Cyberscope

A TAC Security Company

Audit Report

Fair

September 2025

Sha256

175c0d623dfb745efe08dba40af8170cadcd073c82e4b88269af60d70fd9f34f

Audited by © cyberscope

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
Critical	Highly Likely / High Impact
Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
Minor / Informative	Unlikely / Low to no Impact

Review

Audit Updates

Initial Audit	05 Sep 2025

Source Files

Filename	SHA256
lib.rs	175c0d623dfb745efe08dba40af8170cadcd073c82e4b88269af60d70fd9f34f

Overview

FairTokenVault Contract

The **FairTokenVault** contract is designed to manage the controlled distribution and redemption of a custom token in a secure, programmatic manner. It provides mechanisms for users to buy tokens using SOL and redeem them back, while ensuring that token and SOL balances are correctly maintained. The contract leverages **Solana Program Derived Addresses (PDAs)** to securely hold SOL and token assets, and enforces constraints to ensure only authorized accounts and correct token mints are used. This setup provides a predictable and secure framework for token operations.

Initialization Functionality

The contract allows an **administrator** to initialize a token vault, associating it with a specific mint and designating the vault itself as the token authority. During initialization, the vault is funded and, and the mint PDA ensures programmatic control over the token distribution process. This initialization establishes a predictable and secure starting state for all token operations.

Token Purchase Functionality

The `buy_fair_token` function enables users to acquire tokens by transferring **SOL** to the vault. It verifies that the token vault account is correct and that the token mint matches expectations. Using Solana's account constraints and PDAs, the contract ensures that only legitimate token purchases are processed, preventing misdirected transfers or accidental token creation.

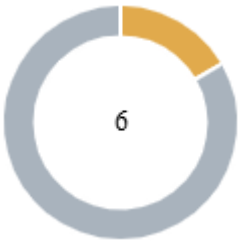
Token Redemption Functionality

The `redeem_fair_token` function allows users to return tokens to the vault in exchange for **SOL**. The function performs lamport transfers between the vault and the user, using checked arithmetic to prevent underflow or overflow issues. The use of PDAs and account constraints ensures that only authorized token redemptions occur.

Finalization Functionality

The contract includes a **finalization feature** that is triggered automatically to lock the token mint. Once finalized, pre-finalization operations such as minting are no longer possible, establishing a fixed and predictable supply of tokens. This ensures that after the launch or distribution phase, the token ecosystem becomes immutable in terms of mint authority and further modifications are restricted.

Findings Breakdown



Critical	0
Medium	1
Minor / Informative	5

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	1	0	0	0
Minor / Informative	5	0	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
Medium	IMAF	Incorrect Minted Amount Finalization	Unresolved
Minor / Informative	CCR	Contract Centralization Risk	Unresolved
Minor / Informative	MT	Mints Tokens	Unresolved
Minor / Informative	PAO	Potential Arbitrage Opportunities	Unresolved
Minor / Informative	UUA	Unrevoked Update Authority	Unresolved
Minor / Informative	UDLM	Unsafe Direct Lamports Manipulation	Unresolved

IMAF - Incorrect Minted Amount Finalization

Criticality	Medium
Location	lib.rs#L271
Status	Unresolved

Description

If `finalize_sale` is triggered within the `redeem_fair_token` function, then the `amount_to_redeem` variable is passed as the input for `redeemed_this_transaction`. The `finalize_sale` function then calculates the amount `to_be_minted`. If the `total_minted` amount is greater than the `MIN_SUPPLY`, additional tokens will be minted although the `MIN_SUPPLY` has been achieved. This could inflate the supply of the token above the expected limits.

```rust
let to_be_minted: u64 = MIN_SUPPLY
    .saturating_sub(total_minted)
    .saturating_sub(bought_this_transaction)
    .saturating_add(redeemed_this_transaction);
```

Recommendation

The team should ensure that the `redeemed_this_transaction` is handled correctly during the calculation of the amount `to_be_minted`.

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	lib.rs#L26
Status	Unresolved

Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

```rust
pub fn initialize(ctx: Context<Initialize>,
sale_end: i64) -> Result<()>
```

Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

MT - Mints Tokens

Criticality	Minor / Informative
Location	lib.rs#L99
Status	Unresolved

Description

Before finalization, users are able to mint tokens by calling the `buy_fair_token` function. As a result, the contract tokens will be highly inflated.

```rust
token::mint_to(cpi_ctx, lamports_sent)?;
```

Recommendation

The team could introduce a `MAX_SUPPLY` limit to ensure that tokens minted will not be more than a reasonable amount.

PAO - Potential Arbitrage Opportunities

Criticality	Minor / Informative
Location	lib.rs#L80,160
Status	Unresolved

Description

The contract allows users to exchange SOL for the Fair Token and vice versa through the `buy_fair_token` and `redeem_fair_token` functions at a fixed 1:1 ratio. While this ensures a stable exchange rate within the contract, it does not account for potential fluctuations of the token price on external markets or DEXs. As a result, users could exploit discrepancies between the fixed rate and the market price, creating arbitrage opportunities that could be profitable at the expense of the contract's reserves.

```rust
pub fn buy_fair_token(ctx: Context<BuyFairToken>,
lamports_sent: u64) -> Result<()>

pub fn redeem_fair_token(ctx: Context<RedeemFairToken>,
amount_to_redeem: u64) -> Result<()>
```

Recommendation

The team could consider implementing a dynamic pricing mechanism or integrate price oracles to adjust the on-chain rate, mitigating the risk of arbitrage and protecting the contract's reserves from being drained.

UUA - Unrevoked Update Authority

Criticality	Minor / Informative
Location	lib.rs#L53
Status	Unresolved

Description

The `token_vault_account` is initialized with a mint PDA as its authority, but the mint's update authority is not revoked after initialization. This means that, even after deployment, the program or any entity that can sign as the mint authority could potentially change token metadata. While the current program flow does not provide direct access to external actors, leaving the update authority unrevoked introduces an unnecessary risk if key compromise or PDA miscalculation occurs.

Rust
```
let ix = spl_token::instruction::set_authority(
    &ctx.accounts.token_program.key(),
    &ctx.accounts.mint.key(),
    None,
    AuthorityType::FreezeAccount,
    &ctx.accounts.mint_authority.key(),
    &[],
)?;
invoke_signed(
    &ix,
    &[
        ctx.accounts.mint.to_account_info(),

ctx.accounts.mint_authority.to_account_info(),
    ],
    signer,

)?;
```

Recommendation

Revoke or set the mint update authority to a PDA-controlled or zeroed-out address after initialization to ensure no further changes to the mint configuration can occur. This eliminates the risk of unauthorized minting or metadata updates and strengthens the integrity of the token distribution.

UDLM - Unsafe Direct Lamports Manipulation

Criticality	Minor / Informative
Location	lib.rs#L223
Status	Unresolved

Description

The smart contract manually modifies lamports of the vault and redeemer accounts using `try_borrow_mut_lamports` . This bypasses the Solana runtime safety checks provided by the System Program, including ownership verification, signature enforcement, and reentrancy protection. Direct lamports manipulation is considered unsafe and can lead to potential vulnerabilities such as unauthorized SOL withdrawal if account ownership checks are bypassed and inconsistent state if lamports underflow or overflow unexpectedly.

```rust
{
    let vault_ai =
ctx.accounts.sol_vault.to_account_info();
    let redeemer_ai =
ctx.accounts.redeemer.to_account_info();
    let from_lamports = vault_ai.lamports();
    let to_lamports = redeemer_ai.lamports();
    let new_from = from_lamports
        .checked_sub(amount_to_redeem)
        .ok_or(ErrorCode::VaultSOLInsufficient)?;
    let new_to = to_lamports
        .checked_add(amount_to_redeem)
        .ok_or(ErrorCode::VaultSOLInsufficient)?;
    **vault_ai.try_borrow_mut_lamports()? =
new_from;
    **redeemer_ai.try_borrow_mut_lamports()? =
new_to;
}
```

Recommendation

The team should replace direct lamports assignment with a safe transfer via the Solana System Program using `invoke_signed` for PDAs.

Summary

Fair Token contract implements an exchange mechanism. This audit investigates security issues, business logic concerns and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment 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 Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a TAC blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

A **TAC Security** Company

The Cyberscope team

cyberscope.io