Sky: LZ Solana Token Bridge
Security Review

Cantina Managed review by:

Christoph Michel, Lead Security Researcher
Mario.eth, Lead Security Researcher

October 28, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings are a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Sky Protocol is a decentralised protocol developed around the USDS stablecoin.

From Sep 22nd to Sep 26th the Cantina team conducted a review of sky-oapp-oft on commit hash c4e4f8ca. The team identified a total of **8** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	1	0	1
Gas Optimizations	0	0	0
Informational	7	6	1
Total	**8**	**6**	**2**

The Cantina Managed team reviewed Sky's `sky-oapp-oft` holistically on commit hash b5c6ef52 and concluded that all findings were addressed and no new vulnerabilities were identified.

3 Findings

3.1 Low Risk

3.1.1 Solana ⇔ EVM rate limiter differences

Severity: Low Risk

Context: peer_config.rs#L30

Description: The Solana rate limiter is different from the EVM one in several ways:

1. When setting a new `capacity` (`limit` in EVM), Solana resets the available capacity to the new capacity, performing a full refill. On EVM, when calling `setRateLimits` the rate limits are not fully reset, they are just refreshed using the previous decay rate up to the current timestamp.
2. On Solana, the `rate_limiter_type` ("GROSS" or "NET") can be set individually per `(remote_eid, inbound/outbound)`, whereas on EVM it applies to both inbound and outbound per `remote_eid`. It's unclear what the purpose of setting one of inbound/outbound to "NET" but leaving the other to "GROSS" is.
3. On Solana, the decay is measured in `refill_per_second`, on EVM, it's implicitly measured via `limit / window`.

It is expected that the rate limit configuration on the outbound source and inbound destination is the same. Slightly different behavior can lead to unexpected temporary disruptions / reverts as the available capacity across chains gets out of sync more easily.

Recommendation: Decide on what the intended behavior should be and consider unifying how the rate limiters work across the chains. For 3), consider choosing a `limit` and `window` on EVM such that `limit % window == 0`, so you can model the same refill per second behavior on both chains, without encountering rounding errors.

Sky: Acknowledged. We will fix this in the future.

Cantina Managed: Acknowledged.

3.2 Informational

3.2.1 Solana ⇔ EVM `quoteOFT` `OFTLimits` discrepancy

Severity: Informational

Context: quote_oft.rs#L37

Description: The EVM code now uses the rate limits to set the `quoteOFT`'s `OFTLimit`:

```
uint256 minAmountLD = 0;
(/*uint256 currentAmountInFlight*/, uint256 maxAmountLD) = getAmountCanBeSent(_sendParam.dstEid);
oftLimit = OFTLimit(minAmountLD, maxAmountLD);
```

Currently, Solana does not use the rate limit to cap the max amount in its `quote_oft` instruction:

```
let oft_limits = OFTLimits { min_amount_ld: 0, max_amount_ld: 0xffffffffffffffff };
```

Recommendation: Consider implementing similar behavior on Solana for `quote_oft`'s `OFTLimits`.

Sky: Fixed in commit bbffa1d0.

Cantina Managed: Fix verified.

3.2.2 `RateLimiter` variable naming could be more precise

Severity: Informational

Context: peer_config.rs#L27

Description: Solana's rate limiter consists of the following `struct`:

```
pub struct RateLimiter {
    pub capacity: u64,
    pub tokens: u64,
    pub refill_per_second: u64,
    pub last_refill_time: u64,
    pub rate_limiter_type: RateLimiterType,
}
```

Recommendation: `RateLimiter.tokens` is a very generic term, here, it refers to what EVM's rate limiter calls "available capacity" (`limit - inflight`), meaning how many tokens can still be bridged. Consider renaming `tokens` to `available_capacity`, and `refill(&mut self, extra_tokens: u64)` to `refill(&mut self, extra_available_capacity: u64)`.

Sky: Fixed in commits a7a90e01 and 5ad5cb6b.

Cantina Managed: Fix verified.

3.2.3 `compose_msg` function name could be more precise

Severity: Informational

Context: msg_codec.rs#L42

Description: When sending a message with a `compose_msg` for `send_compose`, the message is extended with the `[sender_32_bytes, compose_msg]`. Sometimes, the code refers to this entire extension as the `compose_msg`, sometimes only to the actual inner `compose_msg` which is confusing.

The `msg_codec::compose_msg(msg)` returns the full extension of `sender` and `compose_msg`. The `compose_msg_codec::compose_msg(msg)` returns the *inner* actual `compose_msg`, `compose_msg_codec::compose_from(msg)` returns the sender.

Recommendation: Consider renaming `msg_codec::compose_msg` to `msg_codec::compose_msg_with_sender` or similar to reflect that it returns the full extended message prepended by the sender.

Sky: Fixed in commit 29bba09f.

Cantina Managed: Fix verified.

3.2.4 `oft_fee` does not match actual received fees for mint-and-burn-type adapter and fee-on-transfer tokens

Severity: Informational

Context: send.rs#L107-L121, quote_oft.rs#L49

Description: For mint-and-burn type adapters, the fee calculation ignores any transfer fees and only computes the OFT bridging fee `oft_fee`. However, this `oft_fee` is transferred and might therefore be susceptible to a transfer-fee if the token2022 contract has this extension.

The actual received `oft_fee` might be less than the calculated `oft_fee`. This is currently not an issue as the reduced fee can still be withdrawn and there is no explicit fee accounting.

A related issue is that `quote_oft` returns the calculated `oft_fee`.

Recommendation: Consider clarifying in the code that the actual received bridging fee might be less than the computed `oft_fee` for the mint-and-burn-type adapter case if the token2022 has a fee on transfer. Also clarify this in `quote_oft`'s `FeeDetails`.

Sky: Fixed in PR 33.

Cantina Managed: Fix verified.

3.2.5 `compute_fee_and_adjust_amount` fee computations not aligned with EVM and can be optimized

Severity: Informational

Context: quote_send.rs#L73-L105

Description:

> Note that the Solana OFT adapter supports fee-on-transfer tokens (token2022 with the `TransferFeeConfig` extension) whereas EVM does not support them. Therefore, Solana performs additional fee-on-transfer fee computations compared to EVM.
> "WARNING: The default OFTAdapter implementation assumes LOSSLESS transfers, ie. 1 token in, 1 token out." - OFTAdapterDoubleSidedRLFee.sol

The `compute_fee_and_adjust_amount` computes the `(amount_sent_ld, amount_received_ld, oft_fee_ld)` differently based on the type of the OFT adapter:

- Adapter (escrow): Recalculates `amount_sent_ld` given the transfer fee. Removes dust twice. Removes dust from `oft_fee`.
- Native (mint-and-burn): Removes dust twice. Removes dust from `oft_fee`. Ignores transfer fees as they don't apply to mint/burn.

EVM computes the fees in a simplified manner by removing dust once after the fees are applied:

```
// pseudo code
amountSentLD = _amountLD;
uint256 fee = getFee(_dstEid, _amountLD); // _amountLD * feeBps / 1e4
amountReceivedLD = _removeDust(_amountLD - fee);
oft_fee = amountSentLD - amountReceivedLD;
```

Recommendation: Given a transfer fee of zero, the EVM and Solana OFT fee computations should align. Consider an alternative fee computation that could look similar to the following:

```
// pseudo code, untested
// native (mint-and-burn)
let amount_sent_ld = amount_ld;
let oft_fee_ld = calculate_fee(
    amount_sent_ld,
    oft_store.default_fee_bps,
    fee_bps,
);
let amount_received_ld = remove_dust(amount_sent_ld - oft_fee_ld);
let oft_fee_ld = amount_sent_ld - amount_received_ld; // oft_fee + dust

// adapter (escrow)
let amount_sent_ld = amount_ld; // simpler, just what the user specified
let amount_receive_ld_post_transfer = get_post_fee_amount_ld(amount_sent_ld); // apply token2022 transaction
↪  fees
let oft_fee = calculate_fee(amount_receive_ld_post_transfer); // take oapp fee on actually received amount
let amount_received_ld = remove_dust(amount_receive_ld_post_transfer - oft_fee); // finally remove dust once
↪  after both fees are subtracted
let oft_fee = amount_receive_ld_post_transfer - amount_received_ld; // readjust `oft_fee` s.t. oft_fee + dust
```

Sky: Acknowledged, there will be no fix PR for Sky because the effort to make everything aligned is too big to do it as part of Sky project and requires cross team, cross VM collaboration internally in LayerZero.

Cantina Managed: Acknowledged.

3.2.6 Minor improvements for a more idiomatic Rust

Severity: Informational

Context: *(See each case below)*

Description: We've identified minor improvements that would make the code more idiomatic in Rust:

- In init_oft.rs#L51:

```
ctx.accounts.oft_store.endpoint_program =
    if let Some(endpoint_program) = params.endpoint_program {
        endpoint_program
    } else {
        ENDPOINT_ID
    };
```

  can be rewritten as:

```
ctx.accounts.oft_store.endpoint_program = params.endpoint_program.unwrap_or(ENDPOINT_ID);
```

- In quote_send.rs#L108:

```
let final_fee_bps = if let Some(bps) = fee_bps { bps as u128 } else { default_fee_bps as u128 };
```

  can be rewritten as:

```
let final_fee_bps = fee_bps.unwrap_or(default_fee_bps) as u128;
```

Recommendation: Consider updating the code to follow more idiomatic Rust patterns.

Sky: Fixed in commit 3a288929.

Cantina Managed: Fix verified.

3.2.7 Incorrect comments in `lz_receive_types.rs`

Severity: Informational

Context: lz_receive_types.rs#L36-L37, lz_receive_types.rs#L101, lz_receive_types.rs#L111

Description: The comments describing how the accounts for `accounts_for_clear` and `accounts_for_compose` are structured within the accounts array returned by this function are incorrect. The number of accounts returned by `get_accounts_for_send_compose` and `get_accounts_for_clear` is incorrectly specified in the comments:

- On L101, the comment `// remaining accounts 0..9` is incorrect, as `get_accounts_for_clear` returns 8 accounts:

```
vec![
    LzAccount { pubkey: endpoint_program, is_signer: false, is_writable: false },
    LzAccount { pubkey: *receiver, is_signer: false, is_writable: false },
    LzAccount { pubkey: oapp_registry_account, is_signer: false, is_writable: false },
    LzAccount { pubkey: nonce_account, is_signer: false, is_writable: true },
    LzAccount { pubkey: payload_hash_account, is_signer: false, is_writable: true },
    LzAccount { pubkey: endpoint_settings_account, is_signer: false, is_writable: true },
    LzAccount { pubkey: event_authority_account, is_signer: false, is_writable: false },
    LzAccount { pubkey: endpoint_program, is_signer: false, is_writable: false },
]
```

  So the comment should be updated to: `// remaining accounts 0..7`.

- On L111, the comment `// remaining accounts 9..16` is incorrect, as `get_accounts_for_send_compose` returns 7 accounts:

```
vec![
    LzAccount { pubkey: endpoint_program, is_signer: false, is_writable: false },
    LzAccount { pubkey: *from, is_signer: false, is_writable: false },
    LzAccount { pubkey: Pubkey::default(), is_signer: true, is_writable: true },
    LzAccount { pubkey: composed_message_account, is_signer: false, is_writable: true },
    LzAccount { pubkey: SYSTEM_ID, is_signer: false, is_writable: false },
    LzAccount { pubkey: event_authority_account, is_signer: false, is_writable: false },
    LzAccount { pubkey: endpoint_program, is_signer: false, is_writable: false },
]
```

  So the comment should be updated to: `// remaining accounts 8..14`.

Recommendation: Consider updating the comments to reflect the correct positions of the accounts within the returned vector.

Sky: Fixed in PR 24.

Cantina Managed: Fix verified.