soken

Smart Contract Audit Report

Play Agent

soken

1. Audit details

Token Name	Play Agent
Contract address	EQCK-u6Y0WjEVCn1wExkhMNXD43r_Kgfoyf-wwh3s2uMKta9
Contract URL	https://tonscan.com/EQCK-u6Y0WjEVCn1wExkhMNXD43r_Kgfoyf-wwh3s2uMKta9?tab=source
Language	FunC
Platform	TON
Date	8 February 2025

General Overview

The audited smart contract is a TON Jetton master contract responsible for minting, tracking, and managing Jetton balances for users through their individual Jetton wallets. The contract includes admin privileges, token minting, burning, and wallet address calculations.

The contract follows standard Jetton principles, but has security concerns related to centralized control, minting vulnerabilities, and unrestricted updates. If these issues are not addressed, the contract could be used in a scam or rug-pull scenario.

1. Security Risks Identified

1.1 Centralized Ownership Risks

- Issue: The admin has complete control over minting and modifying contract parameters.
- Risk: This could lead to token inflation, admin abuse, or rug-pull scenarios.
- Recommendation: Implement multi-signature authentication or governance mechanisms to reduce centralization risks.

1.2 Unrestricted Token Minting

- Issue: The contract allows unlimited minting without any verification.
- Risk: The admin can create an infinite number of tokens, diluting token value and harming investors.
- Recommendation: Introduce minting caps or require DAO-based approvals for minting operations.

1.3 Burn Underflow Risk

- Issue: The burn function does not check if the total supply becomes negative.
- Risk: Token supply can be broken if excessive burning is allowed.
- Recommendation: Implement a safety check to prevent underflow.

1.4 Metadata Manipulation

- Issue: The admin can change token metadata (content) at any time.
- Risk: This can mislead investors by altering the contract’s details after deployment.
- Recommendation: Consider making metadata immutable post-deployment.

2. Technical Analysis

2.1 Smart Contract Overview

The contract consists of:

Storage:
- total_supply: Tracks total Jettons minted.
- admin_address: Defines the contract owner.
- content: Stores Jetton metadata.
- jetton_wallet_code: The Jetton wallet contract code.

Functions:
- mint_tokens(): Mints new Jettons.
- burn_notification(): Handles token burning.
- recv_internal(): Processes internal messages (Jetton logic).
- get_wallet_address(): Provides user Jetton wallet address.

2.2 High-Risk Issues (Detailed)

H-01: Unlimited Token Minting by Admin

Issue: The admin can mint unlimited tokens at any time.

```
throw_unless(73, equal_slices(sender_address, admin_address));  ;; Only admin
can mint

save_data(total_supply + jetton_amount, admin_address, content,
jetton_wallet_code);
```

Impact: This can lead to token inflation, making it easy for an admin to print tokens and dump them on buyers (rug pull).
Recommendation: Implement minting limits or DAO-based approval for new minting events.

H-02: No Minting Verification

Issue: The mint_tokens() function does not verify that the amount minted matches the declared amount in master_msg.
Impact: The contract can mint more tokens than expected, leading to inflation or hidden token creation.
Recommendation: Add a validation check:

```
    throw_unless(76, amount == jetton_amount);
```

H-03: Admin Can Change Token Metadata

Issue: The op == 4 operation allows the admin to change metadata (content) at any time.
Impact: This allows bait-and-switch scams, where the admin changes token details after investors have purchased tokens.
Recommendation: Make metadata immutable after deployment.

H-04: Burn Underflow Risk

Issue: The contract does not check if total supply goes negative when burning tokens.
Impact: If someone burns more than available, the contract may break.
Recommendation: Add an underflow check:

```
    throw_unless(77, total_supply >= jetton_amount);
```

2.3. Medium-Risk Issues

M-01: No Transfer Restrictions

The contract allows tokens to be minted to any address, including:
- Invalid TON addresses.
- Blacklisted wallets.
- Contracts that cannot receive Jettons.

Recommendation: Validate addresses before minting.

M-02: No Multi-Signature Security

The admin is a single point of failure.
If compromised, the attacker could print unlimited tokens.
Recommendation: Use multi-sig or DAO governance.

M-03: Gas Inefficiencies

The contract recalculates wallet addresses multiple times instead of storing them.
Recommendation: Cache addresses in storage.

3. Recommendations

Issue	Recommendation
H-01: Unlimited Minting	Implement minting limits or DAO-based approval.
H-02: No Minting Verification	Ensure amount == jetton_amount.
H-03: Metadata Manipulation	Make content immutable after deployment.
H-04: Burn Underflow	Add a check to prevent total_supply from going negative.
M-01: No Address Validation	Ensure valid Jetton wallet addresses.
M-02: No Multi-Sig	Require multi-signature admin approval.
M-03: Gas Optimizations	Cache wallet addresses to reduce costs.

4. Conclusion

The TON Jetton contract adheres to the standard Jetton token model but poses security and centralization risks due to unrestricted minting, centralized admin control, and metadata manipulation. Implementing governance mechanisms, minting caps, and enhanced security checks can significantly improve the contract's trustworthiness and decentralization.

Final Verdict: **Moderate Risk (5/10)**

If the identified issues are fixed, the contract can be considered safe for use.
If left unchanged, it could be used for a scam or rug pull.

SOKEN.IO

WEBSITE: WWW.SOKEN.IO

TELEGRAM: @SOKEN_SUPPORT

X (TWITTER): @SOKEN_TEAM