Cyberscope

Audit Report

DROGON

June 2024

Repository https://github.com/0xEl-Loco/drogon_burner

Commit 92a8ae1ad7b6c509052930c01e6d0de7bc745e77

Audited by © cyberscope

Table of Contents

Review

Network	SOL
Repository	https://github.com/0xEl-Loco/drogon_burner
Commit	92a8ae1ad7b6c509052930c01e6d0de7bc745e77

Audit Updates

Initial Audit	06 Jun 2024

Source Files

Filename	SHA256
lib.rs	66e0933dfb20bfd1a67b666310d145631d2ef7381cbce384dbb7fdabd0db19ee

Overview

The drogon_burn program is designed to manage the initialization and burning process of a token on the Solana blockchain. This program includes several key functions that ensure the correct setup and execution of the token burning schedule. The program's primary functionality revolves around initializing accounts, transferring tokens to an escrow account, scheduling burn events, and executing the burning of tokens based on a predefined schedule.

The program includes an initialization function for the "Drogon Account," which ensures that the account is set up correctly by an authorized key. This process includes verifying that the account has not been previously initialized, setting the initializer, and marking the account as initialized. Another function handles the transfer of a specified amount of tokens from a user's wallet to an escrow account. This transfer is authorized by a specified key, and checks are in place to ensure sufficient balance before the transfer occurs.

Additionally, the program sets up a burn schedule that defines the specific times and amounts of tokens to be burned. This schedule is created during the initialization of the burn schedule and is stored in a "Burn Schedule Account." The schedule is defined in terms of events, each specifying the cumulative number of tokens to be burned at a particular timestamp.

The actual burning of tokens is handled by a dedicated function that checks the current timestamp against the burn schedule. If the conditions are met, tokens are burned from the escrow account and the total burned amount is updated. The program also includes a utility function to view the next burn event based on the current timestamp.

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	4

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	4	0	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	IRU	Inefficient Reference Usage	Unresolved
●	PDTO	Potential Data Types Optimization	Unresolved
●	PSAM	Potential Space Allocation Miscalculation	Unresolved
●	PCR	Program Centralization Risk	Unresolved

IRU - Inefficient Reference Usage

Criticality	Minor / Informative
Location	lib.rs#L56
Status	Unresolved

Description

There is an inefficient usage of references in the token amount comparison operation within the `initialize_transfer_to_escrow` function. Specifically, the code compares references to the values of `wallet_to_withdraw_from.amount` and `amount`, instead of comparing the values directly. This unnecessary reference usage can lead to reduced readability and potential inefficiencies in the code.

```
require!(
    &ctx.accounts.wallet_to_withdraw_from.amount >= &amount,
    ErrorCode::InsufficientBalance
);
```

Recommendation

It is recommended to refactor the comparison operation to use the values directly, rather than their references. This change will improve the readability and efficiency of the code. By ensuring that the comparison is performed on the actual values, the code will be clearer and more maintainable, adhering to best practices in Rust programming. This adjustment will contribute to the overall optimization and quality of the program.

PDTO - Potential Data Types Optimization

Criticality	Minor / Informative
Location	lib.rs#L247,353
Status	Unresolved

Description

In the current implementation of the program, several variables and struct fields are defined using larger data types than may be necessary. Using unnecessarily large data types can result in inefficient use of storage and increased gas costs.

```
pub struct BurnEventData {
    pub event_number: u64,
    pub timestamp: i64,
    pub burn_stage: u8,
    pub cumulative_burned: u64,
    pub burned_at_event: u64,
}

pub struct DrogonAccount {
    pub total_burned: u64,
    pub initializer: Pubkey,
    pub escrow_wallet_account: Pubkey,
    pub wallet_to_withdraw_from: Pubkey,
    pub token_mint: Pubkey,
    pub burn_schedule_account: Pubkey,
    pub initiation_time: i64,
    pub drogon_initialized: bool,
    pub tokens_transfered_to_escrow: bool,
    pub burn_schedule_initialized: bool,
}
```

Recommendation

To enhance storage efficiency and reduce gas costs, it is recommended to review all variables and struct fields across the program and assess whether smaller data types can be used without compromising the functionality. This optimization will result in more efficient storage usage and potentially lower gas costs during program execution.

PSAM - Potential Space Allocation Miscalculation

Criticality	Minor / Informative
Location	lib.rs#L322
Status	Unresolved

Description

The space allocation for the `BurnScheduleAccount` might be incorrectly calculated. The current space calculation is `4 + (73 * 40)`, assuming each `BurnEventData` element occupies 40 bytes. However, the accurate calculation shows that each `BurnEventData` element should occupy 33 bytes, plus the necessary overhead. This discrepancy suggests a potential over-allocation or misunderstanding of the required space, which could lead to inefficiencies or errors in the program's operation.

```
#[account(init, payer = sender, seeds =
[b"burn_schedule_account"], bump, space = 4 + (73 * 40))]
pub burn_schedule_account: Account<'info, BurnScheduleAccount>,
```

Recommendation

It is recommended to verify and correct the space allocation for the `BurnScheduleAccount`. Ensure that each `BurnEventData` element's size is accurately calculated and that the total space reflects the actual requirements, including the 4 bytes for vector length prefix and 8 bytes required by Anchor for account discrimination. This adjustment will optimize space usage and ensure the correct functionality of the program.

PCR - Program Centralization Risk

Criticality	Minor / Informative
Location	lib.rs#15,32,39,81
Status	Unresolved

Description

The programs'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. Specifically, the program's functionality and behavior are heavily dependent on external parameters or configurations. The functions that initialize the initial values must be called by a specific authorized account for the program to function as intended. This creates a centralization risk because the correct initialization and subsequent operations rely on a single point of control. If the authorized account is compromised or becomes unavailable, it could disrupt the operation of the program.

Additionally, the escrow wallet that receives the transferred tokens could introduce further centralization risks. As currently implemented, there are no explicit constraints preventing the escrow wallet from transferring the tokens before they are burned. This introduces a risk where the tokens could be moved or misused, potentially undermining the program's integrity and the expected burning process.

```
pub fn initialize_drogon_account(ctx:
Context<InitializeDrogonAccount>) -> Result<()> {
    // Authorization Check
    let authorized_key =
Pubkey::from_str(AUTHORIZED_KEY).map_err(|_|
ErrorCode::Unauthorized)?;
    require!(ctx.accounts.sender.key() == authorized_key,
ErrorCode::Unauthorized);

    // Check if drogon account is already initialized
    require!(
        !ctx.accounts.drogon_account.drogon_initialized,
        ErrorCode::AlreadyInitialized
    );

    // Initialize Drogon account
    let drogon_account = &mut ctx.accounts.drogon_account;
    drogon_account.initializer =
ctx.accounts.sender.to_account_info().key();
    drogon_account.total_burned = 0;
    drogon_account.wallet_to_withdraw_from =
ctx.accounts.wallet_to_withdraw_from.to_account_info().key();
    drogon_account.token_mint =
ctx.accounts.token_mint.to_account_info().key();
    drogon_account.escrow_wallet_account =
ctx.accounts.escrow_wallet_account.to_account_info().key();

    // Mark as initialized
    ctx.accounts.drogon_account.drogon_initialized = true;
    Ok(())
}

...
```

Recommendation

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

Summary

The Drogon Burn program implements a token-burning mechanism on the Solana blockchain. 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 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.

The Cyberscope team

https://www.cyberscope.io