패러티 멀티시그 컨트렉트 해킹 심층분석

in DLab

2017. 7. 26

Contents

1. Overview by 정순형(Kevin)
2. Understanding Library, Link, Using A For B 디렉티브 by 김휘경(Zoe)
3. Proxy Pattern - Upgradable Contract Design by 노언석(Andy)
4. Weakness of Parity MultiSig contract by 이진용(Jay)
5. ~~Exploit Parity MultiSig Demo by 정순형(Kevin) - 추후 게시~~
6. Conclusion
7. Presenter Info

1. 사건 개요(1/6 - 뉴스와 개요)

- 해커는 150,000 이더 탈취(edgeless, swarm city, aeternity)
- 화이트 해커 over 300,000 이더 보존
- https://www.bleepingcomputer.com/news/security/hacker-uses-parity-wallet-vulnerability-to-steal-30-million-worth-of-ethereum/

- https://www.hackread.com/hackers-steal-32-million-in-ethereum-from-3-multisig-wallets/

1.Overview (2/6 - 해커 주소)

- https://etherscan.io/address/0xb3764761e297d6f121e79c32a65829cd1ddb4d32

1. Overview (3/6 - 화이드햇그룹)

- https://etherscan.io/address/0x1dba1131000664b884a1ba238464159892252d3a

1.Overview (4/6 - 스웜시티)

- 44,055 ETH
- 공식성명 : https://press.swarm.city/swarm-city-keyholders-upgrade-to-a-new-multisig-wallet-contract-f2649fb289a8
- 트렌젝션 : https://etherscan.io/tx/0x97f7662322d56e1c54bd1bab39bccf98bc736fcb9c7e61640e6ff1f633637d38

1.Overview (5/6 - aeternity)

- Around 82,000 ETH (of 102,000 ETH) sent
- 공식성명 : https://blog.aeternity.com/parity-multisig-wallet-hack-47cc507d964d
- 컨트렉트 : https://etherscan.io/address/0xbec591de75b8699a3ba52f073428822d0bfc0d7e#internaltx

1. Overview (6/6 - 엣지리스)

- 26 793 ETH
- 공식답변 : https://medium.com/@tomasdraksas/edgeless-response-to-parity-hack-3e35e20ba85c
- 트렌젝션 : https://etherscan.io/tx/0x0e0d16475d2ac6a4802437a35a21776e5c9b681a77fef1693b0badbb6afdb083

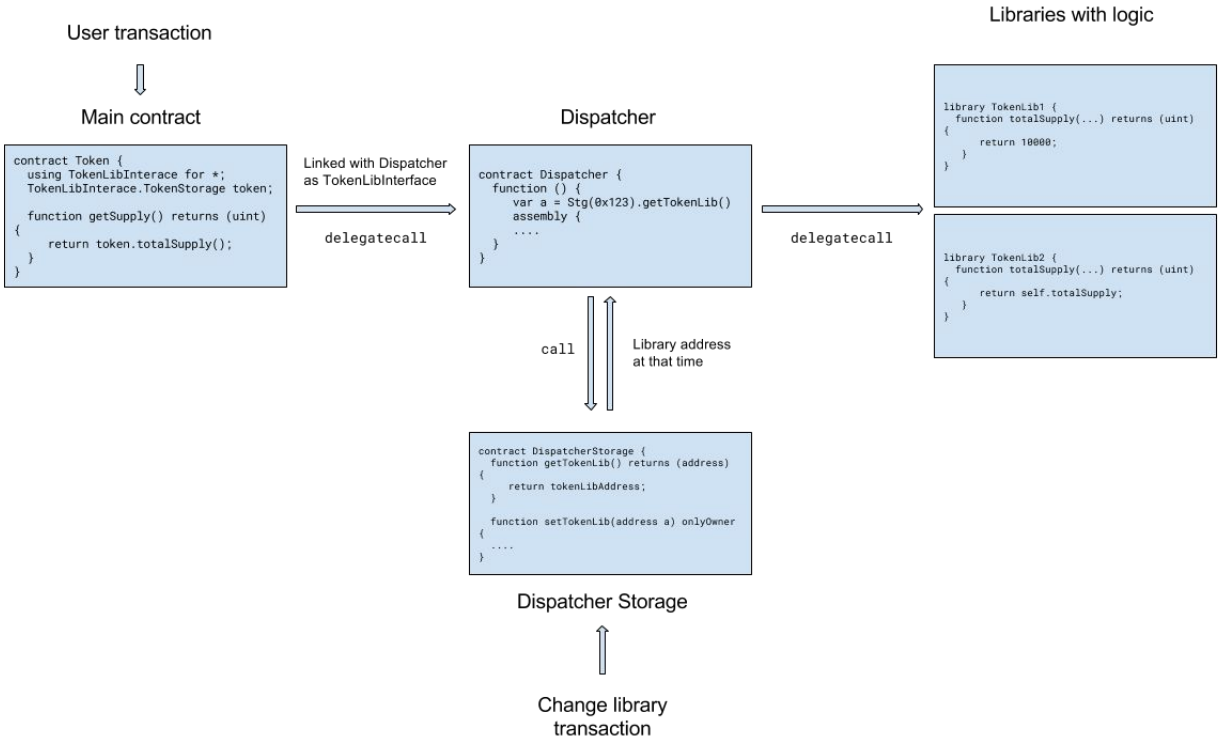
2. Libraries

```
pragma solidity ^0.4.11;

library Set {
  // We define a new struct datatype that will be used
to
  // hold its data in the calling contract.
  struct Data { mapping(uint => bool) flags; }

  // Note that the first parameter is of type "storage
  // reference" and thus only its storage address and
not
  // its contents is passed as part of the call.  This
is a
  // special feature of library functions.  It is
idiomatic
  // to call the first parameter 'self', if the function
can
  // be seen as a method of that object.
  function insert(Data storage self, uint value)
      returns (bool)
  {
      if (self.flags[value])
          return false; // already there
      self.flags[value] = true;
      return true;
  }
  function remove(Data storage self, uint value)
      returns (bool)
  {
      if (!self.flags[value])
          return false; // not there
      self.flags[value] = false;
      return true;
  }

  function contains(Data storage self, uint value)
      returns (bool)
  {
      return self.flags[value];
  }
}


contract C {
    Set.Data knownValues;

    function register(uint value) {
        // The library functions can be called without a
        // specific instance of the library, since the
        // "instance" will be the current contract.
        require(Set.insert(knownValues, value));
    }
    // In this contract, we can also directly access
knownValues.flags, if we want.
}
```

2. Using A For B

```
pragma solidity ^0.4.11;

// This is the same code as before, just without
comments
library Set {
  struct Data { mapping(uint => bool) flags; }

  function insert(Data storage self, uint value)
      returns (bool)
  {
      if (self.flags[value])
        return false; // already there
      self.flags[value] = true;
      return true;
  }

  function remove(Data storage self, uint value)
      returns (bool)
  {
      if (!self.flags[value])
          return false; // not there
      self.flags[value] = false;
      return true;
  }

  function contains(Data storage self, uint value)
      returns (bool)
  {
      return self.flags[value];
  }
}


contract C {
    using Set for Set.Data; // this is the crucial change
    Set.Data knownValues;

    function register(uint value) {
     // Here, all variables of type Set.Data have
     // corresponding member functions.
     // The following function call is identical to
     // Set.insert(knownValues, value)
        require(knownValues.insert(value));
    }
}
```

3. Proxy pattern



3. Proxy pattern

/lib/test/test.js

```
contract('TestProxyLibrary', () => {
  describe('test', () => {
    it('works', () => {
      var thecontract, dispatcherStorage;
      Example.new()
        .then(example => DispatcherStorage.new(example.address))
        .then(d => {
          dispatcherStorage = d;
          Dispatcher.unlinked_binary = Dispatcher.unlinked_binary
            .replace('1111222233334444555566667777888899990000',
            dispatcherStorage.address.slice(2));
          return Dispatcher.new();
        })
```

/lib/test/Example.sol

```
pragma solidity ^0.4.8;

import "../contracts/LibInterface.sol";

library Example {
  function getUint(LibInterface.S storage s) returns (uint) {
    return s.i;
  }
  function setUint(LibInterface.S storage s, uint i) {
    s.i = i;
  }
}
```

/lib/contract/DispatcherStorage.sol

```
contract DispatcherStorage {
  address public lib;
  mapping(bytes4 => uint32) public sizes;

  function DispatcherStorage(address newLib) {
    sizes[bytes4(sha3("getUint(LibInterface.S storage)"))] = 32;
    replace(newLib);
  }

  function replace(address newLib) /* onlyDAO */ {
    lib = newLib;
  }
}
```

3. Proxy pattern

/lib/contract/Dispatcher.sol

```
import "./DispatcherStorage.sol";

contract Dispatcher {
  function() {
    DispatcherStorage dispatcherStorage = DispatcherStorage(0x111122223333444455556666777788889999000);
    uint32 len = dispatcherStorage.sizes(msg.sig);
    address target = dispatcherStorage.lib();

    assembly {
      calldatacopy(0x0, 0x0, calldatasize)
      let a := delegatecall(sub(gas, 10000), target, 0x0, calldatasize, 0, len)
      return(0, len)
    }
  }
}
```

/lib/test/test.js

```
contract('TestProxyLibrary', () => {
  describe('test', () => {
    it('works', () => {
      var thecontract, dispatcherStorage;
      Example.new()
        .then(example => DispatcherStorage.new(example.address))
        .then(d => {
          dispatcherStorage = d;
          Dispatcher.unlinked_binary = Dispatcher.unlinked_binary
            .replace('111122223333444455556666777788889999000',
            dispatcherStorage.address.slice(2));
          return Dispatcher.new();
        })
```

/lib/contract/DispatcherStorage.sol

```
contract DispatcherStorage {
  address public lib;
  mapping(bytes4 => uint32) public sizes;

  function DispatcherStorage(address newLib) {
    sizes[bytes4(sha3("getUint(LibInterface.S storage)"))] = 32;
    replace(newLib);
  }

  function replace(address newLib) /* onlyDAO */ {
    lib = newLib;
  }
}
```

3. Proxy pattern

/lib/contract/Dispatch.sol

```
import "./DispatcherStorage.sol";

contract Dispatcher {
  function() {
    DispatcherStorage dispatcherStorage = DispatcherStorage(0x111122223333444455556666777788889999000);
    uint32 len = dispatcherStorage.sizes(msg.sig);
    address target = dispatcherStorage.lib();

    assembly {
      calldatacopy(0x0, 0x0, calldatasize)
      let a := delegatecall(sub(gas, 10000), target, 0x0, calldatasize, 0, len)
      return(0, len)
    }
```

/lib/test/test.js

```
    .then(dispatcher => {
      TheContract.link('LibInterface', dispatcher.address);
      return TheContract.new();
    })
    .then(c => {
      thecontract = c;
      return thecontract.set(10);
    })
```

/lib/contract/TheContract.sol

```
import "./LibInterface.sol";

contract TheContract {
  LibInterface.S s;

  using LibInterface for LibInterface.S;

  function get() constant returns (uint) {
    return s.getUint();
  }

  function set(uint i) {
    return s.setUint(i);
  }
}
```

/lib/contract/LibInterface.sol

```
library LibNoProxy {
  struct S { uint i; }

  function getUint(S storage s) returns (uint);
  function setUint(S storage s, uint i);
}
```

3. Proxy pattern

/lib/test/test.js

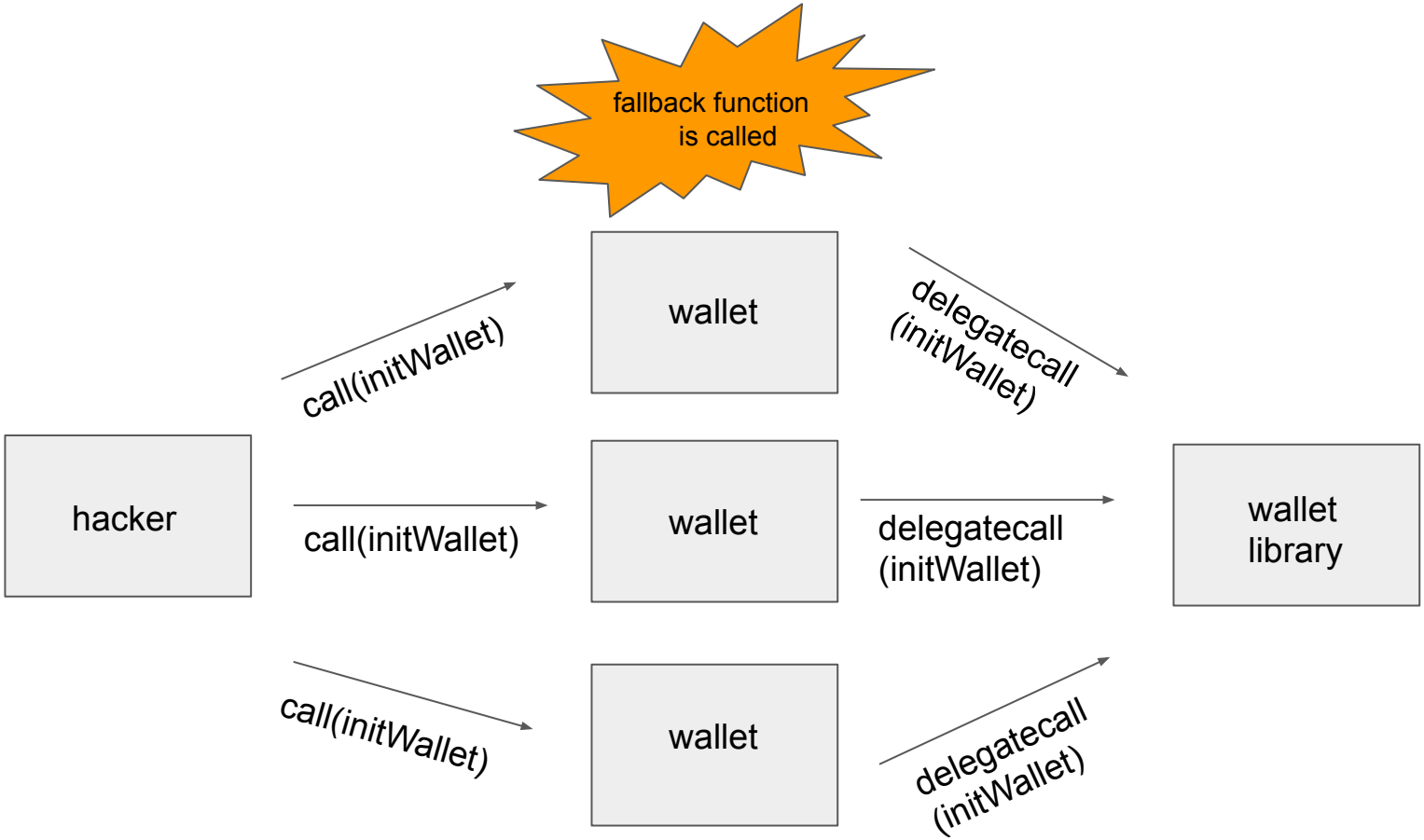
```
    })
    .then(() => Example2.new())
    .then(newExample => dispatcherStorage.replace(newExample.address))
    .then(() => thecontract.get())
    .then(x => assert.equal(x, 10 * 10)); // Example 2 getter multiplies
  });
  it.only('measure gas costs', () => {
  });
});
```

/lib/test/Example2.sol

```
import "../contracts/LibInterface.sol";

library Example2 {
  function getUint(LibInterface.S storage s) returns (uint) {
    return s.i * 10;
  }

  function setUint(LibInterface.S storage s, uint i) {
    s.i = i;
  }
}
```

/lib/contract/DispatcherStorage.sol

```
contract DispatcherStorage {
  address public lib;
  mapping(bytes4 => uint32) public sizes;

  function DispatcherStorage(address newLib) {
    sizes[bytes4(sha3("getUint(LibInterface.S storage)"))] = 32;
    replace(newLib);
  }

  function replace(address newLib) /* onlyDAO */ {
    lib = newLib;
  }
}
```


fallback function
is called
wallet
call(initWallet)
delegatecall
(initWallet)
hacker
call(initWallet)
wallet
delegatecall
(initWallet)
wallet
library
call(initWallet)
wallet
delegatecall
(initWallet)

4. Wallet contract

```
contract Wallet is WalletEvents {
        address constant _walletLibrary = 0xcafecafecafecafecafecafecafecafecafecafe;

         function Wallet(address[] _owners, uint _required, uint _daylimit) {
           bytes4 sig = bytes4(sha3("initWallet(address[],uint256,uint256)"));
           address target = _walletLibrary;

           uint argarraysize = (2 + _owners.length);
           uint argsize = (2 + argarraysize) * 32;

           assembly {
             mstore(0x0, sig)
             codecopy(0x4,  sub(codesize, argsize), argsize)
             delegatecall(sub(gas, 10000), target, 0x0, add(argsize, 0x4), 0x0, 0x0)
           }
         }
        …
}
```

4. Wallet contract

```
contract Wallet is WalletEvents {

        function() payable {
          if (msg.value > 0)
            Deposit(msg.sender, msg.value);
          else if (msg.data.length > 0)
            _walletLibrary.delegatecall(msg.data);
        }
      …
}
```

4. WalletLibrary contract (before bug fix)

```
contract WalletLibrary is WalletEvents {

        function initWallet(address[] _owners, uint _required, uint _daylimit) {
          initDaylimit(_daylimit);
          initMultiowned(_owners, _required);
         }
...
}
```

4. WalletLibrary contract (before bug fix)

```
function initDaylimit(uint _limit) {
  m_dailyLimit = _limit;
  m_lastDay = today();
 }

function initMultiowned(address[] _owners, uint _required) {
  m_numOwners = _owners.length + 1;
  m_owners[1] = uint(msg.sender);
  m_ownerIndex[uint(msg.sender)] = 1;
  for (uint i = 0; i < _owners.length; ++i)
  {
    m_owners[2 + i] = uint(_owners[i]);
    m_ownerIndex[uint(_owners[i])] = 2 + i;
  }
  m_required = _required;
 }
```

버그픽스

- https://github.com/paritytech/parity/commit/02d462e2636f1898df3e7556364260c594b112e6

4. WalletLibrary contract (after bug fix)

```
contract WalletLibrary is WalletEvents {

        modifier only_uninitialized { if (m_numOwners > 0) throw; _; }

        function initWallet(address[] _owners, uint _required, uint _daylimit) only_uninitialized {
          initDaylimit(_daylimit);
          initMultiowned(_owners, _required);
         }
...
}
```

4. WalletLibrary contract (after bug fix)

```
function initDaylimit(uint _limit) internal {
  m_dailyLimit = _limit;
  m_lastDay = today();
 }

function initMultiowned(address[] _owners, uint _required) internal {
  m_numOwners = _owners.length + 1;
  m_owners[1] = uint(msg.sender);
  m_ownerIndex[uint(msg.sender)] = 1;
  for (uint i = 0; i < _owners.length; ++i)
  {
    m_owners[2 + i] = uint(_owners[i]);
    m_ownerIndex[uint(_owners[i])] = 2 + i;
  }
  m_required = _required;
 }
```

4. WalletLibrary contract (after additional bug fix)

```
function initDaylimit(uint _limit) only_uninitialized {
  m_dailyLimit = _limit;
  m_lastDay = today();
 }

function initMultiowned(address[] _owners, uint _required) only_uninitialized {
  m_numOwners = _owners.length + 1;
  m_owners[1] = uint(msg.sender);
  m_ownerIndex[uint(msg.sender)] = 1;
  for (uint i = 0; i < _owners.length; ++i)
  {
    m_owners[2 + i] = uint(_owners[i]);
    m_ownerIndex[uint(_owners[i])] = 2 + i;
  }
  m_required = _required;
 }
```

function visibility

- 화이트보드로 설명

5. Exploit Demo

추후 게시 예정

6. Conclusion and Discussion

- Be careful use fallback func and delegate call
- Please, Understand EVM, especially Storage
- Do not fully depend on smart contract
- Use hardware wallet
- Platform risk

7. Presenter Info

정순형(Kevin Jeong, 철학자) kevin.j@onther.io

김휘경(Zoe Kim) zoe.k@onther.io

노언석(Andy Roh) andy.r@onther.io

이진용(Jay Lee) jay.l@onther.io