Smart Contract Code Review and Security Analysis Report

Customer: SONM
Date: July 13, 18

This document contains confidential information about IT systems and intellectual properties of the customer, as well as information about potential vulnerabilities and methods of their exploitation.
This confidential information is for internal use by the customer only and shall not be disclosed to third parties.

Document:

Name:	Smart Contract Code Review and Security Analysis Report for SONM
Date:	13.07.2018

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by SONM (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer`s smart contract and its code review conducted between June 28th, 2018 - July 13th, 2018.

Scope

The scope of the project is SONM smart contracts, which can be found on github by links below:

- Blacklist

https://github.com/sonm-io/core/blob/master/blockchain/source/contracts/Blacklist.sol

- ProfileRegistry

https://github.com/sonm-io/core/blob/master/blockchain/source/contracts/ProfileRegistry.sol

- OracleUSD

https://github.com/sonm-io/core/blob/master/blockchain/source/contracts/OracleUSD.sol

- SNM

https://github.com/sonm-io/core/blob/master/blockchain/source/contracts/SNM.sol

- SimpleGatekeeperWithLimit

https://github.com/sonm-io/core/blob/master/blockchain/source/contracts/SimpleGatekeeperWithLimit.sol

- SimpleGatekeeperWithLimitLive

https://github.com/sonm-io/core/blob/master/blockchain/source/contracts/SimpleGatekeeperWithLimitLive.sol

Commit 4ccf67358be704f76be54dd28d85cbd4cc801a43

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered (the full list includes them but is not limited to them):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit

- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

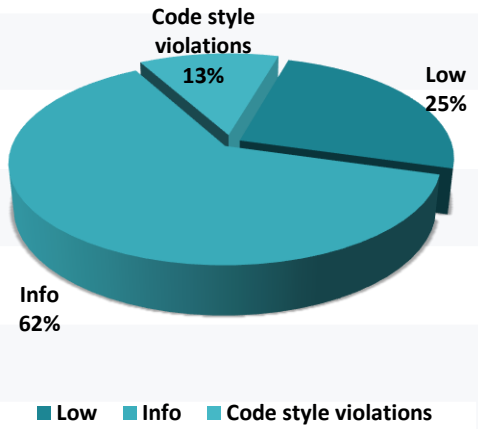
Executive Summary

According to the assessment, Customer`s smart contracts are well secured and have only some low security issues that can't have significant impact on security.

Our team performed analysis of code functionality, manual audit and automated checks with solc, Mythril and remix IDE (see Appendix B pic 1-27). All found issues during automated analysis were manually reviewed and applicable vulnerabilities are presented in Audit Overview section. General overview is presented in AS-IS section and all found issues can be found in Audit overview section.

We found 2 low-level vulnerabilities, outlined 3 important informational statements and 1 code style issue.

Graph 1. Vulnerabilities distribution



Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens lose etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Info	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

AS-IS overview

SNM contract overview

SNM.sol contract describes custom ERC20 token with next parameters:

- name – SONM token
- symbol – SNM
- decimals – 18

SNM contract constructor sets:

- totalSupply_ to 444 * 1e6 * 1e18
- balances[msg.sender] to totalSupply_

Blacklist contract overview

Blacklist contract constructor sets:

- owner to a msg.sender

Blacklist.sol has 1 modifier:

- OnlyMarket – checks if market is not 0x0; checks if msg.sender is a market or msg.sender is a master.

Blacklist.sol has 6 functions:

- Check – checks if person is in a blacklist
- Add – adds provided address to a blacklist. Has onlyMarket modifier.
- Remove – removes provided address from a blacklist.
- AddMaster – adds master. Has onlyOwner modifier.
- RemoveMaster – removes master. Has onlyOwner modifier.
- SetMarketAddress– sets market address to a specified address. Has onlyOwner modifier.

Market contract overview

Market contract constructor sets:

- token to SNM(_token)
- bl to Blacklist(_blacklist)
- oracle to OracleUSD(_oracle)
- pr to ProfileRegistry(_profileRegistry)
- benchmarksQuantity to _benchmarksQuantity
- netflagsQuantity to _netflagsQuantity

Market.sol has 37 functions:

- PlaceOrder – creates order on the market.
- CancelOrder – cancels order with ORDER_ACTIVE status.
- QuickBuy – creates an Ask order and calls OpenDeal.
- OpenDeal – opens the deal with a specified dealID.
- CloseDeal – closes the deal with a specified dealID.
- Bill – makes payments on the transaction..
- CreateChangeRequest – creates a new ChangeRequest, if there is a similar one on the other hand (Supplier or Customer), the transaction conditions change..
- CancelChangeRequest – changes RequesStatus to a REQUEST_CANCELED or REQUEST_REJECTED.
- RegisterWorker – registers the transaction author as Worker for the specified master.
- ConfirmWorker – confirms registration of the specified worker for the master - the author of the transaction.
- RemoveWorker – removes the specified worker for the specified master.
- GetOrderInfo – returns information about the order with a specified orderID.
- GetOrderParams – returns orderStatus and dealID about the order with a specified orderID.
- GetDealInfo – returns information about the deal with a specified dealID.
- GetDealParams – returns parameters for the deal with a specified dealID.
- GetMaster – returns the specified master Worker. Returns Worker, if the master is not specified.
- GetChangeRequestInfo – returns information about the change request with a specified changeRequestID.
- GetDealsAmount – returns dealAmount.
- GetOrdersAmount – returns orderAmount.
- GetChangeRequestsAmount – returns requestsAmount.
- GetBenchmarksQuantity – returns benchmarksQuantity.
- GetNetflagsQuantity – returns netflagsQuantity.
- InternalBill – implements bill logic for specified dealID.
- ReserveNextPeriodFunds – reserves funds for next period for specified dealID.
- RefundRemainingFunds – refunds remaining funds for specified dealID.
- IsSpot – checks if deal duration for specified address is 0.
- CalculatePayment – calculates payment for specified price and period.
- AddToBlacklist – allows consumer to add to a blacklist.
- InternalCloseDeal – allows to close a deal.
- ResizeBenchmarks – resizes benchmark with specified value.
- ResizeNetflags – resizes netflags.
- SetProfileRegistryAddress – sets profile registry address to a specified address. Has onlyOwner modifier.
- SetBlacklistAddress – sets blacklist address to a specified address. Has onlyOwner modifier.
- SetOracleAddress – sets oracle address to a specific address. Has onlyOwner modifier.
- SetBenchmarksQuantity – sets benchmark quantity. Has onlyOwner modifier.
- SetNetflagsQuantity – sets netflagsQuantity. Has onlyOwner modifier.
- KillMarket – transfers tokens to owner and kills market. Has onlyOwner modifier.

ProfileRegistry contract overview

ProfileRegistry contract constructor sets:

- owner to msg.sender
- validators[msg.sender] to -1

ProfileRegistry.sol has 1 modifier:

- onlySonm – checks if GetValidatorLevel(msg.sender) is -1.

ProfileRegistry.sol has 11 functions:

- AddValidator – creates validator for specified address. Has onlySonm and whenNotPaused modifiers.
- RemoveValidator – deletes validator for specified address. Has onlyMarket modifier. Has onlySonm and whenNotPaused modifiers.
- GetValidatorLevel – returns validator level for specified address.
- CreateCertificate – creates certificate. Has whenNotPaused modifier.
- RemoveCertificate – removes certificate. Has whenNotPaused modifier.
- GetCertificate – returns certificate for specified id.
- GetAttributeValue – returns certificate attribute value.
- GetAttributeCount – returns certificate attribute count.
- GetProfileLevel – returns identity level for specified address.
- AddSonmValidator – creates Sonm validator for specified address.Has onlyOwner modifier.
- RemoveSonmValidator – removes Sonm validator for specified address. Has onlyOwner modifier.

SimpleGatekeeperWithLimit contract overview

SimpleGatekeeperWithLimit contract constructor sets:

- token to StandardToken(_token)
- owner to msg.sender
- freezingTime to _freezingTime

SimpleGatekeeperWithLimit.sol has 3 functions:

- ChangeKeeperLimit – changes keeper`s dayLimit for specified address. Has onlyOwner modifier.
- FreezeKeeper – freezes keeper for specified address.
- UnfreezeKeeper – unfreezes keeper for specified address. Has onlyOwner modifier.
- PayIn – transfers tokens to the current contract.
- Payout – transfers tokens to a specified address and registers the payout. Has onlyOwner modifier.

- SetFreezingTime – sets freezing time. Has onlyOwner modifier.
- GetFreezingTime – returns freezing time.
- underLimit – checks if there is enough left - if so, adds provided value to spentToday and return true. Resets the spend limit if we're on a different day to last time.
- today – returns current block timestamp as seconds since unix epoch divided by 24 hours.
- kill – destroys contract and sends all funds to owner. Has onlyOwner modifier.

SimpleGatekeeperWithLimitLive contract overview

SimpleGatekeeperWithLimitLive contract constructor sets:

- token to SNMMasterchain(_token)
- owner to msg.sender
- freezingTime to _freezingTime

SimpleGatekeeperWithLimit.sol has 3 functions:

- ChangeKeeperLimit – changes keeper`s dayLimit for specified address. Has onlyOwner modifier.
- FreezeKeeper – freezes keeper for specified address.
- UnfreezeKeeper – unfreezes keeper for specified address. Has onlyOwner modifier.
- PayIn – transfers tokens to the current contract.
- Payout – transfers tokens to a specified address and registers the payout. Has onlyOwner modifier.
- SetFreezingTime – sets freezing time. Has onlyOwner modifier.
- GetFreezingTime – returns freezing time.
- underLimit – checks if there is enough left - if so, adds provided value to spentToday and return true. Resets the spend limit if we're on a different day to last time.
- today – returns current block timestamp as seconds since unix epoch divided by 24 hours.
- kill – destroys contract and sends all funds to owner. Has onlyOwner modifier.

Audit overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

1. Not using abi.encodePacked in a contracts is bad practice. In current versions of compilers, it is recommended to use bytes32 txHash = keccak256 (abi.encodePacked (_to, _txNumber, _value)); instead of bytes32 txHash = keccak256 (_to, _txNumber, _value); in SimpleGatekeeperWithLimit.sol (line 81); in SimpleGatekeeperWithLimitLive.sol (line 81) (see Appendix A pic 1 for evidence).
2. Calling events without emit is obsolete, events are invoked without emit in SimpleGatekeeperWithLimit.sol (line 119); SNMMasterchain.sol (lines 73, 98, 103, 134) (see Appendix A pic 2 for evidence).

Lowest / Code style / Info

Code style issues

3. Poor conneinting in Market.sol - Line 196 contains a comment // this line contains err. Comment should be removed in the final version (see Appendix A pic 3 for evidence).

Informational statements

Informational statements are audit team findings that doesn't have any security issues. However, they are presented in report to clarify and outline functionality and business requirements.

4. FreezeKeeper function is public in SimpleGatekeeperWithLimit.sol and SimpleGatekeeperWithLimitLive.sol, i.e. any account with a limit greater than 0 can freeze any account with a limit greater than 0 (see Appendix A pic 4 for evidence). We were informed by Customer that this is expected behavior of the contract.
5. Obsolete constructions in SNMMasterchain.sol: (see Appendix A pic 5 for evidence).
   - in the contract old versions of OpenZeppelin contracts are copied: SafeMath.sol, Math.sol, ERC20Basic, BasicToken, ERC20, StandardToken;
   - use function SNMMasterchain instead of constructor ();
   - old version of the Solidity compiler (0.4.24 is recommended);

   SNMMasterchain.sol contract is already deployed and no changes can be made to it.
6. Unlimited tokens can be mint from the ICO account in SNMMasterchain.sol (see Appendix A pic 6 for evidence).
   SNMMasterchain.sol contract is already deployed and no changes can be made to it.
7. Profiles do not stop working after freezing. The AddSonmValidator, RemoveSonmValidator functions do not have a modifier whenNotPaused, i.e. they can be run while profile is frozen (see Appendix A pic 7 for evidence). Customer haven't defined expected behavior yet, but it considers current behavior as OK.
8. Bad practice ordering is used in several Customer contracts. Changing parameters after transferring tokens or ether is a bad practice, which can lead to several vulnerabilities, including reentrancy. On lines 228, 648, 655, 702 Market.sol; lines 104, 128 SimpleGatekeeperWithLimit.sol; line 96 SimpleGatekeeperWithLimitLive.sol (see Appendix A pic 8 for evidence). It is recommended not to change parameters after transfers. However, there is no existing attacks for current code so the issue is considered as informational.

Conclusion

During the audit the contract was manually reviewed and analyzed with static analysis tools. As-is description was described.

Audit team have found some low to lowest security issues during the audit and audit report contains all necessary information related to them.

Because the vulnerabilities found are low level as maximum, the code is considered secure and no major fixes are required.

Disclaimers

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding to several independent audits and a public bug bounty program to ensure the security of the smart contracts.

Technical Disclaimer

Smart contract build on the top of Ethereum blockchain means that a lot of features could be covered by tests, but Turing completeness of Solidity programming language realization leaves some space for unexpected runtime exceptions.

Appendix A. Evidences

Pic. 1. abi.encodePacked not used

```
81          bytes32 txHash = keccak256(_to, _txNumber, _value);
```

Pic. 2. Calling events without emit

```
73      Transfer(msg.sender, _to, _value);
```

Pic. 3. Poor conneinting in Market.sol

```
196             // this line contains err.
197             require(token.transferFrom(msg.sender, this, lockedSum));
198         }
```

Pic. 4. FreezeKeeper function is public

```
55      function FreezeKeeper(address _keeper) public {
56          // check access of sender
57          require(keepers[msg.sender].dayLimit > 0);
58          // check that freezing keeper has limit
59          require(keepers[_keeper].dayLimit > 0);
60          keepers[_keeper].frozen = true;
61          emit KeeperFreezed(_keeper);
62      }
```

Pic. 5. SNMMasterchain obsolete constructions

```
51  contract ERC20Basic {
52    uint public totalSupply;
53    function balanceOf(address who) constant returns (uint);
54    function transfer(address to, uint value);
55    event Transfer(address indexed from, address indexed to, uint value);
56  }
57
58  contract BasicToken is ERC20Basic {
59    using SafeMath for uint;
60
61    mapping(address => uint) balances;
62
63    modifier onlyPayloadSize(uint size) {
64       if(msg.data.length < size + 4) {
65         throw;
66       }
67       _;
68    }
69
70    function transfer(address _to, uint _value) onlyPayloadSize(2 * 32) {
71      balances[msg.sender] = balances[msg.sender].sub(_value);
72      balances[_to] = balances[_to].add(_value);
73      Transfer(msg.sender, _to, _value);
74    }
75
76    function balanceOf(address _owner) constant returns (uint balance) {
77      return balances[_owner];
78    }
79
80  }
```

```
82  contract ERC20 is ERC20Basic {
83    function allowance(address owner, address spender) constant returns (uint);
84    function transferFrom(address from, address to, uint value);
85    function approve(address spender, uint value);
86    event Approval(address indexed owner, address indexed spender, uint value);
87  }
88
89  contract StandardToken is BasicToken, ERC20 {
90
91    mapping (address => mapping (address => uint)) allowed;
92
93    function transferFrom(address _from, address _to, uint _value) {
94      var _allowance = allowed[_from][msg.sender];
95      balances[_to] = balances[_to].add(_value);
96      balances[_from] = balances[_from].sub(_value);
97      allowed[_from][msg.sender] = _allowance.sub(_value);
98      Transfer(_from, _to, _value);
99    }
100
101   function approve(address _spender, uint _value) {
102     allowed[msg.sender][_spender] = _value;
103     Approval(msg.sender, _spender, _value);
104   }
105
106   function allowance(address _owner, address _spender) constant returns (uint remaining) {
107     return allowed[_owner][_spender];
108   }
109
110 }
```

```
122    function SNMMasterchain(address _ico) {
123      ico = _ico;
124    }
```

Pic. 6. Unlimited mint

```
127     function mint(address _holder, uint _value) external {
128       require(msg.sender == ico);
129       require(_value != 0);
130       require(totalSupply + _value <= TOKEN_LIMIT);
131
132       balances[_holder] += _value;
133       totalSupply += _value;
134       Transfer(0x0, _holder, _value);
135     }
```

Pic. 7. No WhenNotPaused modifier for functions

```
142     function AddSonmValidator(address _validator) onlyOwner public returns (bool) {
143         validators[_validator] = -1;
144         return true;
145     }
146
147     function RemoveSonmValidator(address _validator) onlyOwner public returns (bool) {
148         require(GetValidatorLevel(_validator) == -1);
149         validators[_validator] = 0;
150         return true;
151     }
```

Pic 8. Bad practice transaction ordering

```
228           require(token.transfer(msg.sender, orders[orderID].frozenSum));
229           orders[orderID].orderStatus = OrderStatus.ORDER_INACTIVE;

648               require(token.transfer(deal.masterID, deal.blockedBalance));
649               deals[dealID].lastBillTS = block.timestamp;
650               deals[dealID].totalPayout = deals[dealID].totalPayout.add(deal.blockedBalance);
651               deals[dealID].blockedBalance = 0;

655         require(token.transfer(deal.masterID, paidAmount));
656         deals[dealID].blockedBalance = deals[dealID].blockedBalance.sub(paidAmount);
657         deals[dealID].totalPayout = deals[dealID].totalPayout.add(paidAmount);
658         deals[dealID].lastBillTS = block.timestamp;

702             token.transfer(deals[dealID].consumerID, deals[dealID].blockedBalance);
703             deals[dealID].blockedBalance = 0;

35          require(token.transfer(_to, _value));
36          paid[txHash] = true;
```

Appendix B. Automated tools reports

Pic. 1 Solc Blacklist automated report

```
max@max-VirtualBox:~/solidity/projects/sonm$ solc -o . --bin --abi --overwrite Blacklist.sol
max@max-VirtualBox:~/solidity/projects/sonm$
```

Pic. 2 Mythril Blacklist automated report

```
max@max-VirtualBox:~/solidity/projects/sonm$ myth -x Blacklist.sol
The analysis was completed successfully. No issues were detected.

max@max-VirtualBox:~/solidity/projects/sonm$
```

Pic. 3 Solc ProfileRegistry automated report

```
max@max-VirtualBox:~/solidity/projects/sonm$ solc -o . --bin --abi --overwrite ProfileRegistry.sol
max@max-VirtualBox:~/solidity/projects/sonm$
```

Pic. 4 Mythril ProfileRegistry automated report

```
max@max-VirtualBox:~/solidity/projects/sonm$ myth -x ProfileRegistry.sol
The analysis was completed successfully. No issues were detected.

max@max-VirtualBox:~/solidity/projects/sonm$
```

Pic. 5 Solc SNM automated report

```
max@max-VirtualBox:~/solidity/projects/sonm$ solc -o . --bin --abi --overwrite SNM.sol
max@max-VirtualBox:~/solidity/projects/sonm$
```

Pic. 6 Mythril SNM automated report

```
max@max-VirtualBox:~/solidity/projects/sonm$ myth -x SNM.sol
==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: increaseApproval(address,uint256)
PC address: 5383
A possible integer overflow exists in the function `increaseApproval(address,uint256)`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: SafeMath.sol:48

a + b

--------------------

==== Exception state ====
Type: Informational
Contract: Unknown
Function name: increaseApproval(address,uint256)
PC address: 5396
A reachable exception (opcode 0xfe) has been detected. This can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. This is acceptable in most situations. Note ho
wever that `assert()` should only be used to check invariants. Use `require()` for regular input checking.
--------------------
In file: SafeMath.sol:49

assert(c >= a)

--------------------


max@max-VirtualBox:~/solidity/projects/sonm$
```

Pic. 7 Solc OracleUSD automated report

```
max@max-VirtualBox:~/solidity/projects/sonm$ solc -o . --bin --abi --overwrite OracleUSD.sol
max@max-VirtualBox:~/solidity/projects/sonm$
```

Pic. 8 Mythril OracleUSD automated report

```
max@max-VirtualBox:~/solidity/projects/sonm$ myth -x OracleUSD.sol
The analysis was completed successfully. No issues were detected.

max@max-VirtualBox:~/solidity/projects/sonm$
```

Pic. 9 Solc Market automated report

```
max@max-VirtualBox:~/solidity/projects/sonm$ solc -o . --bin --abi --overwrite Market.sol
max@max-VirtualBox:~/solidity/projects/sonm$
```

Pic. 10 Mythril Market automated report

```
max@max-VirtualBox:~/solidity/projects/sonm$ myth -x Market.sol
==== Exception state ====
Type: Informational
Contract: Unknown
Function name: _function_0x03988f84
PC address: 699
A reachable exception (opcode 0xfe) has been detected. This can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. This is acceptable in most situations. Note however t
hat `assert()` should only be used to check invariants. Use `require()` for regular input checking.
--------------------
In file: Market.sol:137

mapping(uint => Deal) public deals

--------------------

==== Exception state ====
Type: Informational
Contract: Unknown
Function name: GetDealParams(uint256)
PC address: 1376
A reachable exception (opcode 0xfe) has been detected. This can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. This is acceptable in most situations. Note however t
hat `assert()` should only be used to check invariants. Use `require()` for regular input checking.
--------------------
In file: Market.sol:556

function GetDealParams(uint dealID) view public
    returns (
        uint duration,
        uint price,
        uint endTime,
        DealStatus status,
        uint blockedBalance,
        uint totalPayout,
        uint lastBillTS
    ){
        return (
        deals[dealID].duration,
        deals[dealID].price,
        deals[dealID].endTime,
        deals[dealID].status,
        deals[dealID].blockedBalance,
        deals[dealID].totalPayout,
        deals[dealID].lastBillTS
        );
    }

------------------
```

Pic. 11 Mythril Market automated report

```
==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: GetDealInfo(uint256)
PC address: 1666
A possible integer overflow exists in the function `GetDealInfo(uint256)`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: Market.sol:534

function GetDealInfo(uint dealID) view public
    returns (
        uint64[] benchmarks,
        address supplierID,
        address consumerID,
        address masterID,
        uint askID,
        uint bidID,
        uint startTime
    ){
        return (
        deals[dealID].benchmarks,
        deals[dealID].supplierID,
        deals[dealID].consumerID,
        deals[dealID].masterID,
        deals[dealID].askID,
        deals[dealID].bidID,
        deals[dealID].startTime

        );
    }

--------------------
```

Pic. 12 Mythril Market automated report

```
==== Exception state ====
Type: Informational
Contract: Unknown
Function name: GetChangeRequestInfo(uint256)
PC address: 1836
A reachable exception (opcode 0xfe) has been detected. This can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. This is acceptable in most situations. Note however t
hat `assert()` should only be used to check invariants. Use `require()` for regular input checking.
--------------------
In file: Market.sol:585

function GetChangeRequestInfo(uint changeRequestID) view public
    returns (
        uint dealID,
        OrderType requestType,
        uint price,
        uint duration,
        RequestStatus status
    ){
        return (
        requests[changeRequestID].dealID,
        requests[changeRequestID].requestType,
        requests[changeRequestID].price,
        requests[changeRequestID].duration,
        requests[changeRequestID].status
        );
    }

--------------------
```

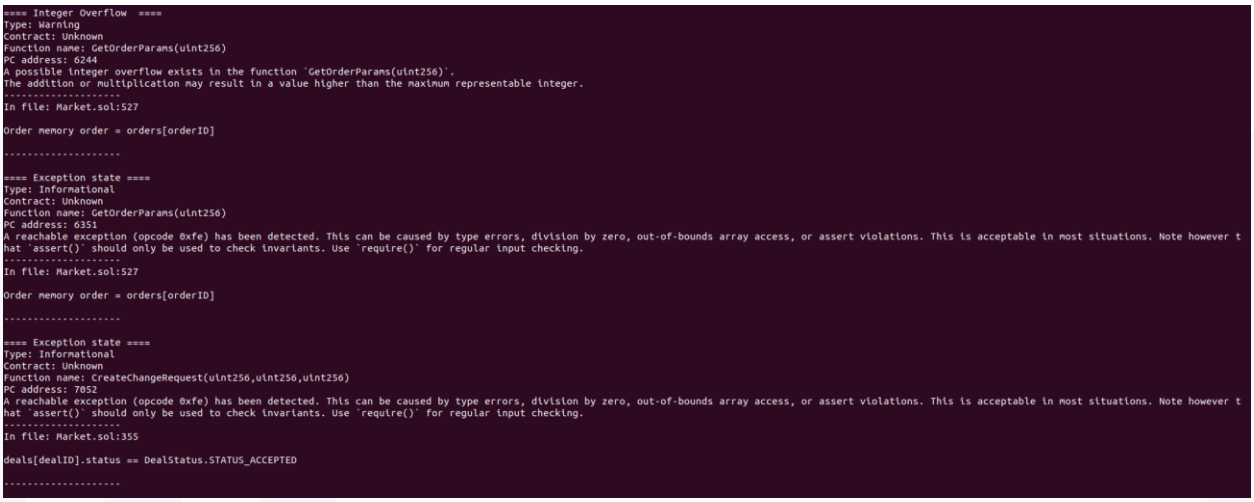
Pic. 13 Mythril Market automated report

```
==== Exception state ====
Type: Informational
Contract: Unknown
Function name: GetChangeRequestInfo(uint256)
PC address: 1868
A reachable exception (opcode 0xfe) has been detected. This can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. This is acceptable in most situations. Note however t
hat `assert()` should only be used to check invariants. Use `require()` for regular input checking.
--------------------
In file: Market.sol:585

function GetChangeRequestInfo(uint changeRequestID) view public
    returns (
        uint dealID,
        OrderType requestType,
        uint price,
        uint duration,
        RequestStatus status
    ){
        return (
        requests[changeRequestID].dealID,
        requests[changeRequestID].requestType,
        requests[changeRequestID].price,
        requests[changeRequestID].duration,
        requests[changeRequestID].status
        );
    }

--------------------

==== Message call to external contract ====
Type: Warning
Contract: Unknown
Function name: SetOracleAddress(address)
PC address: 4693
This contract executes a message call to an address provided as a function argument. Generally, it is not recommended to call user-supplied addresses using Solidity's call() construct. Note that attackers might
leverage reentrancy attacks to exploit race conditions or manipulate this contract's state.
--------------------
In file: Market.sol:767

OracleUSD(_newOracle).getCurrentPrice()

--------------------
```

Pic. 14 Mythril Market automated report

```
==== State change after external call ====
Type: Warning
Contract: Unknown
Function name: SetOracleAddress(address)
PC address: 4826
The contract account state is changed after an external call. Consider that the called contract could re-enter the function before this state change takes place. This can lead to business logic vulnerabilities.
--------------------
In file: Market.sol:768

oracle = OracleUSD(_newOracle)

--------------------

==== Exception state ====
Type: Informational
Contract: Unknown
Function name: GetOrderParams(uint256)
PC address: 5944
A reachable exception (opcode 0xfe) has been detected. This can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. This is acceptable in most situations. Note however t
hat `assert()` should only be used to check invariants. Use `require()` for regular input checking.
--------------------
In file: Market.sol:527

Order memory order = orders[orderID]

--------------------

==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: GetOrderParams(uint256)
PC address: 6242
A possible integer overflow exists in the function `GetOrderParams(uint256)`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: Market.sol:527

Order memory order = orders[orderID]

--------------------
```

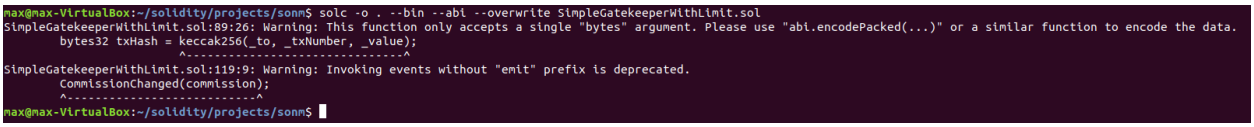
Pic. 15 Mythril Market automated report



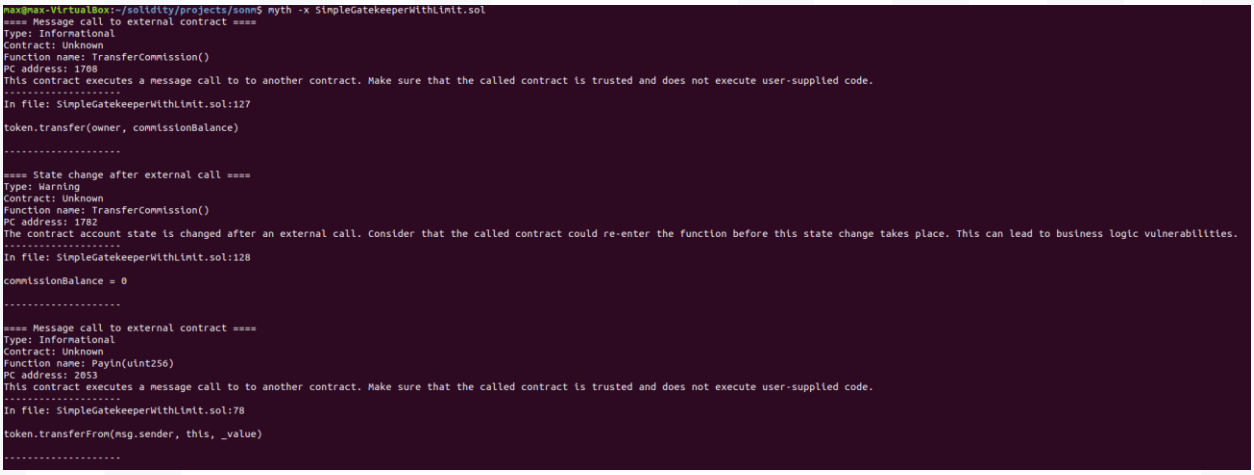
Pic. 16 Mythril Market automated report

```
==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: GetDealInfo(uint256)
PC address: 10788
A possible integer overflow exists in the function `GetDealInfo(uint256)`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: Market.sol:544

return (
        deals[dealID].benchmarks,
        deals[dealID].supplierID,
        deals[dealID].consumerID,
        deals[dealID].masterID,
        deals[dealID].askID,
        deals[dealID].bidID,
        deals[dealID].startTime

        )

--------------------

==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: GetDealInfo(uint256)
PC address: 10790
A possible integer overflow exists in the function `GetDealInfo(uint256)`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: Market.sol:544

return (
        deals[dealID].benchmarks,
        deals[dealID].supplierID,
        deals[dealID].consumerID,
        deals[dealID].masterID,
        deals[dealID].askID,
        deals[dealID].bidID,
        deals[dealID].startTime

        )

--------------------


max@max-VirtualBox:~/solidity/projects/sonm$
```

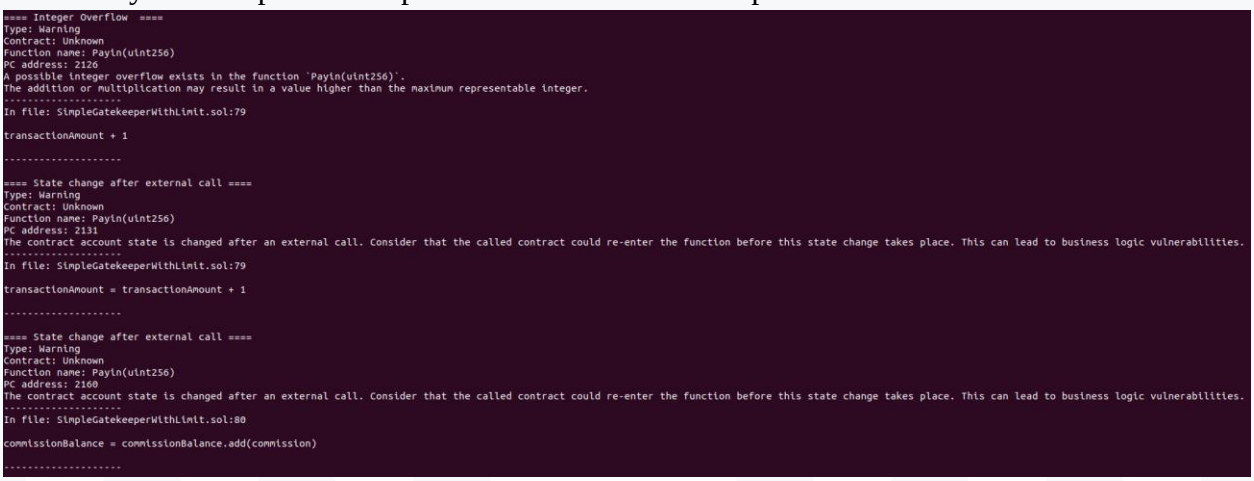
Pic. 17 Solc SimpleGatekeeperWithLimit automated report



Pic. 18 Myhtril SimpleGatekeeperWithLimit automated report



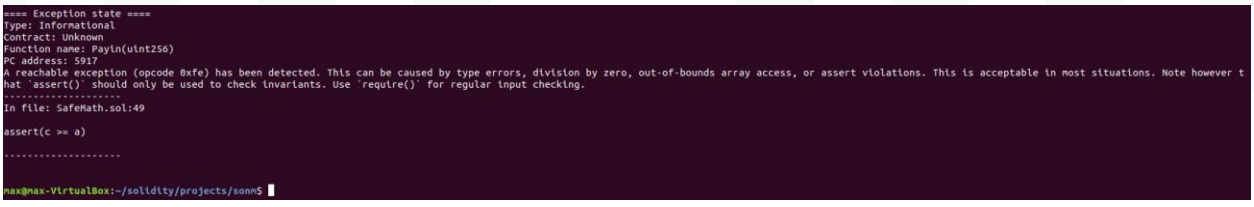
Pic. 19 Myhtril SimpleGatekeeperWithLimit automated report



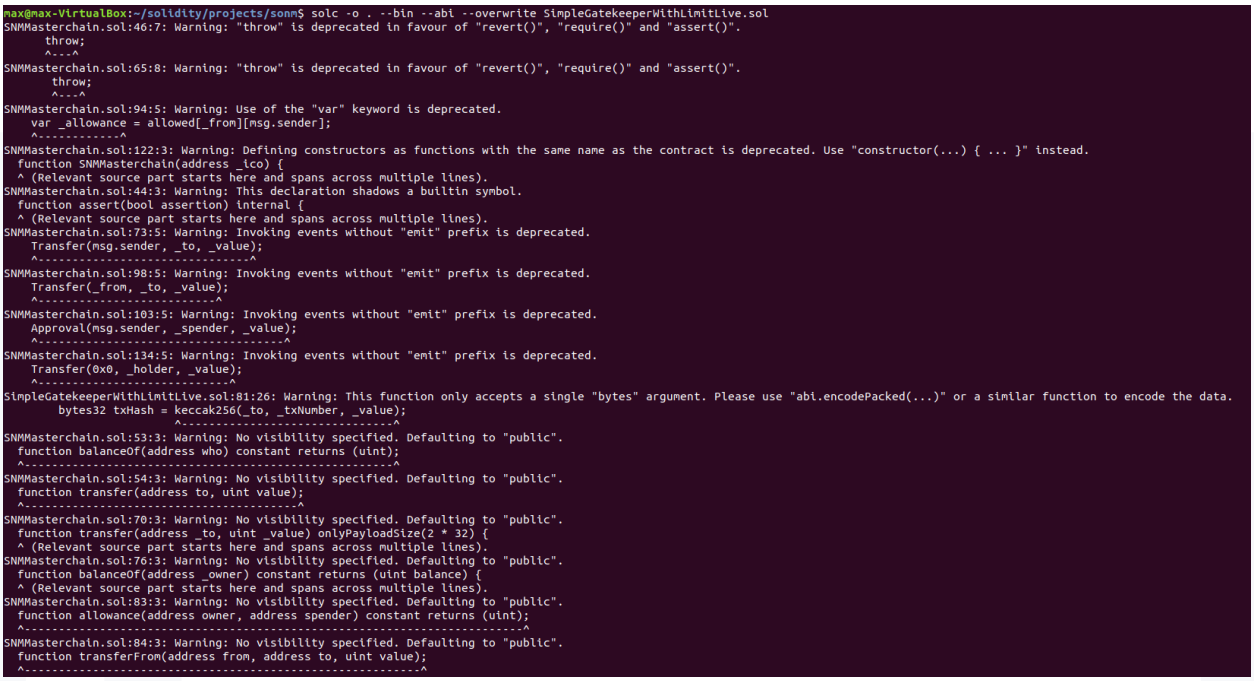
Pic. 20 Myhtril SimpleGatekeeperWithLimit automated report

Pic. 21 Myhtril SimpleGatekeeperWithLimit automated report

Pic. 22 Myhtril SimpleGatekeeperWithLimit automated report



Pic. 23 Solc SimpleGatekeeperWithLimitLive automated report



```
max@max-VirtualBox:~/solidity/projects/sonm$ solc -o . --bin --abi --overwrite SimpleGatekeeperWithLimitLive.sol
SNMMasterchain.sol:46:7: Warning: "throw" is deprecated in favour of "revert()", "require()" and "assert()".
      throw;
      ^---^
SNMMasterchain.sol:65:8: Warning: "throw" is deprecated in favour of "revert()", "require()" and "assert()".
       throw;
       ^---^
SNMMasterchain.sol:94:5: Warning: Use of the "var" keyword is deprecated.
    var _allowance = allowed[_from][msg.sender];
    ^-----------^
SNMMasterchain.sol:122:3: Warning: Defining constructors as functions with the same name as the contract is deprecated. Use "constructor(...) { ... }" instead.
  function SNMMasterchain(address _ico) {
  ^ (Relevant source part starts here and spans across multiple lines).
SNMMasterchain.sol:44:3: Warning: This declaration shadows a builtin symbol.
  function assert(bool assertion) internal {
  ^ (Relevant source part starts here and spans across multiple lines).
SNMMasterchain.sol:73:5: Warning: Invoking events without "emit" prefix is deprecated.
    Transfer(msg.sender, _to, _value);
    ^-------------------------------^
SNMMasterchain.sol:98:5: Warning: Invoking events without "emit" prefix is deprecated.
    Transfer(_from, _to, _value);
    ^--------------------------^
SNMMasterchain.sol:103:5: Warning: Invoking events without "emit" prefix is deprecated.
    Approval(msg.sender, _spender, _value);
    ^-----------------------------------^
SNMMasterchain.sol:134:5: Warning: Invoking events without "emit" prefix is deprecated.
    Transfer(0x0, _holder, _value);
    ^----------------------------^
SimpleGatekeeperWithLimitLive.sol:81:26: Warning: This function only accepts a single "bytes" argument. Please use "abi.encodePacked(...)" or a similar function to encode the data.
       bytes32 txHash = keccak256(_to, _txNumber, _value);
                        ^-------------------------------^
SNMMasterchain.sol:53:3: Warning: No visibility specified. Defaulting to "public".
  function balanceOf(address who) constant returns (uint);
  ^------------------------------------------------------^
SNMMasterchain.sol:54:3: Warning: No visibility specified. Defaulting to "public".
  function transfer(address to, uint value);
  ^----------------------------------------^
SNMMasterchain.sol:70:3: Warning: No visibility specified. Defaulting to "public".
  function transfer(address _to, uint _value) onlyPayloadSize(2 * 32) {
  ^ (Relevant source part starts here and spans across multiple lines).
SNMMasterchain.sol:76:3: Warning: No visibility specified. Defaulting to "public".
  function balanceOf(address _owner) constant returns (uint balance) {
  ^ (Relevant source part starts here and spans across multiple lines).
SNMMasterchain.sol:83:3: Warning: No visibility specified. Defaulting to "public".
  function allowance(address owner, address spender) constant returns (uint);
  ^-------------------------------------------------------------------------^
SNMMasterchain.sol:84:3: Warning: No visibility specified. Defaulting to "public".
  function transferFrom(address from, address to, uint value);
  ^----------------------------------------------------------^
```

Pic. 24 Solc SimpleGatekeeperWithLimitLive automated report

```
SNMMasterchain.sol:85:3: Warning: No visibility specified. Defaulting to "public".
  function approve(address spender, uint value);
  ^--------------------------------------------^
SNMMasterchain.sol:93:3: Warning: No visibility specified. Defaulting to "public".
  function transferFrom(address _from, address _to, uint _value) {
  ^ (Relevant source part starts here and spans across multiple lines).
SNMMasterchain.sol:101:3: Warning: No visibility specified. Defaulting to "public".
  function approve(address _spender, uint _value) {
  ^ (Relevant source part starts here and spans across multiple lines).
SNMMasterchain.sol:106:3: Warning: No visibility specified. Defaulting to "public".
  function allowance(address _owner, address _spender) constant returns (uint remaining) {
  ^ (Relevant source part starts here and spans across multiple lines).
SNMMasterchain.sol:122:3: Warning: No visibility specified. Defaulting to "public".
  function SNMMasterchain(address _ico) {
  ^ (Relevant source part starts here and spans across multiple lines).
SNMMasterchain.sol:28:3: Warning: Function state mutability can be restricted to pure
  function max64(uint64 a, uint64 b) internal constant returns (uint64) {
  ^ (Relevant source part starts here and spans across multiple lines).
SNMMasterchain.sol:32:3: Warning: Function state mutability can be restricted to pure
  function min64(uint64 a, uint64 b) internal constant returns (uint64) {
  ^ (Relevant source part starts here and spans across multiple lines).
SNMMasterchain.sol:36:3: Warning: Function state mutability can be restricted to pure
  function max256(uint256 a, uint256 b) internal constant returns (uint256) {
  ^ (Relevant source part starts here and spans across multiple lines).
SNMMasterchain.sol:40:3: Warning: Function state mutability can be restricted to pure
  function min256(uint256 a, uint256 b) internal constant returns (uint256) {
  ^ (Relevant source part starts here and spans across multiple lines).
SNMMasterchain.sol:44:3: Warning: Function state mutability can be restricted to pure
  function assert(bool assertion) internal {
  ^ (Relevant source part starts here and spans across multiple lines).
max@max-VirtualBox:~/solidity/projects/sonm$
```

Pic. 25 Myhtril SimpleGatekeeperWithLimitLive automated report

Pic. 26 Myhtril SimpleGatekeeperWithLimitLive automated report

Pic. 27 Myhtril SimpleGatekeeperWithLimitLive automated report