red_envelope Smart Contract

August
2024

SMART CONTRACT AUDIT REPORT

ExVul

www.exvul.com

Table of Contents

1. EXECUTIVE SUMMARY

Exvul Web3 Security was engaged by **red_envelope** to review smart contract implementation. The assessment was conducted in accordance with our systematic approach to evaluate potential security issues based upon customer requirement. The report provides detailed recommendations to resolve the issue and provide additional suggestions or recommendations for improvement.

Medium risk and low risk findings are primarily related to the management of privileged roles , parameters check and project logic.

The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

1.1 Methodology

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology [10] which is the gold standard in risk assessment using the following risk models:

- Likelihood: represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- Impact: measures the technical loss and business damage of a successful attack.
- Severity: determine the overall criticality of the risk.

Likelihood can be: High, Medium and Low and impact are categorized into for: High, Medium, Low, Informational. Severity is determined by likelihood and impact and can be classified into five categories accordingly, Critical, High, Medium, Low, Informational shown in table 1.1.

Likelihood	Informational	Low	Medium	High
High	*Informational*	Medium	High	Critical
Medium	*Informational*	Low	Medium	High
Low	*Informational*	Low	Low	Medium
	IMPACT			

Table 1.1 Overall Risk Severity

To evaluate the risk, we will be going through a list of items, and each would be labelled with a severity category. The audit was performed with a systematic approach guided by a comprehensive assessment list carefully designed to identify known and impactful security issues. If our tool or analysis does not identify any issue, the contract can be considered safe regarding the assessed item. For any discovered issue, we might further deploy contracts on our private test environment and run tests to confirm the findings. If necessary, we would additionally build a PoC to demonstrate the possibility of exploitation. The concrete list of check items is shown in Table 1.2.

- Basic Coding Bugs: We first statically analyze given smart contracts with our proprietary static code analyzer for known coding bugs, and then manually verify (reject or confirm) all the issues found by our tool.
- Code and business security testing: We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.
- Additional Recommendations: We also provide additional suggestions regarding the coding and development of smart contracts from the perspective of proven programming practices.

Category	Assessment Item
Basic Coding Assessment	Apply Verification Control
	Authorization Access Control
	Forged Transfer Vulnerability
	Forged Transfer Notification
	Numeric Overflow
	Transaction Rollback Attack
	Transaction Block Stuffing Attack
	Soft Fail Attack
	Hard Fail Attack
	Abnormal Memo
	Abnormal Resource Consumption
	Secure Random Number
Advanced Source Code Scrutiny	Asset Security
	Cryptography Security
	Business Logic Review
	Source Code Functional Verification

Category	Assessment Item
	Account Authorization Control
	Sensitive Information Disclosure
	Circuit Breaker
	Blacklist Control
	System API Call Analysis
	Contract Deployment Consistency Check
Additional Recommendations	Semantic Consistency Checks
	Following Other Best Practices

Table 1.2: The Full List of Assessment Items

To better describe each issue we identified, we categorize the findings with Common Weakness Enumeration (CWE-699) [14], which is a community-developed list of software weakness types to better delineate and organize weaknesses around concepts frequently encountered in software development.

2. FINDINGS OVERVIEW

2.1 Project Info And Contract Address

Project Name: **red_envelope**

Audit Time: August 7th, 2024 – August 14th , 2024

Language: FUNC

File Name	MD5
red_envelope.fc	2028C66D32AA01887BB444D425B31F22

2.2 Summary

Severity	Found	
Critical	0	
High	0	
Medium	4	🟨🟨🟨🟨
Low	2	🟦🟦
Informational	1	🟩

2.3 Key Findings

Medium risk and low risk findings are primarily related to the management of privileged roles , parameters check and project logic.

ID	Severity	Findings Title	Status	Confirm
NVE-001	Medium	Unlimited issue	Fixed	Confirmed
NVE-002	Medium	Privileged role issues	Fixed	Confirmed

ID	Severity	Findings Title	Status	Confirm
NVE-003	Medium	Receive_tokens function	Fixed	Confirmed
NVE-004	Medium	grab_with_sig function	Fixed	Confirmed
NVE-005	Low	Pass in any parameters	Fixed	Confirmed
NVE-006	Informational	Code redundancy	Fixed	Confirmed

Table 2.1: Key Audit Findings

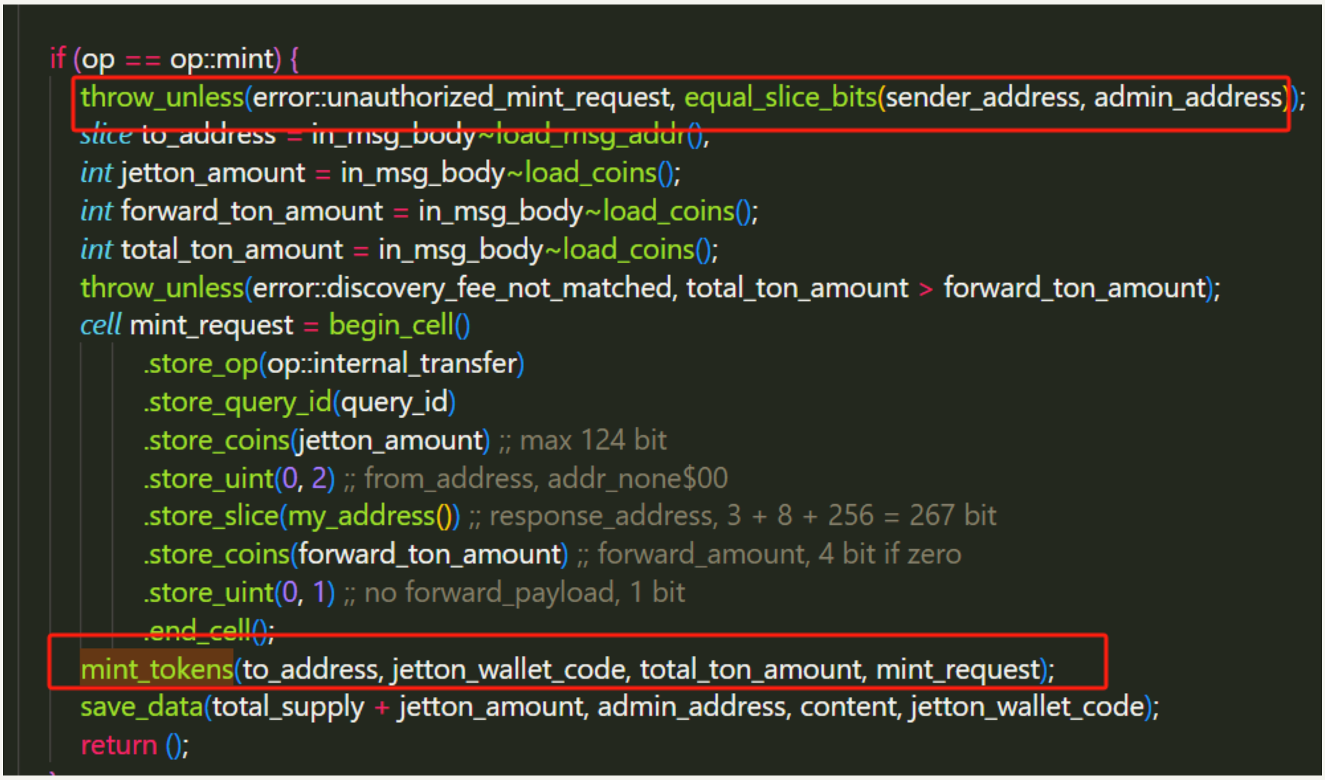
3. DETAILED DESCRIPTION OF FINDINGS

3.1 Unlimited issue

ID:	NVE-001	**Location:**	jetton-minter.func
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	High

Description:

As shown in the figure below, The contract admin privileged role can issue tokens through the mint_tokens() function and there is no upper limit. If the admin address is an EOA address, the leakage of the private key may lead to malicious issue.



```
if (op == op::mint) {
    throw_unless(error::unauthorized_mint_request, equal_slice_bits(sender_address, admin_address));
    slice to_address = in_msg_body~load_msg_addr();
    int jetton_amount = in_msg_body~load_coins();
    int forward_ton_amount = in_msg_body~load_coins();
    int total_ton_amount = in_msg_body~load_coins();
    throw_unless(error::discovery_fee_not_matched, total_ton_amount > forward_ton_amount);
    cell mint_request = begin_cell()
            .store_op(op::internal_transfer)
            .store_query_id(query_id)
            .store_coins(jetton_amount) ;; max 124 bit
            .store_uint(0, 2) ;; from_address, addr_none$00
            .store_slice(my_address()) ;; response_address, 3 + 8 + 256 = 267 bit
            .store_coins(forward_ton_amount) ;; forward_amount, 4 bit if zero
            .store_uint(0, 1) ;; no forward_payload, 1 bit
            .end_cell();
    mint_tokens(to_address, jetton_wallet_code, total_ton_amount, mint_request);
    save_data(total_supply + jetton_amount, admin_address, content, jetton_wallet_code);
    return ();
}
```

Figure 3.1.1 mint_tokens function

Recommendations:

ExVul Web3 Labs recommends adding the cap of token.

Result: Confirmed

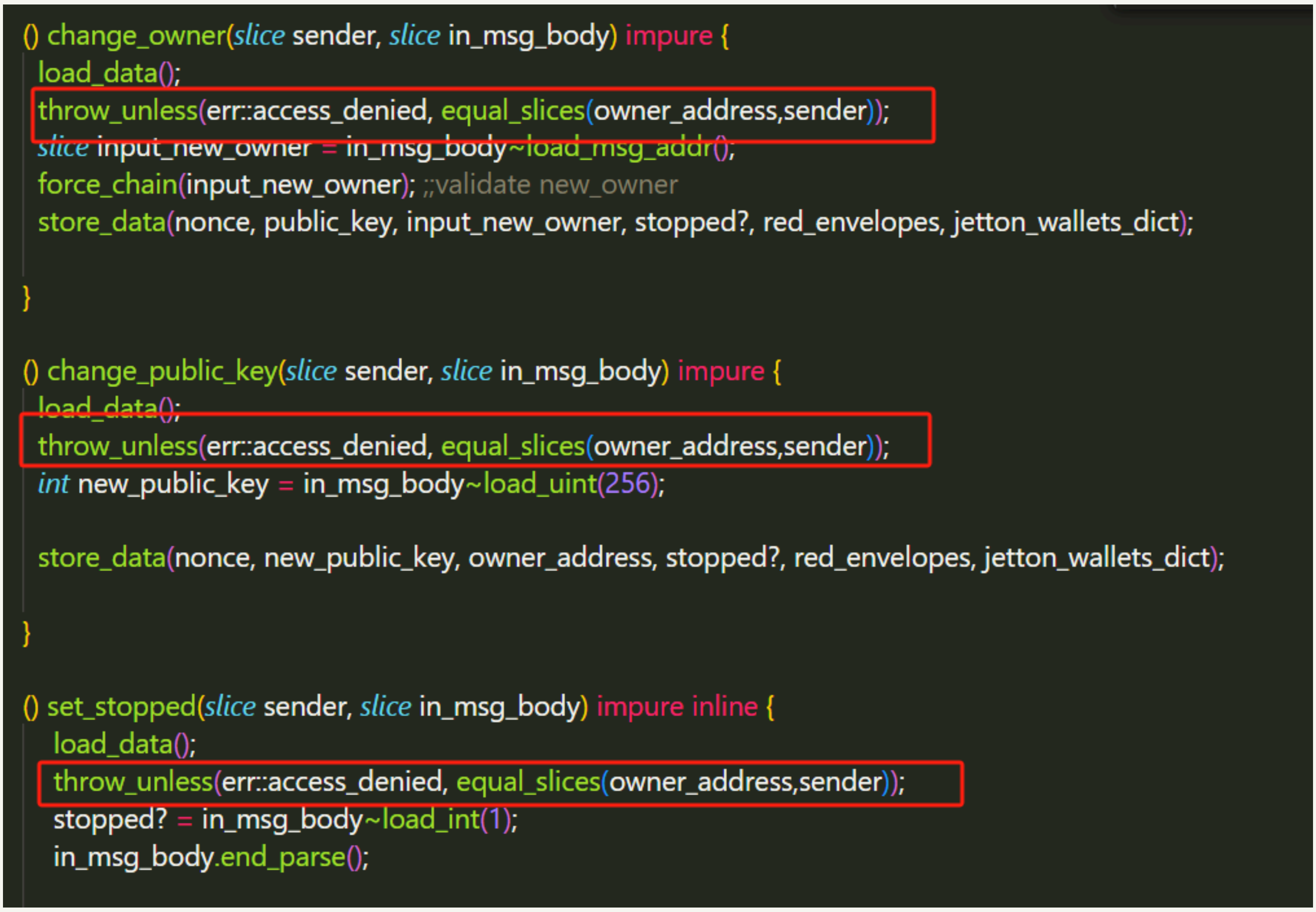
Fix Result: Fixed

3.2 Privileged role issues

ID:	NVE-002	**Location:**	red_envelope_token.func
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	High

Description:

As shown in the figure below, the owner privileged role can call change_owner, change_public_key and set_stopped functions to transfer permissions, change public key address and set paused function. If the privileged role is maliciously manipulated, the project may be seriously damaged.



```
() change_owner(slice sender, slice in_msg_body) impure {
  load_data();
  throw_unless(err::access_denied, equal_slices(owner_address,sender));
  slice input_new_owner = in_msg_body~load_msg_addr();
  force_chain(input_new_owner); ;;validate new_owner
  store_data(nonce, public_key, input_new_owner, stopped?, red_envelopes, jetton_wallets_dict);

}

() change_public_key(slice sender, slice in_msg_body) impure {
  load_data();
  throw_unless(err::access_denied, equal_slices(owner_address,sender));
  int new_public_key = in_msg_body~load_uint(256);

  store_data(nonce, new_public_key, owner_address, stopped?, red_envelopes, jetton_wallets_dict);

}

() set_stopped(slice sender, slice in_msg_body) impure inline {
    load_data();
    throw_unless(err::access_denied, equal_slices(owner_address,sender));
    stopped? = in_msg_body~load_int(1);
    in_msg_body.end_parse();
```

Figure 3.2.1 Part of the code

Recommendations:

ExVul Web3 Labs recommends the owner privileged roles is managed using multi-signatures.

Result: Confirmed

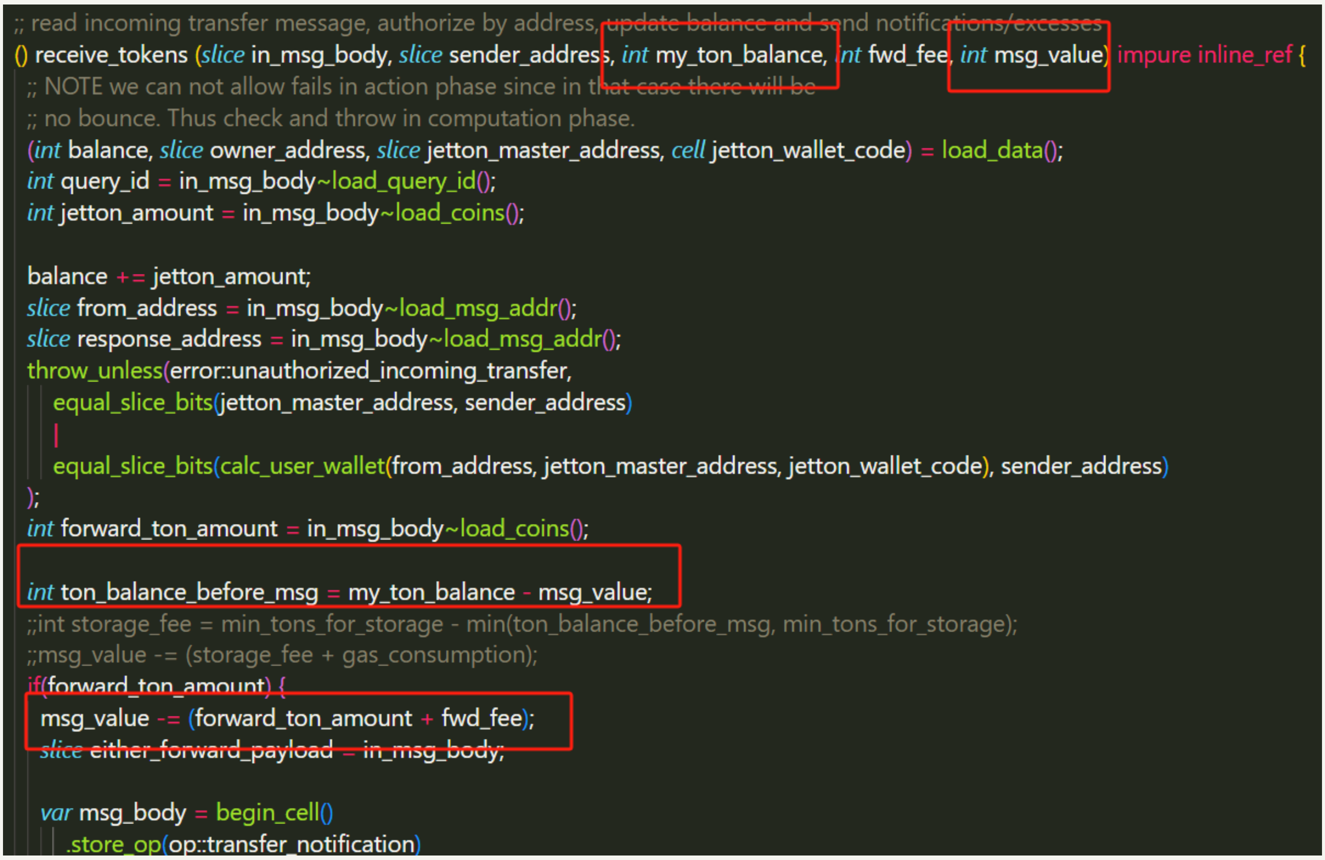
Fix Result: Ignore

3.3 Receive_tokens function

ID:	NVE-003	**Location:**	jetton-wallet.func
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	High

Description:

As shown in the figure below, In the receive_tokens() function, since msg_value and my_ton_balance are both incoming parameters, if the incoming msg_value is a negative number, it will affect the ton_balance_before_msg value. If forward_ton_amount + fwd_fee value is greater than msg_value, the calculated result will show msg_value less than zero.



```
;; read incoming transfer message, authorize by address, update balance and send notifications/excesses
() receive_tokens (slice in_msg_body, slice sender_address, int my_ton_balance, int fwd_fee, int msg_value) impure inline_ref {
  ;; NOTE we can not allow fails in action phase since in that case there will be
  ;; no bounce. Thus check and throw in computation phase.
  (int balance, slice owner_address, slice jetton_master_address, cell jetton_wallet_code) = load_data();
  int query_id = in_msg_body~load_query_id();
  int jetton_amount = in_msg_body~load_coins();

  balance += jetton_amount;
  slice from_address = in_msg_body~load_msg_addr();
  slice response_address = in_msg_body~load_msg_addr();
  throw_unless(error::unauthorized_incoming_transfer,
    equal_slice_bits(jetton_master_address, sender_address)
    |
    equal_slice_bits(calc_user_wallet(from_address, jetton_master_address, jetton_wallet_code), sender_address)
  );
  int forward_ton_amount = in_msg_body~load_coins();

  int ton_balance_before_msg = my_ton_balance - msg_value;
  ;;int storage_fee = min_tons_for_storage - min(ton_balance_before_msg, min_tons_for_storage);
  ;;msg_value -= (storage_fee + gas_consumption);
  if(forward_ton_amount) {
    msg_value -= (forward_ton_amount + fwd_fee);
    slice either_forward_payload = in_msg_body;

    var msg_body = begin_cell()
      .store_op(op::transfer_notification)
```

Figure 3.3.1 addOwner function

Recommendations:

ExVul Web3 Labs recommends checking the my_ton_balance is greater than msg_value and (forward_ton_amount + fwd_fee) are greater than or equal to msg_value.

Result: Confirmed

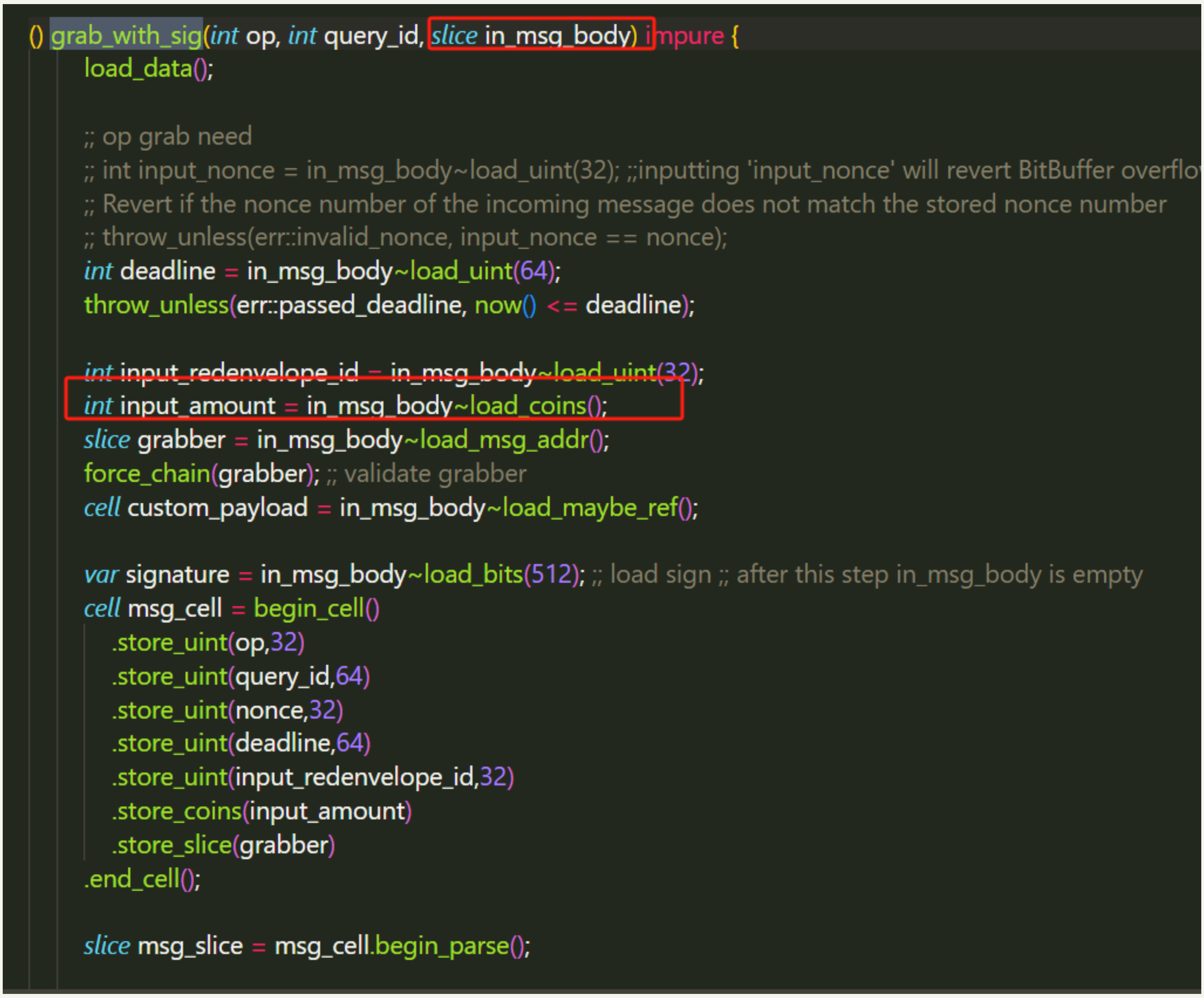
Fix Result: Fixed

3.4 grab_with_sig function

ID:	NVE-004	**Location:**	red_envelope_token.func
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Medium	**Impact:**	Medium

Description:

As shown in the figure below, the grab_with_sig() function is used to grab red envelopes. The value of each red envelope grabbed is input_amount, which is passed in by the caller. This may result in a larger number of people grabbing red envelopes for the first time.



```
() grab_with_sig(int op, int query_id, slice in_msg_body) impure {
    load_data();

    ;; op grab need
    ;; int input_nonce = in_msg_body~load_uint(32); ;;inputting 'input_nonce' will revert BitBuffer overflo
    ;; Revert if the nonce number of the incoming message does not match the stored nonce number
    ;; throw_unless(err::invalid_nonce, input_nonce == nonce);
    int deadline = in_msg_body~load_uint(64);
    throw_unless(err::passed_deadline, now() <= deadline);

    int input_redenvelope_id = in_msg_body~load_uint(32);
    int input_amount = in_msg_body~load_coins();
    slice grabber = in_msg_body~load_msg_addr();
    force_chain(grabber); ;; validate grabber
    cell custom_payload = in_msg_body~load_maybe_ref();

    var signature = in_msg_body~load_bits(512); ;; load sign ;; after this step in_msg_body is empty
    cell msg_cell = begin_cell()
        .store_uint(op,32)
        .store_uint(query_id,64)
        .store_uint(nonce,32)
        .store_uint(deadline,64)
        .store_uint(input_redenvelope_id,32)
        .store_coins(input_amount)
        .store_slice(grabber)
    .end_cell();

    slice msg_slice = msg_cell.begin_parse();
```

Figure 3.4.1 removeOwner function

Recommendations:

ExVul Web3 Labs recommends including red envelope amount verification during signature verification.

Result: Confirmed

Fix Result: Fixed

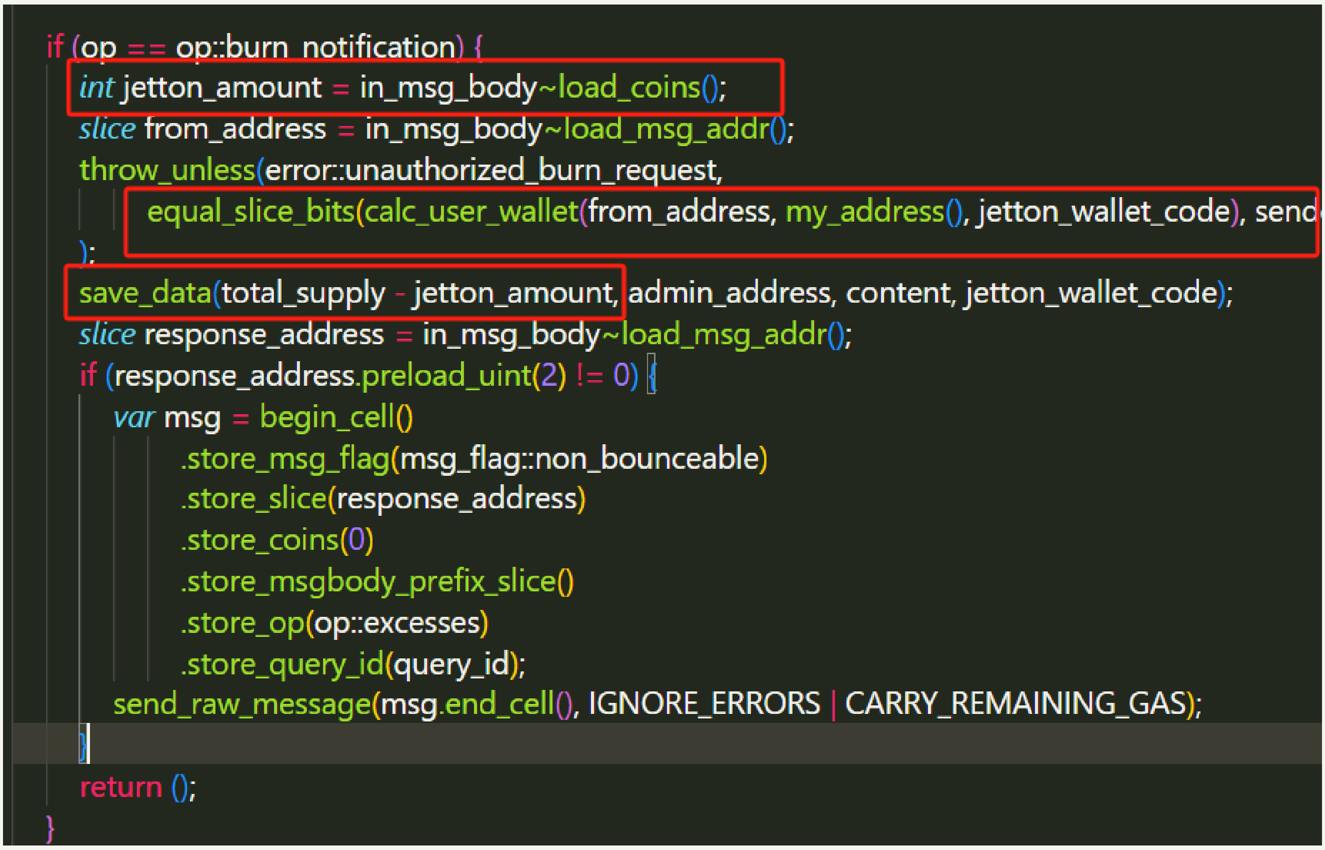
3.5 Pause function and blacklist function

ID:	NVE-005	Location:	jetton-minter.func

Severity:	Low	**Category:**	Business Issues
Likelihood:	Informational	**Impact:**	Low

Description:

As shown in the figure below,When burning tokens, if the incoming burn amount is 0, the operation of burning tokens will have no effect..



```
if (op == op::burn_notification) {
    int jetton_amount = in_msg_body~load_coins();
    slice from_address = in_msg_body~load_msg_addr();
    throw_unless(error::unauthorized_burn_request,
        equal_slice_bits(calc_user_wallet(from_address, my_address(), jetton_wallet_code), send
    );
    save_data(total_supply - jetton_amount, admin_address, content, jetton_wallet_code);
    slice response_address = in_msg_body~load_msg_addr();
    if (response_address.preload_uint(2) != 0) {
        var msg = begin_cell()
            .store_msg_flag(msg_flag::non_bounceable)
            .store_slice(response_address)
            .store_coins(0)
            .store_msgbody_prefix_slice()
            .store_op(op::excesses)
            .store_query_id(query_id);
        send_raw_message(msg.end_cell(), IGNORE_ERRORS | CARRY_REMAINING_GAS);
    }
    return ();
}
```

Figure 3 .5.1 Part of the code

The same problem exists with the following files and functions：

jetton-wallet.func: send_tokens(); Transfer will be invalid.

red_envelope_token.func: create_redenvelope(); Create red envelope will be invalid.

Recommendations:

ExVul Web3 Labs recommends adding amount check.

Result: Confirmed

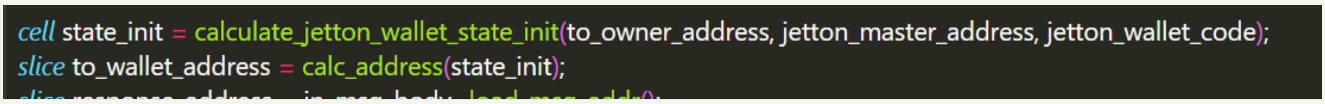
Fix Result: Fixed

3.6 Smaller signature threshold

ID:	NVE-006	**Location:**	jetton-wallet.func
Severity:	Low	**Category:**	Business Issues
Likelihood:	Informational	**Impact:**	Medium

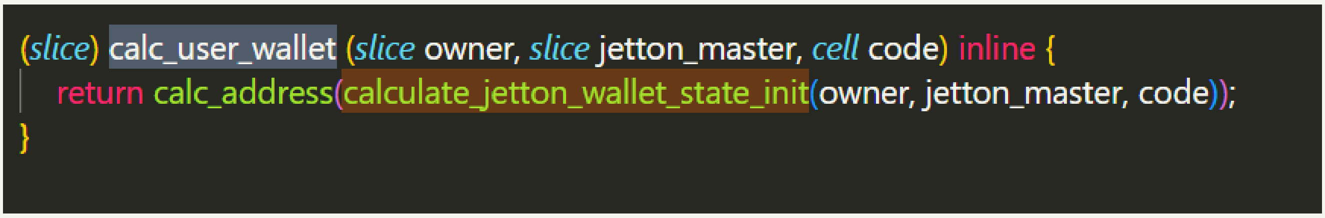
Description:

As shown in the figure below, During the call of the send_tokens() function, calculate_jetton_wallet_state_init() and calc_address() are called in sequence to obtain the address. Since the calc_user_wallet() function executes these two functions together, so calc_user_wallet() can be used directly.



```
cell state_init = calculate_jetton_wallet_state_init(to_owner_address, jetton_master_address, jetton_wallet_code);
slice to_wallet_address = calc_address(state_init);
slice response_address = in_msg_body~load_msg_addr();
```

Figure 3.6.1 Part of the code



```
(slice) calc_user_wallet (slice owner, slice jetton_master, cell code) inline {
    return calc_address(calculate_jetton_wallet_state_init(owner, jetton_master, code));
}
```

Figure 3.6.2 Part of the code

Result: Confirmed

Fix Result: Fixed

4. CONCLUSION

In this audit, we thoroughly analyzed **red_envelope** smart contract implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been communicated to the project leader. We therefore consider the audit result to be **PASSED**. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

5. APPENDIX

5.1 Basic Coding Assessment

5.1.1 Apply Verification Control

- Description: The security of apply verification
- Result: Not found
- Severity: Critical

5.1.2 Authorization Access Control

- Description: Permission checks for external integral functions
- Result: Not found

- Severity: Critical

5.1.3 Forged Transfer Vulnerability

- Description: Assess whether there is a forged transfer notification vulnerability in the contract
- Result: Not found
- Severity: Critical

5.1.4 Transaction Rollback Attack

- Description: Assess whether there is transaction rollback attack vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.5 Transaction Block Stuffing Attack

- Description: Assess whether there is transaction blocking attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.6 Soft Fail Attack Assessment

- Description: Assess whether there is soft fail attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.7 Hard Fail Attack Assessment

- Description: Examine for hard fail attack vulnerability
- Result: Not found
- Severity: Critical

5.1.8 Abnormal Memo Assessment

- Description: Assess whether there is abnormal memo vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.9 Abnormal Resource Consumption

- Description: Examine whether abnormal resource consumption in contract processing.
- Result: Not found
- Severity: Critical

5.1.10 Random Number Security

- Description: Examine whether the code uses insecure random number.
- Result: Not found
- Severity: Critical

5.2 Advanced Code Scrutiny

5.2.1 Cryptography Security

- Description: Examine for weakness in cryptograph implementation.
- Results: Not Found
- Severity: High

5.2.2 Account Permission Control

- Description: Examine permission control issue in the contract
- Results: Not Found
- Severity: Medium

5.2.3 Malicious Code Behavior

- Description: Examine whether sensitive behavior present in the code
- Results: Not found
- Severity: Medium

5.2.4 Sensitive Information Disclosure

- Description: Examine whether sensitive information disclosure issue present in the code.
- Result: Not found
- Severity: Medium

5.2.5 System API

- Description: Examine whether system API application issue present in the code
- Results: Not found
- Severity: Low

6. DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without ExVul's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts ExVul to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. ExVul's position is that each company and individual are responsible for their own due diligence and continuous security. ExVul's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

7. REFERENCES

[1] MITRE. CWE- 191: Integer Underflow (Wrap or Wraparound).

https://cwe.mitre.org/data/ definitions/191.html.

[2] MITRE. CWE- 197: Numeric Truncation Error.

https://cwe.mitre.org/data/definitions/197. html.

[3] MITRE. CWE-400: Uncontrolled Resource Consumption.

https://cwe.mitre.org/data/ definitions/400.html.

[4] MITRE. CWE-440: Expected Behavior Violation.

https://cwe.mitre.org/data/definitions/440. html.

[5] MITRE. CWE-684: Protection Mechanism Failure.

https://cwe.mitre.org/data/definitions/ 693.html.

[6] MITRE. CWE CATEGORY: 7PK - Security Features.

https://cwe.mitre.org/data/definitions/ 254.html.

[7] MITRE. CWE CATEGORY: Behavioral Problems.

https://cwe.mitre.org/data/definitions/438. html.

[8] MITRE. CWE CATEGORY: Numeric Errors.

https://cwe.mitre.org/data/definitions/189.html.

[9] MITRE. CWE CATEGORY: Resource Management Errors.

https://cwe.mitre.org/data/ definitions/399.html.

[10] OWASP. Risk Rating Methodology.

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

www.exvul.com

contact@exvul.co

@exvul

github.com/ExVul

ExVul