BLOCKSEC

# Security Audit

# Report for bgw-swap-aggregator-solana

**Date:** May 18, 2026 **Version:** 1.0

**Contact:** contact@blocksec.com

# Contents

## Report Manifest

| Item | Description |
| --- | --- |
| Client | bitgetwallet |
| Target | bgw-swap-aggregator-solana |

## Version History

| Version | Date | Description |
| --- | --- | --- |
| 1.0 | May 18, 2026 | First release |

## Signature

Digitally signed by: BlockSec (https://blocksec.com)

SHA1 digest of public key: BAB106D3708BC6DE0963CE05E9478C24B326AE1C

Date: 2026-05-18 11:26:04+0000

**About BlockSec** BlockSec focuses on the security of the blockchain ecosystem and collaborates with leading DeFi projects to secure their products. BlockSec is founded by top-notch security researchers and experienced experts from both academia and industry. They have published multiple blockchain security papers in prestigious conferences, reported several zero-day attacks of DeFi applications, and successfully protected digital assets that are worth more than 14 million dollars by blocking multiple attacks. They can be reached at Email, Twitter and Medium.

# Chapter 1 Introduction

## 1.1 About the Audit Target

| Information | Description |
|---|---|
| Type | Smart Contract |
| Language | Rust |
| Approach | Semi-automatic and manual verification |

The audit target (hereinafter referred to as the Target) of this audit is the code repository [1] of bgw-swap-aggregator-solana of bitgetwallet.

This update introduces partner-integrated swap entrypoints for the Solana swap aggregator, enabling protocol fees to be split between the platform and a designated integration partner. It adds partner fee parameters, partner-specific fee collection for both SPL tokens and native SOL, and new swap wrappers across v1, v2, and v3 routes. The update also reorganizes swap modules into versioned directories, exposes the new instructions in the program interface, adds fee split logging, and includes a custom heap allocator for Solana builds.

Note this audit only focuses on the smart contracts in the following directories/files. Code prior to and including the baseline version 0, where applicable, is outside the scope of this audit and assumes to be reliable and secure.

- programs/bgwaggregator/src/lib.rs
- programs/bgwaggregator/src/common/errors.rs
- programs/bgwaggregator/src/common/logs.rs
- programs/bgwaggregator/src/fee/collect_fee.rs
- programs/bgwaggregator/src/swap/swap_core.rs
- programs/bgwaggregator/src/swap/v1/swap_sol_spl_to_b.rs
- programs/bgwaggregator/src/swap/v1/swap_spl_spl_to_b.rs
- programs/bgwaggregator/src/swap/v1/swap_spl_white_to_b.rs
- programs/bgwaggregator/src/swap/v2/swap_spl_sol_v2_to_b.rs
- programs/bgwaggregator/src/swap/v3/swap_sol_spl_v3_to_b.rs
- programs/bgwaggregator/src/swap/v3/swap_spl_sol_v3_to_b.rs
- programs/bgwaggregator/src/swap/v3/swap_spl_spl_v3_to_b.rs
- programs/bgwaggregator/src/swap/v3/swap_spl_white_v3_to_b.rs

Other files are not within the scope of the audit. Additionally, all dependencies of the Target are considered reliable in terms of both functionality and security, and are therefore not included in the audit scope.

The auditing process is iterative. Specifically, we would audit the commits that fix the discovered issues. If there are new issues, we will continue this process. The commit SHA values during the audit are shown in the following table. Our audit report is responsible for the code in the initial version (Version 1), as well as new code (in the following versions) to fix

[1] https://github.com/bitgetwallet/bgw-swap-aggregator-solana

issues in the audit report. Code prior to and including the baseline version (Version 0), where applicable, is outside the scope of this audit and assumes to be reliable and secure.

| Project | Version | Commit Hash |
|---|---|---|
| bgw-swap-aggregator-solana | Version 0 | d2d1a6d6e5399a601d8375ebc5f46f8acdc333d6 |
| | Version 1 | 23f6dcc72683305fa81c1f4f585ead8176b66ea6 |

## 1.2 Disclaimer

This audit report does not constitute investment advice or a personal recommendation. It does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Any entity should not rely on this report in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset.

This audit report is not an endorsement of any particular project or team, and the report does not guarantee the security of any particular project. This audit does not give any warranties on discovering all security issues of the Target, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues. As one audit cannot be considered comprehensive, we always recommend proceeding with independent audits and a public bug bounty program to ensure the security of the Target.

The scope of this audit is limited to the code mentioned in Section 1.1. Unless explicitly specified, the security of the language itself (e.g., the solidity language), the underlying compiling toolchain and the computing infrastructure are out of the scope.

## 1.3 Procedure of Auditing

We perform the audit according to the following procedure.

- **Vulnerability Detection** We first scan the Target with automatic code analyzers, and then manually verify (reject or confirm) the issues reported by them.
- **Semantic Analysis** We study the business logic of the Target and conduct further investigation on the possible vulnerabilities using an automatic fuzzing tool (developed by our research team). We also manually analyze possible attack scenarios with independent auditors to cross-check the result.
- **Recommendation** We provide some useful advice to developers from the perspective of good programming practice, including gas optimization, code style, and etc.

We show the main concrete checkpoints in the following.

### 1.3.1 Security Issues

* Access control
* Permission management
* Whitelist and blacklist mechanisms
* Initialization consistency

- Improper use of the proxy system
- Reentrancy
- Denial of Service (DoS)
- Untrusted external call and control flow
- Exception handling
- Data handling and flow
- Events operation
- Error-prone randomness
- Oracle security
- Business logic correctness
- Semantic and functional consistency
- Emergency mechanism
- Economic and incentive impact

### 1.3.2 Additional Recommendation

- Gas optimization
- Code quality and style

**Note** *The previous checkpoints are the main ones. We may use more checkpoints during the auditing process according to the functionality of the project.*

## 1.4 Security Model

To evaluate the risk, we follow the standards or suggestions that are widely adopted by both industry and academy, including OWASP Risk Rating Methodology [2] and Common Weakness Enumeration [3]. The overall *severity* of the risk is determined by *likelihood* and *impact*. Specifically, likelihood is used to estimate how likely a particular vulnerability can be uncovered and exploited by an attacker, while impact is used to measure the consequences of a successful exploit.

In this report, both likelihood and impact are categorized into two ratings, i.e., *high* and *low* respectively, and their combinations are shown in Table 1.1.

Accordingly, the severity measured in this report are classified into three categories: **High**, **Medium**, **Low**. For the sake of completeness, **Undetermined** is also used to cover circumstances when the risk cannot be well determined.

Furthermore, the status of a discovered item will fall into one of the following five categories:

- **Undetermined** No response yet.
- **Acknowledged** The item has been received by the client, but not confirmed yet.
- **Confirmed** The item has been recognized by the client, but not fixed yet.
- **Partially Fixed** The item has been confirmed and partially fixed by the client.
- **Fixed** The item has been confirmed and fixed by the client.

[2] https://owasp.org/www-community/OWASP_Risk_Rating_Methodology

[3] https://cwe.mitre.org/

**Table 1.1:** Vulnerability Severity Classification

| Impact | | Likelihood: *High* | Likelihood: *Low* |
|---|---|---|---|
| | *High* | High | Medium |
| | *Low* | Medium | Low |

# Chapter 2 Findings

In total, we found **one** potential security issue. Besides, we have **three** recommendations and **one** note.

- Low Risk: 1
- Recommendation: 3
- Note: 1

| ID | Severity | Description | Category | Status |
|---|---|---|---|---|
| 1 | Low | Unrestricted partner fee recipient and partner fee allocation in partner fee-sharing swap flows | Security Issue | Confirmed |
| 2 | - | Ignored WSOL account closure failure | Recommendation | Confirmed |
| 3 | - | Lack of partner fee distribution information in swap events | Recommendation | Confirmed |
| 4 | - | Lack of mutable constraint on `proxy_pda` in SPL to SOL swaps | Recommendation | Confirmed |
| 5 | - | Potential centralization risks | Note | - |

The details are provided in the following sections.

## 2.1 Security Issue

### 2.1.1 Unrestricted partner fee recipient and partner fee allocation in partner fee-sharing swap flows

**Severity** Low

**Status** Confirmed

**Introduced by** `Version 1`

**Description** The `collect_fee_sol_to_b()` and `collect_fee_spl_to_b()` functions support partner fee-sharing swap flows by splitting the collected fee into a protocol portion and a partner portion according to the externally supplied `partner_fee_rate`. The partner fee is then transferred directly to the supplied partner account.

While `fee_to` is validated against `admin_info.fee_to_array`, the protocol does not enforce any on-chain restriction on the supplied partner account or its associated `partner_fee_rate`. In particular, `partner_fee_rate` is allowed to reach `FEE_RATE_BASE`, enabling the partner portion to consume the entire collected fee amount.

As a result, if an integrating backend or payer accepts user-controlled partner parameters when constructing partner fee-sharing swap transactions, the entire fee amount may be redirected to a user-specified partner account instead of being retained by the protocol.

```
218   pub fn collect_fee_sol_to_b<'info>(
219       admin_info: &AdminInfo,
220       fee_params: &FeeParamsToB,
221       amount: u64,
222       sender_or_proxy: &AccountInfo<'info>,
```

```
223         fee_to: &AccountInfo<'info>,
224         partner: &AccountInfo<'info>,
225         system_program: &AccountInfo<'info>,
226         proxy_pda_index: u8,
227         is_pre_collect: bool,
228     ) -> Result<u64> {
229         let (protocol_fee_amount, partner_fee_amount) =
230             pre_check_to_b(admin_info, fee_params, amount, &fee_to.key())?;
231 
232         if is_pre_collect {
233             if protocol_fee_amount > 0 {
234                 transfer_sol_from_user(
235                     sender_or_proxy.to_account_info(),
236                     fee_to.to_account_info(),
237                     system_program.to_account_info(),
238                     protocol_fee_amount,
239                 )?;
240             }
241             if partner_fee_amount > 0 {
242                 transfer_sol_from_user(
243                     sender_or_proxy.to_account_info(),
244                     partner.to_account_info(),
245                     system_program.to_account_info(),
246                     partner_fee_amount,
247                 )?;
248             }
249         } else {
250             if protocol_fee_amount > 0 {
251                 transfer_sol_from_proxy_pda(
252                     sender_or_proxy.to_account_info(),
253                     fee_to.to_account_info(),
254                     system_program.to_account_info(),
255                     protocol_fee_amount,
256                     proxy_pda_index,
257                 )?;
258             }
259             if partner_fee_amount > 0 {
260                 transfer_sol_from_proxy_pda(
261                     sender_or_proxy.to_account_info(),
262                     partner.to_account_info(),
263                     system_program.to_account_info(),
264                     partner_fee_amount,
265                     proxy_pda_index,
266                 )?;
267             }
268         }
269 
270         FeeCoreLog::fee_collected_to_b(protocol_fee_amount, partner_fee_amount);
271         Ok(protocol_fee_amount
272             .checked_add(partner_fee_amount)
273             .ok_or(ErrorCode::FeeAmountOverflow)?)
274     }
```

**Listing 2.1:** programs/bgwaggregator/src/fee/collect_fee.rs

```
126  fn pre_check_to_b(
127      admin_info: &AdminInfo,
128      fee_params: &FeeParamsToB,
129      amount: u64,
130      fee_to: &Pubkey,
131  ) -> Result<(u64, u64)> {
132      // Reuse existing pre_check: validates fee_rate bounds, fee_to whitelist, computes
             fee_amount
133      let fee_amount = pre_check(
134          admin_info,
135          &FeeParams { fee_rate: fee_params.fee_rate, deduct_amount: fee_params.deduct_amount },
136          amount,
137          fee_to,
138      )?;
139
140      require!(
141          fee_params.partner_fee_rate <= FEE_RATE_BASE as u16,
142          ErrorCode::ToBInvalidPartnerFeeRate
143      );
144
145      let partner_fee_amount: u64 = (fee_amount as u128)
146          .checked_mul(fee_params.partner_fee_rate as u128)
147          .ok_or(ErrorCode::ToBCalcErrorPartnerAmount)?
148          .checked_div(FEE_RATE_BASE as u128)
149          .ok_or(ErrorCode::ToBCalcErrorPartnerAmount)?
150          .try_into()
151          .map_err(|_| ErrorCode::ToBPartnerAmountOverflow)?;
152      let protocol_fee_amount = fee_amount
153          .checked_sub(partner_fee_amount)
154          .ok_or(ErrorCode::ToBCalcErrorProtocolAmount)?;
155
156      Ok((protocol_fee_amount, partner_fee_amount))
157  }
```

**Listing 2.2:** programs/bgwaggregator/src/fee/collect_fee.rs

```
10  fn pre_check(
11      admin_info: &AdminInfo,
12      fee_params: &FeeParams,
13      amount: u64,
14      fee_to: &Pubkey,
15  ) -> Result<u64> {
16      require!(
17          admin_info.fee_rate_min <= fee_params.fee_rate
18              && fee_params.fee_rate <= admin_info.fee_rate_max
19              && fee_params.fee_rate > 0,
20          ErrorCode::InvalidFeeRate
21      );
22
23      require!(
```

```
24          admin_info.fee_to_array.contains(fee_to),
25          ErrorCode::InvalidFeeTo
26      );
27
28      let fee_amount_u128: u128 = (amount as u128)
29          .checked_mul(fee_params.fee_rate as u128)
30          .ok_or(ErrorCode::CalcErrorFeeAmount)?
31          .checked_div(FEE_RATE_BASE as u128)
32          .ok_or(ErrorCode::CalcErrorFeeAmount)?;
33
34      let fee_amount: u64 = fee_amount_u128
35          .try_into()
36          .map_err(|_| ErrorCode::FeeAmountOverflow)?;
37
38      require!(fee_amount > 0, ErrorCode::AmountErrorFeeAmount);
39
40      Ok(fee_amount)
41  }
```

**Listing 2.3:** programs/bgwaggregator/src/fee/collect_fee.rs

**Impact** Protocol fee revenue can be unintentionally redirected to externally supplied partner accounts during partner fee-sharing swap flows.

**Suggestion** Consider introducing a protocol-controlled partner registry or enforcing backend-side restrictions on allowed partner accounts and partner fee allocation ratios.

**Feedback from the project** The project considers this a design choice and does not limit who can apply to become a partner or what fee-sharing percentage they may set.

## 2.2 Recommendation

### 2.2.1 Ignored WSOL account closure failure

**Status** Confirmed

**Introduced by** `Version 1`

**Description** When swapping SPL tokens for native SOL, the protocol invokes function `close_token_account_from_proxy_pda()` to close the temporary WSOL token account owned by `proxy_pda`. However, the return value of this function call is ignored. If the WSOL account closure fails, execution continues instead of reverting immediately, and the transaction may only fail later during subsequent SOL transfer or balance handling logic, resulting in unnecessary compute unit consumption.

```
175  let _ = close_token_account_from_proxy_pda(
176      ctx.accounts.proxy_pda_wsol.to_account_info(),
177      ctx.accounts.proxy_pda.to_account_info(),
178      ctx.accounts.proxy_pda.to_account_info(),
179      ctx.accounts.output_token_program.to_account_info(),
180      proxy_pda_index
181  );
```

**Listing 2.4:** programs/bgwaggregator/src/swap/v2/swap_spl_sol_free_v2.rs

**Suggestion** Propagate the error from function `close_token_account_from_proxy_pda()` using "`?`" so the transaction reverts immediately when the WSOL account closure fails.

**Feedback from the project** The project considers that no additional check is required here, as the execution will revert if the WSOL closure fails.

### 2.2.2 Lack of partner fee distribution information in swap events

**Status** Confirmed

**Introduced by** `Version 1`

**Description** The `BWSwapAggregator` event only exposes a single `fee_amount` field. In partner fee-sharing swap flows, this value represents the combined fee amount, including both the `protocol_fee` and the `partner_fee`. Although the protocol emits a `msg!` log containing the split fee information, these details are not included in the emitted `Anchor` event. As a result, off-chain indexers relying on events cannot accurately distinguish partner fee-sharing swaps from regular swaps or reliably attribute fee amounts to the protocol and partner respectively.

```
254   emit_aggregator_event(
255       &ctx.accounts.receiver.key(),
256       swap_params.amount_in,
257       receiver_output_token_balance_change,
258       swap_params.quote_amount,
259       min_amount_out,
260       deduct_amount,
261       0,
262       order_id,
263   );
```

**Listing 2.5:** programs/bgwaggregator/src/swap/v2/swap_spl_sol_free_v2.rs

**Suggestion** Extend the event structure to include fields such as `protocol_fee`, `partner_fee`, and `partner`, or emit a dedicated event for partner fee-sharing swap flows to improve off-chain indexing and fee attribution accuracy.

**Feedback from the project** The project notes that this information is emitted through a log inside the fee-splitting function.

### 2.2.3 Lack of mutable constraint on `proxy_pda` in SPL to SOL swaps

**Status** Confirmed

**Introduced by** `Version 1`

**Description** When a user swaps an SPL token for native SOL, the protocol temporarily receives the output asset as WSOL in the token account owned by `proxy_pda`, then invokes function `close_token_account_from_proxy_pda()` to unwrap WSOL and return the underlying native SOL lamports to `proxy_pda` before forwarding SOL to the receiver. As a result, the lamport balance of `proxy_pda` is modified during execution.

However, in the affected instruction contexts, `proxy_pda` is not marked as mutable. Existing integrations appear to function correctly because their off-chain transaction construction logic already explicitly marks `proxy_pda` as writable when building transactions. However, the introduction of partner fee-sharing integrations may increase the likelihood of programmatic or cross-program swap integrations relying on generated IDLs or generic transaction builders instead of protocol-specific off-chain transaction construction logic. In such cases, `proxy_pda` may be treated as read-only according to the declared account metadata, resulting in integration compatibility issues.

```
34   #[account(
35       seeds = [PROXY_PDA_SEEDS[swap_params.proxy_pda_index as usize]],
36       bump = PROXY_PDA_BUMPS[swap_params.proxy_pda_index as usize],
37   )]
38   pub proxy_pda: UncheckedAccount<'info>,
```

**Listing 2.6:** programs/bgwaggregator/src/swap/v2/swap_spl_sol_v2_to_b.rs

**Suggestion** Mark `proxy_pda` as mutable to ensure it can properly receive and transfer native SOL during execution.

**Feedback from the project** The project states that this will remain unchanged for now and may be updated in a future iteration.

## 2.3 Note

### 2.3.1 Potential centralization risks

**Introduced by** `Version 1`

**Description** In this project, several privileged roles (e.g., `admin_info.authority`) can conduct sensitive operations, which introduces potential centralization risks. For example, `admin_info.authority` can set the array of white lists of `fee_to`. If the private keys of the privileged accounts are lost or maliciously exploited, it could pose a significant risk to the protocol.


BOOST WEB3 THROUGH NEXT-GENERATION SECURITY & USABILITY INNOVATIONS