HASHLAB

Smart Contract audit WELL.sol

Abstract

In this report we consider the security of the ERC-20 Token and ICO contracts of WELL.SOL. Our task is to find and describe any security issues in the Smart Contracts.

Analysis technique

We used several publicly available automated Solidity analysis tools, as well as proceed manual analysis. All the issues found by tools were manually checked (rejected or confirmed). Contracts were manually analyzed

Bugs classification:

CRITICAL - problems leading to stealing funds from any of the participants, or making them inaccessible by anyone
SEVERE - problems that can stop, freeze or break the internal logic of the contract
WARNING - non-critical problems that cannot break the contract, but contract code does not match declared in WhitePaper logic
Notes - any other findings

HASHLAB

Smart Contract audit WELL.sol

WARNING

Contract: Well

Function names: transfer(address, uint), transferFrom(address, address, uint)

In well_smart.sol:385

```
require(_to != address(this) && _to != address(0));
```

In well_smart.sol:396

```
require(_to != address(this) && _to != address(0));
```

Description:

Non-critical problems that cannot break the contract, but contract code does not match declared in WhitePaper logic
In function transfer(address, uint) there is a redundant requirement _to != address(0) already exists in ancestor class BasicToken. In function transferFrom(address, uint) there is a redundant requirement _to != address(0) already exists in ancestor class StandartToken. In case of a non-zero address, additional gas consumption occurs. Presumably such case happens less often.

Solution:

```
require(_to != address(this));
```

HASHLAB

Smart Contract audit WELL.sol

WARNING

Contract: Well

Function names: enableTransfer()

In well_smart.sol:422-429

```
/**
* @dev Function to stop transfering tokens.
* @return True if the operation was successful.
*/
function enableTransfer() onlyOwner returns (bool) {

transferEnabled = true;

   return true;
}
```

Description:
The function description contradicts its name and body.

Smart Contract audit WELL.sol

WARNING

Contract: Well

Function names: buyTokens(address)

Description:

Might be better to check for 0 wei transaction as this will reduce gas consumptions if check succeeds.

Solution:

```
require(msg.value != 0)
```

Smart Contract audit WELL.sol

WARNING

Contract: Well

Function names: buyTokens(address)

In well_smart.sol:475

```
mintInternal(beneficiary, tokens);
```

Description:

There is an unchecked return value. It can cause errors if something goes wrong.

Solution:

```
require(mintInternal(beneficiary, tokens));
```

Smart Contract audit WELL.sol

WARNING

Contract: Well

Function names: getBonusByDate()

Description:
Function should be marked as `view` . In such case it promise not to modify the state and can be run free of gas.

Solution:

```
function getBonusByDate() view returns (uint256)
```

HASHLAB

Smart Contract audit WELL.sol

WARNING

Contract: Well

Function names: getBonusByDate()

Description:
Function can be rewritten to use less gas.

Solution:

```
function getBonusByDate() view returns (uint256) { if
(block.timestamp < 1514764800)
return 0;
if (block.timestamp < 1521158400)
return 40;
if (block.timestamp < 1523836800)
return 30;
if (block.timestamp < 1523923200)
return 25;
if (block.timestamp < 1524441600)
return 20;
if (block.timestamp < 1525046400)
return 10;
if (block.timestamp < 1525651200)
      return 5;
   return 0;
}
```

HASHLAB

Smart Contract audit WELL.sol

Vulnerability checking

We have checked smart contracts for vulnerabilities described above. None of our checks showed vulnerability. So, the probability of their appearance in the smart contracts is low.

We have scanned contract for common and several specific vulnerabilities. Here is a part of them:

Reentrancy (not found)

Timestamp Dependence (not found)

Gas Limit and Loops (not found)

DoS with (Unexpected) Throw (not found)

DoS with Block Gas Limit (not found)

Transaction-Ordering Dependence (not found)

Exception disorder (not found)

Gasless send (not found)

HΔSHLΔB

Smart Contract audit WELL.sol

Conclusion

After complete manual analysis of smart contracts and vulnerability checking we conclude that this contract has high code quality and security. However, we found a list of WARNING issues which are not affect the smart contract in general and can be easily solved using our decisions