SmartDec

YOP Security Analysis

This report is private.

Published: January 14, 2021

Abstract

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

Disclaimer

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

Summary

In this report, we considered the security of YOP token smart contract. We performed our audit according to the procedure described below.

The initial audit showed a code logic issue that results in maximum cap to be `10 ** 8` times lower than it should be. Also, two low severity issues were found.

After the initial audit, the contract was updated to the second version. All the issues mentioned in the initial audit were fixed.

Then, the contract was deployed. We have checked the deployed code and compared it with updated documentation. No changes that endanger project security were found. Token distribution between the groups corresponds with the documentation. However, the code cannot guarantee that token distribution inside the groups will match the documentation.

General recommendations

We do not have any further recommendations.

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- manual audit
  - we manually analyze code base for security vulnerabilities
  - we assess overall project structure and quality
- report
  - we reflect all the gathered information in the report

Project overview

Project description

For the audit, we were provided with a file with the token smart contract of YOP project **YOP.sol**, sha1sum 725fa9444afecf6941ec1d499e3a2a3ee6f6f27c, and the documentation **DefiYOP Deck 2020 V0.2.pdf**, sha1sum 50ec8e992ec53b6dfaab6794a41839c827fbd33c.

The total LOC of audited sources is 239.

Second version of the code

After the initial audit, the code of the contract was updated. For the recheck, we were provided with **YOP.sol** file, sha1sum 3f16b37f96a274b66aacfb7f268d113ea8c0dc6f.

Deployed code

After the recheck, the contract was deployed.

The documentation was updated:

- **Yoponomics_Jan_2021.pdf**,

  sha1sum 8799f42fe7885b66e683eca4d652183a5ff918e5
- **YOP_Deck_Jan_2021.pdf**,

  sha1sum 086b6a5a5df9e9484a85d23d76ae3e8ea4cd12ff

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the 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 project operation in current implementation. We highly recommend addressing them.

Code logic (fixed)

In the current implementation, `maxCap` is set to `88888888`:

```
uint256 constant maxCap = 88888888;
```

However, `_decimals` value is set to `8`. Therefore, all the interfaces will consider maximum cap to be `0.88888888` tokens. To set maximum cap to `88888888` tokens, it should be declared as `88888888 * (10 ** 8)`. In this case, it is better to put `8` into a constant.

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

Low severity issues

Low severity issues can influence project operation in future versions of code. We recommend taking them into account.

Inconsistent comment (fixed)

The comment at lines 379–380 states:

```
Sets the values for {name} and {symbol}, initializes {decimals} with a
default value of 18.
```

But `_decimals` value is set to `8`. Moreover, the comment at line 382 states:

```
To select a different value for {decimals}, use {_setupDecimals}.
```

However, `_setupDecimals()` function is not used in the code.

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

Compiler version not fixed (fixed)

The version of the compiler is not fixed in the contract.

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

```
pragma solidity ^0.7.4; // bad: compiles with 0.7.4 and above
pragma solidity 0.7.4; // good: compiles with 0.7.4 only
```

We recommend following the latter example, as future compiler versions may handle certain language constructions in a way the developers have not foreseen.

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

Notes

ERC20 approve issue

There is ERC20 approve issue: changing the approved amount from a nonzero value to another nonzero value allows a double spending with a front-running attack.

We recommend instructing users to follow one of two ways:

- not to use `approve()` function directly and to use `increaseApproval()`/`decreaseApproval()` functions instead
- 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

Token distribution

According to the documentation, tokens are distributed the following way:

10,666,668	Private Sale	12%	Early Backers, Long Term Holders
2,666,669	Liquidity Reserve	3%	Uniswap Pool
13,333,333	Future Token Sale	15%	VC, Strategic Growth Partners
8,888,888	Team	10%	2 Year Linear Release V0 onwards
13,333,333	Marketing / Airdrop	15%	Building Community Engagement
13,333,333	YOP Rewards	15%	YOP Rewards in App
8,888,888	Advisors and Diplomats	10%	Advisory Tokens and Diplomat Reward Plan
8,888,888	Dev Pool	10%	Bug Bounties, Dev Contests
8,888,888	Reserve	10%	Balance Burnt Post V2

In the code, there are only four groups specified:

```
// 27% token sale
// 53% marketing and Dev rewards
// 10% team
// 10% reserve
```

Private Sale and Future Token Sale add up to 27%. Liquidity Reserve, Marketing, YOP rewards, Advisors, and Dev Pool add up to 53%. Reserve is 10%. Thus, these totals match the three functions in the contract.

However, there is no guarantee that token distribution inside these groups will match the documentation.

This analysis was performed by SmartDec.

Evgeny Marchenko, Senior Security Engineer
Igor Sobolev, Security Engineer
Boris Nikashin, Analyst
Alexander Seleznev, Head of Blockchain Department

January 14, 2021