BLOCKAPEX

SMART CONTRACT SECURITY

V 1.0

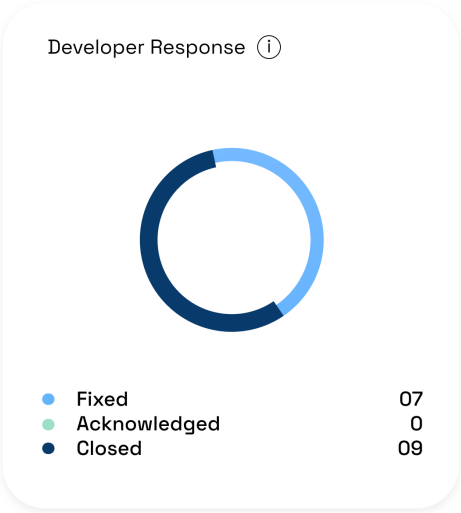
DATE: 16th AUG 2024

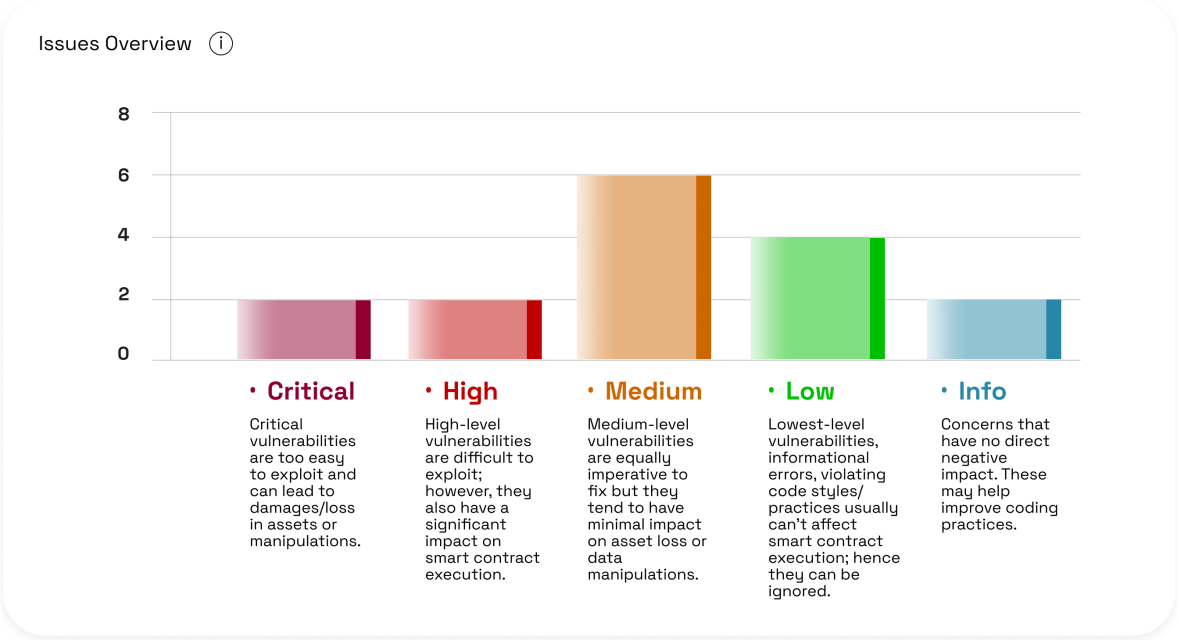
PREPARED FOR: ENSOFI

Contents

1 Executive Summary

EnsoFi engaged BlockApex for a security review of the EnsoFi Lending Protocol. A team of two security researchers reviewed the source code of the Lending Borrowing module for 10-days of effort. The details of project scope, complexity and coverage are provided in the the following sections of this report.





1.1 Summary of Findings Identified

S.No	Severity	Findings	Status
#1	CRITICAL	Potential Reinitialization Attack	FIXED
#2	CRITICAL	Erroneous status update leads to funds locking	FIXED
#3	HIGH	Precision Errors due to Unsafe Typecasting Result in Lost Funds	CLOSED
#4	HIGH	Potential Frontrunning Scenario	FIXED
#5	MEDIUM	Mismatch Health Ratio	FIXED
#6	MEDIUM	Interest Miscalculation: Charged For Full Loan Duration Despite Early Repayment	CLOSED
#7	MEDIUM	Centralization In the EnsoFi Lending Protocol	CLOSED
#8	MEDIUM	Liquidation Lacks Health Ratio and Contract Expiry Verification	FIXED
#9	MEDIUM	Improper Account Type in InitSettingAccount	FIXED
#10	MEDIUM	Missing internal Discriminator Spacing	FIXED
#11	LOW	Missing Constraints on Various Instances	CLOSED
#12	LOW	Insufficient Input Validations for withdraw_amount	CLOSED

S.No	Severity	Findings	Status
#13	LOW	Insufficient Validations during Setting Account Initialization	CLOSED
#14	LOW	Insufficient Validations for Setting Account Editting	CLOSED
#15	INFO	Missing Check During LendOffer Creation	CLOSED
#16	INFO	Missing Event field in edit_setting_account	CLOSED

1.2 Scope

1.2.1 In Scope

Overview of the EnsoFi Lending:

The EnsoFi Lending module facilitates lending and borrowing within the EnsoFi ecosystem, leveraging a combination of decentralized smart contracts and centralized relayers for operations. Decentralized smart contracts handle core lending, borrowing, and collateral management functions, while centralized relayers and servers facilitate communication and execute admin-controlled actions based on emitted events. This ensures that assets remain secure and efficiently managed on their native chains. The Wormhole infrastructure is not being used at the moment but is included in the roadmap of EnsoFi Protocol including a mix of on-chain and off-chain processes manage the various functionalities. Currently, all functionality is based on Solana, and cross-chain lending is not yet implemented. Consequently, the audit did not include attack vectors related to cross-chain lending.

Files in scope:

- enso-finance/programs/enso-lending/src/**

Intial Audit Commit Hash: 408b298ace3d7a1fb99967395910581df3774ebf

Final Audit Commit Hash: 408b298ace3d7a1fb99967395910581df3774ebf

Note: Any external calls to composable smart contracts are assumed to be safe.

Key Functionalities

1. **Lending and Borrowing**:
    1. Implement `create_loan_offer`, `deposit_collateral_loan_offer`, and `repay_loan_offer` to handle the lifecycle of loan offers on Solana.
    2. Manage core lending and borrowing functionalities through decentralized smart contracts.
2. **Collateral Management**:
    1. Ensure assets remain on their native chains, with Wormhole messages used solely for verification and accounting.
    2. Manage collateral deposits and withdrawals through functions such as `deposit_collateral_loan_offer`, `withdraw_collateral`, and `liquidate_collateral`.
3. **Interest Rate Management**:
    1. Enable mutually agreed interest rates between lenders and borrowers through functions like `create_lend_offer` and `edit_lend_offer`.
    2. Maintain a constant interest rate for the duration of the lending period.

4. **Price Feeds and Health Ratio**:
    1. Integrate with Pyth Network for accurate and timely price feeds, ensuring financial stability within the platform.
    2. Implement a health ratio mechanism to monitor and manage the risk of borrower positions based on their collateral.
5. **Liquidation Process**:
    1. Define a liquidation threshold and process to handle positions that fall below the health ratio threshold or reach the end of the contract period.
    2. Ensure that collateral is sold and leftover funds are managed appropriately between lenders and borrowers.
6. **Event-Driven Operations**:
    1. Emit events during key operations, which the centralized relayer listens to in order to perform necessary admin actions.
    2. Ensure the centralized server acts responsibly based on the events emitted by the smart contracts.

1.2.2 Out of Scope

Any features or functionalities not explicitly mentioned in the “In Scope” section are considered outside the scope of this security review.

1.3 Centralization Risks

Some of the components functionalities within the Ensofi Lending Protocol pose centralization risk of critical severity potentialy leading to permanent loss of users’ funds and protocol features’ unavailability via DoS. This and similar potentially vulnerabilites arising due to centralization are described in the findings titled, **Centralization In the Solana Program** and **Missing Constraints on Various Instances** which are detailed in subsequent sections of this audit report.

1.4 Methodology

The codebase was audited using a filtered audit technique. A band of two (2) auditors scanned the codebase in an iterative process for a time spanning 10 days. Starting with the recon phase, a basic understanding of the code was developed, and the auditors worked on developing presumptions for the developed codebase and the relevant documentation/ whitepaper. Furthermore, the audit moved on with the manual code reviews to find logical flaws in the codebase complemented with code optimizations, software, and security design patterns, code styles and best practices.

1.5 Status Description

Acknowledged: The issue has been recognized and is under review. It indicates that the relevant team is aware of the problem and is actively considering the next steps or solutions.

Fixed: The issue has been addressed and resolved. Necessary actions or corrections have been implemented to eliminate the vulnerability or problem.

Closed: This status signifies that the issue has been thoroughly evaluated and acknowledged by the development team. While no immediate action is being taken.

2 Findings and Risk Analysis

2.1 Potential Reinitialization Attack

Severity: Critical

Status: Fixed

Location :

1. src/contexts/init_setting_account.rs#L22-L33

Description To initialize the `setting_account`, `init_setting_account`() is called. The constraint `init_if_needed` is used to initialize the `setting_account`. As per the anchor docs, "The init_if_needed constraint does the same thing as the init constraint, only if the account has already been initialized the instruction will still run." This mean the possiblility of reinitialization. If the same `tier_id` is passed it will overwrite its data with new data.

Impact In case of reinitialization the data is overwritten which can messup with the underlaying `setting_account` params and the lending/borrowing functionality under the hood.

Proof of Concept The current implementation allows reinitialization at any point of time, which allows data overwriting, ultimately leading to data loss. The contract allows succesfully calling `init_setting_account` twice which should ideally fail and prevent the admin to overwrite an active `setting_account` without first closing it.

Recommendation It is worth noting that since there is `edit_setting_account` function which allows the editing of the `setting_account`, hence instead of using `init_if_needed`, `init` constraint should be used. The `init` constraint ensures that instruction can only be called once per account.

2.2 Erroneous status update leads to funds locking

Severity: Critical

Status: Fixed

Location :

1. src/contexts/system_repay_loan_offer.rs

Description The EnsoFi Lending Protocol allows borrowers to repay their loans to the system before expiry. Once the user calls the `repay_loan_offer` in order to eventually recieve their collateral back after paying the borrowed assets, the EnsoFi contract updates the borrower's `LoanOfferStatus` from `FundsTransferred` to `BorrowerPaid` instead of `Repay`. This erroneous status update causes discrepancy in the lifecycle of a loan as the BorrowerPaid status is expected at the last step where the repayment has been done as well as the system knows that the Loan Offer is now ready to be finished. Therefore, the status update to finishing a loan offer via user repayment is impossible since the constraint for the call `system_repay_loan_offer` requires the `Repay` status for the `LoanOffer` as implemented in the codebase here:

```
    constraint = loan_offer.status == LoanOfferStatus::Repay @ RepayOfferError::
        InvalidOfferStatus,
```

Impact With the wrong status update it is now possible for the system to not pay the Borrower back leading to permanent loss of funds.

Recommendation The `LoanOfferStatus` update during the `repay_loan_offer` call should update the borrower's offer status to `Repay` instead of `BorrowerPaid` as here, buggy code snippet below:

`self.loan_offer.status = LoanOfferStatus::BorrowerPaid;` **// @audit wrong status update here**

2.3 Precision Errors due to Unsafe Typecasting Result in Lost Funds

Severity: High

Status: Closed

Location :

1. src/contexts/repay_loan_offer.rs#L139
2. src/contexts/system_liquidate_loan_offer.rs#L139

Description When performing the `RepayLoanOffer`, `total_repay` is calculated. The formula used to calculated the `repay_amount` is `(setting_account.amount as f64 + interest_amount + borrower_fee_amount) as u64`. When casting is performed into `u64` , precision loss happens due to rounding down and the result returned is less than anticipated. Similar is the case during `system_liquidate_loan_offer` in which repay amount is calculated as `(collateral_swapped_amount as f64 - self.loan_offer.borrow_amount as f64 - interest_loan_amount - borrower_fee_amount)as u64`

Impact Due to Precision loss the amount returned is less than what actually should be retunrned. Borrower should be returned the actual amount.

Recommendation It is recommended to apply some scaling factor and do calculation to minimize the precision loss.

Developer Response

1. `system_repay_loan_offer` is removed
2. No change

Currently all calculation is based on LAMPORT unit, so precision loss here is only 10^-9.

2.4 Potential Frontrunning Scenario

Severity: High

Status: Closed

Location :

1. src/contexts/edit_loan_offer.rs
2. src/contexts/create_loan_offer.rs

Description When creating a `lend_offer` Lender have to specify an interest rate. Later this can also be changed via `edit_lend_offer`. When a borrower create a `loan_offer` , the interest is fetched from the `lend_offer.interest`. Now this is critical because the interest charged to borrower is based on this. Borrower can be frontrunned by the lender via the `edit_lend_offer` and forced to use the new interest amount causing financial damage.

Impact The impact of this is high as financial damage can be caused to the borrower.

Proof of Concept :

Attack Scenario

1. User Post a lucrative lend offer with low interest rate.
2. Borrower put up the loan offer.
3. Lender Front-runs the loan offer transaction and call `edit_lend_offer` with higher interest rate.
4. Borrower is forced to pay higher interest rate on the loan amount causing financial damage , or if 100% interest is set then full collateral loss.

Recommendation It is recommended to put a `cooldown period` after editing the `lend_offer` , this should be checked in the `loan_offer` creation too. Within the cooldown period the offer from the borrower should not be matched with the `lend_offer`.

2.5 Mismatch Health Ratio

Severity: Medium

Status: Fixed

Location :

1. src/common/constant.rs

Description In the Enso Docs the `minimum_borrow_health_ration` is defined as `1.2` while in the code its defined as `1.1`. As per the docs: "When a contract's heath ratio goes below 1.2 or a contract expires, we will execute the Liquidation Event." This means that the `minimum_borrow_health_ratio` should be `1.2` not `1.1`.

Recommendation It is recommended to either update the docs or code. Code and Docs should match.

2.6 Interest Miscalculation: Charged For Full Loan Duration Despite Early Repayment

Severity: Medium

Status: Closed

Location :

1. src/contexts/repay_loan_offer.rs#L133

Description When a `setting_account` is initialized, duration of the loan is provided. When a borrower borrows the amount and then returns it within the duration period. E.g Loan Duration is 14 Days , returned in 5 Days. The duration considered during calculation of the interest amount is the full duration of the loan.

Impact The `interest_amount` paid by the user is on the full duration but the amount utilized was for less duration.

Recommendation It is recommended to calculate `time_borrowed` based on the `loan_starting` and `loan_repay` timestamp.

Developer Response

"No change. It is our requirements."

2.7 Centralization In the EnsoFi Lending Protocol

Severity: Medium

Status: Closed

Location :

1. src/contexts/system_liquidate_loan_offer.rs#L58
2. src/contexts/system_finish_loan_offer.rs#L57

Description When the system is performing the liquidation, `collateral_swapped_amount` is passed in the `system_liquidate_loan_offer`. This is externally supplied value which can be an arbitrary value. There is a risk of centralization due to this as it can be anything.

Similary is the case in the `system_cancel_lend_offer`() where `waiting_interest` is passed into the function. There is no formula or calculation available in the docs and code on how the value of `waiting_interest` will be calculated. This shows a centralization approach which is frowned upon in Defi Space.

Recommendation For collateral swapping it is recommended to perform a swap via CPI on DEX like Jupiter and return appropriate funds to the user. In case of `waiting_interest` a formula should be written into the code and interest should be calculated based on it rather passing into the function.

Ref: docs.jup.ag/docs/apis/cpi

Developer Response

No change.

- Waiting interest is calculated in realtime depending on our 3rd parties lending platform. Have no formula for that.
- About using cpi for swap, we notice that transaction can be failed at runtime

"Due to Solana's transaction limit of 1232 bytes, swaps via CPI will likely fail at runtime since Jupiter routes may involve multiple DEXes in order to reduce price impact."

- We're using jupiter now, but we may expand to other plaform in future. So we decide to swap by our own, then submit that tx when process finish (by system_liquidate_loan_offer)

2.8 Liquidation Lacks Health Ratio and Contract Expiry Verification

Severity: Medium

Status: Fixed

Location :

1. src/contexts/liquidate_collateral.rs#L35

Description When a liquidation event is hit `start_liquidate_contract` is called by the system. There is no check for the health ratio in this function which ensures that the `health_ratio` is indeed below the threashold. i.e `1.2` moreover there is also a missing check whether the contract has indeed expired or not.

Impact The check for the health ratio is one of the important factor for liquidation. If this check is not present then the system is freely allowed to liquidate anyone at anytime since there is no particular check against this.

Recommendation It is recommended to put these checks to ensure that if these conditions are met then in that case liquidations should start otherwise borrower can be liquidated at any point.

2.9 Improper Account Type in InitSettingAccount

Severity: Medium

Status: Fixed

Location :

1. src/contexts/init_setting_account.rs#L18-L20
2. src/contexts/edit_setting_account.rs#L16-L18

Description System Owner needs to initilize the `setting_account` for the lending/borrowing functionality. In this `lend_price_feed_account` and `collateral_price_feed_account` are passed the type applied on these is `AccountInfo<&#x27;info>`. As per Pyth documentation `Account<&#x27;info, PriceUpdateV2>` should be used since it ensure that the account passed to their instruction is owned by the Pyth Pull Oracle program.

Impact Validations are important in Solana contract hence if invalid account is passed here oracle won't work properly.

Recommendation It is recommended to use `Account<&#x27;info, PriceUpdateV2>` type , it will automatically performs the checks.

2.10 Missing internal Discriminator Spacing

Severity: Medium

Status: Fixed

Location :

1. init_settings_account
2. create_lend_offer
3. create_loan_offer
4. create_loan_offer_native

Description The Anchor's InitSpace macro used throughout the EnsoFi Lending Module requires the initialized accounts to allocate space for the account's data when the size of an account cannot be predicted. However, while using the INIT_SPACE macro, the docs strictly state that, "In addition to the space for the account data, you have to add 8 to the space constraint for Anchor's internal discriminator.", which is not applied in the current implementation of EnsoFi Lending Module.

Recommendation Add 8 bytes for anchor's internal spacing while declaring the space attribute for account macros, e.g.:

```
    #[account(
      init,
      payer = borrower,
      space = 8 + LoanOfferAccount::INIT_SPACE,
      seeds = [
        ENSO_SEED.as_ref(),
        LOAN_OFFER_ACCOUNT_SEED.as_ref(),
        borrower.key().as_ref(),
        offer_id.as_bytes(),
        crate::ID.key().as_ref()
      ],
      bump
    )]
    pub loan_offer: Account&lt;&#x27;info, LoanOfferAccount&gt;,
```

References :

- Link to Anchor Lang Docs - click here

2.11 Missing Constraints on Various Instances

Severity: Low

Status: Closed

Location :

1. src/contexts/system_cancel_lend_offer.rs
2. src/contexts/system_finish_loan_offer.rs
3. src/contexts/system_repay_loan_offer.rs
4. src/contexts/system_update_loan_offer.rs

Description In various instances execution is needed to be done by the system i.e `OPERATE_SYSTEM_PUBKEY` . Checks on the address passed as `Signer` are needed. We found following context where these checks are missing.

1. `SystemCancelLendOffer` : system need to cancel the lend offer after the lender requested it. Here there is a missing constraint on the signer i.e it should be the `OPERATE_SYSTEM_PUBKEY`
2. `SystemFinishLoanOffer`: system need to call `system_finish_loan_offer`() to conclude the loan offer , Here there is a missing constraint on the signer i.e it should be the `OPERATE_SYSTEM_PUBKEY`
3. `SystemRepayLoadOfferNative`: system need to call this to transfer the `collateral_amount` to borrower, Here there is a missing constraint on the signer i.e it should be the `OPERATE_SYSTEM_PUBKEY`
4. `SystemUpdateLoanOffer`: system needs to call this to transfer the `borrow_amount` to borrower , Here there is a missing constraint on the signer i.e it should be the `OPERATE_SYSTEM_PUBKEY`

Recommendation It is recommended to add the constraints so that only system signer is able to call the functions which are required to be called.

Developer Response

No change

2.12 Insufficient Input Validations for withdraw_amount

Severity: Low

Status: Closed

Location :

1. src/contexts/system_withdraw_native.rs#L66

Description Borrower can call `withdraw_collateral`() to request withdraw collateral keeping in view the `health_ratio`. In turn the `system_transfer_collateral_request_withdraw` is called to transfer the collateral to borrower. Since `withdraw_amount` is passed into this function its imperative to put a check on it. It was observed that there is a missing check for the `withdraw_amount` being equal to `loan_offer.request_withdraw_amount` which is set when `withdraw_collateral` is called.

Recommendation It is recommended to put the check so if in anycase a wrong amount is passed into the `system_transfer_collateral_request_withdraw`() it should not allow the transfer of amount.

Developer Response

- We've removed this function. No use in this version.

2.13 Insufficient Validations during Setting Account Initialization

Severity: Low

Status: Closed

Location :

1. src/contexts/init_setting_account.rs#L43-L56

Description System Owner needs to initilize the `setting_account` for the lending/borrowing functionality. There are no validations applied on the fields such as `lender_fee_percent`, `borrower_fee_percent`, `amount` and `duration`. If `0` or `MAX value` is passed it would be set. Moreover there are missing checks on further attributes `receiver`, `lend_mint_asset`, `collateral_mint_asset`.

Impact Undesired values if passed can have bad impact on the setting_account. Basic validation for zero-values are must unless values such as fees are required to be set as zero for promotional purposes.

Recommendation It is recommended to put validations on the fields and restrict undesired inputs. Receiver should be checked if its a valid `hot_wallet_account`. `lend_mint_asset` and `collateral_mint_asset` should also be verified against whitetlisted addresses.

Developer Response

- No change. this function is only use by system wallet.

2.14 Insufficient Validations for Setting Account Editting

Severity: Low

Status: Closed

Location :

1. src/contexts/edit_setting_account.rs

Description Owner can edit the `setting_account` and update the following state variables without any bounding conditions; `amount`, `duration`, `lender_fee_percent`, `borrower_fee_percent`, `receiver`, `lend_mint_asset`, `collateral_mint_asset`, `lend_price_feed` and `collateral_price_feed`. Validations to be applied on these parameters so correct data is updated.

Recommendation It is recommended to put validations on the fields and restrict undesired inputs. Receiver should be checked if its a valid `hot_wallet_account`. `lend_mint_asset` and `collateral_mint_asset` should also be verified against whitetlisted addresses.

Developer Response

- No change. This function is only use by system wallet."

2.15 Missing Check During LendOffer Creation

Severity: Info

Status: Closed

Location :

1. src/contexts/edit_lend_offer.rs#L29-L29

Description During the editing of `lending_offer` there is check if the interest is less than `0` but if the interest is same as before it would successfully update the offer which seems redundant and cause waste of fees.

Recommendation It is recommended to put a check if the interest is same as before , transaction should not be fullfilled.

2.16 Missing Event field in edit_setting_account

Severity: Info

Status: Closed

Location :

1. src/contexts/edit_setting_account.rs#L81

Description When the `setting_account` is edited, an event is emited with the updated field attributes. It was observed that `borrower_fee_percent` is missing from the event.It should be added as events are normally monitored to trigger more functionalities.

Recommendation It is recommended to add the necessary field in the event.

Disclaimer:

The smart contracts provided by the client with the purpose of security review have been thoroughly analyzed in compliance with the industrial best practices till date w.r.t. Smart Contract Weakness Classification (SWC) and Cybersecurity Vulnerabilities in smart contract code, the details of which are enclosed in this report.

This report is not an endorsement or indictment of the project or team, and they do not in any way guarantee the security of the particular object in context. This report is not considered, and should not be interpreted as an influence, on the potential economics of the token (if any), its sale, or any other aspect of the project that contributes to the protocol's public marketing.

Crypto assets/ tokens are the results of the emerging blockchain technology in the domain of decentralized finance and they carry with them high levels of technical risk and uncertainty. No report provides any warranty or representation to any third-party in any respect, including regarding the bug-free nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the reports in any way, including to make any decisions to buy or sell any token, product, service, or asset. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and is not a guarantee as to the absolute security of the project.

Smart contracts are deployed and executed on a blockchain. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. The scope of our review is limited to a review of the programmable code and only the programmable code, we note, as being within the scope of our review within this report. The smart contract programming language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer or any other areas beyond the programming language's compiler scope that could present security risks.

This security review cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While BlockApex has done their best in conducting the analysis and producing this report, it is important to note that one should not rely on this report only - we recommend proceeding with several independent code security reviews and a public bug bounty program to ensure the security of smart contracts.