Continuation Team

TonRaffles

Security Assessment

November 6, 2023

TonRaffles

The security assessment was prepared by Continuation Team.

Subject of Security Assessment:

- **[raff_jetton]** (jetton-minter) - Jetton contract (TEP-74+TEP-89) with additional logic for binding the number of jettons available for minting to time, an additional administrator (DAO) responsible for minting jettons, and a function to mint available jettons to a list of wallets specified by the minting administrator.
- **[locker]** - Jetton locking and vesting contract

Codebase:

- **[raff_jetton]** - b5bc38aa39ee71ed5f28ddd53bb7a9c76f6c6607
- **[locker]** - 59de57c9f0456205757d4907054d0dd0e2962602

AUDIT SCOPE

- raff_jetton/contracts/jetton-utils.fc
- raff_jetton/contracts/jetton_minter.fc
- raff_jetton/contracts/jetton_wallet.fc
- raff_jetton/contracts/opcodes.fc
- raff_jetton/contracts/params.fc
- lock/contracts/constants.fc
- lock/contracts/locker.fc

FINDINGS

ID	Title	Status
1	Absence of pull-over-push pattern when changing administrator and additional administrator responsible for minting jettons.	Acknowledged
2	Lack of handling bounced messages in the `jetton-minter` contract.	Acknowledged
3	Lack of checking the sufficiency of funds for executing the call chain when minting jettons.	Acknowledged
4	Sending a message to the `jetton-wallet` with an unknown OP and message body when minting jettons.	Acknowledged
5	Lack of `end_parse` when calling all operations in `jetton-minter`.	Acknowledged
6	Attaching an amount equal to 1000 gas when minting jettons.	Resolved
7	Calculation of a constant value (in the context of the `locker` contract) on each execution.	Resolved

RECOMMENDATIONS

ID	Title	Status
-	Common recommendation	Acknowledged
-	Recommendation from the **Continuation Team**	Resolved

1 Absence of pull-over-push pattern when changing administrator and additional administrator responsible for minting jettons.

Locations:

- **raff_jetton/contracts/jetton_minter.fc#L120**
- **raff_jetton/contracts/jetton_minter.fc#L139**

Status: **Acknowledged** *(ignored, because this is a problem in the standard implementation)*

Description

When calling operations `3 (change_admin)` and `101 (change_minter)`, previously set `admin_address` and `minter` are overridden without ensuring they can perform transactions.

Recommendation

In such cases, it is recommended to use the ***pull-over-push*** pattern, where a new administrator is first proposed, and they must accept their new status, thereby ensuring they can perform transactions.

It's worth noting that since the practice of resetting the administrator address (assigning `addr_none`) is common in the TON Blockchain, the ability to assign `addr_none` should be added. In this case, instead of the pull-over-push pattern, a timer (e.g., 24 hours) can be used during which the administrator change can be canceled. This will protect against accidental changes.

2 Lack of handling bounced messages in the `jetton-minter` contract.

Location: **raff_jetton/contracts/jetton_minter.fc#L74-76**

Status: **Acknowledged** *(ignored, because this is a problem in the standard implementation)*

Description

When calling operations `op::mint` and `103 (request mint)`, `total_supply` is increased and a message is sent to `jetton-wallet`. If a bounce occurs from `jetton-wallet`, jetton loss may occur.

Recommendation

Implement a handler for bounced messages in the `jetton-minter` contract, similar to what is done in the `jetton-wallet` contract.

3 Lack of checking the sufficiency of funds for executing the call chain when minting jettons.

Location: **raff_jetton/contracts/jetton_minter.fc#L84-L95**

Status: **Acknowledged** *(ignored, because this is a problem in the standard implementation)*

Description

When calling the `op::mint` operation, there is no check for the sufficiency of the funds sent to cover the gas cost when receiving the message in `jetton-wallet`.

This can lead to a loss of funds, as total-supply will increase, and the wallet may not have enough gas to record the coins.

Recommendation

Add a check similar to the one used in the `send_tokens` function in the `jetton-wallet` contract.

4 Sending a message to the jetton-wallet with an unknown OP and message body when minting jettons.

Location: **raff_jetton/contracts/jetton_minter.fc#L90-L92**

Status: **Acknowledged** *(ignored, because this is a problem in the standard implementation)*

Description

When calling the `op::mint` operation, a message is sent to `jetton-wallet` with an unknown OP and message schema. This can lead to unexpected consequences, including token loss.

Recommendation

It is recommended to construct the message for `jetton-wallet` onchain, as done in the `send_tokens` function in the `jetton-wallet` contract. This allows control over the data structure.

Another solution may involve parsing and verifying the structure of incoming messages, which might even be more gas-efficient.

5 Lack of `end_parse` when calling all operations in `jetton-minter`

Location: **raff_jetton/contracts/jetton_minter.fc**

Status: **Acknowledged** *(ignored, because this is a problem in the standard implementation)*

Description

The absence of `end_parse` leaves us uncertain that the incoming message conforms to the working schema of this contract.

Recommendation

It is recommended to add `end_parse` every time parsing of the incoming payload ends, unless, of course, it should end with a slice according to the scheme.

6 Attaching an amount equal to 1000 gas when minting jettons.

Location: **raff_jetton/contracts/jetton_minter.fc#L165**

Status: **Resolved**

Recommendation

If interaction with other contracts is intended (possibly not even on your side), it is advisable to attach an amount equal to 10,000 gas. This will be sufficient for minimal processing of the request on the receiving contract. Additionally, it is desirable to add some operation code and `query_id` to the payload.

7 Calculation of a constant value (in the context of the `locker` contract) on each execution.

Location: **lock/contracts/locker.fc#L20**

Status: **Resolved**

Recommendation

The calculated address will be the same on each contract call. In this case, it's advisable to either add a separate function that runs on the first contract execution, or calculate this value at contract initialization, as it's not difficult to compute offchain.

Common recommendation.

Use constants for operation codes, error codes, and other immutable values in your code.

This practice is fundamental in promoting clean, maintainable code. By employing constants, you not only enhance code readability but also pave the way for more straightforward code reuse and a reduction in the likelihood of introducing elusive errors. This approach fosters self-explanatory, consistent code, making it comprehensible for developers and ensuring robust, error-resistant software. Embracing constants is an integral part of adhering to best practices in software engineering.

Status: **Acknowledged**

Recommendation from the Continuation Team

Consider adding code to the `jetton-wallet` contract that enables the withdrawal of ”stray” TON and jettons. This enhancement will allow for rectifying situations where tokens become stuck in the jetton wallet. This is crucial because it prevents the loss of valuable tokens due to unforeseen circumstances.

Good example.

Status: **Resolved**

Conclusion

During the security assessment, two smart contracts, `raff_jetton` and `locker`, were audited. Minor issues were identified, and recommendations were provided for improvements.

The majority of the identified issues were related to the standard implementation of `TEP-74 (Jetton)`, which served as the foundation for the `raff_jetton` contract. These minor issues were largely disregarded due to their alignment with the standard implementation.

Errors associated with the modifications made by the **TonRaffles team** to the standard contract were addressed and rectified, and minor imperfections in the `locker` contract were also fixed.

Additionally, the recommendation to enhance the security of the `jetton-wallet` contract was taken into account and implemented.

The security of the `raff_jetton` and `locker` contracts is currently at a fairly good level, despite the unresolved issues. User experience should be fairly good, especially with the security enhancements made to the `jetton-wallet` contract.

Continuation Team

2023