OFFSIDE LABS

Optimex Vault

Smart Contract Security Assessment

March 2025

Prepared for:

Optimex

Prepared by:

Offside Labs

Sirius Xie

Tim Li

Contents

1 About Offside Labs

Offside Labs stands as a pre-eminent security research team, comprising highly skilled hackers with top - tier talent from both academia and industry.

The team demonstrates extensive and diverse expertise in modern software systems, which encompasses yet are not restricted to *browsers*, *operating systems*, *IoT devices*, and *hypervisors*. Offside Labs is at the forefront of innovative domains such as *cryptocurrencies* and *blockchain technologies*. The team achieved notable accomplishments including the successful execution of remote jailbreaks on devices like the **iPhone** and **PlayStation 4**, as well as the identification and resolution of critical vulnerabilities within the **Tron Network**.

Offside Labs actively involves in and keeps contributing to the security community. The team was the winner and co-organizer for the *DEFCON CTF*, the most renowned CTF competition in Web2. The team also triumphed in the **Paradigm CTF 2023** in Web3. Meanwhile, the team has been conducting responsible disclosure of numerous vulnerabilities to leading technology companies, including *Apple*, *Google*, and *Microsoft*, safeguarding digital assets with an estimated value exceeding **$300 million**.

During the transition to Web3, Offside Labs has attained remarkable success. The team has earned over **$9 million** in bug bounties, and **three** of its innovative techniques were acknowledged as being among the **top 10 blockchain hacking techniques of 2022** by the Web3 security community.

2 Executive Summary

Introduction

Offside Labs completed a security audit of *Optimex* smart contracts, starting on March 4, 2025, and concluding on March 26, 2025.

Project Overview

Optimex enables a DeFi layer on Bitcoin. Optimex Swap is a non-custodial, RFQ-based Bitcoin trading protocol designed for high liquidity and fast settlement. The Optimex Vault temporarily holds assets during trades, ensuring security and compatibility across Bitcoin, Ethereum, and Solana.

On Ethereum and Solana, it uses smart contracts for deposits, trade settlements, and allows users to reclaim unsettled assets. Market Makers also complete payments through on-chain contracts.

Audit Scope

The assessment scope contains mainly the smart contracts of the sol-smartcontract program and the evm-smartcontract program for the *Optimex* project.

The audit is based on the following specific branches and commit hashes of the codebase repositories:

- sol-smartcontract
  - Codebase: `https://github.com/optimex-xyz/sol-smartcontract`
  - Commit Hash: bc19ea33ef3c8ac9e1428830ef45f54d74015419
- evm-smartcontract
  - Codebase: `https://github.com/optimex-xyz/evm-smartcontract`
  - Commit Hash: e83246c8345bdcee78f61176b1276ec65453b9ea

We listed the files we have audited below:

- sol-smartcontract
  - programs/**/*.rs
- evm-smartcontract
  - contracts/asset-chain/Payment.sol
  - contracts/asset-chain/Protocol.sol
  - contracts/asset-chain/TokenVault.sol
  - contracts/asset-chain/NativeVault.sol
  - contracts/asset-chain/utils/BaseVault.sol

Findings

The security audit revealed:

- 0 critical issue
- 0 high issue
- 1 medium issues
- 3 low issues
- 6 informational issues

Further details, including the nature of these issues and recommendations for their remediation, are detailed in the subsequent sections of this report.

3 Summary of Findings

ID	Title	Severity	Status
01	Missing Check Before Closing TokenAccount May Cause IX Failure	Medium	Fixed
02	Integer Truncation in Deposit IX Leading to Validation Bypass	Low	Fixed
03	Rent of WhitelistToken Can Be Stolen by Malicious Operators Upon Account Closure	Low	Acknowledged
04	Missing Validation for Pubkeys Can Lead to Permanent Token Lockup	Low	Acknowledged
05	Missing Validation on total_fee in set_total_fee IX	Informational	Fixed
06	Possible Incorrect Error Type Used in withdraw_total_fee IX	Informational	Fixed
07	No Event Emitted in Payment and Deposit	Informational	Acknowledged
08	Missing Validation on trade_id and Other Parameters in payment IX	Informational	Acknowledged
09	Possible Payment Failures Due to Deliberate User Rejection	Informational	Acknowledged
10	Possible Payment Failures Due to Trade Timeout	Informational	Acknowledged

4 Key Findings and Recommendations

4.1 Missing Check Before Closing TokenAccount May Cause IX Failure

Severity: Medium	Status: Fixed
Target: Solana Smart Contract	Category: Logic Error

Description

In the `close_finished_trade` IX, the ATA of `TradeVault` will be closed. However, before closing this token account, there is no check to verify whether its amount is zero.

If someone directly transfers 1 token to this account via SPL Token Transfer, the `token::close_account` instruction will fail with `TokenError::NonNativeHasBalance`, causing the `close_finished_trade` IX to fail as well.

Impact

An attacker can deposit a small amount of dust tokens into the malicious `TradeVault` ATA, preventing this token account from being closed. Additionally, there are no other IXs in the program that support transferring dust from the TradeVault ATA to other accounts. As a result, the rent for the `TradeDetail` and `TradeVault` accounts associated with this trade cannot be refunded to the user due to the failed execution of the `close_finished_trade` IX.

Recommendation

It is recommended to check whether there are any remaining tokens in the account before closing it. If there are, it is advisable to transfer out the excess tokens first.

Mitigation Review Log

Fixed in the commit 0829ba4de9cc3d3d0b59a0e6a7a5db87ce9f1574.

4.2 Integer Truncation in Deposit IX Leading to Validation Bypass

Severity: Low	Status: Fixed
Target: Solana Smart Contract	Category: Code QA

Description

In the `deposit` IX, the `bytes_to_u64_number` function is used to convert the 32-byte `amount_in` parameter from the IX into a `u64 amount`.

```
31      // Find first non-zero byte from left (big-endian)
32      let mut start_idx = 0;
33      while start_idx < amount_in_bytes.len() && amount_in_bytes[start_idx]
          ↪ == 0 {
34          start_idx += 1;
35      }
36
37      // Take last 8 bytes that contain the value
38      let value_bytes = if amount_in_bytes.len() - start_idx >= 8 {
39          let end_idx = amount_in_bytes.len();
40          let start_idx = end_idx - 8;
41          &amount_in_bytes[start_idx..end_idx]
```

utils.rs#L31-L41

Within this function, it first identifies the first non-zero high-order byte in big-endian format and then checks if the sequence of bytes from this point to the end exceeds 8 bytes. If it does, only the last 8 bytes are taken as the final `u64 amount`.

Impact

Suppose `amount_in` is `0x100000002`, this truncation will cause `bytes_to_u64_number` to return `2`. Malicious users might exploit this truncation to bypass the `whitelist_token.amount > number_from_bytes` check and deposit a small `amount` into the `TradeVault`. If the off-chain Solver/Oracle does not perform sufficient validation on the results after a deposit, an attacker could exploit this integer truncation issue to swap a small deposit on the Solana Vault side for a large amount of output tokens.

Recommendation

In Solana SPL, both `Mint.supply` and `TokenAccount.amount` use `u64` as their default types. Generally, standard SPL Mint amounts will not exceed `u64::MAX`.

Here are some recommendations:

- Add a check or error handling in `bytes_to_u64_number` to address cases where `amount_in > u64::MAX`.
- Verify whether `TradeInfo.from_chain` is Solana.

Since this audit does not cover off-chain programs, it is suggested to also add sufficient checks off-chain to ensure the validity and correctness of the deposit input parameters, preventing misuse by attackers.

Mitigation Review Log

Fixed in the commit 0ab2f6bc0f6bc50e8b7c94c926faaa4d348eb8cf.

4.3 Rent of WhitelistToken Can Be Stolen by Malicious Operators Upon Account Closure

Severity: Low	Status: Acknowledged
Target: Solana Smart Contract	Category: Logic Error

Description

The `WhitelistToken` account is created by an operator from `config.operators` in the `add_or_update_whitelist` IX. This account will be closed in the `remove_whitelist` IX, and the rent will be transferred to the operator who invokes the `remove_whitelist` IX.

Since there are multiple operators in `config`, a `WhitelistToken` created by operator A can be closed by operator B, resulting in the rent paid by operator A being transferred to operator B upon account closure.

Impact

If there is a malicious operator in `config` or an operator falls victim to a phishing attack, all `WhitelistToken` account rents could be taken by this operator.

Recommendation

If not all operators in `config` can be fully trusted, it is recommended to return the rent to the original rent payer when closing the `WhitelistToken` account.

Mitigation Review Log

Optimex Team: The rent fee is small, and adding an additional account into context is not preferred to avoid unnecessary complexity.

Offside Labs: Ensure that all operators involved in the program are verified and trustworthy.

4.4 Missing Validation for Pubkeys Can Lead to Permanent Token Lockup

Severity: Low	Status: Acknowledged
Target: Solana Smart Contract	Category: Data Validation

Description

In the `deposit` IX, there is no validation for `TradeDetailInput.mpc_pubkey` and `TradeDetailInput.refund_pubkey`. If either of these pubkeys is set to `Pubkey::default`, the user's swap may not complete as expected.

Impact

- When `TradeDetailInput.mpc_pubkey` is `Pubkey::default`, the settlement cannot be executed, preventing `TradeVault` from transferring tokens to the PMM.
- When `TradeDetailInput.refund_pubkey` is `Pubkey::default`, the claim cannot be executed. If the swap does not go through, the user will be unable to withdraw the tokens deposited into `TradeVault`.
- If both `TradeDetailInput.mpc_pubkey` and `TradeDetailInput.refund_pubkey` are set to `Pubkey::default`, the tokens deposited by the user into `TradeVault` will be permanently locked.

Recommendation

It is recommended to add validation for `TradeDetailInput.mpc_pubkey` and `TradeDetailInput.refund_pubkey` to ensure that both pubkeys are valid.

Mitigation Review Log

Optimex Team: This validation is unnecessary. Even if we verify that mpc_pubkey and refund_pubkey aren't set to their default values, numerous other scenarios could still result in permanent fund loss - for example, if the address is mistakenly set to the Token Program. When constructing the instruction, these addresses are required parameters that clients must provide. The responsibility should be on the client to validate and correctly pass these values, rather than relying on contract-level validation.

4.5 Informational and Undetermined Issues

Missing Validation on total_fee in set_total_fee IX

Severity: Informational	Status: Fixed
Target: Solana Smart Contract	Category: Data Validation

In the `set_total_fee` IX, the program sets the fee amount collected by the protocol for `TradeDetail` during transfers. However, it does not check the relationship between `set_total_fee_args.amount` from the IX parameters and `TradeDetail.amount`. If the `total_fee` set in the IX is greater than `TradeDetail.amount`, it will cause an integer underflow in subsequent `settlement` IX, making it unable to execute.

Possible Incorrect Error Type Used in withdraw_total_fee IX

Severity: Informational	Status: Fixed
Target: Solana Smart Contract	Category: Code QA

In the `withdraw_total_fee IX` , if `to_user != fee_receiver.receiver` , the IX will throw an error with the type `InvalidRefundPubkey` . Based on the literal meaning of this error type, it is more relevant to the `claim IX` . It is recommended to use a more appropriate error type here.

No Event Emitted in Payment and Deposit

Severity: Informational	Status: Acknowledged
Target: Solana Smart Contract	Category: Code QA

The program defines the `Deposited` and `PaymentTransferred` events, but they are not emitted in the corresponding `deposit` and `payment` IXs. If the on-chain contract and off-chain program rely on events emitted by the contract for data transmission, it is recommended to emit the corresponding events to these IXs.

Mitigation Review Log:

Optimex Team: We don't use events as a source for tracking because we cannot fully trust these logs.

Missing Validation on trade_id and Other Parameters in payment IX

Severity: Informational	Status: Acknowledged
Target: EVM & Solana Smart Contract	Category: Data Validation

In the Solana `payment IX` , the transfer is executed directly based on the input parameters.

For `PaymentArgs.token` , it only requires the token to be in the `WhitelistToken List` , without verifying whether the token matches the one required by the Trade corresponding to `trade_id` .

For `PaymentArgs.amount` , there is no check to ensure that the amount is within the expected range for the Trade corresponding to `trade_id` .

Additionally, since this IX is `permissionless` , any user can invoke it. If a malicious user initiates an unexpected payment for a specific `trade_id` , please ensure that there are sufficient checks in the off-chain program to prevent the PMM associated with the `trade_id` from being affected (slashed).

This issue exists in both EVM and Solana contracts.

Mitigation Review Log:

Optimex Team: The payment contract on the asset chain does not contain detailed information about trades occurring on different chains. Instead, the PMM must correctly specify

the tradeId and other details, such as paymentAmount, on their own. Any incorrect information that does not match the recorded trades stored in the L2 Protocol is likely to be ignored by the MPC, potentially resulting in a permanent loss.

Possible Payment Failures Due to Deliberate User Rejection

Severity: Informational	Status: Acknowledged
Target: EVM & Solana Smart Contract	Category: Logic Error

A vulnerability exists where users can deliberately cause ETH payment failures during PMM transactions. For instance, a user may implement a fallback function that always reverts, as illustrated below:

```
contract NoReceiver {
    fallback() external payable {
        revert("No receiver");
    }
}
```

When interacting with such contracts, the PMM's transaction fails due to the deliberate rejection of ETH.

Additionally, blockchain network congestion may also prevent the payment from being completed as expected.

The off-chain components observe that the PMM has failed to make the payment, which can result in the PMM being unjustly penalized (slashed), even though the user induced the failure. Furthermore, since users can reclaim their deposit after a trade times out, they can exploit this vulnerability to force a PMM slashing at almost no cost.

This could also lead to the protocol using the security deposit to `compensate` the user.

For user-initiated payment failures, enhance the off-chain handling of such cases and improve the PMM slashing mechanism.

Mitigation Review Log:

Optimex Team: PMMs will not be penalized if the error is not caused by them. We have a logging system in place to track trades within our protocol. Any failures, missed payments, or late payments will be thoroughly reviewed before making a decision. As of now, the reviews are conducted manually to ensure fairness for all participants. However, we are continuously evaluating ways to enhance automation while maintaining the integrity and reliability of the process.

Possible Payment Failures Due to Trade Timeout

Severity: Informational	Status: Acknowledged
Target: EVM & Solana Smart Contract	Category: Logic Error

A user can deliberately set a short `timeout` value in deposit.

After the PMM executes the payment but before the `settlement` is fully completed, the trade may expire. The expiration may be due to the timeout occurring shortly after the payment is completed or due to blockchain network congestion at the time, preventing the transaction from being submitted and confirmed. At that moment, the user can invoke the permissionless `claim` to retrieve the deposited funds, potentially disrupting the intended funds flow and compromising the protocol's state.

From the perspective of the on-chain contract, the PMM's payment is not part of the user Trade's state machine. This separation of payment and settlement into non-atomic operations makes this scenario possible.

A malicious user can carefully set up a `TradeInput timeout` to take effect after receiving the PMM's payment. This allows them to claim their deposit when the settlement fails with a timeout error, leading to potential double processing or inconsistency.

The protocol might end up in an invalid state, as funds intended for settlement are prematurely withdrawn.

Here is a Proof of Concept:

Assume the current ETH : BTC exchange rate is 100:1.

A malicious user has 100 ETH and initiates a swap for 1 BTC at time T0.

- **[T0]** The user deposits 100 ETH and sets the Trade Timeout to T1.
- **[T1]** Due to network congestion, the PMM completes the payment at T1, and the user receives 1 BTC.
- **[T1 + 1]** The MPC Node attempts to settle but fails with a timeout error.
- **[T1 + 2]** The user claims back the 100 ETH.

Now, the user has a total of 100 ETH and 1 BTC. Ignoring network costs, the malicious user profits by 1 BTC from this operation.

Since the detailed design and implementation of the off-chain component are not within the audit scope, only potential solutions can be suggested here based on the documentation.

- Do not allow users to specify the timeout value during deposit, or set a minimum safe timeout to ensure that the PMM can complete the payment before the minimum timeout is reached, leaving a sufficient time window for MPC Nodes to complete the settlement.
- Further optimizing the permissionless design of the claim by allowing the settlement committee to return user assets.

Mitigation Review Log:

Optimex Team: The contract does not impose restrictions in such cases. Instead, the MPC is responsible for validation. Once a trade is submitted to our protocol, the MPC will verify it and will not confirm the deposit if issues arise. As a result, the PMM will not be affected. If the trade fails to complete within the timeout period, the user can simply claim a refund. Currently, a minimum of 24 hours is required. The MPC strictly validates data submitted to the Vault.

5 Disclaimer

This report reflects the security status of the project as of the date of the audit. It is intended solely for informational purposes and should not be used as investment advice. Despite carrying out a comprehensive review and analysis of the relevant smart contracts, it is important to note that Offside Labs' services do not encompass an exhaustive security assessment. The primary objective of the audit is to identify potential security vulnerabilities to the best of the team's ability; however, this audit does not guarantee that the project is entirely immune to future risks.

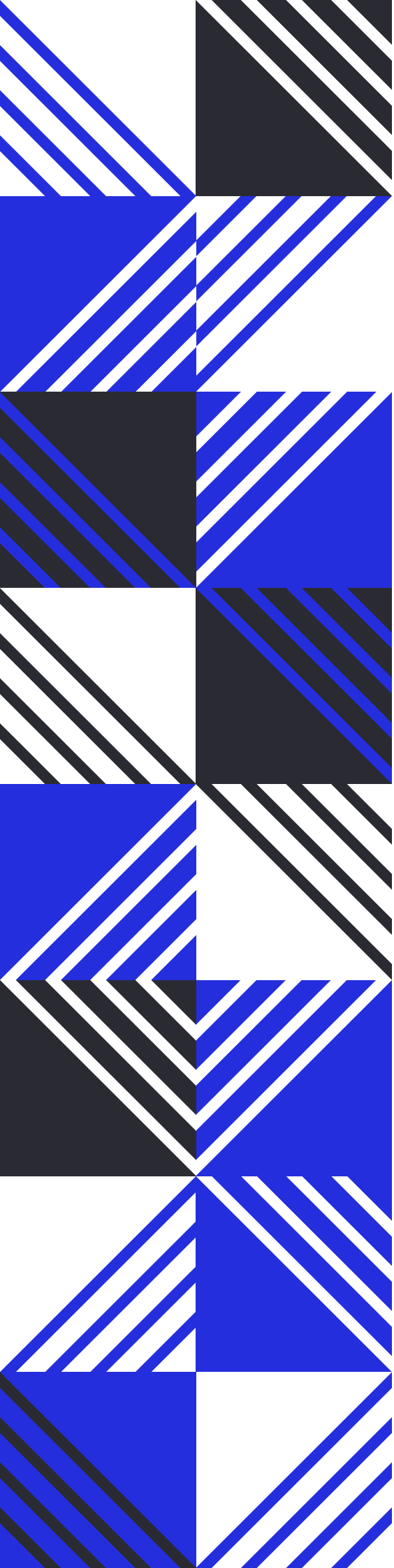
Offside Labs disclaims any liability for losses or damages resulting from the use of this report or from any future security breaches. The team strongly recommends that clients undertake multiple independent audits and implement a public bug bounty program to enhance the security of their smart contracts.

The audit is limited to the specific areas defined in Offside Labs' engagement and does not cover all potential risks or vulnerabilities. Security is an ongoing process, regular audits and monitoring are advised.

Please note: Offside Labs is not responsible for security issues stemming from developer errors or misconfigurations during contract deployment and does not assume liability for centralized governance risks within the project. The team is not accountable for any impact on the project's security or availability due to significant damage to the underlying blockchain infrastructure.

By utilizing this report, the client acknowledges the inherent limitations of the audit process and agrees that the firm shall not be held liable for any incidents that may occur after the completion of this audit.

This report should be considered null and void in case of any alteration.



OFFSIDE LABS

https://offside.io/

https://github.com/offsidelabs

https://twitter.com/offside_labs