SALUS SECURITY

DEC 2025

CODE SECURITY ASSESSMENT

GATE

Overview

Project Summary

- Name: GateChain - Solana vault
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/gatechain/perps/tree/main/contract/solana-vault
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	GateChain - Solana vault
Version	v2
Type	Solidity
Dates	Dec 09 2025
Logs	Dec 04 2025; Dec 09 2025

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	4
Total informational issues	4
Total	8

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	The `rate_limiter` is implemented but not applied	Low	Business Logic	Acknowledged
2	The program initialization may be front run	Low	Business Logic	Acknowledged
3	The `oapp_lz_receive_types` may return incorrect accounts	Low	Business Logic	Acknowledged
4	The `oapp_quote` instruction cannot return the `lz_token-fee`	Low	Business Logic	Acknowledged
5	Unused stored bump wastes compute units	Informational	Gas Optimization	Acknowledged
6	Unused `event_cpi` attribute	Informational	Gas Optimization	Acknowledged
7	Asymmetric packaging method for outbound and inbound messages	Informational	Business Logic	Acknowledged
8	Implement two step ownership transfer	Informational	Business Logic	Acknowledged

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. The `rate_limiter` is implemented but not applied

Severity: Low	Category: Business Logic
Target: - src/instructions/oapp_instr/set_rate_limit.rs	

Description

For `Peer` accounts, the protocol implements rate limiting for inbound messages from the source chain. The `oapp_config` admin can set rate limits for `Peer` accounts using the `set_rate_limit(...)` instruction.

src/instructions/oapp_instr/set_rate_limit.rs:L25-L41

```
pub fn apply(ctx: &mut Context<SetRateLimit>, params: &SetRateLimitParams) -> Result<()>
{
    if !params.enabled {
        ctx.accounts.peer.rate_limiter = None;
        return Ok(());
    }

    let mut rate_limiter = ctx.accounts.peer.rate_limiter.clone().unwrap_or_default();

    if let Some(capacity) = params.capacity {
        rate_limiter.set_capacity(capacity)?;
    }
    if let Some(refill_rate) = params.refill_per_second {
        rate_limiter.set_rate(refill_rate)?;
    }
    ctx.accounts.peer.rate_limiter = Some(rate_limiter);
    Ok(())
}
```

However, rate limiting is not used anywhere, and when `oapp_lz_receive(...)` receives inbound messages to withdraw from the vault, it does not consume tokens from the rate limit.

Recommendation

Call the `try_consume` function in the `oapp_lz_receive(...)` instruction to apply rate limiting.

Status

This issue has been acknowledged by the team. The team stated that these functions will not be used.

2. The program initialization may be front run

Severity: Low	Category: Business Logic

Target:
- src/instructions/oapp_instr/init_oapp.rs

Description

The `init_oapp(...)` instruction is used to initialize the program. It initializes the `oapp_config`, `lz_receive_types`, and `account_list` accounts, and registers the oApp with the LayerZero program.

src/instructions/oapp_instr/init_oapp.rs:L6-L10

```
// Initialize the oapp_config and vault_owner pda
#[derive(Accounts)]
#[instruction(params: InitOAppParams)]
pub struct InitOApp<'info> {
    #[account(mut)]
    pub payer: Signer<'info>,
    //...
}
```

However, this instruction lacks access control, allowing anyone to front-run the program initialization, which can render the program unusable.

Recommendation

Implement access control for the initialization instruction, allowing only a hardcoded address or the program's upgrade authority address to call it.

Status

This issue has been acknowledged by the team. The team stated that the program has already been initialized.

3. The `oapp_lz_receive_types` may return incorrect accounts

Severity: Low	Category: Business Logic

Target:
- src/instructions/oapp_instr/oapp_lz_receive_types.rs

Description

The `oapp_lz_receive_types(...)` instruction is used to return the list of accounts required to execute `oapp_lz_receive(...)`.

src/instructions/oapp_instr/oapp_lz_receive_types.rs:L103-L109

```
    // account 5
    let token_mint;
    if token_pda == ctx.accounts.account_list.usdc_pda {
        token_mint = ctx.accounts.account_list.usdc_mint;
    } else {
        token_mint = Pubkey::from_str("0x0000").unwrap();
    }
```

However, when obtaining `token_mint`, if the `token_pda` is not the PDA stored in the `account_list`, then `token_mint` will be set to the all-zero public key. This causes `token_mint` to return an incorrect value when a cross-chain message withdraws a non-USDC token, and accounts derived from `token_mint` will also return incorrect values. An incorrect account list returned by `oapp_lz_receive_types(...)` can cause the `executor`'s call to `oapp_lz_receive(...)` to fail to execute the message.

Recommendation

Implement access control for the initialization instruction, allowing only a hardcoded address or the program's upgrade authority address to call it.

Status

This issue has been acknowledged by the team.

4. The `oapp_quote` instruction cannot return the `lz_token-fee`

Severity: Low	Category: Business Logic
Target: - src/instructions/oapp_instr/oapp_quote.rs	

Description

The `oapp_quote(...)` instruction is used to quote the fee required by the `deposit(...)` instruction to send a cross-chain message.

src/instructions/oapp_instr/oapp_quote.rs:L63-L69

```
    let endpoint_quote_params = EndpointQuoteParams {
        sender: ctx.accounts.oapp_config.key(),
        dst_eid: ctx.accounts.vault_authority.dst_eid,
        receiver: ctx.accounts.peer.address,
        message: lz_message,
        pay_in_lz_token: false,
        options: options,
    };
```

However, when constructing `EndpointQuoteParams`, `pay_in_lz_token` is forcibly set to false, which prevents the `oapp_quote(...)` instruction from estimating the fee when paying with `lz_token`. If users want to pay the fee using `lz_token`, they cannot directly obtain the fee through the `oapp_quote(...)` instruction.

Recommendation

Modify the parameters accepted by this instruction from `DepositParams` to `QuoteParams`, which includes the fields from `DepositParams` as well as a `pay_in_lz_token` field. Then, construct `EndpointQuoteParams` based on the `pay_in_lz_token` value provided by the user.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

5. Unused stored bump wastes compute units

Severity: Informational	Category: Gas Optimization

Target:
- src/instructions/oapp_instr/set_rate_limit.rs
- src/instructions/oapp_instr/set_accounts_list.rs

Description

The program defines relevant account types with a `bump` field. When the account is initialized, the PDA `bump` is calculated by the program and stored in the account's `bump` field. The `bump` field can be used directly in instructions requiring PDA verification, avoiding recalculation by the program and saving compute units.

src/instructions/oapp_instr/set_rate_limit.rs:L11-L15

```
pub struct SetRateLimit<'info> {
    //...
    #[account(
        mut,
        seeds = [PEER_SEED, &oapp_config.key().to_bytes(),
&params.dst_eid.to_be_bytes()],
        bump
    )]
    pub peer: Account<'info, Peer>,
    //...
}
```

src/instructions/oapp_instr/set_accounts_list.rs:L18-L23

```
pub struct SetAccountList<'info> {
    //...
    #[account(
        mut,
        seeds = [LZ_RECEIVE_TYPES_SEED, &oapp_config.key().as_ref()],
        bump
    )]
    pub lz_receive_types: Account<'info, OAppLzReceiveTypesAccounts>,
    //...
}
```

However, the `peer` accounts in the `set_rate_limit(...)` instruction and the `lz_receive_types` accounts in the `set_accounts_list(...)` instruction do not use the stored `bump`. This causes the program to recalculate the `bump`, wasting compute units.

Recommendation

Use the `bump` stored in the account field directly instead of having the program recalculate it.

Status

This issue has been acknowledged by the team.

6. Unused `event_cpi` attribute

Severity: Informational	Category: Gas Optimization

Target:
- src/instructions/vault_instr/withdraw.rs
- src/instructions/oapp_instr/set_accounts_list.rs

Description

Adding the `event_cpi` attribute to `Accounts` allows using the `emit_cpi!()` macro to send events to the program itself via cross-program invocation (CPI).

src/instructions/vault_instr/withdraw.rs:L7-L9

```
#[event_cpi]
#[derive(Accounts)]
pub struct Withdraw<'info> {
    //...
}
```

src/instructions/oapp_instr/oapp_lz_receive.rs:L13-L16

```
#[event_cpi]
#[derive(Accounts)]
#[instruction(params: OAppLzReceiveParams)]
pub struct OAppLzReceive<'info> {
    //...
}
```

The `withdraw(...)` and `oapp_lz_receive(...)` instructions have the `event_cpi` attribute, but during execution they do not use the `emit_cpi!()` macro to emit events; instead, they trigger events using `emit!()`.

Unnecessary `event_cpi` attributes increase transaction size because an extra account must be passed, resulting in wasted gas.

Recommendation

Remove unnecessary `event_cpi` attributes.

Status

This issue has been acknowledged by the team.

7. Asymmetric packaging method for outbound and inbound messages

Severity: Informational	Category: Business Logic

Target:
- src/instructions/msg_codec.rs
- src/instructions/oapp_instr/oapp_lz_receive.rs

Description

For outbound messages, the program implements encoding in `VaultDepositParams`. Fields shorter than 32 bytes are padded with zero bytes and stored as 32-byte big-endian values.

src/instructions/msg_codec.rs:L23-L36

```
impl VaultDepositParams {
    pub fn encode(&self) -> Vec<u8> {
    //...
        buf.extend_from_slice(&to_bytes32(&self.src_chain_id.to_be_bytes()));
        buf.extend_from_slice(&to_bytes32(&self.token_amount.to_be_bytes()));
        buf.extend_from_slice(&to_bytes32(&self.src_chain_deposit_nonce.to_be_bytes()));
        buf
    }
}
```

However, for inbound messages, `AccountWithdrawSol` uses `decode_packed` for decoding in the `oapp_lz_receive(...)` instruction. This means that messages on the source chain are tightly packed (fields shorter than 32 bytes are not padded to 32 bytes). This asymmetric message packaging format may cause confusion during development.

src/instructions/oapp_instr/oapp_lz_receive.rs:L269-L300

```
impl AccountWithdrawSol {
  pub fn decode_packed(encoded: &[u8]) -> Result<Self> {
      //...
      offset += 8;
      let fee = u64::from_be_bytes(encoded[offset..offset + 8].try_into().unwrap());
      offset += 8;
      let chain_id = u64::from_be_bytes(encoded[offset..offset +
8].try_into().unwrap());
      offset += 8;
      let withdraw_nonce = u64::from_be_bytes(encoded[offset..offset +
8].try_into().unwrap());
      //...
  }
}
```

Recommendation

It is recommended to use the same message packaging format for both inbound and outbound messages.

Status

This issue has been acknowledged by the team.

8. Implement two step ownership transfer

Severity: Informational	Category: Business Logic
Target: - src/instructions/oapp_instr/transfer_admin.rs	

Description

The `transfer_admin(...)` instruction is used to transfer the oApp admin authority.

src/instructions/oapp_instr/transfer_admin.rs:L7-L16

```
#[derive(Accounts)]
pub struct TransferOwner<'info> {
    #[account(mut)]
    pub owner: Signer<'info>,

    #[account(
        mut,
        seeds = [crate::instructions::VAULT_AUTHORITY_SEED],
        bump = vault_authority.bump
    )]
    pub vault_authority: Box<Account<'info, VaultAuthority>>,

    #[account(
        seeds = [OAPP_SEED],
        bump = oapp_config.bump,
    )]
    pub oapp_config: Account<'info, OAppConfig>,
}
```

However, this instruction does not implement a two-step ownership transfer. If the oApp admin authority is mistakenly transferred to an address not controlled by the project team, the admin privileges cannot be recovered.

Recommendation

It is recommended to include `new_admin` in `TransferAdmin<'info>` and apply a `Signer` constraint to ensure that `new_admin` is under control.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 5a478c2:

File	SHA-1 hash
errors.rs	419b4e1ae8b52ed581d076f530fed9bec86ff584
events.rs	46fbaeec74432a62859f2f5873685db9af673673
lib.rs	92526b61697f4be06c4e219a6cfaef952c9dd76b
instructions/mod.rs	ca80b9a973ade18d03df35e09686b03c099bf8b2
instructions/msg_codec.rs	ba9df853409471ee68f49c9bcebbe987df07f93b
instructions/seeds.rs	edf11b6fe591f0874ca95bf9a5e822b3d8db5c5b
instructions/type_utils.rs	7f4ac097182cb278136f83c4f6b8a00b0797aac0
state/mod.rs	5ee0aff396c8264eb30fffd4ab3aa62712f66077
instructions/oapp_instr/init_oapp.rs	533641354962ec1cf8268fc72d276a95ddac419b
instructions/oapp_instr/mod.rs	a7b26be3a5e0c8cf0098b55735bbaabb7f4e3122
instructions/oapp_instr/oapp_lz_receive.rs	f12fbf3faa3a47b375344ba09976d64d8c9469c3
instructions/oapp_instr/oapp_lz_receive_types.rs	fbd9d3a18809090e338f9a2f3840dfa1270af40e
instructions/oapp_instr/oapp_quote.rs	1f652038170f0a3b5b3b31b51f0a5be540a43c4b
instructions/oapp_instr/set_accounts_list.rs	3bff4b60a5a24e6bf948d9e191d6c8b9db5e71d5
instructions/oapp_instr/set_delegate.rs	27af75f906cbb5579131c24c04a1fe199c66d927
instructions/oapp_instr/set_enforced_options.rs	f4ce592c391a59a749fc0f7fd338fe485b26a00e
instructions/oapp_instr/set_peer.rs	afc92a2c9df48ea67f3793dbc82b98bd44d4a7db
instructions/oapp_instr/set_rate_limit.rs	784727d25a9b2644eb4371364edca975c208f7ee
instructions/oapp_instr/transfer_admin.rs	c0e226d3aa26f3b781de41c68fea490fb21bea83
instructions/vault_instr/deposit.rs	3066504cfda5b8b2c422aee60e456c19b8700e96
instructions/vault_instr/mod.rs	713b19eb8f46a4accc31deb041eac8b58592c72b
instructions/vault_instr/pause_vault.rs	650c5f609fbb1e3eca688e51fbf9d7a51dba2803
instructions/vault_instr/resume_vault.rs	7bd519c0f9f05b6c5d0dad2da246e80e7a1074c8

instructions/vault_instr/set_broker.rs	5ad44509145a0888c820db6963bb3ae0b07d1a8d
instructions/vault_instr/set_order_delivery.rs	1ca49b3c4c6b2b3d6937b992dbfcdecd72497cca
instructions/vault_instr/set_token.rs	474469502d50d3e1be3f5cae0d0cf5ed66e5b004
instructions/vault_instr/set_vault.rs	c41426c983e7b5f291edda853f89d810203d4d89
instructions/vault_instr/transfer_owner.rs	95d924590e5a1fd5e589762ba9b88afdce4c903e
instructions/vault_instr/withdraw.rs	9119f87056b82b8f38413ca8b2cf9739e5143511
state/oapp_state/enforced_options.rs	1f336c16944c22871a842b2f98741c4b4dbb0a3a
state/oapp_state/mod.rs	5cea5feb6e0492f7f939baea045c4eedf79ba3cb
state/oapp_state/oapp_config.rs	396fcb91ccfdfdc9dcb0c6a90573e3de3b50295b
state/oapp_state/peer.rs	840ccd14685fb570d344ab7913632510a7f11da6
state/vault_state/allowed_broker.rs	65075b3157b25d6f864bc3539624c8b985e0afab
state/vault_state/allowed_token.rs	d417519c3b577b11acbd357c0aaaaeb6cd61b0bf
state/vault_state/mod.rs	0ac2939d837cc62f08dca80f51344cd8f856f9b9
state/vault_state/vault_authority.rs	8a1d6ff8cf3da8ee97bd3d38735a361fc038209d