Smart Contract Security Audit Report

HydroSwap

SECBIT

August 20, 2018

1. *Introduction*

HydroSwap is an exchange contract based on 0x. It is designed to trade ERC20 tokens and ETH. SECBIT Labs conducted an audit from August 12th, 2018 to August 14th, 2018, including an analysis of the contract in 3 areas: **code bugs**, **logic flaws** and **risk assessment**. The assessment shows that HydroSwap has no critical security risks, and SECBIT team has some tips on potential risks (see part 4 for details).

Type	Description	Level
Potential Risk	HydroSwap requires extra fees for trading	Low

2. Contract Information

This part describes basic contract information and code structure.

2.1 Basic Information

The following list shows basic information of HydroSwap:

Name	HydroSwap
Line	77
Source	The project team
File Hash	62561c7c55c4864e00288ac5e3ab2f671ce436e0
Stage	Deploying Ready
Dependencies	WETH, Exchange(0x), TokenTransferProxy(0x)

2.2 Function List

The following content shows the functions included in HydroSwap:

Name	Lines	Description
constructor	6	Constructor, sets the addresses of WETH, Exchange(0x), TokenTransferProxy(0x)
swap	6	Exchange tokens or ETH

3. Contract Analysis

This part describes details of contract code assessment, including functions of HydroSwap and dependent contracts.

3.1 HydroSwap

HydroSwap has two functions: constructor and swap()

1. constructor

   Sets exchangeAddress, tokenProxyAddress, wethAddress, those addresses cannot be reset afterwards.

   exchangeAddress: address of token exchanging contract

   tokenProxyAddress: address of token transfer proxy contract

   wethAddress: address of WETH token contract

2. swap()

   This is the core of HydroSwap. A token transferring contract based on 0x.

   There are 2 roles in 0x: taker and maker. Maker creates order forms and signs, taker fills the form. This contract acts as an intermediary replacing 0x takers, which simplifies the process of maker signing and transferring between ETH and tokens.

1. Taker (function caller) transfers ETH/tokens to HydroSwap.
    - Transfers to WETH contract if it is ETH, HydroSwap gains the corresponding number of WETH.
    - Approves a sufficient balance before calling this method if it is token.
2. The current contract authorizes the proxy contract of transferring certain tokens.
3. The current contract fills the order form in Exchange contract to complete a transfer and throws an exception if fails.
4. The current contract transfers the preset ETH or tokens to taker.

3.2 Dependent Contracts

1. ETH tokenized contract (WETH)

   Transfers ETH to this contract and get the corresponding WETH, or converts WETH to ETH and transfers ETH out like tokens.
2. 0x token transfer contract (Exchange)

   Token transfer contract. The only function relevant to HydroSwap is `fillOrder()`.
3. 0x proxy contract (TokenTransferProxy)

   The proxy contract approving transfers in Exchange contract.

4. Audit Detail

This part describes the process and detailed results of the audit, also demonstrates the problems and potential risks.

4.1 Audit Process

The audit strictly followed the audit specification of SECBIT Lab. We analyzed the project from code bug, logical implementation and potential risks. The process consists of four steps:

- Fully analysis of contract code line by line.
- Evaluation of vulnerabilities and potential risks revealed in the contract code.
- Communication on assessment and confirmation.
- Audit report writing.

4.2 Audit Result

After scanning with SECBIT Solidity Static Analysis Extension & sf-checker (internal version) developed by SECBIT Labs and Mythril (a security analysis tool, version 0.8.19), the auditing team performed a manual assessment. The team inspected the contract line by line and the result could be categorized into fourteen types:

Number	Classification	Result
1	Normal functioning of features defined by the contract	√
2	No obvious bug (e.g. overflow, underflow)	√
3	Pass Solidity compiler check with no potential error	√
4	Pass common tools check with no obvious vulnerability	√
5	No obvious gas-consuming operation	√
6	No risk in low level call (call, delegatecall, callcode) and in-line assembly	√
7	No deprecated or outdated usage	√
8	Explicit implementation, visibility, variable type and Solidity version number	√
9	No redundant code	√
10	No potential risk manipulated by timestamp and network environment	√
11	Explicit business logic	√
12	Implementation consistent with annotation and other info	√
13	No hidden code about any logic that is not mentioned in design	√
14	No ambiguous logic	√

4.3 Risks

SECBIT team found the following risk after assessing HydroSwap:

- HydroSwap requires extra fees for trading
    - Level: **Low**
    - Description:

      Transfer senders and receivers need to pay for extra fees in 0x token transactions. Part of the fee on takers are set to be paid by HydroSwap, while in actual cases, the real taker(the caller of HydroSwap) does not pay any fee in exchanging tokens with the contract. Thus, all takers' fees are paid by HydroSwap.
        - The transfer would fail when HydroSwap does not possess enough ZRX tokens for fees.
        - If the fee is of high value, there might be attacks of constructing certain trades to extract fees instead.

5. Conclusion

SECBIT team had found no critical code bug or flaw after analyzing HydroSwap. HydroSwap is a token exchange contract based on 0x, fulfilling the function of trading tokens and ETH ingeniously. Meanwhile, the contract reveals a potential risk as demonstrated above.

Disclaimer

SECBIT smart contract audit service assesses the contract's correctness, security and performability in code quality, logic design and potential risks. The report is provided "as is", without any warranties about the code practicability, business model, management system's applicability and anything related to the contract adaptation. This audit report is not to be taken as an endorsement of the platform, team, company or investment.

APPENDIX

Vulnerability/Risk Level Classification

Level	Description
High	Severely damage the contract's integrity and allow attackers to steal ethers and tokens, or lock ethers inside the contract.
Medium	Damage contract's security under given conditions and cause impairment of benefit for stakeholders.
Low	Cause no actual impairment to contract.
Info	Relevant to practice or rationality of the contract, could possibly bring risks.

SECBIT Lab is devoted to construct a common-consensus, reliable and ordered blockchain economic entity.

 http://www.secbit.io

 audit@secbit.io

 @secbit_io