SmartDec

MithrilOre Smart Contracts Security Audit

Abstract

In this report we consider the security of the MithrilOre project. Our task is to find and describe security issues in the smart contracts of the platform.

Procedure

In our analysis we consider MithrilOre whitepaper (sha1sum aad251eb397bceaa6164bbb9c306efb9067ad336 003-Mithril-Ore-Rebrand-Professional-White-Paper-revisions-11-20-17.pdf) and smart contracts code (contract address is 0xfedfe337b4fff461d4b0d74da0307ee90b614cff).
We perform our audit according to the following procedure:

- automated analysis
  - we scan project's smart contracts with our own Solidity static code analyzer SmartCheck
  - we scan project's smart contracts with several publicly available automated Solidity analysis tools such as Remix, Oyente and Securify (beta version since full version was unavailable at the moment this report was made)
  - we manually verify (reject or confirm) all the issues found by tools
- manual audit
  - we manually analyze smart contracts for security vulnerabilities
  - we check smart contracts logic and compare it with the one described in the whitepaper
- report
  - we reflect all the gathered information in the report

The latest version of the code

We have performed the check of the fixed vulnerabilities in the latest version of code — sha1sum fbd703924c51261f9bf26eefe2bb86e8140a05c0 source.sol.

Disclaimer

The audit does not give any warranties on the security of the code. One audit can not 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. Besides, security audit is not an investment advice.

Checked vulnerabilities

We have scanned MithrilOre smart contracts for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we 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
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

About The Project

Project Architecture

The project was downloaded from Etherscan (contract address is 0xfedfe337b4fff461d4b0d74da0307ee90b614cff) as one source file (source.sol). The file contains

- `SafeMath` contract
- `IERC20` interface
- `MithrilOre` contract that implements the `IERC20` interface

Compilation is successful. The project does not contain tests and deployment scripts.

Code Logic

MithrilOre is a ERC20 token (with several discrepancies described below):
name — "Mithril Ore", symbol — MORE, decimals — 2, totalSupply — 500,000.
There are no ERC20 standard violations in the latest version of the code.

Automated Analysis

We used several publicly available automated Solidity analysis tools. Here are the combined results of their analysis. All the issues found by tools were manually checked (rejected or confirmed).
There are no confirmed issues in the latest version of code.

Tool	*Rule*	false positives	true positives
SmartCheck	Constant functions		10
	ERC20 approve		1
	No payable fallback	1	
	Pragmas version		1
	Unchecked math	3	
	Visibility		13
Total (SmartCheck)		4	25
Remix	Gas requirement of function	3	
	Variables have very similar names	1	
Total (Remix)		4	
Securify*		0	0
Oyente		0	0
Total		**8**	**25**

Securify* — beta version, full version is unavailable.
Securify, Oyente — these tools do not support 0.4.18 compiler version; the specified version was changed in the code to 0.4.16 and 0.4.17 respectively for these tools.

Cases when these issues lead to actual bugs or vulnerabilities are described in the next section.

Manual Analysis

Contracts were completely manually analyzed. The contracts don't implement crowdsale logic, so only ERC20 token properties were checked and compared with whitepaper. Besides, the results of automated analysis were manually verified. All confirmed issues are described below.

Critical issues

Critical issues seriously endanger smart contracts security. We highly recommend fixing them.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence smart contracts operation in current implementation. We highly recommend addressing them.

ERC20 standard violation

Strict discrepancy:

- the `transferFrom` function does not allow transferring 0 tokens, although the standard states: “Note Transfers of 0 values MUST be treated as normal transfers and fire the Transfer event.”
  The issue has been fixed and is not present in the latest version of the code..

Non-strict discrepancy:

- The standard says: "A token contract which creates new tokens SHOULD trigger a `Transfer` event with the `_from` address set to 0x0 when tokens are created". But there is no `Transfer` event in `MithrilOre` constructor in the code.
  The issue has been fixed and is not present in the latest version of the code.

ERC20 approve issue

There is ERC20 approve issue (MithrilOre.sol, line 98). We recommend adding `increaseApproval/decreaseApproval` functions. These functions can be found in OpenZeppelin contracts (see StandardToken.sol).
We recommend instructing users not to use `approve` directly and to use `increaseApproval`/`decreaseApproval` functions instead (or to change the approved amount to 0, wait for the transaction to be mined, and then to change the approved amount to the desired value) — link
Changing the approved amount from a nonzero value to another nonzero value allows a double spending with a front-running attack.
The issue has been fixed and is not present in the latest version of the code.

Potential tokens loss

The `transfer` and `transferFrom` functions do not check that the tokens are transferred to the non-zero address. This can lead to the loss of tokens, because tokens transferred to a zero address can not be revoked.
The issue has been fixed and is not present in the latest version of the code.

No tests

Provided code does not contain tests. Testing is crucial for code security. We highly recommend not only to cover the code with tests but also to make sure that the coverage is sufficient. The code also does not contain deployment scripts.

Low severity issues

Low severity issues can influence smart contracts operation in future versions of code. We recommend to take them into account.

Redundant code

- lines 78-80: redundant checks

```
require(balanceOf[msg.sender] >= _value);
        require(balanceOf[_to] + _value >= balanceOf[_to]);
        require(balanceOf[msg.sender] >= _value && balanceOf[_to] +
_value >= balanceOf[_to]);
```

  First two checks make third one redundant.

- lines 87-91: redundant check

```
require(allowance [_from][msg.sender] >= _value
            && balanceOf[_from] >= _value
            && _value > 0
            );
```

Use of `require` is redundant. If the condition is true, the SafeMath functions `add/sub` will throw an exception. We recommend use SafeMath functions (in particular, use `sub` instead of "-", line 94) without implementing redundant checks to save on the gas.
The issue has been fixed and is not present in the latest version of the code.

Pragmas version

Solidity source files indicate the versions of the compiler they can be compiled with.
Example:

```
pragma solidity ^0.4.18; // bad: compiles w 0.4.18 and above
pragma solidity 0.4.18; // good : compiles w 0.4.18 only
```

We recommend following the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee. Besides, we recommend using the latest compiler version (0.4.20 at the moment).
The issue has been fixed and is not present in the latest version of the code.

Codestyle issues

Codestyle issues influence code conciseness and readability and in some cases may lead to bugs in future. We recommend to take them into account.

Code Style

The code violates the Style Guide for Solidity badly (Solium and Solhint find a lot of style guide violations, see the output in Appendix).
The issue has been fixed and is not present in the latest version of the code.

Constant functions

We recommend to use `view` instead of `constant`, which will be deprecated for functions.

- line 9:

```
function mul(uint256 a, uint256 b) internal constant returns
(uint256)
```

- line 15:

```
function div(uint256 a, uint256 b) internal constant returns
(uint256)
```

- line 22:

```
function sub(uint256 a, uint256 b) internal constant returns
(uint256)
```

- line 27:

```
function add(uint256 a, uint256 b) internal constant returns
(uint256)
```

- line 35:

```
function totalSupply() constant returns (uint256 totalSupply)
```

- line 36:

```
function balanceOf(address _owner) constant returns (uint256
balance);
```

- line 40:

```
function allowance(address _owner, address _spender) constant
returns (uint256 remaining)
```

If a function is not supposed to modify the state or read from state, consider declaring it as `pure`.

The issue has been fixed and is not present in the latest version of the code.

Constants readability

The `initialSupply` (source.sol, line 63) value is defined not very clearly for reader.

```
initialSupply = 50000000;
```

We recommend defining it in the following format:

```
initialSupply = 500000 * (10 ** decimals);
```

The issue has been fixed and is not present in the latest version of the code.

Implicit visibility level

In many places in the code there are functions with implicit visibility level.

- lines 35, 71:

```
function totalSupply() constant returns (uint256 totalSupply)
```

- lines 36, 74:

```
function balanceOf(address _owner) constant returns (uint256
balance
```

- lines 37, 77:

```
function transfer(address _to, uint256 _value) returns (bool
success)
```

- lines 38, 86:

```
function transferFrom(address _from, address _to, uint256 _value)
returns (bool success)
```

- lines 39, 98:

```
function approve(address _spender, uint256 _value) returns (bool
success)
```

- lines 40, 104:

```
function allowance(address _owner, address _spender) constant
returns (uint256 remaining)
```

- line 61:

```
function MithrilOre()
```

The default function visibility level in Solidity is public. We recommend specifying visibility level explicitly and correctly to prevent confusion.
The issue has been fixed and is not present in the latest version of the code.

Conclusion

In this report we have considered the security of MithrilOre smart contracts. We performed our audit according to the procedure described above.
The audit showed no critical issues and several medium and low severity issues. Most of them has been fixed in the latest version of the code.

This analysis was performed by SmartDec

COO Sergey Pavlin 

December 26, 2017

Appendix

Compilation output

```
$ solcjs --bin source.sol
source.sol:35:5: Warning: No visibility specified. Defaulting to "public".
function totalSupply() constant returns (uint256 totalSupply);
^-------------------------------------------------------------^

source.sol:36:5: Warning: No visibility specified. Defaulting to "public".
function balanceOf(address _owner) constant returns (uint256 balance);
^--------------------------------------------------------------------^

source.sol:37:5: Warning: No visibility specified. Defaulting to "public".
function transfer(address _to, uint256 _value) returns (bool success);
^--------------------------------------------------------------------^

source.sol:38:5: Warning: No visibility specified. Defaulting to "public".
function transferFrom(address _from, address _to, uint256 _value) returns
(bool success);
^-----------------------------------------------------------------------------
-------------^

source.sol:39:5: Warning: No visibility specified. Defaulting to "public".
function approve(address _spender, uint256 _value) returns (bool success);
^-------------------------------------------------------------------------^

source.sol:40:5: Warning: No visibility specified. Defaulting to "public".
function allowance(address _owner, address _spender) constant returns
(uint256 remaining);
^-----------------------------------------------------------------------------
--------------^

source.sol:61:5: Warning: No visibility specified. Defaulting to "public".
function MithrilOre() {
^
Spanning multiple lines.

source.sol:71:5: Warning: No visibility specified. Defaulting to "public".
function totalSupply() constant returns (uint256 totalSupply){
^
Spanning multiple lines.

source.sol:74:5: Warning: No visibility specified. Defaulting to "public".
function balanceOf(address _owner) constant returns (uint256 balance){
^
Spanning multiple lines.

source.sol:77:5: Warning: No visibility specified. Defaulting to "public".
function transfer(address _to, uint256 _value) returns (bool success) {
^
Spanning multiple lines.

source.sol:86:5: Warning: No visibility specified. Defaulting to "public".
```

```
function transferFrom(address _from, address _to, uint256 _value) returns
(bool success){
^
Spanning multiple lines.

source.sol:98:5: Warning: No visibility specified. Defaulting to "public".
function approve(address _spender, uint256 _value) returns (bool success){
^
Spanning multiple lines.

source.sol:104:5: Warning: No visibility specified. Defaulting to "public".
function allowance(address _owner, address _spender) constant returns
(uint256 remaining){
^
Spanning multiple lines.

source.sol:35:46: Warning: This declaration shadows an existing
declaration.
function totalSupply() constant returns (uint256 totalSupply);
^-----------------^
source.sol:35:5: The shadowed declaration is here:
function totalSupply() constant returns (uint256 totalSupply);
^-----------------------------------------------------------^


source.sol:71:46: Warning: This declaration shadows an existing
declaration.
function totalSupply() constant returns (uint256 totalSupply){
^-----------------^
source.sol:71:5: The shadowed declaration is here:
function totalSupply() constant returns (uint256 totalSupply){
^
Spanning multiple lines.


source.sol:9:3: Warning: Function state mutability can be restricted to
pure
function mul(uint256 a, uint256 b) internal constant returns (uint256) {
^
Spanning multiple lines.

source.sol:15:3: Warning: Function state mutability can be restricted to
pure
function div(uint256 a, uint256 b) internal constant returns (uint256) {
^
Spanning multiple lines.

source.sol:22:3: Warning: Function state mutability can be restricted to
pure
function sub(uint256 a, uint256 b) internal constant returns (uint256) {
^
Spanning multiple lines.

source.sol:27:3: Warning: Function state mutability can be restricted to
pure
function add(uint256 a, uint256 b) internal constant returns (uint256) {
^
Spanning multiple lines.
```

Solium output

source.sol

```
8:0 warning Library 'SafeMath' must be preceded by 2 blank lines. blank-
lines
9:2 error Only use indent of 4 spaces. indentation
9:2 warning FunctionDeclaration must be succeded by 1 blank line blank-
lines
13:0 error Only use indent of 4 spaces. indentation
15:2 warning FunctionDeclaration must be succeded by 1 blank line blank-
lines
15:2 error Only use indent of 4 spaces. indentation
20:0 error Only use indent of 4 spaces. indentation
22:2 warning FunctionDeclaration must be succeded by 1 blank line blank-
lines
22:2 error Only use indent of 4 spaces. indentation
25:0 error Only use indent of 4 spaces. indentation
27:2 error Only use indent of 4 spaces. indentation
31:0 error Only use indent of 4 spaces. indentation
35:4 error No visibility specified explicitly for totalSupply function.
security/enforce-explicit-visibility
36:4 error No visibility specified explicitly for balanceOf function.
security/enforce-explicit-visibility
37:4 error No visibility specified explicitly for transfer function.
security/enforce-explicit-visibility
38:4 error No visibility specified explicitly for transferFrom function.
security/enforce-explicit-visibility
39:4 error No visibility specified explicitly for approve function.
security/enforce-explicit-visibility
40:4 error No visibility specified explicitly for allowance function.
security/enforce-explicit-visibility
45:0 warning Contract 'MithrilOre' must be preceded by 2 blank lines.
blank-lines
47:29 error 'Token 0.1': String Literals must be quoted with double quotes
only. quotes
48:4 warning Constant name 'name' doesn't follow the UPPER_CASE notation
uppercase
49:4 warning Constant name 'symbol' doesn't follow the UPPER_CASE notation
uppercase
50:4 warning Constant name 'decimals' doesn't follow the UPPER_CASE
notation uppercase
61:4 error No visibility specified explicitly for MithrilOre function.
security/enforce-explicit-visibility
61:4 warning FunctionDeclaration must be succeded by 1 blank line blank-
lines
63:9 error Only use indent of 8 spaces. indentation
71:4 warning FunctionDeclaration must be succeded by 1 blank line blank-
lines
71:4 error No visibility specified explicitly for totalSupply function.
security/enforce-explicit-visibility
71:65 error Function 'totalSupply': Opening brace must be preceded by only
a single space. lbrace
74:4 warning FunctionDeclaration must be succeded by 1 blank line blank-
lines
74:4 error No visibility specified explicitly for balanceOf function.
security/enforce-explicit-visibility
74:73 error Function 'balanceOf': Opening brace must be preceded by only a
single space. lbrace
77:4 warning FunctionDeclaration must be succeded by 1 blank line blank-
lines
```

```
77:4 error No visibility specified explicitly for transfer function.
security/enforce-explicit-visibility
82:6 error Only use indent of 8 spaces. indentation
83:6 error Only use indent of 8 spaces. indentation
84:6 error Only use indent of 8 spaces. indentation
86:4 error No visibility specified explicitly for transferFrom function.
security/enforce-explicit-visibility
86:4 warning FunctionDeclaration must be succeded by 1 blank line blank-
lines
86:92 error Function 'transferFrom': Opening brace must be preceded by only
a single space. lbrace
88:24 error There should be a maximum of single space and no comments
between left side and '&&'. operator-whitespace
88:50 error Operator "&&" should be on the line where left side of the
Binary expression ends. operator-whitespace
91:0 error Only use indent of 8 spaces. indentation
93:6 error Only use indent of 8 spaces. indentation
94:6 error Only use indent of 8 spaces. indentation
95:6 error Only use indent of 8 spaces. indentation
96:6 error Only use indent of 8 spaces. indentation
98:4 warning FunctionDeclaration must be succeded by 1 blank line blank-
lines
98:4 error No visibility specified explicitly for approve function.
security/enforce-explicit-visibility
98:77 error Function 'approve': Opening brace must be preceded by only a
single space. lbrace
104:4 error No visibility specified explicitly for allowance function.
security/enforce-explicit-visibility
104:4 warning FunctionDeclaration must be succeded by 1 blank line blank-
lines
104:93 error Function 'allowance': Opening brace must be preceded by only a
single space. lbrace
106:0 error Only use indent of 4 spaces. indentation
✖ 39 errors, 15 warnings found.
```

Solhint output

```
1:17 warning Compiler version must be fixed compiler-fixed
8:1 error Definition must be surrounded with two blank line indent two-
lines-top-level-separator
9:3 error Expected indentation of 4 spaces but found 2 indent
10:5 error Expected indentation of 8 spaces but found 4 indent
11:5 error Expected indentation of 8 spaces but found 4 indent
12:5 error Expected indentation of 8 spaces but found 4 indent
13:3 error Expected indentation of 4 spaces but found 2 indent
15:3 error Expected indentation of 4 spaces but found 2 indent
17:5 error Expected indentation of 8 spaces but found 4 indent
19:5 error Expected indentation of 8 spaces but found 4 indent
20:3 error Expected indentation of 4 spaces but found 2 indent
22:3 error Expected indentation of 4 spaces but found 2 indent
23:5 error Expected indentation of 8 spaces but found 4 indent
24:5 error Expected indentation of 8 spaces but found 4 indent
25:3 error Expected indentation of 4 spaces but found 2 indent
27:3 error Expected indentation of 4 spaces but found 2 indent
28:5 error Expected indentation of 8 spaces but found 4 indent
29:5 error Expected indentation of 8 spaces but found 4 indent
30:5 error Expected indentation of 8 spaces but found 4 indent
31:3 error Expected indentation of 4 spaces but found 2 indent
```

```
34:1 error Definition must be surrounded with two blank line indent two-
lines-top-level-separator
35:28 warning Explicitly mark visibility in function func-visibility
36:40 warning Explicitly mark visibility in function func-visibility
37:52 warning Explicitly mark visibility in function func-visibility
38:71 warning Explicitly mark visibility in function func-visibility
39:56 warning Explicitly mark visibility in function func-visibility
40:58 warning Explicitly mark visibility in function func-visibility
41:5 warning Event and function names must be different no-simple-event-
func-name
45:1 error Definition must be surrounded with two blank line indent two-
lines-top-level-separator
47:30 error Use double quotes for string literals quotes
48:28 error Constant name must be in capitalized SNAKE_CASE const-name-
snakecase
49:28 error Constant name must be in capitalized SNAKE_CASE const-name-
snakecase
50:27 error Constant name must be in capitalized SNAKE_CASE const-name-
snakecase
61:5 error Definitions inside contract / library must be separated by one
line separate-by-one-line-in-contract
61:27 warning Explicitly mark visibility in function func-visibility
63:10 error Expected indentation of 8 spaces but found 9 indent
71:28 warning Explicitly mark visibility in function func-visibility
71:66 error Open bracket must be indented by other constructions by space
bracket-align
74:40 warning Explicitly mark visibility in function func-visibility
74:5 error Definitions inside contract / library must be separated by one
line separate-by-one-line-in-contract
74:74 error Open bracket must be indented by other constructions by space
bracket-align
77:5 error Definitions inside contract / library must be separated by one
line separate-by-one-line-in-contract
77:52 warning Explicitly mark visibility in function func-visibility
77:5 warning Event and function names must be different no-simple-event-
func-name
82:7 error Expected indentation of 8 spaces but found 6 indent
83:7 error Expected indentation of 8 spaces but found 6 indent
84:7 error Expected indentation of 8 spaces but found 6 indent
86:5 error Definitions inside contract / library must be separated by one
line separate-by-one-line-in-contract
86:93 error Open bracket must be indented by other constructions by space
bracket-align
86:71 warning Explicitly mark visibility in function func-visibility
88:23 error Expression indentation is incorrect. Required no spaces before
[ expression-indent
93:7 error Expected indentation of 8 spaces but found 6 indent
94:7 error Expected indentation of 8 spaces but found 6 indent
95:16 error Expression indentation is incorrect. Required no spaces before
( expression-indent
95:7 error Expected indentation of 8 spaces but found 6 indent
96:7 error Expected indentation of 8 spaces but found 6 indent
98:5 error Definitions inside contract / library must be separated by one
line separate-by-one-line-in-contract
98:56 warning Explicitly mark visibility in function func-visibility
98:78 error Open bracket must be indented by other constructions by space
bracket-align
104:58 warning Explicitly mark visibility in function func-visibility
104:94 error Open bracket must be indented by other constructions by space
bracket-align
```

```
106:1 error Expected indentation of 4 spaces but found 0 indent
107:5 error Definitions inside contract / library must be separated by one
line separate-by-one-line-in-contract
107:5 warning Event and function names must be different no-simple-event-
func-name

✖ 64 problems (47 errors, 17 warnings)
```