TokenContractUpgradeWithPoS

고태건 김경모 김해인 배소정 이영환

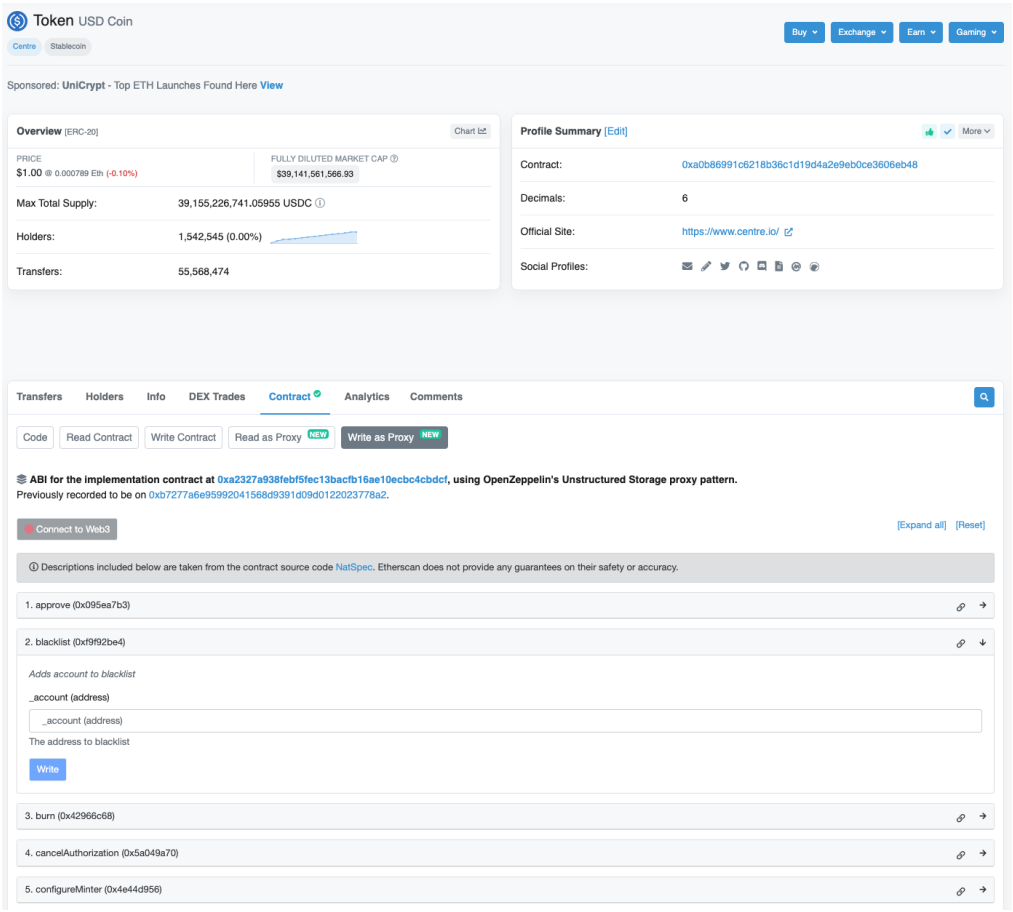
▼ 표제

지분 증명을 활용한 토큰 스마트 컨트랙트 업그레이드

▼ Motivation

▼ USDC 동결

- Tornado Cash : 암호화폐 익명화 서비스
- 미 재무부 해외자산통제국(OFAC)에서 제재
  - Circle은 USDC 동결
- 동결 방식
  - `blacklist` 함수를 통해 동결
    - https://etherscan.io/token/0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48#readProxyContract



    - Tornado Cash를 사용한 주소들을 트래킹하다가 권한 소유자가 실시간으로 등록

▼ Ankr BNB token(aBNBc) 해킹

- 해커가 개인키를 탈취해서 aBNBc를 10^13개 추가 민팅
- 이를 이용해 DEX pool을 다 drain

▼ Upgradable Smart Contract

- USDC
  - 토큰을 처음 만들 때, blacklist라는 함수가 이미 있었을까?
- aBNBc
  - 토큰 컨트랙트에 민팅 함수는 원래 막아놓음. 근데 어떻게 추가 민팅?
- Upgradable Smart Contract!
  - 스마트 컨트랙트는 업그레이드(바꿔치기)가 가능하다.
  - Proxy Pattern (EIP-1967)
- 가능한 것은 알고 있었는데 토큰에서도 사용하다니
- 심지어 어떤 체인들은 기본적으로 가능
  - Cosmos(migrate)와 Near(deploy)의 smart contract update
  - default까지는 아닌데 쉽게 가능 뒤에서 다시 설명

▼ Pros and Cons of Contract Upgrade

▼ Pros

- 배포 이후 취약점 수정 보완
  - 빠른 버그 수정 가능
- 로직 업그레이드(추가 기능)
  - 시간을 두고 기능 개발 가능

▼ Cons

- 탈중앙성과 보안성을 의미하는 코드 불변성에 위반
- 개발자를 믿어야 함(신뢰 문제)
- 해커나 악성 행위자의 공격이 용이(제어 권한 탈취 후 로직 변경 가능)
- 업그레이드를 위한 구조 등이 오히려 버그나 취약점을 만들 수 있음
- 개발 단에서 개발자를 안심하게 함(군기 상실)
- 어딧 받고 컨트랙트 업그레이드하면 무슨 소용

▼ Token Contract

- 별로 매력적이지 않은 Pros
  - 로직이 복잡하지도 않음
  - 기능 추가가 잦지도 않음
- 중요한 Cons

- 교환이나 가치 저장의 대상이 되는 토큰은 신뢰 문제가 최소화 되어야하고 공격으로부터도 최대한 안전할 필요가 있음

(Circle은 책임 집단이 명확하므로 조금 논외 …)

▼ How can we upgrade a smart contract?

- Contract Migration
  - 유저가 인터랙션 하는 컨트랙트 주소를 직접 변경
  - 원래 컨트랙트가 가지고 있던 유저의 상태를 그대로 다 복사해와야하는 단점도 있음
- Proxy Pattern

(Stores address of the logic contract)

  - 프록시 컨트랙트는 “상태”를 저장하고, 상호작용할 “로직 컨트랙트의 주소”를 가지고 있음
  - 컨트랙트 업그레이드 할때 로직 컨트랙트의 주소를 바꿈
  - 흔히 쓰이는 방식
- 업데이트 권한 : “Admin”
  - 일반적으로 하나의 키페어
  - 이더리움 공식 문서에서도 Multi-sig를 이용하거나 DAO member로 업그레이드 권한을 분산하는 방법 등을 짧게 제시해놓음

▼ TokenContractUpgradeWithPoS

- 토큰 컨트랙트는 더 높은 코드 안정성과 높은 신뢰(낮은 신뢰 비용)가 필요
  - 그러나 표준 업데이트 등에 의하여 업그레이드가 필요하긴 함
- PoS를 통해 Multi-sig와 DAO member 모두 인증 가능

▼ 예시

1. 개발자가 로직 컨트랙트를 디플로이
   a. 당연히 코드도 공개
2. 프록시 컨트랙트에서 로직 컨트랙트 변경을 신청
   a. 어떤 주소, 투표 기간 등을 입력
   b. Quorum 등은 프록시 컨트랙트에 이미 있어야 함

▼ 구현

▼ Ethereum

- 현재 EIP-1967을 이용하는 프록시 컨트랙트들은 solo admin을 설정하고 컨트랙트 업데이트 시 admin인지를 확인
- 컨트랙트상에서 admin인지 확인하는 부분을 투표로 구현

▼ Cosmos, Near Protocol 등 여타 체인

- Cosmos : Smart contract initation 할 때 owner field가 컨트랙트 어드민 권한
- Near protocol : 비슷한 방식
- Application Layer 내 구현이지만, contract 단에서 자연스러운 변경이 불가능
  - 체인 커뮤니티 내에서 최소 multi-sig는 기본으로 지원해줘야하지 않을까하는 개인적인 의견

▼ 우려점

- 버그에 대한 대처
  - 토큰 컨트랙트의 경우 빠른 대처보다 보안성과 신뢰성이 더 중요
- 거버넌스 참여 요인 필요
  - (USDC와 같이 가치 보증 단체가 중앙화된 경우에는 애초에 해당 로직이 필요하지 않음)
  - 거버넌스 리워드
  - 낮은 최소 정족수
    - = 낮은 보안성 …
  - **Dispute period**

▼ Reference

Upgrading smart contracts | ethereum.org

Smart contracts on Ethereum are self-executing programs that run in the Ethereum Virtual Machine (EVM). These programs are immutable by design, which prevents any updates to the business logic once the contract is deployed. While immutability is necessary for trustlessness,

https://ethereum.org/en/developers/docs/smart-contracts/upgrading/