dasp top 10

1. Reentrancy (재진입)

**function** **withdraw**(uint \_amount) {

require(balances[msg.sender] >= \_amount);

msg.sender.call.value(\_amount)();

balances[msg.sender] -= \_amount;

}

(Msg.sender.call.value() 는 무슨의미일까?)

Reentrancy가 발생하면 무한으로 발생하는지 아니면 1회만 작동할거 3회 작동한다던지?

동기 비동기는 뭘까? -> 비동기는 함수가 실행중에 다른함수를 실행하면 멈추는게 비동기. 다른함수 실행하면서 자기다음코드로 넘어가는것은 동기.

예시

Withdraw()함수로 외부주소의 잔고를 추적하고 자금을 인출할수있음.

악의적인 계약은 withdraw()를 사용하여 전체 잔고를 인출하려고함.

피해를 받는 계약은 악의적인 계약에게 이더를전송하기위해

call.value(amount)() 라는 low level 함수를 악의적인 계약의 잔고를업데이트 하기 전에 사용함

악의적인 계약은 payable fallback() 함수로 자금을 받고 피해받는계약의 withdraw()함수를 다시 호출한다.

두번째 실행은 자금의 전송을 일으킴. 그러나 악의적인 계약의 잔고는 첫번째 인출에서 업데이트 되지 않음. 결과적으로 악의적인 계약은 두번째에서 모든 잔고를 인출해감.

1. Access control
2. **function** **initContract**() **public** {
3. owner = msg.sender;
4. }

함수를 호출한 사람이 주인으로 저장되게 만들어놨는데 public 이라 아무나 다쓸수있었음. 그래서 librariy 계약에 정의하는거로 변경함. 사용자는 delegateCall을 통해 libarary의 함수를 호출하여 초기화 한다. 근데 이 예제에서는 이 함수가 이미 초기화된것인지 확인하지않는다는것, 게다가 스마트컨트랙트라서 library를 초기화 하거나 없앨수 있다는것..

3.Arithmetic

3번예제는 if문을 평생 충족하지못하는것 아닌가?

4번 예제에서 왜 255이상의 숫자에 도달하지못하고 가스가 떨어지면 멈추는것인지?

**function** **withdraw**(uint \_amount) {

require(balances[msg.sender] - \_amount > 0);

msg.sender.transfer(\_amount);

balances[msg.sender] -= \_amount;

}

쉽게말해 오버플로 언더플로

Withdraw 함수는 잔고가 남아있는만큼 인출해주는 함수이다.

공격자가 피해자의 잔고보다 큰수를 인출하려고 시도함.

그럼 withdraw가 또 인출해주고 잔고는 -가 되므로 언더플로우로 어마어마한잔고가 생기게됨.

**function** **popArrayOfThings**() {

require(arrayOfThings.length >= 0);

arrayOfThings.length--;

}

이 예제는 배열의 길이가 uint로 표현된다는 사실로 인해 발생되는 off-by-one error이다.

function **votes**(**uint** postId, **uint** upvote, **uint** downvotes) {

**if** (upvote - downvote < 0) {

deletePost(postId)

}

}

이 예제는 uint 두개의 연산값이 uint인 첫번째 예의 변형이다.

**for** (**var** i = 0; i < somethingLarge; i ++) {

// ...

}

이 예제는 곧 사라질 var에 대한 예제인데 var은 가장 작은 자료형으로 변형되기때문에 uint8로 변경되고 만약 255보다 큰 숫자를 반복하길 원했다면 거기까지 도달할수가 없고 가스가 떨어지면 멈출것이다.

4.Unchecked Low Level Calls

5.Denial of Services

6.Bad Randomness

7.Front Running

8.Time Manipulation

9.Short Addresses

10. Unknown Unknowns