Follows Solana Program Code Review and Security Analysis

Client: Follows

Date: 14th Feb. 2024

Version: 1.0

Table of Contents

Security Review Information

Repository	follows-solana
Initial commit	f111978c69bae66d427df0a8333c5f89d6c11581
Final commit	e2647a4c57aefff9ca3da465e5420353dd818e4c
Scope	programs/follows-solana/*
Reviewer	meltedblocks @ Torii Security
Version	Final Report (v. 1.0)
Date	14th Feb. 2024

Executive summary

As of February 14, 2024, our comprehensive security review of the Follows protocol has been concluded. Initially, the assessment identified **1** high-severity and **3** low-severity vulnerabilities. We are pleased to report that, following our review and the implementation of recommended fixes, all identified issues have been successfully resolved.

Introduction

Torii Security has been commissioned by Follows to conduct a comprehensive security review of their Solana Program, focusing on the robustness, security, and efficiency of the program's implementation. The review aims to critically evaluate the program's architecture and codebase, with the following specific objectives:

- **Verify Protocol Integrity**: Assess the program's operation against its design specifications to ensure it functions correctly and efficiently within the Solana ecosystem. This includes evaluating its interaction with other protocols and services on the Solana network.
- **Identify Security Vulnerabilities**: Uncover potential security weaknesses that could be exploited by malicious actors, including but not limited to, flaws in program logic, transaction handling, and external dependencies.
- **Detect Program Bugs**: Identify bugs and glitches in the code that may result in unintended or erratic program behavior, potentially compromising its performance or security.
- **Provide Improvement Recommendations**: Offer actionable advice to enhance the program's security posture, efficiency, and code clarity, aiming to fortify it against current and future security threats while improving maintainability and scalability.

Exclusions from This Review

While the security review conducted by Torii Security on behalf of Follows provides a comprehensive analysis of the Solana Program's architecture, codebase, and security posture, certain aspects are beyond the scope of this audit. These exclusions are critical for stakeholders to understand, as they may require separate consideration and evaluation. The following areas have not been verified as part of this security review:

- **Deployment and Program Upgrade Process:** The procedures and mechanisms for deploying the Solana Program to the network and subsequent upgrades or modifications to the program are not covered. This includes the validation of deployment scripts, migration strategies, and the security of upgradeable contract mechanisms.
- **Keys Management**: The management, storage, and security practices for keys, including administrator keys and those used for program interactions, are outside the scope of this review. This encompasses both the technical and procedural safeguards in place to protect keys from unauthorized access or misuse.
- **Economic Vulnerabilities**: The review does not delve into the economic aspects or incentive structures of the Follows protocol. Potential vulnerabilities arising from economic models, tokenomics, or financial incentives that could impact the program's security or integrity are not evaluated.

Protocol Overview

The Follows protocol introduces a novel marketplace model within the digital asset space, akin to a trading platform for individuals' shares. At its core, the protocol leverages a bonding curve mechanism to dynamically determine the price of shares, offering a unique approach to valuing participants' stakes in the market. This mechanism ensures that the price of shares increases with demand, providing an automatic pricing model that adjusts based on buy and sell actions.

Key Features

- **Account Initialization**: Users can onboard by initializing their accounts and setting the groundwork for participation in trading activities.
- **First Share Purchase**: The protocol allows users to buy their first share of themselves, marking their entry into the market and enabling them to become tradable entities within the ecosystem.
- **Share Trading**: Users can buy multiple shares of themselves or engage in buying and selling shares of other participants. This feature fosters an interactive market where individuals can invest in the potential of others or increase their own market presence.
- **Taxation Model**: Every transaction within the Follows protocol incurs a 10% tax, distributed as follows (fees can be changed, but total fees cannot exceed 10%):
    - **4% to the Protocol**: A portion of the tax is allocated to the protocol itself, supporting its maintenance and development.
    - **5% to the Shareholder**: A significant share of the tax goes to the individual whose shares are being traded, rewarding them for their participation and success within the market.
    - **0.75% Cashback to the Trader**: Buyers or sellers receive cashback as an incentive for their trading activity, enhancing user engagement.
    - **0.25% Referral Bonus**: The protocol supports a referral system, wherein referrers gain a continuous reward from transactions involving the users they have introduced to the platform.

Risks

The security review of the Follows protocol has identified several risks. It is crucial to understand that the risks identified in this section of the report are associated with operational and procedural aspects of the Follows protocol that were not within the purview of our security review. These risks pertain to administrative controls, deployment practices, fee distribution configurations, and the integration of external components - areas that extend beyond the core protocol code and architecture, which were the primary focus of our analysis.

1. **Fee Distribution Configuration**:
    a. *Risk Description*: While the total transaction fee is capped at 10%, the protocol permits the adjustment of fee distributions through the **set_fee** instruction. Improper configuration could disrupt the intended economic balance.
    b. *Mitigation Suggestion*: Establish strict guidelines and oversight for fee distribution changes, potentially including community governance mechanisms for decision-making.

2. **Fee Receiver Configuration**:
    a. *Risk Description*: The ability for admins to set the fee receiver address poses a risk of misdirection of funds if set to an incorrect or malicious address.
    b. *Mitigation Suggestion*: Implement verification procedures for address changes, such as secondary confirmations or a delay period allowing for address change reviews.
3. **Deployment and Upgradability**:
    a. *Risk Description*: The deployment procedures, including the possibility of deploying the program as upgradable, were not audited. This leaves room for unauthorized or harmful upgrades by the protocol owners.
    b. *Mitigation Suggestion*: Conduct a thorough audit of deployment procedures. For upgradable contracts, consider implementing a transparent and secure upgrade process, such as time-locked upgrades and community review periods.
4. **Frontend and Integration Review**:
    a. *Risk Description*: The absence of a review for the frontend and other integrations means that it has not been verified whether the slippage calculation from the user side is correctly implemented, potentially affecting transaction efficiency and fairness.
    b. *Mitigation Suggestion*: Perform comprehensive testing and auditing of frontend applications and integration points, mainly focusing on critical functionalities like slippage calculations and user interactions.

Methodology

Issue Severity Classification

This report differentiates identified issues into distinct severity levels, each reflecting the potential impact on the system's security and overall functionality.

Severity	Description
Critical	Issues that present an immediate and severe threat, such as significant financial loss, irreversible locking of funds, or catastrophic system failure. These vulnerabilities require urgent remediation.
High	Bugs or vulnerabilities that could disrupt the correct operation of the system, potentially leading to incorrect states or temporary denial of service. Prompt attention and corrective action are necessary.
Medium	Issues that indicate deviations from best practices or suboptimal use of system primitives. While they may not pose immediate security threats, these issues could lead to vulnerabilities or inefficiencies if unaddressed.
Low	Minor concerns that have a negligible impact on system security or functionality. These may include inefficiencies or minor deviations from best practices that are unlikely to affect the system's operation significantly.
Informational	Suggestions related to design decisions, potential enhancements, or optimizations that do not have a direct impact on security. Implementing these recommendations may improve aspects such as usability or code readability but is not essential for system security.

Issue Status Definitions

Each issue is assigned a status reflecting its current resolution stage.

Status	Description
Pending	The issue has been identified but not yet reviewed or addressed by the development team.
Acknowledged	The development team has recognized the issue but has not completed its resolution.
Resolved	The issue has been fully addressed, with implemented changes verified for effectiveness.

Findings Summary

ID	Title	Severity	Status
TS-H1	Lack of Slippage Control in buy_token Instruction Leads to Potential MEV and Sandwich Attacks	High	**Fixed**
TS-L1	Lack of Two-Step Ownership Verification in the change_admin Instruction	Low	**Fixed**
TS-L2	Inadequate Handling of Token Supply Invariant in Price Determination Function	Low	**Fixed**
TS-L3	Transaction Failures Due to Insufficient Lamports for Rent in Fee Accounts	Low	**Fixed**
TS-I1	Possibility of Inaccurate Net Worth Calculation Due to Initial Token Allocation Mechanism	Informational	**Accepted**
TS-I2	Inefficiency and Limited Functionality with Custom Print Logging	Informational	**Accepted**
TS-I3	Redundant and Costly Account Initialization Checks	Informational	**Fixed**
TS-I4	Absence of Event Logging in Key Administrative Functions	Informational	**Fixed**

Detailed Findings

TS-H1 - Lack of Slippage Control in Buy Token Instruction Leads to Potential MEV and Sandwich Attacks

Classification

Severity: High
Status: Fixed

Description

The ***buy_token*** instruction within the protocol currently requires users to specify the number of tokens they wish to purchase. Token price is derived from the bonding curve formula introduced by the Follows protocol.

It was observed, that the ***buy_token*** function is missing slippage control. This omission significantly increases the risk of Miner Extractable Value (MEV) exploitation, particularly through sandwich attacks on the Solana blockchain, where rapid transaction ordering (e.g., via Jito) can exacerbate these risks.

In the potential attack scenario, attackers can manipulate the market price by creating the Jito bundle - executing a buy order immediately before the user's transaction and selling the tokens right after at a higher price, thus profiting from the induced price slippage:

1. Attacker buys X amount of tokens.
2. User buys Y amount of tokens with increased price.
3. Attacker sells X amount of tokens with profit.

It should be noted that ***sell_token*** instruction has slippage control implemented.

Recommendation

To mitigate this risk and protect users from potential price manipulation and MEV attacks, it is recommended to introduce slippage control. It can be achieved by introducing the ***max_price*** argument in the ***buy_token*** instruction to fortify the maximum lamports that the user is willing to pay for the exact amount of tokens.

Remediation

Issue was fixed in the d78e87 commit by introducing ***max_price*** parameter in the ***buy_token*** function.

TS-L1 - Lack of Two-Step Ownership Verification in the change_admin Instruction

Classification

Severity: Low
Status: **Fixed**

Description

The ***change_admin*** instruction is pivotal in managing the administrative rights within the protocol. However, the instruction's current implementation poses a security concern due to its inadequate verification process. The protocol merely compares the new administrator's public key against a predefined default key, neglecting the crucial step of requiring the proposed administrator to authenticate the transaction through signature verification. This shallow verification process fails to confirm the proposed administrator's control over the corresponding private keys, introducing a severe risk.

The most pressing danger of this oversight is the potential transfer of administrative control to an entity without access to the corresponding private keys. Such a scenario could irreversibly lock the protocol's administrative functions, including the inability to adjust fees or modify the fee receiver. This vulnerability not only compromises the protocol's operational integrity but also places its entire ecosystem at risk of stagnation and loss of trust.

Recommendation

To mitigate the identified security risks and strengthen the administrative change process, the following recommendation is proposed:

Implement Transaction Signing Requirement: Modify the ***change_admin*** instruction to require that the new admin be a signer of the transaction. This ensures that the new admin has access to and control over the private keys associated with their public key, thereby reducing the risk of locking the program admin access.

Alternatively:
Introduce Two-Step Ownership Verification:

- Step 1: The current admin initiates the change of administration but the change is staged and not finalized. The transaction should include the public key of the proposed new admin.
- Step 2: The new admin must accept the role by signing a separate transaction. This acceptance confirms their willingness to take on the responsibilities and verifies their control over the necessary keys. Only after this acceptance is the administrative change finalized.

Remediation

Issue was fixed in the d78e87 commit by introducing a two step ownership pattern.

TS-L2 - Inadequate Handling of Token Supply Invariant in Price Determination Function

Classification

Severity: Low
Status: **Fixed**

Description

The function tasked with determining the price of tokens within the protocol (***get_token_price***) includes an invariant check designed to ensure the token supply remains non-negative. Despite this safeguard, an oversight exists in its current implementation. Specifically, when the function compares the total supply to **0** and the amount to **1**, it erroneously returns a price of **0** instead of correctly identifying this situation as an error and halting transaction processing. This behavior not only violates the fundamental invariant that the token supply should never drop below zero but also introduces a serious flaw in the token economics, potentially leading to mispricing and exploitation.

```
programs/follows-solana/src/states/token_info.rs
if current_supply == 0u64 && amount == 1u64 {
    return Ok(0u64);
 }
```

Recommendation

It is recommended to modify the existing invariant check within the price determination function to ensure it actively throws an error upon detecting a violation of the token supply invariant.

Remediation

Issue was fixed in the e2647a commit by throwing an error on invariant check.

TS-L3 - Transaction Failures Due to Insufficient Lamports for Rent in Fee Accounts

Classification

Severity: Low
Status: **Fixed**

Description

Within the Solana ecosystem, accounts must maintain a minimum balance of lamports to satisfy rent requirements, necessitating a balance greater than the minimum rent threshold of approximately *0.00089088* SOL for account initialization. This fundamental requirement poses a challenge during the execution of token buy and sell instructions, particularly when fee-receiving accounts (such as those designated for referrals, protocol fee collection, and token creators) are not properly initialized or have been closed.

The problem manifests significantly in token transactions, where the distribution of fees from the token's price must exceed the rent exemption threshold. If the designated fee-receiving accounts lack the necessary lamport balance due to being uninitialized or previously closed (with all funds withdrawn), transactions are at risk of failing. This results in a limitation on the minimum quantity of tokens that can be purchased, dictated by the need to ensure that the transaction fees allocated do not fall below the rent exemption amount, leading to a potential **InsufficientFundsForRent** error.

Scenario 1: Fee Receiver Account Issues
When a fee receiver account is not initialized or has been closed, the minimum quantity of tokens that must be purchased to avoid transaction failure is artificially raised. For instance, if the account designated to receive fees from token sales is compromised, any attempt to buy fewer than 8 tokens may fail due to insufficient funds to cover the rent requirement. What is more, selling the last 8 tokens would not be possible in such a scenario.

Scenario 2: Referral Account Closure
In situations where a referral or creator account has been inadvertently or maliciously closed, the minimum token purchase threshold increases further. Under these conditions, any attempt to purchase fewer than 19 tokens is likely to fail, again due to the **InsufficientFundsForRent** error, significantly impacting the accessibility and fluidity of token transactions. What is more, selling the last 19 tokens would not be possible in such a scenario.

Recommendation

It’s recommended to either:

- **Implement Pull-Over-Push Pattern:** To address the issue of fee distribution, it is recommended to adopt a pull-over-push pattern. This means that instead of directly sending fees to individual wallets, fees should be collected in a Program Derived Address (PDA) within the program. Authorized entities can then withdraw the accumulated fees from this PDA, ensuring that transactions do not fail due to insufficient lamports in fee accounts.

- **(Alternative) Utilize Wrapped SOL (WSOL):** Another solution is to transition from using lamports directly to utilizing Wrapped SOL (WSOL), the Solana equivalent of Wrapped Ethereum (WETH). By conducting fee transfers through the Token Program using WSOL, many of the issues related to lamport management and rent payment can be mitigated. This approach simplifies transactions and reduces the risk of failure due to insufficient funds in fee accounts.
- **(Alternative) Verify Fee Receivers:** Alternatively, to avoid transaction failures, introduce the following checks:
    - Introduce operational procedure to keep lamports for rent in protocol-owned fee received account, so this account is always rent-exempt or ideally use PDA for protocol fees and introduce ***withdraw_fee*** instruction that checks for minimum amount of lamports left in the account.
    - On buy and sell transactions, verify that the receiver account (referral and token creator) is going to have enough lamports to pay the rent, otherwise do not send fees to such accounts.

Remediation

Issue was fixed in the e2647a commit. Now fees are transferred to wallets only if lamports in such account are more than fee exempt value, avoiding transaction reverts.

TS-I1 - Possibility of Inaccurate Net Worth Calculation Due to Initial Token Allocation Mechanism

Classification

Severity: Informational
Status: Accepted

Description

In the existing token initialization process within Follows protocol, the token owner is automatically awarded 1 token upon the creation of the token. However, the token owner is unable to sell this initially allocated token. This restriction can lead to discrepancies in the calculation of the token owner's net worth, as the value of this unsellable token may incorrectly inflate the owner's asset valuation. This misrepresentation is particularly significant in scenarios where token value comprises a substantial portion of the owner's net worth or in financial reporting and analysis where precision is paramount.

Recommendation

It is recommended to enhance documentation and user interfaces to clearly indicate that the initial token awarded upon token creation is not sellable and should be excluded from net worth calculations.

TS-I2 - Inefficiency and Limited Functionality with Custom Print Logging

Classification

Severity: Informational
Status: Accepted

Description

The audited Solana program relies on custom print messages for logging instead of utilizing Anchor's event. Anchor events offer a more structured, efficient way to log data by encoding it directly into transaction logs, facilitating easy decoding through the Anchor SDK. The current approach with custom print messages not only incurs higher computational costs due to string formatting but also complicates data analysis and debugging due to lack of structured data.

```
programs/follows-solana/src/instructions/buy_token.rs
pub fn log_buy_info(&self, amount: u64, price: u64) {
    msg!(
        "Buyer: {}, bought: {} shares of token: {} and paid a price of: {}
lamports at a price factor of {}",
        self.buyer_wallet.key(),
        amount,
        self.token_info.key(),
        price,
        self.token_info.alpha as f32 / 10f32,
    );
```

```
programs/follows-solana/src/instructions/sell_token.rs
pub fn log_sell_info(&self, amount: u64, price: u64) {
    msg!(
        "Seller: {}, sold: {} shares of token: {} at a price of: {}
lamports and a price factor of {}",
        self.seller_wallet.key(),
        amount,
        self.token_info.key(),
        price,
        self.token_info.alpha as f32 / 10f32,
    );
}
```

Recommendation

Consider code refactoring to use Anchor events for logging important information during instruction execution.

TS-I3 - Redundant and Costly Account Initialization Checks

Classification

Severity: Informational
Status: **Fixed**

Description

The audit has identified redundant checks for account initialization and unnecessary public key comparisons within the code. These practices are superfluous for two main reasons: firstly, the Anchor framework's ***init*** macro inherently performs account initialization checks, preventing reinitialization attacks. Secondly, the Anchor ***AccountLoader*** mechanism automatically verifies the initialization status of accounts when a specific account type is provided by the user. Additionally, public key comparisons are notably costly in terms of compute units (CPI) on Solana, making these operations inefficient and unnecessarily expensive.

Unnecessary checks were observed in the following places in the audited codebase:

```
programs/follows-solana/src/instructions/buy_token.rs
#[account(
    seeds = [
        FOLLOWS_PROGRAM,
        buyer_wallet.key().as_ref(),
        TOKEN_DATA_V1
    ],
    bump = token_info_buyer.bump,
    constraint = token_info_buyer.creator_wallet != Pubkey::default(),
    constraint = token_info_buyer.creator_wallet == buyer_wallet.key(),
    constraint = token_info_buyer.current_supply > 0,
)]
pub token_info_buyer: Box<Account<'info, TokenInfoV1>>,
```

```
programs/follows-solana/src/instructions/change_admin.rs
#[account(
    mut,
    seeds = [
        FOLLOWS_PROGRAM,
        PLATFORM_DATA_V1
    ],
    bump = platform_info.bump,
    has_one = admin_wallet,
    constraint = platform_info.admin_wallet != Pubkey::default(),
)]
pub platform_info: Box<Account<'info, PlatformInfoV1>>,
```

```
programs/follows-solana/src/instructions/sell_token.rs
#[account(
    mut,
    seeds = [
        FOLLOWS_PROGRAM,
        token_info.key().as_ref(),
        seller_wallet.key().as_ref(),
        HOLDER_DATA_V1
    ],
    bump = holder_info.bump,
    constraint = holder_info.token_address == token_info.key(),
    constraint = holder_info.holder_wallet == seller_wallet.key(),
    constraint = holder_info.holder_wallet != Pubkey::default(),
    constraint = holder_info.balance > 0,
)]
pub holder_info: Box<Account<'info, HolderInfoV1>>,
```

```
programs/follows-solana/src/states/token_info.rs firstnumber=48 hlines=48}
self.referred_by
    .assert_keys_eq(&Pubkey::default(), "Referrer already set");
...
self.creator_wallet
    .assert_keys_eq(&Pubkey::default(), "TokenInfoV1::Already
Initialized");

programs/follows-solana/src/instructions/init_token.rs firstnumber=30
hlines=30}
#[account(
    init,
    payer = creator_wallet,
    space = TokenInfoV1::SIZE_BYTES,
    seeds = [
        FOLLOWS_PROGRAM,
        creator_wallet.key().as_ref(),
        TOKEN_DATA_V1
    ],
    bump,
)]
pub token_info: Box<Account<'info, TokenInfoV1>>,

#[account(
    init,
    payer = creator_wallet,
```

```
    space = HolderInfoV1::SIZE_BYTES,
    seeds = [
        FOLLOWS_PROGRAM,
        token_info.key().as_ref(),
        creator_wallet.key().as_ref(),
        HOLDER_DATA_V1
    ],
    bump,
)]
pub holder_info: Box<Account<'info, HolderInfoV1>>,
```

Recommendation

Simplify the codebase by removing redundant account initialization checks and unnecessary public key comparisons. Leveraging Anchor's ***init*** macro and ***AccountLoader*** verification offers sufficient protection and efficiency. Streamlining these aspects will enhance code clarity, reduce compute unit consumption, and improve the overall performance and maintainability of the code.

Remediation

Issue was fixed in the e2647a commit. Unnecessary chekcs were removed.

TS-I4 - Absence of Event Logging in Key Administrative Functions

Classification

Severity: Informational
Status: **Fixed**

Description

An oversight has been identified in the code concerning the lack of event logging for several essential administrative operations, including modifying admin privileges, adjusting fees, and changing the platform fees receiver account. Events play a crucial role in tracking and auditing changes within a system, providing transparency and accountability for administrative actions. Their absence in these key operations can lead to challenges in monitoring system changes, diagnosing issues, and ensuring the integrity of administrative actions.

Recommendation

It is recommended to introduce event logging for all significant administrative operations within the system.

Remediation

Issue was fixed in the e2647a commit. Events logging were introduced in setting admin and setting fees operations.

Disclaimer

This report has been prepared in accordance with the current best practices and standards applicable at the time of its preparation. It is intended to provide an analysis and evaluation of the subject matter based on the information available, including any potential vulnerabilities, issues, or risks identified during the assessment process. The scope of this analysis is limited to the data, documents, and materials provided for review, and the conclusions drawn are based on the status of the information at the time of the report.

No representation or warranty, express or implied, is made as to the absolute completeness or accuracy of the information contained in this report. It is important to acknowledge that the findings, conclusions, and recommendations presented are subject to change should there be any modifications to the subject matter. This report should not be viewed as a conclusive or exhaustive evaluation of the subject matter's safety, functionality, or reliability. Stakeholders are encouraged to conduct their own independent reviews, assessments, and validations to ensure the integrity and security of the evaluated subject.

The authors and auditors of this report disclaim any liability for any direct, indirect, incidental, or consequential damages or losses that may result from reliance on this report or its contents. It is the responsibility of the stakeholders to ensure the ongoing monitoring and evaluation of the subject matter to mitigate potential risks or vulnerabilities.

The nature of technology and digital systems means that they are inherently subject to risks, including but not limited to vulnerabilities, bugs, and security threats that may not be foreseeable at the time of this report. The dynamic and evolving nature of technological standards, security practices, and threat landscapes means that absolute security cannot be guaranteed. Despite thorough analysis and evaluation, unforeseen vulnerabilities may exist, and new threats may emerge subsequent to the issuance of this report.

The authors and auditors make no guarantee regarding the impenetrability or infallibility of the subject matter under evaluation. Stakeholders are advised to implement comprehensive security measures, including but not limited to regular updates, patches, and monitoring, to safeguard against potential threats and vulnerabilities.