Code Review and Security Analysis Report
May 23, 18

Report For: Origin Sport

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 of Origin Sport contract
Source code version	https://github.com/OriginSport/originsport-token commit: d01bd61f9da8ab8a3d1bd15d16b0cff6a61664e3
Version	Audit report v2, final version.
Date:	23/05/18

Origin Sport contract

Executive Summary

Hacken OÜ (Consultant) was contracted by Origin Sport (Customer) to conduct an Smart Contract Code Review and Security Analysis against token and crowd sale smart contracts.
We thank Origin Sport for giving us the opportunity to audit your smart contract code. We looked that source code and now share our results.

This report summary represents the findings of a security assessment of Origin Sport remastered contract conducted between the periods 18th of May 2018 to 23th of May 2018.

Automated analysis

We scan project's smart contracts with several publicly available automated Solidity analysis tools such as Remix, Mythril and Solhint. We manually verify (reject or confirm) all the issues found by tools.

Manual audit

Contracts were completely manually analyzed; their logic was checked. Besides, the results of automated analysis were manually verified and considered.

Report

We report all the issues found to the developer during the audit process, we check the issues fixed by the developer and reflect all the gathered information in the report

Checked vulnerabilities

We have scanned Origin Sport 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
- Byte array
- 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

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.

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.

Analysis

Manual audit

The full architecture consists of crowdsale logic and token implementation. Additional logic is the airdrop. It's allow owner to make a drop for recipients.

The base of crowdsale is the OriginSportTokenSale.sol contract.This is pausable contract. The OriginSportTokenSale.sol contract has basic functional like own initialization, token initialization, buying tokens, some utility modifiers, custom validation functions and time settings with getting rate by date. Also, contract has fallback function, which allows token to be purchased by directly sending ether to this smart contract.

Apart from crowdsale architecture there is a token architecture. It based on StandardToken, Ownable and Burnable token interface implementation. Contract allow owner to add address to transfer tokens, allow all users use transfer, transferFrom and batch transfer functions with modifier to prevent from transfer with invalid address, burn function with modifier to prevent burn while untransferable.

Also, in this project publicly available contracts are Zeppelin contracts and libraries, like SafeMath.

Under the development hood is truffle, development framework for Ethereum. This allows import another contracts and keep modular project structure.

The project compiles successfully.

Test coverage

Project has tests for the business logic and blockchain interaction. Tests covered each contract and describe different use cases. Contract deployment implemented with contract mocks for flexibility. Most of all of critical functions covered by tests.

Initial testing configuration:

```
------------------------------------------------
publicSaleStartTime : 1527141600
publicSaleStartTime : Thu May 24 2018 09:00:00 GMT+0300 (EEST)
publicSaleEndTime   : 1528178400
publicSaleEndTime   : Tue Jun 05 2018 09:00:00 GMT+0300 (EEST)
------------------------------------------------
```

```
  Contract: OriginSportToken
    OriginSportToken initialization
      ✓ creation: should have an balance of 300 million tokens
      ✓ creation: test correct setting of vanity information (175ms)
    OriginSportToken transfer test
      ✓ transfers: should transfer amount to user1 with admin having amount (91ms)
      ✓ transfers: should transfer correctly (117ms)
      ✓ transfers: normal user can not transfer correctly (55ms)
      ✓ transfers: should handle zero-transfers normally (82ms)
      ✓ transfers: normal user can not transfer token when transferable is false (49ms)
      ✓ transfers: normal user can transfer correctly after add user1 to whitelist (247ms)
      ✓ transfers: normal user can transfer correctly after enable transfer (224ms)
      ✓ transfers: normal user can batch transfer correctly after enable transfer (373ms)
      ✓ transfers: should fail when trying to transfer greater than TOTAL_SUPPLY to user1 (44ms)
      ✓ transfers: should throw  when trying to transfer to 0x0 (53ms)
    OriginSportToken approvals test
      ✓ approvals: msg.sender should approve 100 to user1 (80ms)
      ✓ approvals: msg.sender approves user1 of 100 and withdraws 20 once (203ms)
    OriginSportToken burn test
      ✓ burn: total supply would decrease to 200 million (122ms)

  Contract: OriginSportTokenSale Test
    OriginSportToken sale initialization
      ✓ check property is all correct (149ms)
      ✓ contribute when token sale is not start will failed (61ms)
    Assume token sale is started and check basic contribute function
      ✓ check this sale contract can transfer tokens
      ✓ contribute when token sale is started will success (177ms)
      ✓ contribute amount less than minimum will failed (58ms)
      ✓ if contract paused, every contribute will failed (162ms)
      ✓ after contract unpaused, contribute will return normal (187ms)
      ✓ check rate in first 5 days
      ✓ check rate in second 5 days (1186ms)
      ✓ check rate in last 5 days (1096ms)
      ✓ if sale has reached end time, contribute will failed (2127ms)


  26 passing (12s)
```

Testing contracts by use cases:
Everything passed correctly.
Critical issues

Critical issues seriously endanger smart contracts security. We highly recommend fixing them.
The analysis showed no critical issues.

Medium severity issues

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

1. Pragmas version

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

```
pragma solidity 0.4.17; // old: compiles w 0.4.17
pragma solidity 0.4.24; // new: compiles w 0.4.24
```

We recommend using the latest compiler version.

2. Functions transfer and transferFrom of ERC-20 Token should throw

Functions of ERC-20 Token Standard should throw in special cases:

- transfer should throw if the _from account balance does not have enough tokens to spend
- transferFrom should throw unless the _from account has deliberately authorized the sender of the message via some mechanism

Recommendation: The ERC20 standard recommends throwing exceptions in functions transfer and transferFrom.

Low severity issues

The analysis showed several low severity issues that were eliminated by the developers during the audit process

1. Invoking events without "emit" prefix is deprecated

OriginSportToken.sol:47

```
function OriginSportToken(address admin) validAddress(admin) public {
  require(msg.sender != admin);
  whitelistedTransfer[admin] = true;
  totalSupply_ = INITIAL_SUPPLY;
  balances[admin] = totalSupply_;
  Transfer(address(0x0), admin, totalSupply_);
  transferOwnership(admin);
}
```

OriginSportTokenSale.sol:78

```
function buyTokens(address _beneficiary) payable whenNotPaused inProgress public {
  require(msg.value >= MINIMAL_CONTRIBUTION);
  require(!hardCapReached());
  uint rate = getRate();
  uint orsAmount = msg.value.mul(rate);
  token.transfer(_beneficiary, orsAmount);
  forwardFunds();
  weiRaised = weiRaised.add(msg.value);
  tokenSold = tokenSold.add(orsAmount);
  LogContribute(_beneficiary, msg.value, orsAmount);
  if (weiRaised > HARD_CAP) {
    finalizeSale();
  }
}
```

OriginSportTokenSale.sol:127

```
function finalizeSale() internal {
  finalized = true;
  Finalized(tokenSold, weiRaised);
}
```

2. Defining constructors as functions with the same name as the contract is deprecated.

OriginSportTokenSale.sol:45

```
function OriginSportTokenSale(uint _startTime, uint _endTime, address _token, address
_wallet) public {
  require(_endTime > _startTime + 10 days);
  require(_startTime >= now);
  startTime = _startTime;
  endTime = _endTime;
  token = OriginSportToken(_token);
  wallet = _wallet;
}
```

OriginSportToken.sol:42

```
function OriginSportToken(address admin) validAddress(admin) public {
  require(msg.sender != admin);
  whitelistedTransfer[admin] = true;
  totalSupply_ = INITIAL_SUPPLY;
  balances[admin] = totalSupply_;
  Transfer(address(0x0), admin, totalSupply_);
  transferOwnership(admin);
}
```

Style Guide Violations

Not necessary to resolve but increase readability and enforces style guide standards accepted by the global community. An example image from the contract has been added for each violation but similar violations may repeat themselves within the contract.

```
contracts/OriginSportTokenSale.sol
  11:24  error    Constant name must be in capitalized SNAKE_CASE                const-name-snakecase

contracts/OriginSportToken.sol
  15:26  error    Constant name must be in capitalized SNAKE_CASE        const-name-snakecase
  16:26  error    Constant name must be in capitalized SNAKE_CASE        const-name-snakecase
  17:26  error    Constant name must be in capitalized SNAKE_CASE        const-name-snakecase
  42:44  error    Visibility modifier must be first in list of modifiers    visibility-modifier-order
  56:53  error    Visibility modifier must be first in list of modifiers    visibility-modifier-order
  63:29  error    Visibility modifier must be first in list of modifiers    visibility-modifier-order
  115:3  warning  Event and function names must be different             no-simple-event-func-name
```

Automated audit

Mythril

Successfully scanned contract without any concerns.

Conclusion

In general, code quality is high and our feedback is very positive.