L20: Lab - Token Analysis

Module B: Ethereum & Smart Contracts

Blockchain & Cryptocurrency Course

December 2025

Learning Objectives

By the end of this lab, you will be able to:

- Analyze real-world token contracts on Etherscan (USDC, DAI)
- Interpret token holder distribution and centralization metrics
- Track large transfers and identify whale activity
- Examine upgrade events and governance actions
- Deploy and verify a custom ERC-20 token with advanced features
- Interact with your deployed token via Etherscan

Lab Overview

Part 1: Analyzing USDC (30 minutes)

- Contract architecture and proxy pattern
- Admin roles and permissions
- Blacklist mechanism analysis
- Recent upgrade events

Part 2: Analyzing DAI (20 minutes)

- Permit function (EIP-2612) implementation
- Minting authorization via Maker Vat
- Holder distribution comparison with USDC

Part 3: Deploy Custom Token (30 minutes)

- Deploy ERC-20 with minting, burning, and pausing
- Verify contract on Etherscan
- Test all functions via Etherscan UI

Part 1: USDC Contract Architecture

Step 1: Locate USDC on Etherscan

1. Go to **etherscan.io**
2. Search: `0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48`
3. Click **Contract** tab

Observations:

- Contract name: **FiatTokenProxy**
- Compiler: Solidity 0.6.12
- Verified source code available
- Implements: ERC20, Pausable, Blacklistable, Ownable

Step 2: Identify Proxy Pattern

- Look for `implementation()` function in Read Contract
- Current implementation address: (changes with upgrades)
- Click implementation address to see actual logic contract

USDC: Admin Roles Analysis

Step 3: Examine Admin Functions

In Read Contract tab, find:

- `owner()`: Returns owner address (Circle)
- `masterMinter()`: Can add/remove minters
- `pauser()`: Can pause all transfers
- `blacklister()`: Can freeze accounts

Task: Record Admin Addresses

1. Click `owner()` - note address
2. Click `masterMinter()` - note address
3. Click `pauser()` - note address
4. Click `blacklister()` - note address
5. Are they the same or different? (Multi-sig recommended)

Question: What are the implications of these centralized roles for DeFi protocols using USDC?

USDC: Blacklist Mechanism

Step 4: Test Blacklist Function

In Read Contract:

1. Scroll to `isBlacklisted(address account)`
2. Enter a random address - should return `false`
3. Enter known sanctioned address (e.g., Tornado Cash router): `0x...`
4. Check if `true` (Circle blacklists OFAC-sanctioned addresses)

Find Blacklist Events:

1. Click **Events** tab
2. Filter by `Blacklisted` event
3. Browse recent blacklist actions
4. Click transaction hash to see details

Discussion:

- How does blacklisting affect DeFi composability?
- What happens if a blacklisted address receives USDC?

USDC: Upgrade History

Step 5: Track Upgrade Events

1. In **Events** tab, filter by `Upgraded` event
2. Note timestamp and new implementation address for each upgrade
3. Click on implementation address to see new contract version

Example Upgrades (as of 2025):

- V1 → V2 (2018): Added permit() function
- V2 → V2.1 (2020): Gas optimizations
- V2.1 → V2.2 (2023): EIP-3009 transferWithAuthorization

Task:

- How much notice did Circle give before upgrades?
- Were upgrades done via timelock or direct?
- Compare source code differences between versions

USDC: Holder Distribution

Step 6: Analyze Token Holders

1. Click **Holders** tab on USDC token page
2. Review top 10 holders

Typical Distribution (as of 2025):

- Rank 1: Binance (centralized exchange) - approximately 10%
- Rank 2: Coinbase (centralized exchange) - approximately 8%
- Rank 3: Aave V3 (DeFi lending pool) - approximately 5%
- Rank 4-10: Mix of exchanges and DeFi protocols

Calculate Concentration:

- Top 10 holders: Sum percentages (typically 40-50%)
- Compare to DAI (next section)

Question: What does high concentration in exchanges vs DeFi protocols tell us about usage patterns?

Part 2: DAI Contract Analysis

Step 1: Locate DAI Contract

1. Search Etherscan: `0x6B175474E89094C44Da98b954EedeAC495271d0F`
2. Click **Contract** tab
3. Note: No proxy pattern (DAI uses migration instead)

Step 2: Examine Permit Function

- In Read Contract, scroll to `PERMIT_TYPEHASH()`
- This implements EIP-2612 gasless approvals
- Users sign approval off-chain, submit on-chain

Task: Compare to USDC

- Does USDC have `permit()`? (Check Read Contract)
- Which version added it? (Check upgrade history)

DAI: Minting Authorization

Step 3: Find Authorized Minters

1. In Read Contract, find `wards(address)` function
2. This mapping tracks authorized addresses (ward = authorized)
3. Enter Maker Vat address: `0x35D1b3F3D7966A1DFe207aa4514C12a259A0492B`
4. Should return `1` (authorized)

Observations:

- Only Maker Vat can mint DAI (via collateralized debt positions)
- No single admin can arbitrarily mint
- More decentralized than USDC's masterMinter model

Step 4: Check for Blacklist

- Search Read Contract for "blacklist" - none found
- DAI cannot freeze user addresses (tradeoff: regulatory risk)

DAI: Holder Distribution Comparison

Step 5: Analyze DAI Holders

1. Click **Holders** tab on DAI token page
2. Compare top 10 to USDC

Typical DAI Distribution:

- Higher percentage in DeFi protocols (Aave, Compound, Uniswap)
- Lower percentage in centralized exchanges
- Maker DSR (savings rate contract) often in top 5

Comparison Table:

Metric	USDC	DAI
Top 10 concentration	approximately 45%	approximately 40%
CEX holdings	approximately 25%	approximately 10%
DeFi holdings	approximately 20%	approximately 30%
Blacklist capability	Yes	No

Tracking Whale Activity

Step 6: Monitor Large Transfers

For USDC or DAI:

1. Go to token page, click **Transactions** tab
2. Filter by **ERC-20 Transfers**
3. Sort by **Quantity** (descending)

Identify Patterns:

- **Exchange Deposits:** User → Exchange (sell pressure?)
- **Exchange Withdrawals:** Exchange → User (potential accumulation)
- **DeFi Interactions:** User → Aave/Compound (borrowing/lending)
- **Whale Swaps:** Large DEX swaps (Uniswap, Curve)

Tools for Advanced Analysis:

- Etherscan Labels: Identify known addresses (exchanges, protocols)
- Whale Alert (Twitter bot): Real-time alerts for ¿1M transfers
- Nansen: Paid analytics for labeled wallet tracking

Key Takeaways

1. **Etherscan Analysis:** Read/Write Contract tabs enable complete contract interaction and transparency
2. **USDC Architecture:** Upgradeable proxy with centralized admin roles (owner, minter, pauser, blacklister)
3. **DAI Architecture:** Non-upgradeable with decentralized minting via Maker Vat, no blacklist capability
4. **Holder Distribution:** USDC favors CEX, DAI favors DeFi protocols, indicating different use cases
5. **Whale Tracking:** Large transfers reveal market sentiment (deposits = sell, withdrawals = accumulation)
6. **Custom Token Deployment:** OpenZeppelin contracts enable rapid development of feature-rich tokens
7. **Access Control:** Role-based permissions (MINTER, PAUSER) provide granular security

Discussion Questions

1. How would you design a stablecoin that balances decentralization (like DAI) with regulatory compliance (like USDC)?
2. What are the risks of relying on USDC in a DeFi protocol, given Circle's blacklist capability?
3. How can users verify that a token's claimed features (e.g., "no team mint") match the deployed contract?
4. What metrics would you use to assess whether a token is sufficiently decentralized?
5. How might regulators view tokens with pausable transfers vs non-pausable transfers?

Module B Summary: Ethereum and Smart Contracts

Key Concepts Covered:

- **L13:** Ethereum architecture, EVM, account model, Merkle Patricia Trie
- **L14:** Gas mechanics, EIP-1559, optimization strategies
- **L15:** Solidity fundamentals, data types, functions, events, inheritance
- **L16:** Remix IDE, contract deployment, Etherscan verification
- **L17:** ERC-20 standard, allowance mechanism, real-world tokens (USDC, DAI)
- **L18:** ERC-721 NFTs, ERC-1155 multi-token, metadata and IPFS
- **L19:** Token lifecycle, minting strategies, access control, upgradeability
- **L20:** Token analysis on Etherscan, deployment and verification

Next Module: DeFi Protocols (Uniswap, Aave, Compound)