FYEO

Ondo

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

Security Review Update: September 8, 2025

Reviewer: balthasar@gofyeo.com

Ondo Security Review Update

New security issues, 2

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 adds a new on-chain “sanity check” feature and its supporting state and role-management paths: administrators can initialize per-token sanity parameters (a deviation percentage and a lifetime), designate who may update them, and the token flow now enforces price deviation limits and staleness checks separately for buys and sells. Finally, the transfer hook and constants were hardened: a seed name was standardized and the hook now validates that the blocklist account really belongs to the finance program and carries the expected account discriminator before using its data.

RESIZE SKIPPED WHEN RENT_DIFF == 0

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

Description

When adding an element to blocklist/whitelist the code only calls `account_info.resize(new_account_size)?` inside the `if rent_diff != 0` branch. If the account already holds enough lamports (`rent_diff == 0`) the resize is **skipped**, so the account is not enlarged even though `new_account_size` increased. This prevents actually storing the new `Pubkey` entry while the rent state is already satisfied.

Proof of Issue

File name: programs/ondo-finance/src/instructions/admin_operations.rs
Line number: 79, 138

```
if rent_diff != 0 {
    ...

    account_info.resize(new_account_size)?;
}
```

Severity and Impact Summary

Adding to blocklist/whitelist can fail to actually expand account data when the account already has sufficient lamports. The operation will *appear* to succeed (no error path taken) but the data layout won't have space for the new entry.

Recommendation

Always call `resize(new_account_size)?`.

Code Improvements

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

Description

Several small maintainability / observability issues around the new sanity-check & role code: a misleading copy-pasted comment, missing events for parameter setters, conversions between `u64` and `i64` for `lifetime`, and duplicated `is_buy`/`else` lifetime-check logic with unhandled `None` cases. All are minor but worth fixing to improve clarity, safety, and observability.

Proof of Issue

File name: programs/ondo-finance/src/instructions/roles_operations.rs
Line numbers: 978

```
// Instruction for adding a role to the GmtokenFactory
#[derive(Accounts)]
pub struct OndoSanitySetterRemoveRole<'info> {
```

Copy & Paste comment.

File name: programs/ondo-finance/src/instructions/roles_operations.rs
Line numbers: 1028

```
impl<'info> SetSanityCheck<'info> {

    pub fn set_percentage_bp(&mut self, percentage_bp: u64) -> Result<()> {...}

    pub fn set_lifetime(&mut self, lifetime: u64 ) -> Result<()> {...}

    pub fn set_sanity_check_params(&mut self, percentage_bp: u64, lifetime: u64
) -> Result<()> {...}
```

File name: programs/ondo-finance/src/instructions/token_manager.rs
Line numbers: 489, 505

```
let lifetime_i64 =
i64::try_from(self.sanity_check_account.lifetime).map_err(|_|
OndoError::MathOverflow)?;
```

This conversion is verbose. The variable could be stored as an i64.

File name: programs/ondo-finance/src/instructions/token_manager.rs
Line numbers: 486

```
if is_buy {
    match self.token_limit_account.buy_last_updated {
        Some(last_updated) => {
            let lifetime_i64 =
i64::try_from(self.sanity_check_account.lifetime)...
            let elapsed_days =
Clock::get()?.unix_timestamp.checked_sub(last_updated)?...
            if elapsed_days > lifetime_i64 { return
Err(OndoError::LifetimeExpired.into()); }
```

```
        }
        None => {
            //return Err(OndoError::InvalidPrice.into()) TBD
        }
    }
} else {
    match self.token_limit_account.sell_last_updated {
        Some(last_updated) => {
            let lifetime_i64 =
i64::try_from(self.sanity_check_account.lifetime)...
            let elapsed_days =
Clock::get()?.unix_timestamp.checked_sub(last_updated)?...
            if elapsed_days > lifetime_i64 { return
Err(OndoError::LifetimeExpired.into()); }
        }
        None => {
            //return Err(OndoError::InvalidPrice.into()) TBD
        }
    }
}
```

The sanity_check function repeats almost identical logic for buy_last_updated and sell_last_updated. Both None branches contain a TBD comment and currently do nothing (commented //return Err(…) TBD). This duplication and missing handling create maintainability problems and potential undefined behavior when None should be treated as an error.

Severity and Impact Summary

No immediate risk — primarily affects developer ergonomics and slightly increases error surface.

Recommendation

- Replace/remove the copy-pasted comment with an accurate short comment or none.
- Emit concise events for each setter (and combined setter) to improve observability.
- Store `lifetime` as a signed type (`i64`) to match `Clock::unix_timestamp`.
- Refactor duplicated `is_buy` / `else` blocks into a small helper and implement explicit handling for the `None` case.

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:

120eedb2190b20e16907f184ac6ca8cfbbf7b734

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.