Airdrop Contract

deBridge

HALBORN

Report a Bug

Airdrop Contract - deBridge

Prepared by: HALBORN Last Updated 10/16/2024 Date of Engagement by: July 29th, 2024 - October 1st, 2024

Summary

100% ⓘ OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
3	0	0	0	0	3

1. Introduction

deBridge engaged Halborn to conduct a security assessment on their updated **Claiming Solana program** beginning on 07/29/2024, and ending on 01/10/2024. The security assessment was scoped to the Solana Programs provided in points-claimer-solana GitHub repository. Commit hashes and further details can be found in the Scope section of this report.

The Claiming Program allows users to claim tokens based on their point score, supporting addresses from both Solana and Ethereum Virtual Machine (EVM) chains. The eligibility and amount of tokens that can be claimed are defined in a data source which specifies the addresses and their corresponding claimable amounts. The program uses a Merkle tree for secure and efficient verification of claims.

The deBridge team provided three updated versions of the program. The first updated version of the claiming program introduces several new features:

- The program authority can now create multiple "seasonal" allocations, allowing users to claim their tokens after specific start slots.
- The program authority can also blacklist specific users for a given season.
- The terms and conditions of signature verification for Solana users were modified.
- The code base has been refactored and modularized into multiple files.

The second updated version adds the following feature to the program:

- The claim instruction using EVM address now has full and short variants of terms to be signed.

The third updated version introduces only minor, non-functional changes.

2. Assessment Summary

`Halborn` was provided 5 weeks for the engagement and assigned one full-time security engineer to review the security of the Solana Programs in scope. The engineer is a blockchain and smart contract security expert with advanced smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of the assessment is to:

- Identify potential security issues within the Solana Programs.
- Ensure that smart contract functionality operates as intended.

In summary, `Halborn` did not identify any major security concerns and all informational issues have been acknowledged and addressed by `deBridge team`. The informational issue "Possibility to use incorrect mint to claim tokens" has been addressed and solved.

The remaining two informational issues were acknowledged:

- Proof length as env variable risks token claim failures
- State authority can withdraw tokens without restrictions

3. Test Approach And Methodology

Halborn performed a combination of a manual review of the source code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the program assessment. While manual testing is recommended to uncover flaws in business logic, processes, and implementation; automated testing techniques help enhance coverage of programs 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 assessment:

- Research into the architecture, purpose, and use of the platform.

- Manual program source code review to identify business logic issues.

- 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.

- Scanning dependencies for known vulnerabilities (`cargo audit`).

- Local Anchor testing (`anchor test`)

4. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

4.1 EXPLOITABILITY

ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

ATTACK COST (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

ATTACK COMPLEXITY (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

METRICS:

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

4.2 IMPACT

CONFIDENTIALITY (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

INTEGRITY (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

AVAILABILITY (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

DEPOSIT (D):

Measures the impact to the deposits made to the contract by either users or owners.

YIELD (Y):

Measures the impact to the yield generated by the contract for either users or owners.

METRICS:

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

4.3 SEVERITY COEFFICIENT

REVERSIBILITY (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

TABLE OF CONTENTS

SCOPE (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

5. SCOPE

FILES AND REPOSITORY

(a) Repository: points-claimer-solana

(b) Assessed Commit ID: a729b54

(c) Items in scope:

- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/Cargo.toml
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/Xargo.toml
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/build.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/constants.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/error.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/ban.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/claim_evm.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/claim_solana.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/create_season.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/update_state_any_time.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/update_state_before_started.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/withdraw.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/lib.rs

- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/merkle.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/state.rs
- debridge-finance/points-claimer-solana/blob/main/template.txt

Out-of-Scope: debridge-finance/points-claimer-solana/blob/main/.cargo/config, debridge-finance/points-claimer-solana/blob/main/.docker/Dockerfile.contracts, debridge-finance/points-claimer-solana/blob/main/.docker/Dockerfile.reader, debridge-finance/points-claimer-solana/blob/main/.github/workflows/release-tag.yml, debridge-finance/points-claimer-solana/blob/main/.github/workflows/rust.yml, debridge-finance/points-claimer-solana/blob/main/.gitignore, debridge-finance/points-claimer-solana/blob/main/.gitmodules, debridge-finance/points-claimer-solana/blob/main/.prettierignore, debridge-finance/points-claimer-solana/blob/main/Anchor.toml, debridge-finance/points-claimer-solana/blob/main/Cargo.lock, debridge-finance/points-claimer-solana/blob/main/Cargo.toml, debridge-finance/points-claimer-solana/blob/main/README.md, debridge-finance/points-claimer-solana/blob/main/claim-reader/Cargo.toml, debridge-finance/points-claimer-solana/blob/main/claim-reader/build.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/chain_id.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/config.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/evm_consumer.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/main.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/producer.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/solana_parser.rs, debridge-finance/points-claimer-solana/blob/main/claim.proto, debridge-finance/points-claimer-solana/blob/main/cli/abi/gate.json, debridge-finance/points-claimer-solana/blob/main/cli/claim.evm.multisig.ts, debridge-finance/points-claimer-solana/blob/main/cli/claim.evm.ts, debridge-finance/points-claimer-solana/blob/main/cli/claim.sol.ts, debridge-finance/points-claimer-solana/blob/main/cli/constants.ts, debridge-finance/points-claimer-solana/blob/main/cli/createTree.ts, debridge-finance/points-claimer-

solana/blob/main/cli/mt.ts, debridge-finance/points-claimer-solana/blob/main/common/Cargo.toml, debridge-finance/points-claimer-solana/blob/main/common/src/lib.rs, debridge-finance/points-claimer-solana/blob/main/dln-protobuf, debridge-finance/points-claimer-solana/blob/main/docker-compose.yml, debridge-finance/points-claimer-solana/blob/main/extcall_program.json, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/.gitignore, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/Cargo.toml, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/accounts.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/errors.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/events.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/instructions.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/lib.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/typedefs.rs, debridge-finance/points-claimer-solana/blob/main/migrations/deploy.ts, debridge-finance/points-claimer-solana/blob/main/package.json, debridge-finance/points-claimer-solana/blob/main/src/README.md, debridge-finance/points-claimer-solana/blob/main/src/client.ts, debridge-finance/points-claimer-solana/blob/main/src/idl/secp_recover_poc.ts, debridge-finance/points-claimer-solana/blob/main/tests/client.test.ts, debridge-finance/points-claimer-solana/blob/main/tests/secp-recover-poc.ts, debridge-finance/points-claimer-solana/blob/main/tsconfig.json, debridge-finance/points-claimer-solana/blob/main/yarn.lock

FILES AND REPOSITORY

(a) Repository: points-claimer-solana

(b) Assessed Commit ID: 3aa89be

(c) Items in scope:

- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/Cargo.toml
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/Xargo.toml
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/build.rs

- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/constants.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/error.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/event.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/ban.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/claim_evm.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/claim_solana.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/create_season.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/update_state_any_time.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/update_state_before_started.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/withdraw.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/lib.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/merkle.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/state.rs
- debridge-finance/points-claimer-solana/blob/main/Anchor.toml
- debridge-finance/points-claimer-solana/blob/main/Cargo.lock
- debridge-finance/points-claimer-solana/blob/main/Cargo.toml
- debridge-finance/points-claimer-solana/blob/main/common/Cargo.toml
- debridge-finance/points-claimer-solana/blob/main/common/src/lib.rs
- debridge-finance/points-claimer-solana/blob/main/short_template.txt
- debridge-finance/points-claimer-solana/blob/main/template.txt

Out-of-Scope: debridge-finance/points-claimer-solana/blob/main/.cargo/config.toml, debridge-finance/points-claimer-solana/blob/main/.docker/Dockerfile.contracts, debridge-finance/points-claimer-solana/blob/main/.docker/Dockerfile.reader, debridge-finance/points-claimer-solana/blob/main/.github/workflows/release-tag.yml, debridge-finance/points-claimer-solana/blob/main/.github/workflows/rust.yml, debridge-finance/points-claimer-solana/blob/main/.gitignore, debridge-finance/points-claimer-solana/blob/main/.gitmodules, debridge-finance/points-claimer-solana/blob/main/.prettierignore, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/Cargo.toml, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/Xargo.toml, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/build.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/constants.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/error.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/event.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/ban.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/claim_evm.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/claim_solana.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/create_season.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/update_state_any_time.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/update_state_before_started.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/withdraw.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/lib.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/merkle.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/state.rs, debridge-finance/points-claimer-solana/blob/main/dln-protobuf, debridge-finance/points-claimer-solana/blob/main/docker-compose.yml, debridge-finance/points-claimer-solana/blob/main/extcall_program.json, debridge-finance/points-claimer-

solana/blob/main/extcall_program_interface/.gitignore, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/Cargo.toml, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/accounts.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/errors.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/events.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/instructions.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/lib.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/typedefs.rs, debridge-finance/points-claimer-solana/blob/main/migrations/deploy.ts, debridge-finance/points-claimer-solana/blob/main/package.json, debridge-finance/points-claimer-solana/blob/main/src/README.md, debridge-finance/points-claimer-solana/blob/main/src/client.ts, debridge-finance/points-claimer-solana/blob/main/src/idl/secp_recover_poc.ts, debridge-finance/points-claimer-solana/blob/main/tests/client.test.ts, debridge-finance/points-claimer-solana/blob/main/tests/secp-recover-poc.ts, debridge-finance/points-claimer-solana/blob/main/tsconfig.json, debridge-finance/points-claimer-solana/blob/main/yarn.lock, debridge-finance/points-claimer-solana/blob/main/.cargo/config.toml, debridge-finance/points-claimer-solana/blob/main/.docker/Dockerfile.contracts, debridge-finance/points-claimer-solana/blob/main/.docker/Dockerfile.reader, debridge-finance/points-claimer-solana/blob/main/.github/workflows/release-tag.yml, debridge-finance/points-claimer-solana/blob/main/.github/workflows/rust.yml, debridge-finance/points-claimer-solana/blob/main/.gitignore, debridge-finance/points-claimer-solana/blob/main/.gitmodules, debridge-finance/points-claimer-solana/blob/main/.prettierignore, debridge-finance/points-claimer-solana/blob/main/README.md, debridge-finance/points-claimer-solana/blob/main/claim-reader/Cargo.toml, debridge-finance/points-claimer-solana/blob/main/claim-reader/build.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/chain_id.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/config.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/consistency_checker.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/consistency_checker/slot_rollup_iterator.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/consistency_repo.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/evm_consumer.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/main.rs, debridge-finance/points-

claimer-solana/blob/main/claim-reader/src/producer.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/solana_parser.rs, debridge-finance/points-claimer-solana/blob/main/claim.proto, debridge-finance/points-claimer-solana/blob/main/cli/abi/gate.json, debridge-finance/points-claimer-solana/blob/main/cli/claim.evm.multisig.ts, debridge-finance/points-claimer-solana/blob/main/cli/claim.evm.ts, debridge-finance/points-claimer-solana/blob/main/cli/claim.sol.ts, debridge-finance/points-claimer-solana/blob/main/cli/constants.ts, debridge-finance/points-claimer-solana/blob/main/cli/createTree.ts, debridge-finance/points-claimer-solana/blob/main/cli/mt.ts

FILES AND REPOSITORY

(a) Repository: points-claimer-solana

(b) Assessed Commit ID: 7ac31e6

(c) Items in scope:

- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/Cargo.toml
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/Xargo.toml
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/build.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/constants.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/error.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/event.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/ban.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/claim_evm.rs

- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/claim_solana.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/create_season.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/update_state_any_time.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/update_state_before_started.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/withdraw.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/lib.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/merkle.rs
- debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/state.rs
- debridge-finance/points-claimer-solana/blob/main/Anchor.toml
- debridge-finance/points-claimer-solana/blob/main/Cargo.lock
- debridge-finance/points-claimer-solana/blob/main/Cargo.toml
- debridge-finance/points-claimer-solana/blob/main/common/Cargo.toml
- debridge-finance/points-claimer-solana/blob/main/common/src/lib.rs
- debridge-finance/points-claimer-solana/blob/main/short_template.txt
- debridge-finance/points-claimer-solana/blob/main/template.txt

Out-of-Scope: debridge-finance/points-claimer-solana/blob/main/.cargo/config.toml, debridge-finance/points-claimer-solana/blob/main/.docker/Dockerfile.contracts, debridge-finance/points-claimer-solana/blob/main/.docker/Dockerfile.reader, debridge-finance/points-claimer-solana/blob/main/.github/workflows/release-tag.yml, debridge-finance/points-claimer-solana/blob/main/.github/workflows/rust.yml, debridge-finance/points-claimer-solana/blob/main/.gitignore, debridge-finance/points-claimer-solana/blob/main/.gitmodules, debridge-finance/points-claimer-solana/blob/main/.prettierignore, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/Cargo.toml, debridge-finance/points-

claimer-solana/blob/main/programs/points-claimer/Xargo.toml, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/build.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/constants.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/error.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/event.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/ban.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/claim_evm.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/claim_solana.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/create_season.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/update_state_any_time.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/update_state_before_started.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/instructions/withdraw.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/lib.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/merkle.rs, debridge-finance/points-claimer-solana/blob/main/programs/points-claimer/src/state.rs, debridge-finance/points-claimer-solana/blob/main/dln-protobuf, debridge-finance/points-claimer-solana/blob/main/docker-compose.yml, debridge-finance/points-claimer-solana/blob/main/extcall_program.json, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/.gitignore, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/Cargo.toml, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/accounts.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/errors.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/events.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/instructions.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/lib.rs, debridge-finance/points-claimer-solana/blob/main/extcall_program_interface/src/typedefs.rs, debridge-finance/points-claimer-solana/blob/main/migrations/deploy.ts, debridge-finance/points-claimer-solana/blob/main/package.json, debridge-finance/points-claimer-solana/blob/main/src/README.md, debridge-finance/points-claimer-

solana/blob/main/src/client.ts, debridge-finance/points-claimer-solana/blob/main/src/idl/secp_recover_poc.ts, debridge-finance/points-claimer-solana/blob/main/tests/client.test.ts, debridge-finance/points-claimer-solana/blob/main/tests/secp-recover-poc.ts, debridge-finance/points-claimer-solana/blob/main/tsconfig.json, debridge-finance/points-claimer-solana/blob/main/yarn.lock, debridge-finance/points-claimer-solana/blob/main/.cargo/config.toml, debridge-finance/points-claimer-solana/blob/main/.docker/Dockerfile.contracts, debridge-finance/points-claimer-solana/blob/main/.docker/Dockerfile.reader, debridge-finance/points-claimer-solana/blob/main/.github/workflows/release-tag.yml, debridge-finance/points-claimer-solana/blob/main/.github/workflows/rust.yml, debridge-finance/points-claimer-solana/blob/main/.gitignore, debridge-finance/points-claimer-solana/blob/main/.gitmodules, debridge-finance/points-claimer-solana/blob/main/.prettierignore, debridge-finance/points-claimer-solana/blob/main/README.md, debridge-finance/points-claimer-solana/blob/main/claim-reader/Cargo.toml, debridge-finance/points-claimer-solana/blob/main/claim-reader/build.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/chain_id.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/config.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/consistency_checker.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/consistency_checker/slot_rollup_iterator.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/consistency_repo.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/evm_consumer.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/main.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/producer.rs, debridge-finance/points-claimer-solana/blob/main/claim-reader/src/solana_parser.rs, debridge-finance/points-claimer-solana/blob/main/claim.proto, debridge-finance/points-claimer-solana/blob/main/cli/abi/gate.json, debridge-finance/points-claimer-solana/blob/main/cli/claim.evm.multisig.ts, debridge-finance/points-claimer-solana/blob/main/cli/claim.evm.ts, debridge-finance/points-claimer-solana/blob/main/cli/claim.sol.ts, debridge-finance/points-claimer-solana/blob/main/cli/constants.ts, debridge-finance/points-claimer-solana/blob/main/cli/createTree.ts, debridge-finance/points-claimer-solana/blob/main/cli/mt.ts

REMEDIATION COMMIT ID:

- 8a37b39

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW
0	0	0	0

INFORMATIONAL
3

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
STATE AUTHORITY CAN WITHDRAW TOKENS WITHOUT RESTRICTIONS	INFORMATIONAL	ACKNOWLEDGED
PROOF LENGTH AS ENV VARIABLE RISKS TOKEN CLAIM FAILURES	INFORMATIONAL	ACKNOWLEDGED

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
POSSIBILITY TO USE INCORRECT MINT TO CLAIM TOKENS	INFORMATIONAL	SOLVED - 08/06/2024

7. FINDINGS & TECH DETAILS

7.1 STATE AUTHORITY CAN WITHDRAW TOKENS WITHOUT RESTRICTIONS

// INFORMATIONAL

Description

The Claiming Program allows users to claim tokens based on their point score. The tokens are held in a treasury account until they are transferred to eligible users.
The instruction `Withdraw` enables the state authority to withdraw the full amount of tokens from the treasury to prevent any unclaimed tokens to be stuck. However, the `Withdraw` instruction can be invoked by the state authority at any time. If used inappropriately, users will be unable to claim their tokens.

lib.rs

```
7  #[derive(Accounts, AccountsCount)]
8  #[instruction(
9      season: u8,
10 )]
11 pub struct Withdraw<'info> {
12     #[account(
13         seeds = [&SeasonalState::seed(season)],
14         bump = state.bump,
15         constraint = state.season_number == season,
16     )]
17     pub state: Account<'info, SeasonalState>,
18     #[account(
19         mut,
20         token::mint = mint,
21         token::authority = state,
```

```
        token::token_program = token_program,
    )]
    pub treasury: InterfaceAccount<'info, TokenAccount>,
    #[account(
        mut,
        token::mint = mint,
        token::token_program = token_program,
    )]
    pub dest_token_acc: InterfaceAccount<'info, TokenAccount>,
    mint: InterfaceAccount<'info, Mint>,
    pub token_program: Interface<'info, TokenInterface>,

    state_authority: Signer<'info>,
    #[account(
        constraint = program.programdata_address()?
        == Some(program_data.key()) @ Error::WrongProgram,
    )]
    program: Program<'info, PointsClaimer>,
    #[account(
        constraint = program_data.upgrade_authority_address
        == Some(state_authority.key()) @ Error::WrongProgramAuthorit
    )]
    program_data: Account<'info, ProgramData>,
}

pub fn withdraw(ctx: Context<Withdraw>) -> Result<()> {
    token_interface::transfer_checked(
        CpiContext::new_with_signer(
            ctx.accounts.token_program.to_account_info(),
            token_interface::TransferChecked {
                from: ctx.accounts.treasury.to_account_info(),
                mint: ctx.accounts.mint.to_account_info(),
                authority: ctx.accounts.state.to_account_info(),
```

```
                to: ctx.accounts.dest_token_acc.to_account_info(),
            },
            &[&[
                &SeasonalState::seed(ctx.accounts.state.season_numbe
                &[ctx.accounts.state.bump],
            ]],
        ),
        ctx.accounts.treasury.amount,
        ctx.accounts.mint.decimals,
    )?;

    token_interface::close_account(CpiContext::new_with_signer(
        ctx.accounts.token_program.to_account_info(),
        token_interface::CloseAccount {
            account: ctx.accounts.treasury.to_account_info(),
            destination: ctx.accounts.dest_token_acc.to_account_info
            authority: ctx.accounts.state.to_account_info(),
        },
        &[&[
            &SeasonalState::seed(ctx.accounts.state.season_number),
            &[ctx.accounts.state.bump],
        ]],
    ))?;

    Ok(())
}
```

Proof of Concept

1. Create a new season with a given Merkle root.

2. Claim tokens using valid MT proof.
3. Withdraw tokens from the treasury.

BVSS

AO:S/AC:L/AX:L/C:N/I:N/A:N/D:H/Y:N/R:N/S:U (1.5)

Recommendation

To address this issue, it is recommended to implement a mechanism to prevent unconditional withdrawal from the treasury, such as using a multisig account owned by multiple parties as the state authority, or adding a minimum time period after which token withdrawals will be possible.

Remediation

ACKNOWLEDGED: The **deBridge team** confirmed that a multisig authority will be used as the program upgrade authority and therefore the state and withdraw authority. This reduces the risk of incorrect or unauthorized withdrawals.

7.2 PROOF LENGTH AS ENV VARIABLE RISKS TOKEN CLAIM FAILURES

// INFORMATIONAL

Description

The length of Merkle tree proof can be defined as an environmental variable that is not stored in version control. After building and deploying the program, it won't be possible to track down the proof length set during deployment. Setting the length incorrectly will result in users being unable to claim their tokens and necessity to redeploy the program.

constants.rs

```
7  pub const PROOF_LEN: usize = option::unwrap_or!(
8      option::and_then!(option_env!("PROOF_LEN"), |str| result::ok!(pa
9          str
10     ))),
11     19
12 );
```

BVSS

AO:S/AC:L/AX:L/C:N/I:N/A:C/D:N/Y:N/R:F/S:U (0.5)

Recommendation

To address this issue, it is recommended to use a configuration file that can be stored in a version control system.

Remediation

ACKNOWLEDGED: The **deBridge team** acknowledged the issue.

7.3 POSSIBILITY TO USE INCORRECT MINT TO CLAIM TOKENS

// INFORMATIONAL

Description

The `claimSolana` and `claimEvm` instructions enable users to claim their tokens based on their point score.
Although unlikely, it is possible to invoke both claim instructions with an incorrect mint account and the corresponding treasury and receiver token accounts. This would consequently disable the possibility to claim tokens with the correct mint.
claim_solana.rs

```
19 pub struct ClaimSolana<'info> {
20     #[account(
21         seeds = [&SeasonalState::seed(season)],
22         bump = state.bump,
23         constraint = state.is_working @ Error::StatePaused,
24         constraint = amount <= state.max_drop_per_ix @ Error::MaxDrc
25         constraint = state.starting_slot <= Clock::get()?.slot @ Err
26         constraint = state.season_number == season,
27     )]
28     state: Account<'info, SeasonalState>,
29     #[account(
30         mut,
31         token::mint = mint,
32         token::authority = state,
33         token::token_program = token_program,
34         has_one = mint,
35     )]
36     treasury: InterfaceAccount<'info, TokenAccount>,
37
```

```
    #[account(
        init_if_needed,
        payer = points_owner,
        associated_token::mint = mint,
        associated_token::authority = receiver,
        associated_token::token_program = token_program,
    )]
    receiver_token_acc: InterfaceAccount<'info, TokenAccount>,
    mint: InterfaceAccount<'info, Mint>,
    #[account(
        init,
        seeds = [b"receipt", &[season][..], &points_owner.key().to_b
        bump,
        payer = points_owner,
        space = 8 + Receipt::INIT_SPACE,
    )]
    receipt: Account<'info, Receipt>,
    /// CHECK: signature will be checked in handler
    receiver: AccountInfo<'info>,
    system_program: Program<'info, System>,
    token_program: Interface<'info, TokenInterface>,
    associated_token_program: Program<'info, AssociatedToken>,
    /// CHECK: will be processed in the handler
    ban_marker: AccountInfo<'info>,
    #[account(mut)]
    points_owner: Signer<'info>,
}
```

claim_evm.rs

```
pub struct ClaimEvm<'info> {
    #[account(
```

```
24          seeds = [&SeasonalState::seed(season)],
25          bump = state.bump,
26          constraint = state.is_working @ Error::StatePaused,
27          constraint = amount <= state.max_drop_per_ix @ Error::MaxDrc
28          constraint = state.starting_slot <= Clock::get()?.slot @ Err
29          constraint = state.season_number == season,
30      )]
31      state: Account<'info, SeasonalState>,
32      #[account(
33          mut,
34          token::mint = mint,
35          token::authority = state,
36          token::token_program = token_program,
37          has_one = mint,
38      )]
39      treasury: InterfaceAccount<'info, TokenAccount>,
40      #[account(
41          init_if_needed,
42          payer = payer,
43          associated_token::mint = mint,
44          associated_token::authority = receiver,
45          associated_token::token_program = token_program,
46      )]
47      receiver_token_acc: InterfaceAccount<'info, TokenAccount>,
48      mint: InterfaceAccount<'info, Mint>,
49      #[account(
50          init,
51          seeds = [b"receipt", &[season][..], &address],
52          bump,
53          payer = payer,
54          space = 8 + Receipt::INIT_SPACE,
55      )]
56      receipt: Account<'info, Receipt>,
```

```
    /// CHECK: can be any account
    receiver: AccountInfo<'info>,
    #[account(mut)]
    payer: Signer<'info>,
    system_program: Program<'info, System>,
    token_program: Interface<'info, TokenInterface>,
    associated_token_program: Program<'info, AssociatedToken>,
    /// CHECK: will be processed in the handler
    ban_marker: AccountInfo<'info>,
}
```

Proof of Concept

1. Create season 0 using mintA.
2. Create a new mintB where the authority is the season 0 PDA and corresponding treasury.
3. Claim solana using mintB.
4. Claim solana using mintA.

BVSS

AO:S/AC:L/AX:L/C:N/I:N/A:N/D:C/Y:N/R:P/S:U (1.0)

Recommendation

To address this issue, it is recommended to require using the same mint as saved in the seasonal state account.

Remediation

SOLVED: The **deBridge team** solved the issue by adding the `has_one = mint` anchor constraints in the `ClaimSolana` and `ClaimEvm` instruction contexts to ensure only the expected mint is used

Remediation Hash

https://github.com/debridge-finance/points-claimer-solana/commit/8a37b39e3be1def2cd3016e811c49675ecec92af

8. AUTOMATED TESTING

STATIC ANALYSIS REPORT

Description

Halborn used automated security scanners to assist with detection of well-known security issues and vulnerabilities. Among the tools used was `cargo audit`, a security scanner for vulnerabilities reported to the RustSec Advisory Database. All vulnerabilities published in `https://crates.io` are stored in a repository named The RustSec Advisory Database. `cargo audit` is a human-readable version of the advisory database which performs a scanning on Cargo.lock. Security Detections are only in scope. All vulnerabilities shown here were already disclosed in the above report. However, to better assist the developers maintaining this code, the auditors are including the output with the dependencies tree, and this is included in the cargo audit output to better know the dependencies affected by unmaintained and vulnerable crates.

Cargo Audit Results

ID	CRATE	DESCRIPTION
RUSTSEC-2024-0344	curve25519-dalek	Timing variability in `curve25519-dalek`'s `Scalar29::sub`/ `Scalar52::sub`
RUSTSEC-2022-0093	curve25519-dalek	Double Public Key Signing Function Oracle Attack on `ed25519-dalek`

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the project's integrity and addressing potential vulnerabilities introduced by code modifications.