



**SOLID**Proof  
*Bring trust into your projects*

**Blockchain Security | Smart Contract Audits | KYC**

MADE IN GERMANY

**Bullperks**

**Audit**

**Security Assessment**  
**26. August, 2022**

**For**



**BullPerks**



**SolidProof\_io**



**@solidproof\_io**

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Scope of Work/Verify Claims	8
Modifiers and public functions	9
Critical issues	11
High issues	11
Medium issues	11
Low issues	12
Informational issues	13
Audit Comments	14

# Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

**SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.**

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	06. - 12. August 2022	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>
1.1	26. August 2022	Update

## **Network**

Solana (Rust)

## **Website**

<https://bullperks.com/>

## **Telegram**

<https://t.me/BullPerksAnnouncements>

## **Twitter**

<https://twitter.com/bullperks>

## **Facebook**

<https://www.facebook.com/bullperks>

## **Instagram**

[https://www.instagram.com/bullperks\\_vc/](https://www.instagram.com/bullperks_vc/)

## **Reddit**

[https://www.reddit.com/r/BullPerks\\_VC/](https://www.reddit.com/r/BullPerks_VC/)

## **Medium**

<https://medium.com/bull-perks>

## **Discord**

<https://discord.gg/5avWfavp2n>

## **Youtube**

<https://www.youtube.com/channel/UCIY2Vz-X3vmBLSqbMcPZyMQ>

## **LinkedIn**

<https://www.linkedin.com/company/77364631/admin/>

## Description

The fairest and most community-oriented decentralized VC and multichain launchpad

## Project Engagement

During the 3rd of August 2022, **Bullperks Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

## Logo



# BullPerks

## Contract Link

**v1.0**

- Github
  - [https://github.com/bullperks/claiming\\_contracts\\_solana](https://github.com/bullperks/claiming_contracts_solana)
  - Commit: 664d08ae95392e6491f9b52cb554f85c11a4c112

# Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
<b>Critical</b>	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
<b>High</b>	7 - 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
<b>Medium</b>	4 - 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
<b>Low</b>	2 - 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
<b>Informational</b>	0 - 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

# Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

## **Methodology**

The 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 SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
  - 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 SolidProof 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 those test cases.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

## Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .rs).

We will verify the following claims:

1. Missing signer checks
2. Missing ownership checks
3. Missing rent exemption checks
4. Signed invocation of unverified programs
5. Solana account confusions
6. Re-initiation with cross-instance confusion
7. Arithmetic overflow/underflows
8. Numerical precision errors
9. Loss of precision in calculation
10. Incorrect calculation
11. Casting truncation
12. Exponential complexity in calculation
13. Missing freeze authority checks
14. Insufficient SPL-Token account verification
15. Over/under payment of loans
16. Overall checkup (Smart Contract Security)

## Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

### Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚠
Unverified / Not checked	✗
Not available	—



# Modifiers and public functions

## v1.0

### Implements

```
✓ {} impl Config
|   ⌵ LEN usize
✓ {} impl MerkleDistributor
|   ⌵ space_required fn(periods: &[Period]) -> usize
✓ {} impl TokenTransfer<'_, '_>
|   ⌵ make fn(self) -> Result<()>
✓ {} impl UserDetails
|   ⌵ LEN usize
✓ {} impl Vesting
|   ⌵ apply_change fn(&mut self, change: Change)
|   ⌵ bps_available_to_claim fn(&self, now: u64, user_details: &UserDetails) -> (Decimal, Decimal)
|   ⌵ has_started fn(&self, clock: &Sysvar<Clock>) -> bool
|   ⌵ new fn(schedule: Vec<Period>) -> Result<Self>
|   ⌵ validate fn(&self) -> Result<()>
```

### Public functions

```
initialize_config fn(ctx: Context<InitializeConfig>, bump: u8) -> Result<()>
initialize fn(ctx: Context<Initialize>, args: InitializeArgs) -> Result<()>
init_user_details fn(ctx: Context<InitUserDetails>, bump: u8) -> Result<()>
update_schedule fn(ctx: Context<UpdateSchedule>, args: UpdateScheduleArgs) -> Result<()>
update_root fn(ctx: Context<UpdateRoot>, args: UpdateRootArgs) -> Result<()>
set_paused fn(ctx: Context<SetPaused>, paused: bool) -> Result<()>
add_admin fn(ctx: Context<AddAdmin>) -> Result<()>
remove_admin fn(ctx: Context<RemoveAdmin>) -> Result<()>
withdraw_tokens fn(ctx: Context<WithdrawTokens>, amount: u64) -> Result<()>
claim fn(ctx: Context<Claim>, args: ClaimArgs) -> Result<()>
```

## Structs

```
AddAdmin  
Claim  
ClaimArgs  
Claimed  
Config  
Initialize  
InitializeArgs  
InitializeConfig  
InitUserDetails  
MerkleDistributor  
MerkleRootUpdated  
Period  
RemoveAdmin  
SetPaused  
TokensWithdrawn  
TokenTransfer  
UpdateRoot  
UpdateRootArgs  
UpdateSchedule  
UpdateScheduleArgs  
UserDetails  
Vesting  
WithdrawTokens
```

## Comments

- N/A

**Please check if an OnlyOwner or similar restrictive modifier has been forgotten.**

# Audit Results

**AUDIT PASSED**

## Critical issues

**No critical issues**

## High issues

**No high issues**

## Medium issues

**No medium issues**

## Low issues

Issue	File	Type	Line/ Category	Description
#1	Main	Arithmetic overflow/ underflow	398, 410, 413, 781, 271, 272, 356	<p>It's a common error to think that Rust catches overflows, when in fact this is only true in debug mode. Rust integers have fixed sizes and can only represent values within their supported ranges. If an arithmetic operation results in a higher or lower value, the value will wrap around with two's complement. Citing from the Rust documentation (<a href="https://doc.rust-lang.org/book/ch03-02-data-types.html#integer-overflow">https://doc.rust-lang.org/book/ch03-02-data-types.html#integer-overflow</a>). When you're compiling in release mode with the <code>—release</code> flag, Rust does not include checks for integer overflow that cause panics. Instead, if overflow occurs, Rust performs two's complement wrapping. In short, values greater than the maximum value the type can hold “wrap around” to the minimum of the values the type can hold. In the case of a <code>u8</code>, 256 becomes 0, 257 becomes 1, and so on. The program won't panic but the variable will have a value that probably isn't what you were expecting to have.</p> <p>We suggest that you use some safe math function supported by Rust, ex: <code>checked_add</code>, <code>checked_sub</code> instead of using normal operation <code>#</code>, <code>-</code>.</p>

#2	Main	Warning code	166, 256, 338	<p>There were some warnings in using Rust code.</p> <p>We suggest</p> <ul style="list-style-type: none"> <li>• <code>user_details</code> instead of <code>&amp;user_details</code></li> <li>• <code>admin_slot.is_none()</code> instead of <code>None=admin_slot</code></li> <li>• <code>!self.schedule.is_empty()</code> instead of <code>self.schedule.len()&gt;0</code></li> </ul>
----	------	--------------	---------------	---

## Informational issues

**No informational issues**

# Audit Comments

## August 2022:

Contract was compiled on Ubuntu 18.04 x64 with actual Rust, Solana, NPM and Yarn packages.

Automated testing results:

- ✓ Compiler Optimization Passes
- ✓ Pointer Analysis
- ✓ Building Static Happens-Before Graph
- ✓ Detecting Vulnerabilities

No Vulnerabilities were found in the crate dependencies.  
Not lint mistakes were found against 450 lint rules.

**No unit tests were performed because no corresponding tests were supplied.**

The logo features the words "SolidProof" in a white, elegant script font. The "P" is particularly large and stylized, with a long horizontal stroke that extends to the left. The background is a solid blue color with a faint, large shield emblem. The shield has a checkered pattern on its right side and a solid blue area on its left side.

SolidProof

**Blockchain Security | Smart Contract Audits | KYC**

A small horizontal bar representing the German flag, with black, red, and gold stripes.

MADE IN GERMANY