VRust

Security Assessment

O2Lab VRust Team

28/01/2022 18:21:02

Contents

Summary

This report has been prepared for O2Lab VRust Team to discover issues and vulnerabilities in the source code of the O2Lab VRust Team project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques. The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	O2Lab VRust Team
Platform	Ethereum
Language	Solana
Crate	clearing_house
GitHub Location	https://github.com/parasol-aser/vrust
sha256	Unknown

Audit Summary

Delivery Date	28/01/2022
Audit Methodology	Static Analysis
Key Components	

Vulnerability Summary

Vulnerability Level	Total
Critical	10
Major	0
Medium	0
Minor	0
Informational	0
Discussion	0

Findings

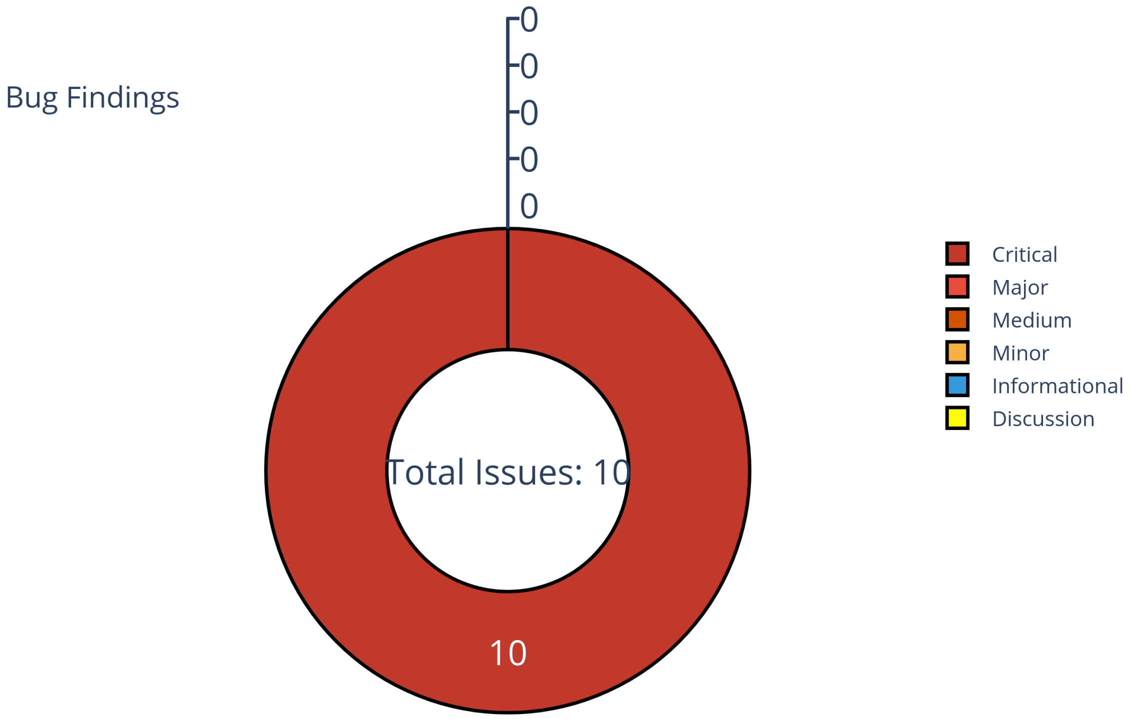

Figure 1: Findings

ID	Title	Category	Severity	Status
INT_CVE_0	Overflow	Missing Owner Check	Critical	UnResolved
INT_CVE_1	Overflow	Missing Owner Check	Critical	UnResolved
INT_CVE_2	Overflow	Missing Owner Check	Critical	UnResolved
INT_CVE_3	Overflow	Missing Owner Check	Critical	UnResolved
INT_CVE_4	Overflow	Missing Owner Check	Critical	UnResolved
INT_CVE_5	Overflow	Missing Owner Check	Critical	UnResolved
INT_CVE_6	Overflow	Missing Owner Check	Critical	UnResolved
INT_CVE_7	Overflow	Missing Owner Check	Critical	UnResolved

ID	Title	Category	Severity	Status
INT_CVE_8	Overflow	Missing Owner Check	Critical	UnResolved
INT_CVE_9	Overflow	Missing Owner Check	Critical	UnResolved

Issue: INT_CVE_0: IntegerCve - Overflow

Category	Severity	Status
Missing Owner Check	Critical	UnResolved

- Location

programs/clearing_house/src/math/position.rs:81:1: 87:2

```rust
pub fn swap_direction_to_close_position(base_asset_amount: i128) -> SwapDirection {
    if base_asset_amount >= 0 {
        SwapDirection::Add
    } else {
        SwapDirection::Remove
    }
}

```

- Call Stack

```
programs/clearing_house/src/math/position.rs
```

- description:

Description of the bug here.

- link:

GitHub Link to be added.

- alleviation:

Some alleviation steps here.

Issue: INT_CVE_1: IntegerCve - Overflow

Category	Severity	Status
Missing Owner Check	Critical	UnResolved

- Location

programs/clearing_house/src/controller/funding.rs:86:1: 192:2

```rust
pub fn update_funding_rate(
    market_index: u64,
    market: &mut Market,
    price_oracle: &AccountInfo,
    now: UnixTimestamp,
    clock_slot: u64,
    funding_rate_history: &mut RefMut<FundingRateHistory>,
    guard_rails: &OracleGuardRails,
    funding_paused: bool,
) -> ClearingHouseResult {
    let time_since_last_update = now
        .checked_sub(market.amm.last_funding_rate_ts)
        .ok_or_else(math_error!())?;

    let (block_funding_rate_update, _) =
        oracle::block_operation(&market.amm, price_oracle, clock_slot, guard_rails, None)?;

    // round next update time to be available on the hour
    let mut next_update_wait = market.amm.funding_period;
    if market.amm.funding_period > 1 {
        let last_update_delay = market
            .amm
            .last_funding_rate_ts
            .rem_euclid(market.amm.funding_period);
        if last_update_delay != 0 {
            let max_delay_for_next_period = market
                .amm
                .funding_period
                .checked_div(3)
                .ok_or_else(math_error!())?;
            if last_update_delay > max_delay_for_next_period {
                // too late for on the hour next period, delay to following period
                next_update_wait = market
                    .amm
                    .funding_period
                    .checked_mul(2)
                    .ok_or_else(math_error!())?
                    .checked_sub(last_update_delay)
                    .ok_or_else(math_error!())?;
            } else {
                // allow update on the hour
                next_update_wait = market
                    .amm
                    .funding_period
                    .checked_sub(last_update_delay)
                    .ok_or_else(math_error!())?;
            }
        }
    }

    if !funding_paused && !block_funding_rate_update && time_since_last_update >= next_update_wait {
        let mark_price_twap = amm::update_mark_twap(&mut market.amm, now, None)?;

        let one_hour_i64 = cast_to_i64(ONE_HOUR)?;
        let period_adjustment = (24_i64)
            .checked_mul(one_hour_i64)
            .ok_or_else(math_error!())?
            .checked_div(max(one_hour_i64, market.amm.funding_period))
            .ok_or_else(math_error!())?;
        // funding period = 1 hour, window = 1 day
        // low periodicity => quickly updating/settled funding rates => lower funding rate payment per interval
        let (oracle_price_twap, price_spread) = amm::calculate_oracle_mark_spread(
            &market.amm,
            price_oracle,
            cast(ONE_HOUR)?,
            clock_slot,
            None,
        )?;
        let funding_rate = price_spread
            .checked_mul(cast(FUNDING_PAYMENT_PRECISION)?)
            .ok_or_else(math_error!())?
            .checked_div(cast(period_adjustment)?)
            .ok_or_else(math_error!())?;

        let (funding_rate_long, funding_rate_short) =
            calculate_funding_rate_long_short(market, funding_rate)?;

        market.amm.cumulative_funding_rate_long = market
            .amm
            .cumulative_funding_rate_long
            .checked_add(funding_rate_long)
            .ok_or_else(math_error!())?;

        market.amm.cumulative_funding_rate_short = market
            .amm
            .cumulative_funding_rate_short
            .checked_add(funding_rate_short)
            .ok_or_else(math_error!())?;

        market.amm.last_funding_rate = funding_rate;
        market.amm.last_funding_rate_ts = now;

        let record_id = funding_rate_history.next_record_id();
        funding_rate_history.append(FundingRateRecord {
            ts: now,
            record_id,
            market_index,
            funding_rate,
            cumulative_funding_rate_long:
↪   market.amm.cumulative_funding_rate_long,
            cumulative_funding_rate_short:
↪   market.amm.cumulative_funding_rate_short,
            mark_price_twap,
            oracle_price_twap,
        });
    }

    Ok(())
}
```

```

```

- Call Stack

```
programs/clearing_house/src/controller/funding.rs
```

- description:

Description of the bug here.

- link:

GitHub Link to be added.

- alleviation:

Some alleviation steps here.

Issue: INT_CVE_2: IntegerCve - Overflow

Category	Severity	Status
Missing Owner Check	Critical	UnResolved

- Location

programs/clearing_house/src/math/funding.rs:144:1: 171:2

```rust
fn _calculate_funding_payment(
    funding_rate_delta: i128,
    base_asset_amount: i128,
) -> ClearingHouseResult<i128> {
    let funding_rate_delta_sign: i128 = if funding_rate_delta > 0 { 1 } else { -1 };

    let funding_rate_payment_mag = cast_to_i128(
        bn::U192::from(funding_rate_delta.unsigned_abs())
            .checked_mul(bn::U192::from(base_asset_amount.unsigned_abs()))
            .ok_or_else(math_error!())?
            .checked_div(bn::U192::from(MARK_PRICE_PRECISION))
            .ok_or_else(math_error!())?
            .checked_div(bn::U192::from(FUNDING_PAYMENT_PRECISION))
            .ok_or_else(math_error!())?
            .try_to_u128()?,
    )?;

    // funding_rate: longs pay shorts
    let funding_rate_payment_sign: i128 = if base_asset_amount > 0 { -1 } else { 1 };

    let funding_rate_payment = (funding_rate_payment_mag)
        .checked_mul(funding_rate_payment_sign)
        .ok_or_else(math_error!())?
        .checked_mul(funding_rate_delta_sign)
        .ok_or_else(math_error!())?;

    return Ok(funding_rate_payment);
}

```

- Call Stack

```
programs/clearing_house/src/math/funding.rs
```

- description:

Description of the bug here.

- link:

GitHub Link to be added.

- alleviation:

Some alleviation steps here.

Issue: INT_CVE_3: IntegerCve - Overflow

Category	Severity	Status
Missing Owner Check	Critical	UnResolved

- Location

programs/clearing_house/src/math/funding.rs:68:1: 128:2

```rust
fn calculate_capped_funding_rate(
    market: &Market,
    symmetric_funding_pnl: i128,
    funding_rate: i128,
) -> ClearingHouseResult<(i128, i128)> {
    let total_fee_minus_distributions_low_bound = market
        .amm
        .total_fee
        .checked_mul(SHARE_OF_FEES_ALLOCATED_TO_CLEARING_HOUSE_NUMERATOR)
        .ok_or_else(math_error!())?
        .checked_div(SHARE_OF_FEES_ALLOCATED_TO_CLEARING_HOUSE_DENOMINATOR)
        .ok_or_else(math_error!())?;

    let this_funding_rate_inflow = -(if funding_rate > 0 {
        calculate_funding_payment_in_quote_precision(funding_rate, market.base_asset_amount_long)
    } else {
        calculate_funding_payment_in_quote_precision(funding_rate, market.base_asset_amount_short)
    }?);

    let funding_rate_pnl_limit =
        if market.amm.total_fee_minus_distributions > total_fee_minus_distributions_low_bound {
            -cast_to_i128(
                market
                    .amm
                    .total_fee_minus_distributions
                    .checked_sub(total_fee_minus_distributions_low_bound)
                    .ok_or_else(math_error!())?,
            )?
        } else {
            0
        };

    // if theres enough in fees, give user's symmetric at a loss funding
    // if theres a little in fees, give the user's assymetric capped
    ↪  outflow funding
    // if theres nothing in fees/inflows, give user's no outflow funding
    let capped_symmetric_funding_pnl = max(symmetric_funding_pnl,
    ↪  funding_rate_pnl_limit);

    let this_funding_rate_outflow = if symmetric_funding_pnl <
    ↪  funding_rate_pnl_limit {
        let funding_rate_pool_size = funding_rate_pnl_limit
            .checked_sub(this_funding_rate_inflow.abs())
            .ok_or_else(math_error!())?;

        if funding_rate < 0 {
            // longs receive
            calculate_funding_rate_from_pnl_limit(
                funding_rate_pool_size,
                market.base_asset_amount_long,
            )?
        } else {
            // shorts receive
            calculate_funding_rate_from_pnl_limit(
                funding_rate_pool_size,
                market.base_asset_amount_short,
            )?
        }
    } else {
        funding_rate
    };

    return Ok((this_funding_rate_outflow, capped_symmetric_funding_pnl));
}

```

- Call Stack

```
programs/clearing_house/src/math/funding.rs
```

- description:

Description of the bug here.

- link:

GitHub Link to be added.

- alleviation:

Some alleviation steps here.

Issue: INT_CVE_4: IntegerCve - Overflow

Category	Severity	Status
Missing Owner Check	Critical	UnResolved

- Location

programs/clearing_house/src/math/collateral.rs:5:1: 17:2

```rust
pub fn calculate_updated_collateral(collateral: u128, pnl: i128) -> ClearingHouseResult<u128> {
    return Ok(if pnl.is_negative() && pnl.unsigned_abs() > collateral {
        0
    } else if pnl > 0 {
        collateral
            .checked_add(pnl.unsigned_abs())
            .ok_or_else(math_error!())?
    } else {
        collateral
            .checked_sub(pnl.unsigned_abs())
            .ok_or_else(math_error!())?
    });
}

```

- Call Stack

```
programs/clearing_house/src/math/collateral.rs
```

- description:

Description of the bug here.

- link:

GitHub Link to be added.

- alleviation:

Some alleviation steps here.

Issue: INT_CVE_5: IntegerCve - Overflow

Category	Severity	Status
Missing Owner Check	Critical	UnResolved

- Location

programs/clearing_house/src/math/repeg.rs:13:1: 53:2

```rust
pub fn calculate_repeg_candidate_pnl(
    market: &Market,
    new_peg_candidate: u128,
) -> ClearingHouseResult<i128> {
    let amm = market.amm;

    let net_user_market_position = market.base_asset_amount;

    let peg_spread_1 = cast_to_i128(new_peg_candidate)?
        .checked_sub(cast(amm.peg_multiplier)?)
        .ok_or_else(math_error!())?;

    let peg_spread_direction: i128 = if peg_spread_1 > 0 { 1 } else { -1 };
    let market_position_bias_direction: i128 = if net_user_market_position > 0 { 1 } else { -1 };

    let pnl_mag = U256::from(
        peg_spread_1
            .unsigned_abs()
            .checked_mul(PRICE_TO_PEG_PRECISION_RATIO)
            .ok_or_else(math_error!())?, // 1e10
    )
    .checked_mul(U256::from(net_user_market_position.unsigned_abs())) //1e13
    .ok_or_else(math_error!())?
    .checked_div(U256::from(
        AMM_RESERVE_PRECISION, // 1e13
    ))
    .ok_or_else(math_error!())?;

    let pnl = cast_to_i128(pnl_mag.try_to_u128()?)?
```

```
        .checked_mul(
            market_position_bias_direction
                .checked_mul(peg_spread_direction)
                .ok_or_else(math_error!())?
                .checked_mul(-1)
                .ok_or_else(math_error!())?,
        )
        .ok_or_else(math_error!())?;

    // 1e10 (PRECISION)
    return Ok(pnl);
}

```

- Call Stack

```
programs/clearing_house/src/math/repeg.rs
```

- description:

Description of the bug here.

- link:

GitHub Link to be added.

- alleviation:

Some alleviation steps here.

Issue: INT_CVE_6: IntegerCve - Overflow

Category	Severity	Status
Missing Owner Check	Critical	UnResolved

- Location

programs/clearing_house/src/controller/position.rs:175:1: 234:2

```rust
pub fn close(
    user: &mut Account<User>,
    market: &mut Market,
    market_position: &mut MarketPosition,
    now: i64,
) -> ClearingHouseResult {
    // If user has no base asset, return early
    if market_position.base_asset_amount == 0 {
        return Ok(());
    }

    let swap_direction = if market_position.base_asset_amount > 0 {
        SwapDirection::Add
    } else {
        SwapDirection::Remove
    };

    let (_base_asset_value, pnl) =
        calculate_base_asset_value_and_pnl(&market_position, &market.amm)?;

    controller::amm::swap_base_asset(
        &mut market.amm,
        market_position.base_asset_amount.unsigned_abs(),
        swap_direction,
        now,
    )?;

    user.collateral = calculate_updated_collateral(user.collateral, pnl)?;
    market_position.last_cumulative_funding_rate = 0;
    market_position.last_funding_rate_ts = 0;

    market.open_interest = market
        .open_interest
        .checked_sub(1)
        .ok_or_else(math_error!())?;

    market_position.quote_asset_amount = 0;

    market.base_asset_amount = market
        .base_asset_amount
        .checked_sub(market_position.base_asset_amount)
        .ok_or_else(math_error!())?;

    if market_position.base_asset_amount > 0 {
        market.base_asset_amount_long = market
            .base_asset_amount_long
            .checked_sub(market_position.base_asset_amount)
            .ok_or_else(math_error!())?;
    } else {
        market.base_asset_amount_short = market
            .base_asset_amount_short
            .checked_sub(market_position.base_asset_amount)
            .ok_or_else(math_error!())?;
    }

    market_position.base_asset_amount = 0;
    market_position.market_index = 0;

    Ok(())
}
```

- Call Stack

```
programs/clearing_house/src/controller/position.rs
```

- description:

Description of the bug here.

- link:

GitHub Link to be added.

- alleviation:

Some alleviation steps here.

Issue: INT_CVE_7: IntegerCve - Overflow

Category	Severity	Status
Missing Owner Check	Critical	UnResolved

- Location

programs/clearing_house/src/controller/position.rs:175:1: 234:2

```rust
pub fn close(
    user: &mut Account<User>,
    market: &mut Market,
    market_position: &mut MarketPosition,
    now: i64,
) -> ClearingHouseResult {
    // If user has no base asset, return early
    if market_position.base_asset_amount == 0 {
        return Ok(());
    }

    let swap_direction = if market_position.base_asset_amount > 0 {
        SwapDirection::Add
    } else {
        SwapDirection::Remove
    };

    let (_base_asset_value, pnl) =
        calculate_base_asset_value_and_pnl(&market_position, &market.amm)?;

    controller::amm::swap_base_asset(
        &mut market.amm,
        market_position.base_asset_amount.unsigned_abs(),
        swap_direction,
        now,
    )?;

    user.collateral = calculate_updated_collateral(user.collateral, pnl)?;
    market_position.last_cumulative_funding_rate = 0;
    market_position.last_funding_rate_ts = 0;

    market.open_interest = market
        .open_interest
        .checked_sub(1)
        .ok_or_else(math_error!())?;

    market_position.quote_asset_amount = 0;

    market.base_asset_amount = market
        .base_asset_amount
        .checked_sub(market_position.base_asset_amount)
        .ok_or_else(math_error!())?;

    if market_position.base_asset_amount > 0 {
        market.base_asset_amount_long = market
            .base_asset_amount_long
            .checked_sub(market_position.base_asset_amount)
            .ok_or_else(math_error!())?;
    } else {
        market.base_asset_amount_short = market
            .base_asset_amount_short
            .checked_sub(market_position.base_asset_amount)
            .ok_or_else(math_error!())?;
    }

    market_position.base_asset_amount = 0;
    market_position.market_index = 0;

    Ok(())
}
```

- Call Stack

```
programs/clearing_house/src/controller/position.rs
```

- description:

Description of the bug here.

- link:

GitHub Link to be added.

- alleviation:

Some alleviation steps here.

Issue: INT_CVE_8: IntegerCve - Overflow

Category	Severity	Status
Missing Owner Check	Critical	UnResolved

- Location

programs/clearing_house/src/controller/amm.rs:16:1: 50:2

```
pub fn swap_quote_asset(
    amm: &mut AMM,
    quote_asset_swap_amount: u128,
    direction: SwapDirection,
    now: i64,
    precomputed_mark_price: Option<u128>,
) -> ClearingHouseResult<i128> {
    amm::update_mark_twap(amm, now, precomputed_mark_price)?;

    let scaled_quote_asset_amount =
     ↪  scale_to_amm_precision(quote_asset_swap_amount)?;
    let round_up = direction == SwapDirection::Remove;
    let unpegged_scaled_quote_asset_amount =
        unpeg_quote_asset_amount(scaled_quote_asset_amount,
 ↪  amm.peg_multiplier, round_up)?;

    if unpegged_scaled_quote_asset_amount < amm.minimum_trade_size {
        return Err(ErrorCode::TradeSizeTooSmall);
    }

    let initial_base_asset_amount = amm.base_asset_reserve;
    let (new_base_asset_amount, new_quote_asset_amount) =
     ↪  amm::calculate_swap_output(
        unpegged_scaled_quote_asset_amount,
        amm.quote_asset_reserve,
        direction,
        amm.sqrt_k,
    )?;

    amm.base_asset_reserve = new_base_asset_amount;
    amm.quote_asset_reserve = new_quote_asset_amount;
```

```rust

    let acquired_base_asset_amount = cast_to_i128(initial_base_asset_amount)?
        .checked_sub(cast(new_base_asset_amount)?)
        .ok_or_else(math_error!())?;

    return Ok(acquired_base_asset_amount);
}

```

- Call Stack

```
programs/clearing_house/src/controller/amm.rs
```

- description:

Description of the bug here.

- link:

GitHub Link to be added.

- alleviation:

Some alleviation steps here.

Issue: INT_CVE_9: IntegerCve - Overflow

Category	Severity	Status
Missing Owner Check	Critical	UnResolved

- Location

programs/clearing_house/src/math/fees.rs:83:1: 95:2

```
fn calculate_token_discount_for_tier(
    fee: u128,
    tier: &DiscountTokenTier,
    discount_token: TokenAccount,
) -> Option<u128> {
    if discount_token.amount >= tier.minimum_balance {
        return Some(
            fee.checked_mul(tier.discount_numerator)?
                .checked_div(tier.discount_denominator)?,
        );
    }
    return None;
}

```

- Call Stack

```
programs/clearing_house/src/math/fees.rs
```

- description:

Description of the bug here.

- link:

GitHub Link to be added.

- alleviation:

Some alleviation steps here.

Appendix

Copied from https://leaderboard.certik.io/projects/aave

Finding Categories

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum” command against the target file.

Disclaimer

Copied from https://leaderboard.certik.io/projects/aave

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. 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 CertiK's prior written consent in each instance.

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 considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report 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 in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK'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 CertiK is subject to dependencies and 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.