OFFSIDE LABS

SolvBTC Vault

Smart Contract Security Assessment

October 2025

Prepared for:

Solv Protocol

Prepared by:

Offside Labs

Sirius Xie

Gongyu Shi

Contents

1 About Offside Labs

Offside Labs is a leading security research team, composed of top talented hackers from both academia and industry.

We possess a wide range of expertise in modern software systems, including, but not limited to, *browsers*, *operating systems*, *IoT devices*, and *hypervisors*. We are also at the forefront of innovative areas like *cryptocurrencies* and *blockchain technologies*. Among our notable accomplishments are remote jailbreaks of devices such as the **iPhone** and **PlayStation 4**, and addressing critical vulnerabilities in the **Tron Network**.

Our team actively engages with and contributes to the security community. Having won and also co-organized *DEFCON CTF*, the most famous CTF competition in the Web2 era, we also triumphed in the **Paradigm CTF 2023** within the Web3 space. In addition, our efforts in responsibly disclosing numerous vulnerabilities to leading tech companies, such as *Apple*, *Google*, and *Microsoft*, have protected digital assets valued at over **$300 million**.

In the transition towards Web3, Offside Labs has achieved remarkable success. We have earned over **$9 million** in bug bounties, and **three** of our innovative techniques were recognized among the **top 10 blockchain hacking techniques of 2022** by the Web3 security community.

https://offside.io/

https://github.com/offsidelabs

https://twitter.com/offside_labs

2 Executive Summary

Introduction

Offside Labs completed a security audit of *SolvBTC Vault* smart contracts, starting on September 12th, 2025, and concluding on September 20th, 2025.

Project Overview

Audit Scope

The assessment scope contains mainly the smart contracts of the solvbtc program for the *SolvBTC Vault* project.

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

- SolvBTC Vault
  - Codebase: `https://github.com/solv-finance/SolvBTC-Solana-Contract`
  - Commit Hash: a1a322587b7f82e0b95ba3bc7e270b92ab35aa2e

We listed the files we have audited below:

- SolvBTC Vault
  - programs/solvbtc/src/contexts/minter_manager_initialize.rs
  - programs/solvbtc/src/contexts/minter_manager_mint.rs
  - programs/solvbtc/src/contexts/minter_manager_transfer_admin.rs
  - programs/solvbtc/src/contexts/minter_manager_update_minter.rs
  - programs/solvbtc/src/contexts/vault_deposit.rs
  - programs/solvbtc/src/contexts/vault_initialize.rs
  - programs/solvbtc/src/contexts/vault_oracle_update.rs
  - programs/solvbtc/src/contexts/vault_request_withdraw.rs
  - programs/solvbtc/src/contexts/vault_update.rs
  - programs/solvbtc/src/contexts/vault_withdraw.rs
  - programs/solvbtc/src/helpers.rs
  - programs/solvbtc/src/lib.rs
  - programs/solvbtc/src/state/minter_manager.rs
  - programs/solvbtc/src/state/vault.rs
  - programs/solvbtc/src/state/withdraw_request.rs

Findings

The security audit revealed:

- 0 critical issue
- 0 high issue
- 2 medium issues
- 1 low issue

- 10 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	Improper Slippage Validation in vault_deposit IX	Medium	Fixed
02	Possible Unexpected Withdrawals Due to Vault Configuration Changes	Medium	Acknowledged
03	Withdrawal Signatures May Be Reused Across Vaults	Low	Acknowledged
04	Missing Update in add_currency/remove_currency IXs	Informational	Fixed
05	Missing Validation of Fee Parameters in vault_initialize IX	Informational	Fixed
06	Lack of user_withdraw_ta Validation in withdraw IX	Informational	Fixed
07	Zero Amount Leads to Invalid WithdrawRequest	Informational	Fixed
08	Missing Parameter Validation When Adding or Removing currency and minter	Informational	Fixed
09	Use of Hard-Coded Fee Instead of MAX_FEE Constant	Informational	Fixed
10	Lack of Token-2022 Transfer Fee Support Can Lead to Vault Losses	Informational	Acknowledged
11	Missing Memo Support for Token-2022 Transfers	Informational	Acknowledged
12	Missing Verifier Parameter Validation in vault_initialize/set_verifier IXs	Informational	Acknowledged
13	DoS Risk Due to ATA Authority Change	Informational	Acknowledged

4 Key Findings and Recommendations

4.1 Improper Slippage Validation in vault_deposit IX

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

Description

In `VaultDeposit.mint_target_tokens`, the provided `min_amount_out` is used for the slippage check.

```rust
    pub fn mint_target_tokens(&mut self, amount: u64, min_amount_out: u64)
    ↪ -> Result<()> {
        let (mint_amount, fee_amount) =
        ↪ Vault::calculate_fee(self.vault.shares_from_deposit(amount)?,
        ↪ self.vault.deposit_fee)?;

        // Slippage protection
        require_gte!(amount, min_amount_out,
        ↪ SolvError::SlippageExceeded);
```

programs/solvbtc/src/contexts/vault_deposit.rs#L64-L68

However, the check incorrectly uses the variable `amount` instead of the intended `mint_amount`.

Impact

This incorrect variable can make the effective slippage tolerance too large or too small compared to the expected range.

Recommendation

It's recommend to replace `amount` with `mint_amount` in the slippage protection.

Mitigation Review Log

Fixed in the commit 195fefe2afbf69ec1a867e79286a140460705e5e.

4.2 Possible Unexpected Withdrawals Due to Vault Configuration Changes

Severity: Medium	Status: Acknowledged
Target: Smart Contract	Category: Data Validation

Description

User withdrawals are a two-step process: the user must first invoke `vault_withdraw_request` and then `vault_withdraw` to receive tokens.

However, since these two operations are non-atomic, certain situations may lead to outcomes that differ from user expectations.

Impact

Scenario 1

1. User invokes the `vault_withdraw_request` IX.
2. The vault admin increases the withdraw fee via the `vault_set_withdraw_fee` IX.
3. The user receives fewer tokens than originally expected.

In this case, the user pays an unexpected withdraw fee, so the final withdrawn token amount is reduced.

Scenario 2

1. User invokes the `vault_withdraw_request` IX to withdraw Mint A.
2. The vault admin removes Mint A from `Vault.deposit_currencies` via the `vault_remove_currency` IX.
3. The user's `WithdrawRequest` fails on the constraint `vault.deposit_currencies.contains(&mint_withdraw.key())`.

Since the vault admin removed Mint A from the vault, the pending `WithdrawRequest` can no longer be completed.

Recommendation

The vault admin must ensure that, when modifying these status values, there are no pending WithdrawRequest.

4.3 Withdrawal Signatures May Be Reused Across Vaults

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

Description

In the `withdraw` IX, the verified message hash is `hash(user, withdraw_token, request_hash, shares, nav)`, which does not include the `vault`.

Impact

Consider a scenario where two Vaults use the same verifier public key, and the user creates identical requests in both Vaults (same `user` / `withdraw_token` / `request_hash` / `shares` / `nav`). The same signature would then pass verification in both Vaults, enabling cross-Vault replay. i.e., in the second Vault, it would allow a permissionless withdrawal.

Recommendation

It's recommend to include the `vault` (and `vault.mint`) in `WithdrawRequest::hash()` to better namespace signatures per Vault and effectively prevent replay.

4.4 Informational and Undetermined Issues

Missing Update in add_currency/remove_currency IXs

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

In the program, most operations that modify `Vault` fields call `Vault.update()` at the end. However, `Vault.add_currency` and `Vault.remove_currency` do not. For consistency, `Vault.update()` should also be called at the end of these operations.

Fixed in the commit 195fefe2afbf69ec1a867e79286a140460705e5e.

Missing Validation of Fee Parameters in vault_initialize IX

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

In `vault_set_deposit_fee` and `vault_set_withdraw_fee`, the new fee parameter is validated to be ≤ `MAX_FEE`. However, in `vault_initialize`, no equivalent check is performed on the input `deposit_fee` / `withdraw_fee`, allowing values greater than `MAX_FEE`. Such oversized fees can cause user deposits and withdrawals to fail and may also lead to NAV being updated beyond expected values.

It's recommend to enforce the `MAX_FEE` limit on `deposit_fee` and `withdraw_fee` during vault initialization as well.

Fixed in the commit 195fefe2afbf69ec1a867e79286a140460705e5e.

Lack of user_withdraw_ta Validation in withdraw IX

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

In the `vault_withdraw_request` IX, when creating the `WithdrawRequest` , `WithdrawRequest.withdraw_token_account` is explicitly specified. However, in the `vault_withdraw` IX, the provided `VaultWithdraw.user_withdraw_ta` is not verified to equal `withdraw_request.withdraw_token_account` .

It is recommended to add a check in the `vault_withdraw` IX that `VaultWithdraw.user_withdraw_ta == withdraw_request.withdraw_token_account` .

Fixed in the commit 195fefe2afbf69ec1a867e79286a140460705e5e.

Zero Amount Leads to Invalid WithdrawRequest

Severity: Informational	Status: Fixed
Target: Smart Contract	Category: Logic Error

In the `vault_withdraw_request` IX, there is no requirement that the `amount` parameter be greater than 0. If `amount` is 0, a `WithdrawRequest` can still be created, but it may not trigger the final `Withdraw` because `withdraw_amount` is 0, causing the rent paid for this account to be non-refundable. It is recommended to require `amount > 0` in the `vault_withdraw_request` IX to avoid creating such abnormal `WithdrawRequest` s.

Fixed in the commit 195fefe2afbf69ec1a867e79286a140460705e5e.

Missing Parameter Validation When Adding or Removing currency and minter

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

When adding or removing a `currency` in the vault, or a `minter` in the mint manager, there is no check to ensure that the provided parameter is not the default public key `Pubkey::default()` . As a result, the corresponding IX may execute successfully and call `self.update` , even though it has no actual effect, and this can be confusing.

Fixed in the commit 195fefe2afbf69ec1a867e79286a140460705e5e and 211aaf6bd1e4a43ec547f6510d5058a9a02dec40.

Use of Hard-Coded Fee Instead of MAX_FEE Constant

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

Some functions apply a hard-coded fee value instead of referencing the `MAX_FEE` constant.

```rust
    let fee: u64 = u128::from(amount)
        .checked_mul(fee as u128)
        .ok_or(ProgramError::ArithmeticOverflow)?
        .checked_div(10000)
```

programs/solvbtc/src/state/vault.rs#L154-L157

```rust
    let max_nav: u64 = u64::try_from(u128::from(self.nav)
        .checked_mul(self.withdraw_fee as u128)
        .ok_or(ProgramError::ArithmeticOverflow)?
        / 10_000)
```

programs/solvbtc/src/state/vault.rs#L138-L141

Fixed in the commit 195fefe2afbf69ec1a867e79286a140460705e5e.

Lack of Token-2022 Transfer Fee Support Can Lead to Vault Losses

Severity: Informational	Status: Acknowledged
Target: Smart Contract	Category: Token

The SolvBTC program supports Token-2022 mints as currencies, and a Token-2022 mint can include the TransferFee extension. In `VaultDeposit.deposit_tokens` , the user is only required to transfer an amount equal to `amount` of the mint.

```rust
    pub fn deposit_tokens(&mut self, amount: u64) -> Result<()> {
        ...
        transfer_checked(ctx, amount, self.mint_token.decimals)
    }
```

programs/solvbtc/src/contexts/vault_deposit.rs#L51-L54

The same `amount` is then used when minting the target token for the user.

```rust
    pub fn mint_target_tokens(&mut self, amount: u64, min_amount_out: u64)
    ↪   -> Result<()> {
        let (mint_amount, fee_amount) =
        ↪   Vault::calculate_fee(self.vault.shares_from_deposit(amount)?,
        ↪   self.vault.deposit_fee)?;
```

programs/solvbtc/src/contexts/vault_deposit.rs#L64-L65

However, this calculation does not account for the Token-2022 transfer fee.

Ignoring the transfer fee can cause the program to record it received more tokens than it actually did, resulting in over-minting of the target token to the user. Later, when the user withdraws, the vault may transfer out more tokens than it actually received.

When calling `mint_target_tokens` , use the post-fee (net) amount—i.e., `amount` minus the transfer fee—as the number of shares, so that the accounting is more accurate.

Missing Memo Support for Token-2022 Transfers

Severity: Informational	Status: Acknowledged
Target: Smart Contract	Category: Token

`Vault.mint` and `Vault.deposit_currencies` might be a Token-2022 Mint. In the `withdraw` IX, the program transfers tokens from vault to the user.

In Token-2022, users can choose to enable the `Memo` extension on their token accounts, which requires a separate Memo instruction to be included before any transfer. And whether the user enables Memo is independent of the Mint. However, current implementation does not handle logic related to the Memo extension. It is recommended to add support for Memo to prevent transfer failures caused by this requirement.

Missing Verifier Parameter Validation in vault_initialize/set_verifier IXs

Severity: Informational	Status: Acknowledged
Target: Smart Contract	Category: Data Validation

In `vault_initialize` and `vault_set_verifier`, when setting `Vault.verifier`, there is no check that the public key is on-curve. If a non-curve point is supplied, signature verification during `withdraw` will fail. When setting a new verifier, it should be validated it on-chain.

DoS Risk Due to ATA Authority Change

Severity: Informational	Status: Acknowledged
Target: Smart Contract	Category: Logic Error

In the current design, deposited tokens are transferred to the treasurer's ATA. During withdrawals, fees are transferred directly to the fee receiver's ATA. The account constraints enforce that these ATAs are owned by the expected treasurer and fee receiver respectively.

```rust
    #[account(
        mut,
        associated_token::authority = vault.treasurer,
        associated_token::mint = mint_token
    )]
    pub treasurer_token_ta: Box<InterfaceAccount<'info, TokenAccount>>,
```

programs/solvbtc/src/contexts/vault_deposit.rs#L27-L32

```rust
    #[account(
        mut,
        associated_token::authority = vault.fee_receiver,
        associated_token::mint = mint_withdraw
    )]
    pub fee_receiver_ta: Box<InterfaceAccount<'info, TokenAccount>>,
```

programs/solvbtc/src/contexts/vault_withdraw.rs#L51-L56

However, for ATAs of token program, it is possible to change its authority. If this is changed, the constraint checks will fail and leads to a DoS.

5 Disclaimer

This audit report is provided for informational purposes only and is not intended to be used as investment advice. While we strive to thoroughly review and analyze the smart contracts in question, we must clarify that our services do not encompass an exhaustive security examination. Our audit aims to identify potential security vulnerabilities to the best of our ability, but it does not serve as a guarantee that the smart contracts are completely free from security risks.

We expressly disclaim any liability for any losses or damages arising from the use of this report or from any security breaches that may occur in the future. We also recommend that our clients engage in multiple independent audits and establish a public bug bounty program as additional measures to bolster the security of their smart contracts.

It is important to note that the scope of our audit is limited to the areas outlined within our engagement and does not include every possible risk or vulnerability. Continuous security practices, including regular audits and monitoring, are essential for maintaining the security of smart contracts over time.

Please note: we are not liable for any security issues stemming from developer errors or misconfigurations at the time of contract deployment; we do not assume responsibility for any centralized governance risks within the project; we are not accountable for any impact on the project's security or availability due to significant damage to the underlying blockchain infrastructure.

By using this report, the client acknowledges the inherent limitations of the audit process and agrees that our firm shall not be held liable for any incidents that may occur subsequent to our engagement.

This report is considered null and void if the report (or any portion thereof) is altered in any manner.

OFFSIDE LABS

https://offside.io/

https://github.com/offsidelabs

https://twitter.com/offside_labs