



Open in app

Get started



Heuristic Wave

Follow

Nov 10, 2018 · 5 min read



Save



Ethernaut Telephone Problem — 이더넷 4단계 문제 해설

문제 해설에 들어가기 전, 이번 포스팅은 이더넷 내에서 콘솔창과 상호작용을 할 줄 알고 기본적인 리믹스 및 메타마스크 사용법이 숙지되어 있다는 가정 하에 해설을 진행합니다.

The Ethernaut

by



1



Heuristic Wave



Telephone Problem

이번 문제는 컨트랙트의 오너십을 가져오는 문제이다. 필자의 후반부 포스팅에서는 문제에서 요구하는 개념의 대한 설명을 먼저 설명하고 솔루션을 제시했지만, 이번에는 문제를 푸는 힘을 기르기 위해 필자가 고민하는 방식대로 문제해설을 하였다.

코드 분석

```
pragma solidity ^0.4.18;

contract Telephone {

    address public owner;

    function Telephone() public {
        owner = msg.sender;
    }

    function changeOwner(address _owner) public {
        if (tx.origin != msg.sender) {
            owner = _owner;
        }
    }
}
```

Telephone이라는 생성자를 보면 owner는 msg.sender라고 선언되어 있고, changeOwner함수는 주소값[_owner]를 매개변수로 넣어서 tx.origin과 msg.sender가 같지않다면 owner가 된다는 코드다. 즉, changeOwner의 함수를 호출하여 조건만 맞춘다면 오너십을 가져올 수 있다.

이 문제에서 출제자는 msg.sender와 tx.origin에 대한 개념을 여러분에게 꼭 알려주고 싶어 이번 문제를 출제했다. 두개의 차이를 정확하게 알고있다면 바로 이번문제를 해결할 수 있다.

본격적으 설명을 하기전에, 본인 스스로 이유을 찾는다면 실력향상에 더 많은 도움이 된다. 사실 이번문제의 시사점은 솔리디티 도큐먼트의 tx.origin의 위험성을 알리는 부분에 실려있는데, 필자의 게시물보다 도큐먼트를 혼자 공부하면서 문제와 씨름하는 것도 좋은 방법이다.

필자와 본격적으로 문제를 풀기전에 시중에 판매되는 이더리움 교재에서 나와있는 설명을 보고가자.

msg.sender

메시지를 호출한 송신자의 주소를 반환

tx.origin

트랜잭션의 송신자를 반환

위 설명 만으로는 msg.sender와 tx.origin의 차이가 감이 잘 잡히지 않을 것이다. 공통점으로는 둘다 주소값을 반환한다는 것이 있다.

이 문제의 구조를 파악하면 해결책을 찾기 수월하니, remix에서 문제로 발급받은 instance address로 telephone 컨트랙트를 호출하여 확인해보자.

필자가 발급받은 컨트랙트를 로딩하여(RED) owner함수로 오너를 확인하면(BLUE) **Level address**가 오너인 것을 알 수 있다. 이말은 즉, Level address (EOA)주소가 msg.sender라는 것이다. 우리가 알고싶어하는 tx.origin도 트랜잭션의 송신자를 반환하는 것이니까 EOA가 반환될 수도 있고 CA가 반환될수 있다고도 생각할 수 있다. (트랜잭션은 계정, 컨트랙트 모두 만들수 있으니까)

(이즈음에서 필자가 약간의 스포일러를 하겠다) 출제자가 우리에게 알려주고 싶은것은 바로 이 차이점이다. msg.sender는 EOA가 될수도 있고 CA가 될수도 있다. 그러나,

tx.origin은 EOA주소만이 가능하다. 여기까지 우리는 문제를 풀 수 있는 실마리를 얻었다. 그러나 아직 그 둘의 완벽한 차이를 알게된 것은 아니다.

실마리를 통해서 우리는 이런 생각을 할 수 있다. tx.origin은 EOA주소만이 가능하고 msg.sender는 CA계정도 가능하다면, CA로부터 해당 컨트랙트(출제문제)를 호출한다면 주어진 조건(tx.origin != msg.sender)을 통과할 수 있을 것이다.

위 아이디어를 적용한 공격자의 코드는 아래와 같다.

```
contract Attacker {
    Telephone telephone;

    function Attacker(address _contract) {
        telephone = Telephone(_contract);
    }

    function attack() public {
        telephone.changeOwner(msg.sender);
    }
}
```

공격자 컨트랙트의 생성자에 문제로 주어진 telephone 컨트랙트를 넣을수 있게 만들고, attack 함수에서 telephone 컨트랙트의 changeOwner함수를 호출하도록하면 이번 문제를 해결 할 수 있다.

필자의 경우 스택오버플로우를 뒤져가며 telephone 문제를 약 7월경에 풀었는데, 8월에 포스팅된 (*이더넷 후반부에서 많은 도움을 받은*) [Nicole Zhu님의 블로그](#)에 tx.origin과 msg.sender에 대한 가장 설명이 잘 되어있는 그림이 있다. 그림을 보면서 자신이 맞게 이해했는지 확인해보자!

그럼, 다음번에는 5단계 Token에서 만나요!

Ethernaut Token Problem — 이더넷 5단계 문제 해설

문제 해설에 들어가기 전, 이번 포스팅은 이더넷 내에서 콘솔창과 상호 작용을 할 줄 알고 기본적인 리믹스 및 메타마스크 사용법이 숙지되어

medium.com

ne Ethernaut

euristic Wav



[About](#) [Help](#) [Terms](#) [Privacy](#)

Get the Medium app

