FYEO

Ondo

Repo: https://github.com/3uild-3thos/ondo-finance

Security Review Update: November 5, 2025

Reviewer: balthasar@gofyeo.com

Ondo Security Review Update

New security issues, 6

After the development team implemented the latest updates, FYEO conducted a review of the modifications. The primary goal of this evaluation was to ensure the continued robustness of the program's security features, safeguarding user funds and maintaining the overall integrity of the program.

General Updates:

The update expands and reworks configuration and constants across the codebase: new chain and admin keys are added for different build targets, token decimals and scaling factors are defined, price scaling and price bounds are changed, and several role identifiers and seeds are introduced or renamed. Defaults and initialization behavior are hardened (rate-used counters are initialized when limits exist, default windows added), and event emissions were added when rate limits, pauses, or sanity settings change. Program IDs and linkage between programs were updated for non-mainnet builds, and various constants and public keys were consolidated into both the main program and the transfer-hook program.

The business logic and safety checks were significantly enhanced. Attestation-based flow was introduced so buy/sell operations now require a compact attestation identifier and on-chain secp256k1 verification via the instructions sysvar, including a digest construction routine and parsing/validation of the secp instruction. Price sanity checks were tightened and exposed to configuration (including storing last price), arithmetic was made safer by moving risky multiplications into u128-buffered helpers, and rate-limiting got a robust linear-decay implementation with overflow protection and clearer failure messages. The transfer-hook was made updatable: it can now compute and reallocate storage for extra account metadata, and the hook program IDs and authority checks were adjusted to match the new constants and admin model.

Attestation message omits user

Finding ID: FYEO-ONDO-01
Severity: **High**
Status: **Open**

Description

The attestation digest (the message that is signed and later verified by the secp256k1 instruction) does not include the `user` public key. The `verify_attestation` function accepts a `_user: Pubkey` parameter but the `calculate_quote_hash` implementation has the `user` line commented out and instead embeds only `asset`, `price`, `amount`, `expiration`, etc. Because the signed message does not bind the attestation to a specific Solana user / public-key, a valid attestation intended for one user can be replayed / used by another user to perform an action on their behalf.

Proof of Issue

File name: programs/ondo-finance/src/instructions/token_manager.rs
Line: 444

```
pub fn verify_attestation(
    &self,
    chain_id: [u8; 32],
    attestation_id: [u8; 16],
    side: u8,
    _user: Pubkey,
    asset: Pubkey,
    price: u64,
    amount: u64,
    expiration: i64,
) -> Result<()> {
    ...
    let quote_hash = self.calculate_quote_hash(
        chain_id,
        attestation_id,
        side,
        asset,
        price,
        amount,
        expiration,
    );
    ...
}

pub fn calculate_quote_hash(
    &self,
    chain_id: [u8; 32],
    attestation_id: [u8; 16],
    side: u8,
    asset: Pubkey,
    price: u64,
    amount: u64,
    expiration: i64,
```

```
) -> [u8; 32] {
    ...
    quote[49..81].copy_from_slice(&asset.to_bytes());
    ...
}
```

Severity and Impact Summary

A valid attestation (signature) for (attestation_id, asset, price, …) can be used by any signer because the attestation doesn't cryptographically bind to the Solana `user` key. This allows attestation forgery: e.g., attacker or third-party can present a quote intended for Alice and have Bob execute the same quote.

Recommendation

Include `user` in the signed message. Modify `calculate_quote_hash` to include `user.to_bytes()` at a fixed offset. Add unit tests that assert an attestation valid for user A is rejected if submitted by user B.

No replay protection: attestation IDs are not recorded

Finding ID: FYEO-ONDO-02
Severity: **High**
Status: **Open**

Description

Although `attestation_id` is passed into `verify_attestation` and included in the quote hash, the program does **not store or mark attestation IDs as consumed** anywhere. That means the same attestation (valid signature) can be replayed multiple times until its `expiration` timestamp, enabling repeated execution of the same authorized action.

Proof of Issue

File name: programs/ondo-finance/src/instructions/token_manager.rs

```
// No storage access / no marking of attestation_id:
let quote_hash = self.calculate_quote_hash(
    chain_id,
    attestation_id,
    side,
    asset,
    price,
    amount,
    expiration,
);
self.verify_secp256k1_ix(...)?;
msg!("✓ Attestation signature verified");
Ok(())
```

Severity and Impact Summary

Any valid attestation can be used repeatedly until `expiration`. If an attestation authorizes a mint, burn, or transfer, it can be executed many times causing loss/theft or token inflation. Off-chain signers normally expect attestations to be one-time; without on-chain consumption the attestation semantics are broken.

Recommendation

Record `attestation_id` on first use.

INCORRECT AUTHORITY USED IN BURN_TOKEN CPI

Finding ID: FYEO-ONDO-03
Severity: **Low**
Status: **Open**

Description

In the `burn_token` instruction, the CPI context uses the `mint_authority` as the authority account instead of the actual token account authority.

Proof of Issue

File name: programs/ondo-finance/src/instructions/token_manager.rs
Line number: 904

```
let cpi_accounts = BurnChecked {
    mint: self.mint.to_account_info(),
    from: self.user_token_account.to_account_info(),
    authority: self.mint_authority.to_account_info(), // To be updated with the
actual authority
};
```

Severity and Impact Summary

This may not work as intended as the user would need to sign to burn their tokens.

Recommendation

Ensure the `authority` matches the token account's actual owner.

Missing zero check for limit_window (possible division by zero)

Finding ID: FYEO-ONDO-04
Severity: **Low**
Status: **Open**

Description

The `limit_window` field is used in arithmetic operations without validation. If `limit_window` is `Some(0)`, subsequent division operations will panic, leading to a denial-of-service or unintended state.

Proof of Issue

File name: programs/ondo-finance/src/instructions/token_manager.rs
Line number: 688

```
let rate_per_second = token_rate_limit
    .checked_div(token_limit_window)
    .ok_or(OndoError::MathOverflow)?;
let remainder = token_rate_limit
    .checked_rem(token_limit_window)
    .ok_or(OndoError::MathOverflow)?;
```

Severity and Impact Summary

Division by zero can cause runtime panics, halting execution of affected transactions.

Recommendation

Validate that `token_limit_window` is non-zero before performing any division or remainder operations. Make sure that the codebase does not accept such configuration in the first place.

HARDCODED ADMIN KEY

Finding ID: FYEO-ONDO-05
Severity: **Informational**
Status: **Open**

Description

A hardcoded admin public key (`ROOT_PUBKEY`) introduces a single point of failure and centralized control risk. The `mainnet` constant is currently a `TODO` placeholder, which can leave the program in an uninitialized or insecure state if deployed accidentally.

Proof of Issue

File name: programs/ondo-finance/src/constants.rs
Line number: 12

```
// Admin Pubkey
#[cfg(feature = "mainnet")]
pub const ROOT_PUBKEY: Pubkey = pubkey!("TODO MAINNET ADMIN PUBKEY");
#[cfg(feature = "devnet")]
pub const ROOT_PUBKEY: Pubkey =
pubkey!("2LmKU1zcaxyXMfMxJKoKbSuFtb2E9kWca2QhEV2KEPXX");
#[cfg(feature = "localnet")]
pub const ROOT_PUBKEY: Pubkey =
pubkey!("4BZNKxYR1jrx6sZS8GVcNtFxvoiZEnWn6NdVH8Pe1Nqq");
```

Severity and Impact Summary

Centralized single key creates governance and compromise risks.

Recommendation

Use a configuration account to store the admin key rather than hardcoding it. Prefer a **multisig** for safety.

Incorrect and inconsistent event emissions

Finding ID: FYEO-ONDO-06
Severity: **Informational**
Status: **Open**

Description

Several emitted events use inconsistent or incorrect field assignments. In `RateLimitTokenSet`, the `limit_window` field mistakenly reuses `rate_limit` instead of `limit_window`, likely a copy-paste error. In addition, duplicate and conflicting `SanityCheckUpdated` events reference different fields (`self.operator.key()` vs `self.mint.key()`).

Proof of Issue

File name: programs/ondo-finance/src/instructions/initialize_token_limit.rs
Line number: 71

```
emit!(crate::events::RateLimitTokenSet {
    token: self.mint.key(),
    limit: if self.token_limit.rate_limit.is_some() {
self.token_limit.rate_limit } else { self.token_limit.default_user_rate_limit
},
    limit_window: if self.token_limit.limit_window.is_some() {
self.token_limit.rate_limit } else { self.token_limit.default_user_limit_window
},
});

emit!(SanityCheckUpdated {
    mint: self.operator.key(),
});

emit!(SanityCheckUpdated {
    mint: self.mint.key(),
});
```

Severity and Impact Summary

Misleading event data can cause off-chain indexers, monitoring tools, or analytics to process incorrect values. Duplicate and inconsistent events reduce auditability and trace accuracy.

Recommendation

Correct the event field mappings. Standardize the `SanityCheckUpdated` event to consistently reference the correct account key.

Commit Hash Reference:

For transparency and reference, the security review was conducted on the specific commit hash for the Ondo repository. The commit hash for the reviewed versions is as follows:

a3688c5cb249e45c93ce6f4ac1b4b1333d68d2dc

Conclusion:

In conclusion, the security aspects of the Ondo program remain robust and unaffected by the recent updates. Users can confidently interact with the protocol, assured that their funds are well-protected. The commitment to security exhibited by the development team is commendable, and we appreciate the ongoing efforts to prioritize the safeguarding of user assets.