Smart Contract Audit Report

Address: 0x6B175474E89094C44Da98b954EedeAC495271d0F

Risk Score: 80/100 - High Risk

Contract Analysis:

Summary of the Dai Smart Contract

This contract implements the **Dai Stablecoin (DAI)**, a decentralized ERC-20 token primarily used in the MakerDAO ecosystem. It includes standard token functionality (transfer, approve, mint, burn) and additional features like **permit-based approvals** (EIP-712) and **authorization controls**.

What Does This Contract Do?

Core Functionality

- **ERC-20 Token**: Implements all standard functions:
  - `transfer`, `transferFrom`, `approve`
  - Events: `Transfer`, `Approval`
  - Metadata: name, symbol, decimals, version

- **Minting & Burning**:
  - `mint(address usr, uint wad)`: Increases the token supply and assigns tokens to an address.
  - `burn(address usr, uint wad)`: Decreases the token supply by burning tokens from an address.

- **Token Aliases**:

  - `push`, `pull`, `move`: Syntactic sugar for transfers, useful in DeFi integrations.

- **Permit (EIP-712)**:

  - Allows off-chain signed approvals without needing an on-chain transaction.

  - Uses domain separation and signature verification.

Risky Functions & Owner Controls

Authorization System

- **`wards` mapping**: Tracks addresses with admin rights.

- **`rely(address)` / `deny(address)`**: Grants or revokes admin rights. Only callable by current admins.

- **`auth` modifier**: Restricts access to admin functions.

Admin-Only Functions

- **`mint`**: Only admins can mint new DAI.

- **`burn`**: Anyone can burn their own DAI, but admins can burn anyone's DAI (if allowed via approval).

- **No direct blacklist or freeze functions**, but **admins can mint unlimited DAI**, which is a **centralization risk**.

Mint / Burn / Blacklist Features

Feature	Implemented?	Notes
Mint	Yes	Only by authorized addresses (`wards`)
Burn	Yes	Anyone can burn their own DAI
Blacklist

Attack Vector:
The smart contract provided resembles a standard ERC-20 token implementation (in this case, likely **Dai Stablecoin**), with additional mechanics for authorization (referred to as **wards**, signifying privileged roles), delegation via `permit`, and some alias functions such as `push`, `pull`, and `move`. It is derived from the DSS (Dai Stablecoin System) used in MakerDAO and exhibits agentic behavior through its composed mechanisms.

Below is an analysis focused on **possible abuse vectors** under certain conditions, focusing on dangerous functions, access control design weaknesses, and realistic adversarial scenarios.

---

1. **Potential Dangerous Functions**

`mint(address, uint)`
- **Role in Contract**: Allows creation of new tokens.
- **Access Control**: Protected by the `auth` modifier only those with `wards[msg.sender] == 1` can execute it.
- **Risk Level**: **Critical**
- **Potential for Abuse**:
  - If an attacker ever gains control of a `ward`-listed address or compromise its private key, they

would have **unlimited minting capabilities**.

  - This can lead to **massive inflation**, devaluing the token supply, and enabling direct financial gain via arbitrage or dumping.

`rely(address)` and `deny(address)`

- **Role in Contract**: Modifies access control by granting or revoking `ward` status.

- **Access Control**: Also guarded by `auth`.

- **Risk Level**: **Critical**

- **Potential for Abuse**:

  - The same wards that allow minting also allow changes to who can mint.

  - If a compromised address adds another malicious address, it could **permanently delegate mint rights** or obfuscate the original attacker.

`permit(...)`

- **Role in Contract**: Enables token approvals via signature without calling `approve()` on-chain using EIP712.

- **Risk Level**: **Significant**

- **Potential for Abuse**:

  - **Phishing or front-running attacks**: A user could be convinced to sign an unsafe permit, allowing a malicious party to drain their tokens if long-term or infinite approvals are accepted.

  - **Nonce management failure**: If a users nonce is replayed via multiple signed permits (due to poor signing applications or phishing), **allowances may be overwritten or misapplied**.

###

Token Behavior:

The token can be classified as: **Stable Utility**.

Explanation:

- **Name and Symbol**: The contract explicitly states the name as "Dai Stablecoin" and symbol as "DAI", both of which are well-known identifiers for a decentralized stablecoin pegged to the US Dollar.
- **Contract Functionality**: The implementation includes standard ERC-20 functions, authorization mechanisms (e.g., `rely`, `deny`), and token minting/burning capabilities. These features are consistent with MakerDAOs Dai stablecoin infrastructure.
- **Decimals**: The `decimals` value is set to **18**, which is typical for ERC-20 tokens like Dai.
- **Licensing & Documentation**: The code includes references to well-known, established open-source licenses and contains mature, battle-tested logic derived from MakerDAO's Deployment Stabilization System (`dss`).
- **Owner Address**: The owner address `0x6B175474E89094C44Da98b954EedeAC495271d0F` is known to be associated with MakerDAO governance, reinforcing that this is a legitimate and established token.
- **Mint/Burn Mechanism**: Controlled by the `auth` modifier, meaning only authorized addresses can mint, ensuring supply is managed a feature consistent with a stablecoin system.
- **Top Holder Share is 85%**: While normally concerning, given the clear context that this is Dai (DAI), it aligns with MakerDAO-reserved balances or institutional allocations used for system stability or governance purposes.

Based on the **code identity, structure, vintage, and ownership**, this is clearly a mainnet Dai (DAI) clone. Therefore, the token is a **Stable Utility** token.

Import Risk:

No imports found in contract.