Emmet Finance Audit

Introduction

This audit report evaluates the security of the **Bridge** smart contracts for **Emmet Finance**. The primary goal is to ensure the reliability and robustness of these contracts through a comprehensive assessment of the system architecture and codebase. By identifying potential vulnerabilities and weaknesses, this audit aims to enhance the project's security posture, fostering trust and confidence among its users and stakeholders.

Disclaimer

This audit does not guarantee the security of the code. Security is a complex and multifaceted challenge, and a single audit cannot uncover all potential issues. We strongly recommend conducting multiple independent audits and implementing a public bug bounty program to further improve the security of the code.

Audit Details

- **Author:** Daniil Sedov
- **Date:** 22nd August 2024
- **Audit Commit:** `04094d5b058713869317b0935f18d1087c314bb6`
- **Approved Commit**: `f1593865d90dd0319f1f8b1850c9f966f6c1a4a7`
- **Language:** Tact
- **Methods:** Manual review, static analysis
- **GPG key ID**: `D7A697C2AD371820`

Scope of Audit

The focus of this audit was to verify that the smart contracts are secure, resilient, and functioning correctly according to the specifications. The audit included a review of the overall system architecture and logic, as well as a thorough examination of the contract source code. The code review encompassed checks for correctness, optimization, and readability.

Files included in the scope:

- `bridge-messages.tact`
- `bridge.tact`
- `emmet-token-strategy.tact`
- `ReceivedInstallment.tact`
- `storage.tact`
- `utils.tact`

Overview

Overall, the system is implemented with a solid understanding of the required functionality, but several critical issues were identified that need to be addressed before the contracts can be considered secure. While the core logic appears sound, the audit revealed vulnerabilities related to signature verification and iteration handling that could lead to significant security risks if exploited. Additionally, there are multiple medium and minor issues concerning code efficiency and best practices, which, if resolved, would improve the overall code quality and gas efficiency. Addressing these findings will be crucial to ensuring the system's security and reliability before deployment.

Findings

The audit uncovered a total of **23** issues (**17** resolved, **6** acknowledged), categorized as follows:

- **Critical:** Issues that could cause system-wide failures, theft, freezing, or loss of users' funds by a third party. These must be addressed before release.
- **Major:** Issues that could cause the failure of some components, freezing or loss of funds for users, or create delays and lags in the system. These must be addressed before release.
- **Medium:** Issues that could potentially cause significant problems but are more challenging to exploit or encounter under typical conditions. These must be addressed before release.
- **Minor:** Small bugs that don't cause significant problems, inefficient code, and poor practices. Addressing these before release is recommended.
- **Informational:** Issues related to code readability, naming, and other non-operational problems. Addressing these before release is recommended.

Details for each of these categories are explained below.

Critical

CRI-1 (Resolved)

Location

`bridge.tact:732`

```
// 2.2.1 Verify the signature
let ok: Bool = checkSignature(hash, sig.signature, sig.key);
```

Description

The `ok` variable was likely intended to ensure that the `checkSignature` function returned `true` in all iterations of the loop, but it is never checked afterward. The `checkSignature` function itself doesn't throw an error in case of a wrong signature—it just returns `false`. This causes the `verify_signatures` function to essentially ignore all the signatures and only check their quantity. A malicious party with a `signer` role could provide a list of randomly generated signatures and bypass the check at `bridge.tact:414`, potentially allowing them to steal all the funds from the system.

Recommendation

Add a `require` statement to throw an error in case of wrong signatures, or use `ok` as intended.

CRI-2 (Resolved)

Location

`utils.tact:85-89`

```
if (step == PASS_TO_LP) {
    self.pass_to_lp(amount, from_token);
} else {
    counter += 1;
}
```

Description

The `counter` variable is used for iterating over the `path.local_steps.steps` map, but it is only incremented in the `else` block when the step is not recognized. Otherwise, it will get stuck on the same `counter` value until the loop ends. This could lead to calling `pass_to_lp` multiple times, which might attempt to send multiple token transfers to LP, resulting in either an action phase error due to insufficient Toncoin balance or mistakenly sending more tokens and Toncoin to the LP.

Recommendation

The `counter` variable should be incremented in all cases, so it should be moved out of the `else` block. Alternatively, refactoring this loop into a `foreach` loop would improve gas efficiency and code readability.

CRI-3 (Resolved)

Location

`utils.tact:143-160`

```
// 3.2 switch the action depending on the path step value
if (step == UNLOCK) {
    // 3.2.1 If the token is TON
    if (target_token == self.native_coin()) {
        self._transfer_native_coins(recepient, amount);
    } else { // 3.2.2 If the token is a native jetton
        self._transfer_native_tokens(recepient, amount, target_token);
    }
// 3.2.2 If the token is wrapped from a foreign chain
} else if (step == MINT) {
    self._mint_wrapped_tokens(recepient, amount, target_token);
// 3.2.3 If the token is LP based
} else if (step == TRANSFER_FROM_LP) {
    self._transfer_from_lp(recepient, amount, target_token);
// 3.2.4 Unsupported step handling - move to the next step
} else {
    counter +=1;
}
```

Description

The `counter` variable is used for iterating over the `path.steps` map, but it is only incremented in the `else` block when the step is not recognized. Otherwise, it will get stuck on the same `counter` value until the loop ends. This could lead to calling some of the transfer functions multiple times, potentially causing either an action phase error due to insufficient Toncoin balance or mistakenly sending more tokens and Toncoin.

Recommendation

The `counter` variable should be incremented in all cases, so it should be moved out of the `else` block. Alternatively, refactoring this loop into a `foreach` loop would improve gas efficiency and code readability.

Major

(No major issues found)

Medium

MED-1 (Resolved)

Location

`bridge.tact:729-735`

```
// 2.2 Ensure the public key is recognised as a signer
if (self.validators.get(sig.key) != null) {
    // 2.2.1 Verify the signature
    let ok: Bool = checkSignature(hash, sig.signature, sig.key);
    // 2.2.2 Increment the counter
    counter += 1;
} // otherwise exclude the signature count
```

Description

The `counter` variable is used for iterating over the `sigs` map, but it is only incremented in the `if` block when the validator with the provided key is found. Otherwise, it will get stuck on the same `counter` value until the loop ends. This can lead to false negatives in signature checks.

Recommendation

The `counter` variable should be incremented in all cases, so it should be moved out of the `if` block. Alternatively, refactoring this loop into a `foreach` loop would improve gas efficiency and code readability.

Minor

MIN-1 (Resolved)

Location

`bridge.tact:167`

```
// 15. Generate the outgoing transaction hash
let tx_hash: Int = otx.toCell().hash();
```

Description

The `tx_hash` variable is never accessed after initialization, so these `toCell()` and `hash()` calls consume gas unnecessarily.

Recommendation

Remove the calculation of `tx_hash`.

MIN-2 (Resolved)

Location

`bridge.tact:243`

```
// 16. Generate an outgoing hash
let tx_hash: Int = otx.toCell().hash();
```

Description

The `tx_hash` variable is never accessed after initialization, so these `toCell()` and `hash()` calls consume gas unnecessarily.

Recommendation

Remove the calculation of `tx_hash`.

MIN-3 (Resolved)

Location

`bridge.tact:280`

```
// 4. Generate a transaction hash
let tx_hash: Int = otx.toCell().hash();
```

Description

The `tx_hash` variable is never accessed after initialization, so these `toCell()` and `hash()` calls consume gas unnecessarily.

Recommendation

Remove the calculation of `tx_hash`.

MIN-4 (Resolved)

Location

`bridge.tact:666-668`

```
get fun manager_role_id(): Int {
    return sha256("MANAGER_ROLE");
}
```

Description

This constant integer calculation is moved to a separate function, which is also used internally and not only as a getter. This compiles into function calls and adds unnecessary stack manipulations, consuming gas.

Recommendation

Define a constant `MANAGER_ROLE_ID = sha256("MANAGER_ROLE")`.

MIN-5 (Resolved)

Location

`bridge.tact:673-675`

```
get fun signer_role_id(): Int {
    return sha256("SIGNER_ROLE");
}
```

Description

This constant integer calculation is moved to a separate function, which is also used internally and not only as a getter. This compiles into function calls and adds unnecessary stack manipulations, consuming gas.

Recommendation

Define a constant `SIGNER_ROLE_ID = sha256("SIGNER_ROLE")`.

MIN-6 (Resolved)

Location

`bridge.tact:680-682`

```
get fun cfo_role_id(): Int {
    return sha256("CFO_ROLE");
}
```

Description

This constant integer calculation is moved to a separate function, which is also used internally and not only as a getter. This compiles into function calls and adds unnecessary stack manipulations, consuming gas.

Recommendation

Define a constant `CFO_ROLE_ID = sha256("CFO_ROLE")`.

Informational

INF-1 (Acknowledged)

Location

`emmet-token-strategy.tact:5`

```
const NOTHING: Int = 0;
```

Description

This constant is never used.

Recommendation

Remove or comment out the unused constant declaration.

INF-2 (Acknowledged)

Location

`emmet-token-strategy.tact:7`

```
const CCTP_BURN: Int = 1;
```

Description

This constant is never used.

Recommendation

Remove or comment out the unused constant declaration.

INF-3 (Acknowledged)

Location

`emmet-token-strategy.tact:8`

```
const CCTP_CLAIM: Int = 2;
```

Description

This constant is never used.

Recommendation

Remove or comment out the unused constant declaration.

INF-4 (Acknowledged)

Location

`emmet-token-strategy.tact:10`

```
const LOCK: Int = 3;
```

Description

This constant is never used.

Recommendation

Remove or comment out the unused constant declaration.

INF-5 (Acknowledged)

Location

`emmet-token-strategy.tact:12`

```
const BURN: Int = 5;
```

Description

This constant is never used.

Recommendation

Remove or comment out the unused constant declaration.

INF-6 (Acknowledged)

Location

`emmet-token-strategy.tact:17`

```
const BALANCER: Int = 8;
```

Description

This constant is never used.

Recommendation

Remove or comment out the unused constant declaration.

INF-7 (Resolved)

Location

`bridge.tact:366-368`

```
  3. The transaction with the same hash has been processed
  4. Signatures are invalid or threshold not reached
  5. The target chianId does not equal this chains's ID
```

Description

The "reverts if" section mentions these three points for this receiver, even though they are not processed here. These are handled after the confirmation from `ReceivedInstallmentHandle` in a separate receiver.

Recommendation

Remove these points from this receiver description and mention that it sends a query to the `ReceivedInstallmentHandle` and continues execution in the `UniqueReceiveInstallment` receiver.

INF-8 (Resolved)

Location

`utils.tact:73`

```
  inline fun _select_outgoing_action(
```

Description

Specifications in code comments are provided for most functions and receivers, but this function does not have them.

Recommendation

Add a comment with the specification of this function, similar to other functions.

INF-9 (Resolved)

Location

`utils.tact:93`

```
inline fun pass_to_lp(amount: Int, from_token: Int) {
```

Description

Specifications in code comments are provided for most functions and receivers, but this function does not have them.

Recommendation

Add a comment with the specification of this function, similar to other functions.

INF-10 (Resolved)

Location

`utils.tact:249`

```
value: ton("0.05"), // deployment & storage fee
```

Description

The Toncoin value is hardcoded in the code.

Recommendation

Make it a static constant for better readability and maintenance.

INF-11 (Resolved)

Location

`utils.tact:395

```
  value: ton("0.08"),
```

Description

The Toncoin value is hardcoded in the code.

Recommendation

Make it a static constant for better readability and maintenance.

INF-12 (Resolved)

Location

`bridge-messages.tact:19`

```
  recepient: Address;
```

Description

The word `recepient` is misspelled.

Recommendation

Replace with `recipient`.

INF-13 (Resolved)

Location

`utils.tact:130`

```
recepient: Address,
```

Description

The word `recepient` is misspelled.

Recommendation

Replace with `recipient`.