Security Assessment

YouSwap

Apr 28th, 2021

Summary

This report has been prepared for YouSwap smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Additionally, this audit is based on a premise that all external smart contracts are implemented safely.

The security assessment resulted in 18 findings that ranged from major to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	YouSwap
Description	Multi-chain interoperability DEX based on AMM model
Platform	Ethereum, Heco
Language	Solidity
Codebase	https://github.com/YouSwap/contracts/tree/4be34be8ee0fe3cafebdae2a2486cff7089f57b0
Commits	4be34be8ee0fe3cafebdae2a2486cff7089f57b0

Audit Summary

Delivery Date	Apr 28, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	18
Critical	0
Major	1
Minor	5
Informational	12
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
YSF	v1-core/YouSwapFactory.sol	26946481b07cbc1e1a9bfb8bca452a0cc3b3a0ca083c8f1bb1de9b9dbc93dbd4
RYS	v1-core/token/Repurchase.sol	51a80a6c29ac34709cddae72cb38e41102d71b159ddafcd3f79755a335ff5fca
YFV	v1-mining/implement/YouswapFactoryV1.sol	c16804e3f1229c798580075603e149ecfd985b67291aed74307b98b705c83fb8
YIV	v1-mining/implement/YouswapInviteV1.sol	c8eb2447899555cb8ac24832a00251beb16e919b020019fdf35d077598bdd66c
ECY	v1-mining/library/ErrorCode.sol	db820fbdb1e07cbce800c9092e8a753170a14c42bac44803d606f1e0d30ed1d8
YSR	v1-periphery/YouSwapRouter.sol	0518dc42dbaa085c4ae4c874fd67730637ac74087070e12ec482de7bec938ad1
YSL	v1-periphery/libraries/YouSwapLibrary.sol	98abec347133e2d4b2005df5d2fedbdfea4405a9b790064dd7d97b36b986c539

Findings

Critical	0 (0.00%)
Major	1 (5.56%)
Minor	5 (27.78%)
Informational	12 (66.67%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
RYS-01	Missing Emit Events	Coding Style	Informational	Declined
RYS-02	Proper Usage of “public” And “external” Type	Gas Optimization	Informational	Declined
RYS-03	Missing Check For The Result of Transfer	Logical Issue	Informational	Declined
RYS-04	Centralization Issue	**Centralization / Privilege**	**Minor**	**Acknowledged**
RYS-05	Addresses On Testnet	Logical Issue	Informational	Acknowledged
YFV-01	Missing Emit Events	Coding Style	Informational	Declined
YFV-02	Incorrect Statement In Require	Logical Issue	Informational	Declined
YFV-03	Add Modifiers For Checking Variables	Logical Issue	Informational	Declined
YFV-04	Check Effect Interaction Pattern Violated	Logical Issue	Minor	Acknowledged
YFV-05	Discussion For YouswapInviteV1.inviteBatch()	Logical Issue	Informational	Declined
YIV-01	Check The Level of The Inviter	Logical Issue	Minor	Declined
YIV-02	Invite Persons Without Allowance	Logical Issue	Minor	Acknowledged
YIV-03	The Invite List Has No Limit	Logical Issue	Informational	Declined

ID	Title	Category	Severity	Status
YSF-01	Missing Zero Address Validation	Logical Issue	Informational	Declined
YSL-01	Hard Code For Init Hash Code	Logical Issue	Minor	Declined
YSR-01	Missing Emit Events	Coding Style	Informational	Declined
YSR-02	Missing Zero Address Validation	Logical Issue	Informational	Declined
YSR-03	Unimplemented Method	Logical Issue	Major	Acknowledged

RYS-01 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	v1-core/token/Repurchase.sol: 85, 90, 122, 132	Declined

Description

Some functions should be able to emit events as notifications to customers because they change the status of sensitive variables or call important processes. This suggestion is not limited to these codes but also applies to other similar codes.

Recommendation

Consider adding an emit after changing the status of variables or calling important processes.

Alleviation

No alleviation.

RYS-02 | Proper Usage of “public” And “external” Type

Category	Severity	Location	Status
Gas Optimization	Informational	v1-core/token/Repurchase.sol: 84, 88, 93, 98, 111, 130	Declined

Description

`public` functions that are never called by the contract could be declared `external`.

Recommendation

Consider using the `external` attribute for functions never called from the contract.

Alleviation

No alleviation.

RYS-03 | Missing Check For The Result of Transfer

Category	Severity	Location	Status
Logical Issue	Informational	v1-core/token/Repurchase.sol: 132	Declined

Description

The function `emergencyWithdraw()` does not check if the transfer process was successful.

Recommendation

Consider using `_safeTransfer()` instead of `transfer()` as below :

```
_safeTransfer(_token, emergencyAddress, IERC20(_token).balanceOf(address(this)));
```

Alleviation

[YouSwap] response: The balance of this contract is checked before the transfer process, the transfer process won't fail generally. Once transfer failed, there is no effect. The revert reason can be queried from blockchain explorer.

RYS-04 | Centralization Issue

Category	Severity	Location	Status
Centralization / Privilege	Minor	v1-core/token/Repurchase.sol: 71~74	Acknowledged

Description

In function `emergencyWithdraw()`, the `owner` can transfer all the `_token` in this contract to `emergencyAddresses`. When will this function be called and what is the `_token`? Please give an introduction to this function.

Alleviation

[YouSwap] response: The `_token` usually refers to `USDT`. This function was provided to retrieve tokens from this contract. For example, users transferred the wrong tokens to this contract.

RYS-05 | Addresses On Testnet

Category	Severity	Location	Status
Logical Issue	Informational	v1-core/token/Repurchase.sol: 70~73	Acknowledged

Description

The addresses `USDT`, `YOU`, `YOU_YSDT`, `destroyAddress` are addresses on testnet `https://ropsten.etherscan.io`, but not mainnet of `Ethereum` or `Heco`.

Recommendation

Consider changing related value to addresses of mainnet.

Alleviation

[YouSwap] response: These addresses will be replaced with the official addresses when deploying.

YFV-01 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	v1-mining/implement/YouswapFactoryV1.sol: 53, 63, 114, 419, 49	Declined

Description

Some functions should be able to emit events as notifications to customers because they change the status of sensitive variables or call important processes. This suggestion is not limited to these codes but also applies to other similar codes.

Recommendation

Consider adding an emit after changing the status of variables or calling important processes.

Alleviation

No alleviation.

YFV-02 | Incorrect Statement In Require

Category	Severity	Location	Status
Logical Issue	Informational	v1-mining/implement/YouswapFactoryV1.sol: 488, 499, 510	Declined

Description

The require statement in function `setName()`, `setMultiple()`, `setPriority()` should be the same as the require statement in function `setRewardPerBlock`, `setRewardTotal` as they are all set the variables of `pollInfo`.

Recommendation

Consider modifying the require statement as below :

```
require((address(0) != poolInfo.lp) && (0 == poolInfo.endBlock),
ErrorCode.POOL_NOT_EXIST_OR_END_OF_MINING);
```

Alleviation

No alleviation.

YFV-03 | Add Modifiers For Checking Variables

Category	Severity	Location	Status
Logical Issue	Informational	v1-mining/implement/YouswapFactoryV1.sol: 66, 93, 117, 336, 346, 455, 469, 485, 496, 507	Declined

Description

Before getting `pollInfo`, need to check if the variable `_pool` is less than `poolInfos`'s length. This can be added in a modifier.

Recommendation

Consider adding a modifier as below :

```
modifier checkPool(uint256 _pool) {
        require(_pool < poolInfos.length, ErrorCode.POOL_NOT_EXIST_OR_END_OF_MINING);
        _;
    }
```

Alleviation

No alleviation.

YFV-04 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	Minor	v1-mining/implement/YouswapFactoryV1.sol: 93~115	Acknowledged

Description

The order of external call/transfer and storage manipulation must follow a check effect interaction pattern. This suggestion is not limited to these codes but also applies to other similar codes.

Recommendation

We advise the client to check if storage manipulation is before the external call/transfer operation by considering the following modification:

```
        PoolInfo storage poolInfo = poolInfos[_pool];
        require((address(0) != poolInfo.lp) && (poolInfo.startBlock <= block.number),
ErrorCode.MINING_NOT_STARTED);
        if (0 < _amount) {
            UserInfo storage userInfo = pledgeUserInfo[_pool][msg.sender];
            require(_amount <= userInfo.amount, ErrorCode.BALANCE_INSUFFICIENT);
        }

        (address _upper1, address _upper2) = invite.inviteUpper2(msg.sender);

        computeReward(_pool);

        provideReward(_pool, poolInfo.rewardPerShare, poolInfo.lp, msg.sender, _upper1,
_upper2);

        if (0 < _amount) {
            subPower(_pool, msg.sender, _amount, _upper1, _upper2);
        }

        setRewardDebt(_pool, poolInfo.rewardPerShare, msg.sender, _upper1, _upper2);

         if (0 < _amount) {
            IERC20(poolInfo.lp).safeTransfer(msg.sender, _amount);

            emit UnStake(_pool, poolInfo.lp, msg.sender, _amount);
        }
```

Alleviation

[YouSwap] response: They are sure the function `IERC20(poolInfo.lp).safeTransfer()` is safe. There is no risk of reentrancy attack in this function.

YFV-05 | Discussion For YouswapInviteV1.inviteBatch()

Category	Severity	Location	Status
Logical Issue	Informational	v1-mining/implement/YouswapFactoryV1.sol: 107~135	Declined

Description

The key for mapping `inviteUserInfoV2` in function `inviteBatch()` is `msg.sender`, If this function is called by another contract, this will cause a problem.

Recommendation

Please confirm that this situation will not happen.

Alleviation

[YouSwap] response: The inviter can be a contract or a user. So this is not a problem.

YIV-01 | Check The Level of The Inviter

Category	Severity	Location	Status
Logical Issue	Minor	v1-mining/implement/YouswapInviteV1.sol: 85~137	Declined

Description

According to [YouSwap]'s response. The users who have two levels of superiors can not invite others. This logic should be checked in functions `acceptInvitation()` and `inviteBatch()`.

Recommendation

Consider adding a logic as below :

```
UserInfo storage inviter = inviteUserInfoV2[_inviter];
if(inviter.upper != ZERO) {
    UserInfo storage upper1 = inviteUserInfoV2[inviter.upper];
    if(upper1.upper != ZERO) {
          require(false, "_inviter has tow levels of superiors, so he can not invite
others!");
    }
}
```

Alleviation

[YouSwap] response: Users can have more than two levels of superiors or lowers, but only the nearest two levels of superiors can get inviting rewards.

YIV-02 | Invite Persons Without Allowance

Category	Severity	Location	Status
Logical Issue	Minor	v1-mining/implement/YouswapInviteV1.sol: 113~141	Acknowledged

Description

In function `inviteBatch()`, the `_invitees` can be added to the invite list without their own consent. In this case, someone may search addresses from the internet and invite these addresses. The addresses to be invited may lose the chance to invite others and get the invite rewards since they have two levels of superiors.

Alleviation

[YouSwap] response: First of all, `inviteBatch()` will spend the fee. If users plan to participate, they can change their addresses to participate if they are invited. This is just a publicity mechanism for the community to actively spend the fee to help and guide a large number of users to participate.

YIV-03 | The Invite List Has No Limit

Category	Severity	Location	Status
Logical Issue	Informational	v1-mining/implement/YouswapInviteV1.sol: 12	Declined

Description

The length of `YouswapInviteV1.inviteUserInfoV1()` can expand with no limit. When the users that are invited grows too large. i.e., more than 10 millions. Traverse them need consume a lot of gas which may exceed the gas limit of the blockchain.

Recommendation

Please make a limit of `YouswapInviteV1.inviteUserInfoV1()`.

Alleviation

[YouSwap] response: There is an invitee limitation. There is no case that needs to traverse a related array.

YSF-01 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	Informational	v1-core/YouSwapFactory.sol: 21, 47, 52	Declined

Description

Addresses should be checked before assignment to make sure they are not zero addresses.

Recommendation

Consider adding a zero check.

Alleviation

No alleviation.

YSL-01 | Hard Code For Init Hash Code

Category	Severity	Location	Status
Logical Issue	Minor	v1-periphery/libraries/YouSwapLibrary.sol: 24	Declined

Description

The variable `YouSwapFactory.initCodeHash` we calculated is `hex'6e6dd6fee5dcabca80a53f8492913bc6349d97ec296c9c105cddcae5e4232a0c'`, which is different from the hash code in `YouSwapLibrary.pairFor()`. This hash code is generated by `YouSwapPair.sol` and the files referenced by `YouSwapPair.sol`. So when these files are changed, the hash code is changed at the same time. Hard code may cause inconsistency.

Recommendation

Consider adding codes in `YouSwapLibrary.sol` as below :

```
    bytes32 public initCodeHash;

    constructor(address _factory) public {
        require(_factory != address(0), "_factory can not be zero address!");
        initCodeHash = YouSwapFactory(_factory).initCodeHash();
    }
```

Then use `initCodeHash` instead of hard code in `YouSwapLibrary.pairFor()` to make sure the hash code is correct even `YouSwapPair.sol` and the files referenced by `YouSwapPair.sol` are changed.

Alleviation

[YouSwap] response: The hash code in `YouSwapLibrary.pairFor()` used the value of variable `YouSwapFactory.initCodeHash` calculated during deploying process. They are sure that the hash code has no problem.

YSR-01 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	v1-periphery/YouSwapRouter.sol: 36	Declined

Description

Some functions should be able to emit events as notifications to customers because they change the status of sensitive variables or call important processes. This suggestion is not limited to these codes but also applies to other similar codes.

Recommendation

Consider adding an emit after changing the status of variables or calling important processes.

Alleviation

No alleviation.

YSR-02 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	Informational	v1-periphery/YouSwapRouter.sol: 27, 36	Declined

Description

Addresses should be checked before assignment to make sure they are not zero addresses.

Recommendation

Consider adding a zero check.

Alleviation

No alleviation.

YSR-03 | Unimplemented Method

Category	Severity	Location	Status
Logical Issue	Major	v1-periphery/YouSwapRouter.sol: 224~226, 344~346	Acknowledged

Description

The function `ISwapMining(swapMining).swap()` hasn't been implemented yet. This protocol can only be used after such functions to be implemented safely.

Recommendation

Consider implementing this function.

Alleviation

[YouSwap] response: This function is used for expansion in the future. The address of `swapMining` is `address(0)` now.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete .

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.