Heuristic Wave Follow
Oct 7, 2018 · 5 min read

Ethernaut Locked Problem — 이더넛 17단계 문제 해설

문제 해설에 들어가기 전, 이더넛 내에서 콘솔창과 상호작용을 할 줄 알고 기본적인 리믹스 및 메타마스크 사용법이 숙지되어 있다는 가정 하에 해설을 진행합니다. 필자의 풀이 방법이 절대적은 풀이 방법은 아니므로 이 점 참고하시기 바랍니다.

The Ethernaut

by

Heuristic Wave

. . .

Locked Problem

이번에도 역시 주어진 조건을 읽어보자. 필자의 초월해석이 담겨 있기 때문에, 원문을 직접 읽는 것이 제일 좋다.

Name Registrar는 잠겨 있으며, 새로운 이름을 등록할 수 없습니다. Registrar를 해제하려면 이번 단계를 넘어서야 한다.

- storage가 어떻게 작동하는지 이해하자
- 로컬 변수의 기본 storage 유형을 이해하자
- storage와 memory의 차이점을 이해하자

이번 문제는 리믹스에 주어진 코드를 붙여넣으면 강력한 힌트가 나온다. 힌트를 만나기전 코드를 보면서 어떻게 해결할지 생각해보자!

코드 분석

Locked.sol

```
pragma solidity ^0.4.23;

// A Locked Name Registrar
contract Locked {
    bool public unlocked = false;  // registrar locked, no
name updates

    struct NameRecord { // map hashes to addresses
        bytes32 name; //
        address mappedAddress;
    }
    // records who registered names
    mapping(address => NameRecord) public
registeredNameRecord;
    mapping(bytes32 => address) public resolve; // resolves
hashes to addresses

    function register(bytes32 _name, address _mappedAddress)
public {
        // set up the new NameRecord
        NameRecord newRecord;
        newRecord.name = _name;
        newRecord.mappedAddress = _mappedAddress;

        resolve[_name] = _mappedAddress;
        registeredNameRecord[msg.sender] = newRecord;

        require(unlocked); // only allow registrations if
contract is unlocked
    }
}
```

주어진 힌트(어떻게 storage가 작동하는가)를 고민하며 storage의 슬롯에 어떤 데이터가 담길지 차례로 보자! 우선, 첫 번째 슬롯에 bool 형태의 false 값이 들어온다. 그다음 NameRecord 구조체가 들어올지 코드를 확인하다 보면 register 함수 안에서 사용된

것이 보인다. 이미 리믹스에서 경고문으로 `storage` 혹은 `memory`라는 키워드를 선언하라고 한다. 기본적으로 키워드가 없다면, 디폴트 값은 storage이다.

다시 말해, NameRecord의 name과 mappedAddress는 storage에 저장이된다.

storage와 memory 키워드의 차이를 모른다면 아래 게시물에서 공부를 하고 문제를 해결해 나가자!

> storage VS memory(영문)
>
> Storage, Memory 구조체 내부의 배열을 초기화 하는 방법(한글)

왜 솔리디티 컴파일러는 경고문으로 storage / memory를 사용하라고 말하는 것일까?

바로 아래와 같은 이슈(스토리지 할당 공격) 때문에 버전업이 되면서 엄격하게 체크를 하기 시작했다. 변수를 어디에 저장할지 명확하게 지정하지 않는다면 0.4.5 버전 이후로는 아예 컴파일 불가하다고 한다.

스토리지 할당 공격에 대하여 간략히 설명을 하자면, 어떠한 변수도 storage에 저장이 되고 이후에 사용되는 구조체도 storage에 저장이 된다면 구조체에 관한 정보가 이미 전역으로 선언한 변수의 슬롯위에 overwrite을 한다는 내용이다.

문제 풀이 과정

위 문제를 우리 문제에 적용하면 slot 0에는 false 값에 해당하는 정보가 들어 있다.

```
false :
0x0000000000000000000000000000000000000000000000000000000000
000000
```

그러나, NameRecord 구조체를 사용하면서 구조체의 정보들이 slot 0 을 overwrite 할 수 있는데, 이때 true에 해당하는 값을 매개변수로 넣고 저장하면 false의 상태를 true로 바꿀 수 있다.

왼쪽 : register 함수에 true에 해당하는 값과 나의 EOA주소를 넣고 트랜잭션을 발생시킴

오른쪽 : 데이터가 overwriter 되며 지정한 값들이 슬롯에 들어가 있는 것을 확인 할 수 있다.

솔리디티가 버전업을 거듭하면서 앞으로 사용할 user들은 이번 문제와 같은 overwrite로 인한 데이터 손실문제를 겪을 경험이 줄어들 것이다. 그러나, 이번 문제에서 hint로 언급한 3가지 시사점은 꼭 익혀두자!

그럼, 다음번에는 18단계 Recovery에서 만나요!