risk of unauthorized access.

**Fix Details:**

- Access control now validates the caller against the stored admin key **(ctx.accounts.admin.key() != ctx.accounts.pool_info.admin)**.
- The admin key can no longer be spoofed, and changes require strict verification.

**Assessment:**

- This fix directly addresses the issue, ensuring only authorized updates. However, residual operational risks (e.g., admin key compromise) persist if external factors are not fully controlled.
- Changed risk to **Low Risk**

# VULNERABILITY ANALYSIS

## ERRORS FOUND

### 4. No Handling for Division by Zero (Fixed)

Passed

The program performs division operations without validating divisors. If apy_denominator is zero, it may cause a panic or undefined behavior.

**Explanation**

- Malicious actors setting apy_denominator to zero.
- Bugs or improper state updates.

**Fix Details:**

- Division operations now include checks to ensure denominators are non-zero (**if pool_info.apy_denominator == 0 { return Err(ErrorCode::InvalidApyDenominator.into()); }**).
- The contract fails gracefully if division by zero is attempted.

**Assessment:**

- This solution eliminates technical risks. There are no remaining vulnerabilities from division by zero.
- Changed risk to **Passed**

# VULNERABILITY ANALYSIS

## ERRORS FOUND

### 5. Stake Seed Conflict Check (Fixed)

Passed

The DeStake function does not ensure that stake_seed is unique. Conflicting seeds could lead to one user's stake being overwritten by another's.

**Explanation**

- Accidental or malicious reuse of stake_seed could disrupt staking data.
- Users may lose their stake due to overwritten accounts.

**Fix Details:**

- Added a check to ensure stake_seed uniqueness before creating a new stake (**if StakeInfo::fetch_by_seed(stake_seed).is_some() { return Err(ErrorCode::StakeSeedConflict.into()); }**).
- Prevents accidental or malicious duplication of stake seeds.

**Assessment:**

- The uniqueness check fully resolves the issue. There are no operational or technical risks left.
- Changed risk to **Passed**

# VULNERABILITY ANALYSIS

## ERRORS FOUND

### 6. Token Transfer Validation: PASSED

Passed

- The program correctly validates token transfers and ensures that accounts are correctly authorized.
- Checks for ownership and sufficient token balances are implemented.

### 7. Stake Initialization and Withdrawal: PASSED

Passed

- The stake initialization process enforces deadlines and minimum staking amounts.
- Token withdrawals are checked for eligibility, preventing unauthorized access.

# ABOUT US

We are a growing crypto security agency offering audits, KYCs and consulting services for some of the top names in the crypto industry.

- **OVER 700 SUCCESSFUL CLIENTS**
- **MORE THAN 1000 SCAMS EXPOSED**
- **MILLIONS SAVED IN POTENTIAL FRAUD**
- **PARTNERSHIPS WITH TOP LAUNCHPADS, INFLUENCERS AND CRYPTO PROJECTS**
- **CONSTANTLY BUILDING TOOLS TO HELP INVESTORS DO BETTER RESEARCH**

SPYWOLF

CRYPTO SECURITY

Audits | KYCs | dApps
Contract Development

To hire us, reach out to
contact@spywolf.co or
t.me/joe_SpyWolf

## FIND US ONLINE

SPYWOLF.CO

@SPYWOLFNETWORK

@SPYWOLFNETWORK

# Disclaimer

This report shows findings based on our limited project analysis, following good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall social media and website presence and team transparency details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report.
While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

**DISCLAIMER:**
By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice.
No one shall have any right to rely on the report or its contents, and SpyWolf and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (SpyWolf) owe no duty of care towards you or any other person, nor does SpyWolf make any warranty or representation to any person on the accuracy or completeness of the report.
The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SpyWolf hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SpyWolf hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SpyWolf, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts, website, social media and team.
No applications were reviewed for security. No product code has been reviewed.