1. Proxy.sol

```
myth analyze /home/profganeshteam/Tools/Contracts/Proxy.sol
profganeshteam@SmartContract:~/Tools/FolderForMythril$
myth analyze /home/profganeshteam/Tools/Contracts/Proxy.sol
==== Delegatecall Proxy To User-Supplied Address ====
SWC ID: 112
Severity: Medium
Contract: Proxy
Function name: constructor
PC address: 412
Estimated Gas Usage: 6522 - 66958
The contract delegates execution to another contract with a user-supplied address.
The smart contract delegates execution to a user-supplied address. Note that callers can execute
arbitrary contracts and that the callee contract can access the storage of the calling contract.
--------------------
In file: /home/profganeshteam/Tools/Contracts/Proxy.sol:11

contractLogic.delegatecall(constructData)

--------------------
Initial State:

Account: [CREATOR], balance: 0x1, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0
```

2. RToken.sol

```
myth analyze /home/profganeshteam/Tools/Contracts/RToken.sol
profganeshteam@SmartContract:~/Tools/FolderForMythril$
myth analyze /home/profganeshteam/Tools/Contracts/RToken.sol
The analysis was completed successfully. No issues were detected.
```

3. MainframeStake.sol

```
myth analyze /home/profganeshteam/Tools/Contracts/MainframeStake.sol
profganeshteam@SmartContract:~/Tools/FolderForMythril$
myth analyze /home/profganeshteam/Tools/Contracts/MainframeStake.sol
```

No result with long time execution.

4. tokensalechallenge.sol

```
myth analyze /home/profganeshteam/Tools/Contracts/tokensalechallenge.sol
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/tokensalechallenge.sol --solv 0.4.21
==== Integer Overflow ====
SWC ID: 101
Severity: High
Contract: TokenSaleChallenge
Function name: buy(uint256)
PC address: 360
Estimated Gas Usage: 5747 - 26032
The binary multiplication can overflow.
The operands of the multiplication operation are not sufficiently constrained. The multiplication
could therefore result in an integer overflow. Prevent the overflow by checking inputs or ensure
sure that the overflow is caught by an assertion.
--------------------
In file: /home/profganeshteam/VerificationTool/Contracts/tokensalechallenge.sol:21
R_TOKEN);
            balanceO
--------------------
Initial State:
Account: [CREATOR], balance: 0x40080000040000000, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x400000010000, nonce:0, storage:{}
Transaction Sequence:
Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0xde0b6b3a7640000
Caller: [ATTACKER], function: buy(uint256), txdata: 0xd96a094a80, value: 0x0
==== Integer Overflow ====
SWC ID: 101
Severity: High
Contract: TokenSaleChallenge
Function name: buy(uint256)
PC address: 442
Estimated Gas Usage: 5747 - 26032
The binary addition can overflow.
The operands of the addition operation are not sufficiently constrained. The addition could
therefore result in an integer overflow. Prevent the overflow by checking inputs or ensure sure
that the overflow is caught by an assertion.
--------------------
In file: /home/profganeshteam/VerificationTool/Contracts/tokensalechallenge.sol:23

+= numTokens;
      }
```

```
        function
--------------------
Initial State:
Account: [CREATOR], balance: 0x20020be0000000000, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x176400870230c0050, nonce:0, storage:{}
Transaction Sequence:
Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0xde0b6b3a7640000
Caller: [SOMEGUY], function: buy(uint256), txdata: 0xd96a094a40, value: 0x0
Caller: [SOMEGUY], function: buy(uint256), txdata: 0xd96a094ac0, value: 0x0
```

5. StakeInterface.sol

```
myth analyze /home/profganeshteam/Tools/Contracts/StakeInterface.sol
profganeshteam@SmartContract:~/Tools/FolderForMythril$ myth analyze
/home/profganeshteam/Tools/Contracts/StakeInterface.sol
mythril.interfaces.cli [ERROR]: input files do not contain any valid contracts
```

6. RTokenStructs.sol

```
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/RTokenStructs.sol
The analysis was completed successfully. No issues were detected.
```

7. RTokenStorage.sol

```
myth analyze /home/profganeshteam/Tools/Contracts/RTokenStorage.sol
mythril.interfaces.cli [ERROR]: input files do not contain any valid contracts
```

8. ReentrancyGuard.sol

```
myth analyze /home/profganeshteam/Tools/Contracts/ReentrancyGuard.sol
mythril.interfaces.cli [ERROR]: input files do not contain any valid contracts
```

9. Proxiable.sol

```
myth analyze /home/profganeshteam/Tools/Contracts/Proxiable.sol
profganeshteam@SmartContract:~/Tools/FolderForMythril$ myth analyze
/home/profganeshteam/Tools/Contracts/Proxiable.sol
The analysis was completed successfully. No issues were detected.
```

10. Ownable1.sol

```
myth analyze /home/profganeshteam/Tools/Contracts/Ownable1.sol
profganeshteam@SmartContract:~/Tools/FolderForMythril$ myth analyze
/home/profganeshteam/Tools/Contracts/Ownable1.sol
mythril.interfaces.cli [ERROR]: input files do not contain any valid contracts
```

11. modifier_reentrancy.sol

```
myth analyze /home/profganeshteam/Tools/Contracts/modifier_reentrancy.sol
profganeshteam@SmartContract:~/Tools/FolderForMythril$ myth analyze /home/profganeshteam/Tools/Contracts/modifier_reentrancy.sol
The analysis was completed successfully. No issues were detected.
```

12. MainframeTokenDistribution.sol

```
myth analyze /home/profganeshteam/Tools/Contracts/MainframeTokenDistribution.sol
```

no result with long time execution

13. MainframeToken.sol

```
myth analyze /home/profganeshteam/Tools/Contracts/MainframeToken.sol
```

no result with long time execution

14. Lockdrop.sol

```
profganeshteam@SmartContract:~$ myth analyze /home/profganeshteam/Tools/UpdatedBenchmarkContracts/Lockdrop.sol
==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Lock
Function name: fallback
PC address: 16
Estimated Gas Usage: 2029 - 36124
A control flow decision is made based on a predictable variable.
The block.timestamp environment variable is used in to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables for random number generation or to make critical control flow decisions.
--------------------
In file: /home/profganeshteam/Tools/UpdatedBenchmarkContracts/Lockdrop.sol:19

 0) }
                case

--------------------
Initial State:

Account: [CREATOR], balance: 0x1, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}
```

```
Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0
Caller: [CREATOR], function: unknown, txdata: 0x, value: 0x0

==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Lock
Function name: fallback
PC address: 23
Estimated Gas Usage: 2029 - 36124
A control flow decision is made based on a predictable variable.
The block.timestamp environment variable is used in to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables for random number generation or to make critical control flow decisions.
--------------------
In file: /home/profganeshteam/Tools/UpdatedBenchmarkContracts/Lockdrop.sol:21

        switch call(gas, sload(0x00), balance(address), 0, 0, 0, 0)
                    case 0 { revert(0, 0) }
                }
            }
        }
}

co

--------------------
Initial State:

Account: [CREATOR], balance: 0x1, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0
Caller: [CREATOR], function: unknown, txdata: 0x, value: 0x0

==== External Call To User-Supplied Address ====
SWC ID: 107
Severity: Medium
Contract: Lock
Function name: fallback
PC address: 45
Estimated Gas Usage: 2029 - 36124
A call to a user-supplied address is executed.
The callee address of an external message call can be set by the caller. Note that the callee can contain arbitrary code and may re-enter any function in this contract. Review the business logic carefully to prevent averse effects on the contract state.
--------------------
In file: /home/profganeshteam/Tools/UpdatedBenchmarkContracts/Lockdrop.sol:21

), balance(address), 0, 0, 0, 0)
                  cas

--------------------
Initial State:

Account: [CREATOR], balance: 0x2, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x421410c04020ffffb, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x52b864a29610ffffc, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0
Caller: [ATTACKER], function: unknown, txdata: 0x, value: 0x0
Caller: [ATTACKER], function: unknown, txdata: 0x, value: 0x0

==== Unprotected Ether Withdrawal ====
SWC ID: 105
Severity: High
Contract: Lock
Function name: fallback
PC address: 45
Estimated Gas Usage: 2029 - 36124
Anyone can withdraw ETH from the contract account.
Arbitrary senders other than the contract creator can withdraw ETH from the contract account without previously having sent an equivalent amount of ETH to it. This is likely to be a vulnerability.
--------------------
In file: /home/profganeshteam/Tools/UpdatedBenchmarkContracts/Lockdrop.sol:21

), balance(address), 0, 0, 0, 0)
```

```
                    cas

--------------------
Initial State:

Account: [CREATOR], balance: 0x2, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x21810c04400c0000, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x56b45422123080003, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0
Caller: [ATTACKER], function: unknown, txdata: 0x, value: 0x0

==== Integer Overflow ====
SWC ID: 101
Severity: High
Contract: Lockdrop
Function name: constructor
PC address: 81
Estimated Gas Usage: 10616 - 55600
The binary addition can overflow.
The operands of the addition operation are not sufficiently constrained. The addition could therefore result in an integer overflow. Prevent the overflow by checking inputs or ensure sure that the overflow is caught by an assertion.
--------------------
In file: /home/profganeshteam/Tools/UpdatedBenchmarkContracts/Lockdrop.sol:47

*
      * @dev              Locks u

--------------------
Initial State:

Account: [CREATOR], balance: 0x1, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0

==== Dependence on predictable environment variable ====
SWC ID: 116
```

Severity: Low
Contract: Lockdrop
Function name: signal(address,uint32,bytes)
PC address: 2847
Estimated Gas Usage: 3346 - 7511
A control flow decision is made based on a predictable variable.
The block.timestamp environment variable is used in to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables for random number generation or to make critical control flow decisions.

In file: /home/profganeshteam/Tools/UpdatedBenchmarkContracts/Lockdrop.sol:101

```
rop has not ended
     */
    m
```

Initial State:

Account: [CREATOR], balance: 0x40400000020, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x422000000000, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0
Caller: [CREATOR], function: signal(address,uint32,bytes), txdata: 0x911fab44000000000000000000000000efd31549c348040a755a0600040045bc4c01000000000
00000000000000000000000000000000000000000000000000001000000000000000000000
00000000000000000000000000000000000000001e, value: 0x0

==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Lockdrop
Function name: signal(address,uint32,bytes)
PC address: 2862
Estimated Gas Usage: 3346 - 7511
A control flow decision is made based on a predictable variable.
The block.timestamp environment variable is used in to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of

```
earlier blocks. Don't use any of those environment variables for random number generation or to
make critical control flow decisions.
--------------------
In file: /home/profganeshteam/Tools/UpdatedBenchmarkContracts/Lockdrop.sol:109

ss from a normal address and

--------------------
Initial State:

Account: [CREATOR], balance: 0x40400000020, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x422000000000, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0
Caller: [CREATOR], function: signal(address,uint32,bytes), txdata:
0x911fab44000000000000000000000000efd31549c348040a755a0600040045bc4c01000000000
000000000000000000000000000000000000000000000000010000000000000000000000
0000000000000000000000000000000000000001e, value: 0x0

==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Lockdrop
Function name: lock(uint8,bytes,bool)
PC address: 3272
Estimated Gas Usage: 46021 - 97379
A control flow decision is made based on a predictable variable.
The block.timestamp environment variable is used in to determine a control flow decision. Note
that the values of variables like coinbase, gaslimit, block number and timestamp are predictable
and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of
earlier blocks. Don't use any of those environment variables for random number generation or to
make critical control flow decisions.
--------------------
In file: /home/profganeshteam/Tools/UpdatedBenchmarkContracts/Lockdrop.sol:101

rop has not ended
         */
       m

--------------------
Initial State:
```

```
Account: [CREATOR], balance: 0x3fffffc03ffe3f3a, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x1, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0
Caller: [SOMEGUY], function: lock(uint8,bytes,bool), txdata:
0xa40d3060000000000000000000000000000000000000000000000000000000000000020000
000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Lockdrop
Function name: lock(uint8,bytes,bool)
PC address: 3287
Estimated Gas Usage: 46021 - 97379
A control flow decision is made based on a predictable variable.
The block.timestamp environment variable is used in to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables for random number generation or to make critical control flow decisions.
--------------------
In file: /home/profganeshteam/Tools/UpdatedBenchmarkContracts/Lockdrop.sol:109

ss from a normal address and


--------------------
Initial State:

Account: [CREATOR], balance: 0x3fffffc03ffe3f3a, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x1, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0
Caller: [SOMEGUY], function: lock(uint8,bytes,bool), txdata:
0xa40d3060000000000000000000000000000000000000000000000000000000000000020000
000000000000000000000000000000000000000000000000000000000000000000000000000000
```

```
0000000000000000000000000000000000000000000000, value: 0x0

==== Exception State ====
SWC ID: 110
Severity: Low
Contract: Lockdrop
Function name: lock(uint8,bytes,bool)
PC address: 3456
Estimated Gas Usage: 44359 - 92211
A reachable exception has been detected.
It is possible to trigger an exception (opcode 0xfe). Exceptions can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. Note that explicit `assert()` should only be used to check invariants. Use `require()` for regular input checking.
--------------------
In file: /home/profganeshteam/Tools/UpdatedBenchmarkContracts/Lockdrop.sol:66

ked(owner, eth, lockAddr, term, edgewareAddr,

--------------------
Initial State:

Account: [CREATOR], balance: 0x0, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x2005000c6b0083f44, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0
Caller: [SOMEGUY], function: lock(uint8,bytes,bool), txdata:
0xa40d306000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000200000000000000000000
0000000000000000000000000000000000000000000000, value: 0x0

==== Exception State ====
SWC ID: 110
Severity: Low
Contract: Lockdrop
Function name: lock(uint8,bytes,bool)
PC address: 3716
Estimated Gas Usage: 1447 - 1542
A reachable exception has been detected.
It is possible to trigger an exception (opcode 0xfe). Exceptions can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. Note that explicit `assert()` should only be used to check invariants. Use `require()` for regular input checking.
```

```
--------------------
In file: /home/profganeshteam/Tools/UpdatedBenchmarkContracts/Lockdrop.sol:86

ow + 183 days;



--------------------
Initial State:

Account: [CREATOR], balance: 0x0, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x800200040060c00, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0
Caller:          [SOMEGUY],          function:          lock(uint8,bytes,bool),          txdata:
0xa40d3060000000000000000000000000000000000000000000000000000000000000000220000
00000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000, value: 0x0
```

15. LibraryLock.sol

```
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/LibraryLock.sol
mythril.interfaces.cli [ERROR]: input files do not contain any valid contracts
```

16. IRTokenAdmin.sol

```
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/IRTokenAdmin.sol
mythril.interfaces.cli [ERROR]: input files do not contain any valid contracts
```

17. IRToken.sol

```
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/IRToken.sol
mythril.interfaces.cli [ERROR]: input files do not contain any valid contracts
```

18. IAllocationStrategy.sol

```
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/IAllocationStrategy.sol
mythril.interfaces.cli [ERROR]: input files do not contain any valid contracts
```

19. guess_the_random_number.sol

```
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/guess_the_random_number.sol --solv 0.4.21
==== Unprotected Ether Withdrawal ====
SWC ID: 105
Severity: High
Contract: GuessTheRandomNumberChallenge
Function name: guess(uint8)
PC address: 259
Estimated Gas Usage: 1451 - 36062
Anyone can withdraw ETH from the contract account.
Arbitrary senders other than the contract creator can withdraw ETH from the contract account
without previously having sent an equivalent amount of ETH to it. This is likely to be a
vulnerability.
--------------------
In file: /home/profganeshteam/VerificationTool/Contracts/guess_the_random_number.sol:24

ther);
            }
        }
}
--------------------
Initial State:
Account: [CREATOR], balance: 0x21c1040142040002, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x42001040440080000, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x200600080103fbfb, nonce:0, storage:{}
Transaction Sequence:
Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0xde0b6b3a7640000
Caller: [ATTACKER], function: guess(uint8), txdata: 0x4ba4c16b, value: 0xde0b6b3a7640000
```

20. dos_simple.sol

```
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/dos_simple.sol --solv 0.4.25
The analysis was completed successfully. No issues were detected.
```

21. dos_number.sol

(Because it ran a long time without any output, so I stopped it manually . That's why there is a reminder of "Keyboard Interrupt" in the following.)

```
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/dos_number.sol --solv 0.4.25
^Cmythril.mythril.mythril_analyzer [CRITICAL]: Keyboard Interrupt
The analysis was completed successfully. No issues were detected.
```

22. dos_address.sol

```
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/dos_address.sol
The analysis was completed successfully. No issues were detected.
```

23. CompoundAllocationStrategy.sol

```
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/CompoundAllocationStrategy.sol
mythril.interfaces.cli [ERROR]: input files do not contain any valid contracts
```

24. Casino.sol

```
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/Casino.sol
```

No result with long time execution.

25. send_loop.sol

```
profganeshteam@SCV:~/VerificationTool$ myth analyze
/home/profganeshteam/VerificationTool/Contracts/send_loop.sol --solv 0.4.24
==== Multiple Calls in a Single Transaction ====
SWC ID: 113
Severity: Low
Contract: Refunder
Function name: refundAll()
PC address: 431
Estimated Gas Usage: 7767 - 78314
Multiple calls are executed in the same transaction.
This call is executed after a previous call in the same transaction. Try to isolate each call, transfer
or send into its own transaction.
--------------------
In file: /home/profganeshteam/VerificationTool/Contracts/send_loop.sol:16
[x].send(refunds[refundAddresses[x]])); // doubly ba
--------------------
Initial State:
Account: [CREATOR], balance: 0x449000c00440c8830, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x40000001, nonce:0, storage:{}
Transaction Sequence:
Caller: [CREATOR], data: [CONTRACT_CREATION], value: 0x0
Caller: [SOMEGUY], function: refundAll(), txdata: 0x38e771ab, value: 0x0
```