

Hashlock - Audit Report

 aiaudit.hashlock.com/audit/a6f207a9-b0a6-4eca-af4f-1233fd1f8fe8

Disclaimer: This AI-generated scan may contain inaccuracies and false positives. For professional audits, consult Hashlock security audit experts.

Token Faucet/Mining Contract

A token mining contract that allows users to claim a fixed amount of ERC20 tokens at regular intervals with daily limits and cooldown periods. Users can mine 1 token per hour up to a maximum of 10 tokens per day.

Access Control

ownable

Scan results

1

Centralized Control Risk with Emergency Withdrawal Function

The ``emergencyWithdraw`` function allows the contract owner to withdraw any amount of tokens without any restrictions, timelock, or multi-signature requirements. This creates a significant centralization risk where the owner could perform a rug pull by draining all contract funds, leaving users unable to mine tokens.

2

Potential Token Transfer Failure with Non-Standard ERC20 Tokens

The contract uses direct ``token.transfer()`` calls and checks the boolean return value. However, some ERC20 tokens (like USDT) do not return a boolean value from transfer functions, which could cause the transaction to revert unexpectedly. Additionally, some tokens may return false instead of reverting on failure, which the contract handles correctly, but others might have different behaviors.

1

Timestamp Manipulation Vulnerability in Daily Reset Logic

The contract uses `block.timestamp / 1 days` to determine the current day for resetting daily mining limits. Miners can manipulate block timestamps within a 15-second window to potentially reset their daily counter early or extend their mining window. While the impact is limited due to the small manipulation window, it could allow users to mine slightly more than intended over day boundaries.

1

Gas Optimization: Redundant Balance Check in Mine Function

The `mine()` function checks the contract's token balance before every transfer, but this check is redundant since the ERC20 transfer will fail anyway if there are insufficient funds. The balance check adds unnecessary gas cost to every mining operation.

1

Integer Division Precision Loss in Day Calculation

The contract uses `block.timestamp / 1 days` for day calculations, which truncates any fractional parts. This creates edge cases around day boundaries where the exact timing of when a new day begins might not align with user expectations, especially in different time zones or when considering leap seconds.

2

Missing Contract Funding Mechanism

The contract has no built-in mechanism for funding itself with tokens. While the `ContractFunded` event is defined, there's no corresponding function to actually fund the contract. The contract relies on external token transfers to its address, but provides no standardized way to track or manage funding.

[Learn More](#)
