



Security Assessment

EarnX

Jul 1st, 2021

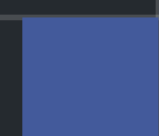


Table of Contents

Summary

Overview

Project Summary

Audit Summary

Vulnerability Summary

Audit Scope

Findings

EXE-01 : Variable `_rOwned` not updated

EXE-02 : Redundant code

EXE-03 : Incorrect error message

EXE-04 : Possible to gain ownership after renouncing the contract ownership

EXE-05 : Privileged ownership

EXE-06 : 3rd party dependencies

EXE-07 : Missing event emitting

EXE-08 : Function and variable naming doesn't match the operating environment

EXE-09 : Typos in the contract

EXE-10 : Potential sandwich attack

EXE-11 : Centralized risk in `addLiquidity`

EXE-12 : Leftover tokens cause by the unoptimized way to perform the one-sided supply.

Appendix

Disclaimer

About

Summary

This report has been prepared for Yearn Classic Finance to discover issues and vulnerabilities in the source code of the EarnX project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing 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.

The security assessment resulted in findings that ranged from critical 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	EarnX
Description	EarnX
Platform	BSC
Language	Solidity
Codebase	https://github.com/YearnClassic/EarnX/blob/main/EarnX.sol
Commit	a3a433e3b9668703619d5158805f5339394bb358

Audit Summary

Delivery Date	Jul 01, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

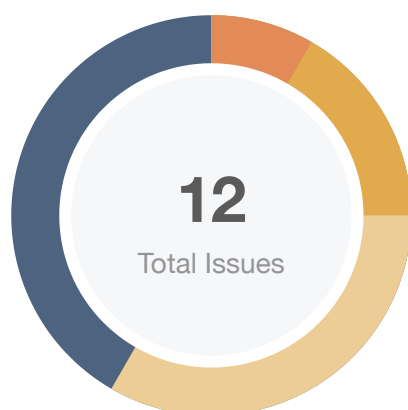
Vulnerability Summary

Vulnerability Level	Total Count	Pending	Partially Resolved	Resolved	Acknowledged	Declined
● Critical	0	0	0	0	0	0
● Major	1	0	0	0	1	0
● Medium	2	0	0	1	1	0
● Minor	4	0	0	2	2	0
● Informational	5	0	0	2	3	0
● Discussion	0	0	0	0	0	0

Audit Scope

ID	file	SHA256 Checksum
EXE	EarnX.sol	d0e4de3a738c637b68230a5ddc918d7bf4f170bf47ecc1ad78567b89f6150fb8

Findings



Critical	0 (0.00%)
Major	1 (8.33%)
Medium	2 (16.67%)
Minor	4 (33.33%)
Informational	5 (41.67%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
EXE-01	Variable <code>_rOwned</code> not updated	Control Flow	Minor	Resolved
EXE-02	Redundant code	Logical Issue	Informational	Resolved
EXE-03	Incorrect error message	Logical Issue	Minor	Resolved
EXE-04	Possible to gain ownership after renouncing the contract ownership	Logical Issue, Centralization / Privilege	Medium	Resolved
EXE-05	Privileged ownership	Centralization / Privilege	Medium	Acknowledged
EXE-06	3rd party dependencies	Control Flow	Minor	Acknowledged
EXE-07	Missing event emitting	Coding Style	Informational	Acknowledged
EXE-08	Function and variable naming doesn't match the operating environment	Coding Style	Informational	Acknowledged
EXE-09	Typos in the contract	Coding Style	Informational	Resolved
EXE-10	Potential sandwich attack	Logical Issue	Informational	Acknowledged
EXE-11	Centralized risk in <code>addLiquidity</code>	Centralization / Privilege	Major	Acknowledged
EXE-12	Leftover tokens cause by the unoptimized way to perform the one-sided supply.	Mathematical Operations	Minor	Acknowledged

EXE-01 | Variable `_rOwned` not updated

Category	Severity	Location	Status
Control Flow	Minor	EarnX.sol: 887~898	Resolved

Description

Variable `_rOwned` is not updated in function `includeAccount()`, which will make the accounts included siphons off the tokens out of the balances of all accounts that are currently holders. Details of this finding can be seen in this article from Pera Finance:

[Link]<https://perafinance.medium.com/safemoon-is-it-safe-though-a-detailed-explanation-of-frictionless-yield-bug-338710649846>

Recommendation

Sample code:

```
function includeAccount(address account) external onlyOwner() {
    require(!_isExcluded[account], "Account is already excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            uint256 currentRate = getRate();
            _rOwned[account] = _tOwned[account].mul(currentRate);
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

Alleviation

The update has been applied to EarnX.sol.

[<https://github.com/YearnClassic/EarnX/commit/8f050c86bdd4a18a789659db378585605f9e8ecc#diff-32502555365e03ac659fb4417d59c756d6f4d78fe4d3ae271bfdd3729fa7826a>]

EXE-02 | Redundant code

Category	Severity	Location	Status
Logical Issue	● Informational	EarnX.sol: 1161~1162	✓ Resolved

Description

The condition `!_isExcluded[sender] && !_isExcluded[recipient]` can be included in `else` .

Recommendation

The following code can be removed:

```
1 ... else if (!_isExcluded[sender] && !_isExcluded[recipient]) {  
2     _transferStandard(sender, recipient, amount);  
3 } ...
```

Alleviation

The update has been applied to EarnX.sol.

[<https://github.com/YearnClassic/EarnX/commit/8f050c86bdd4a18a789659db378585605f9e8ecc#diff-32502555365e03ac659fb4417d59c756d6f4d78fe4d3ae271bfdd3729fa7826a>]

EXE-03 | Incorrect error message

Category	Severity	Location	Status
Logical Issue	● Minor	EarnX.sol: 888	✓ Resolved

Description

The error message in `require(!_isExcluded[account], "Account is already excluded")` does not describe the error correctly.

Recommendation

The message "Account is already excluded" can be changed to "Account is not excluded" .

Alleviation

The update has been applied to EarnX.sol.

[<https://github.com/YearnClassic/EarnX/commit/8f050c86bdd4a18a789659db378585605f9e8ecc#diff-32502555365e03ac659fb4417d59c756d6f4d78fe4d3ae271bfdd3729fa7826a>]

EXE-04 | Possible to gain ownership after renouncing the contract ownership

Category	Severity	Location	Status
Logical Issue, Centralization / Privilege	● Medium	EarnX.sol	✓ Resolved

Description

An owner is possible to gain ownership of the contract even if he calls function `renounceOwnership` to renounce the ownership. This can be achieved by performing the following operations:

1. Call `lock` to lock the contract. The variable `_previousOwner` is set to the current owner.
2. Call `unlock` to unlock the contract.
3. Call `renounceOwnership` to leave the contract without an owner.
4. Call `unlock` to regain ownership.

Recommendation

We advise updating/removing `lock` and `unlock` functions in the contract; or removing the `renounceOwnership` if such a privilege retains at the protocol level. If timelock functionality could be introduced, we recommend using the implementation of Compound finance as reference. Reference: <https://github.com/compound-finance/compound-protocol/blob/master/contracts/Timelock.sol>

Alleviation

[EarnX Team] Removed lock unlock functions and `_previousOwner` variable to eliminate the bug, as of now we did not set the `_previousOwner` variable before ownership renounce so it is not possible to regain the ownership.

[<https://github.com/YearnClassic/EarnX/commit/8f050c86bdd4a18a789659db378585605f9e8ecc#diff-32502555365e03ac659fb4417d59c756d6f4d78fe4d3ae271bfdd3729fa7826a>]

EXE-05 | Privileged ownership

Category	Severity	Location	Status
Centralization / Privilege	● Medium	EarnX.sol	① Acknowledged

Description

The owner of contract `EarnX` has the permission to:

1. change the address that can receive LP tokens,
2. lock the contract,
3. exclude/include addresses from rewards/fees,
4. set `taxFee`, `liquidityFee` and `_maxTxAmount`,
5. start/finalize frictionless yield,
6. enable `swapAndLiquifyEnabled`

without obtaining the consensus of the community.

Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

Alleviation

[EarnX Team] Ownership to the contract is already renounced so it is not possible to alter the `taxFee`, `liquidityFee` and `_maxTxAmount` of the contract.

EXE-06 | 3rd party dependencies

Category	Severity	Location	Status
Control Flow	● Minor	EarnX.sol	ⓘ Acknowledged

Description

The contract is serving as the underlying entity to interact with third party PancakeSwap protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume its functional correctness. However in the real world, 3rd parties may be compromised that led to assets lost or stolen.

Recommendation

We understand that the business logic of the EarnX protocol requires the interaction PancakeSwap protocol for adding liquidity and swap tokens. We encourage the team to constantly monitor the statuses of those 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[EarnX Team]

Acknowledged as ownership to contract is renounced and deployed contract cannot be updated, decided to retain the code base unchanged.

EXE-07 | Missing event emitting

Category	Severity	Location	Status
Coding Style	● Informational	EarnX.sol	📄 Acknowledged

Description

In contract `EarnX`, there are a bunch of functions that can change state variables. However, these functions do not emit events to pass the changes out of the chain.

- `setTaxFeePercent()`
- `setNumTokensSellToAddToLiquidity()`
- `finalizeDxSale()`
- `finalizeListing()`

Recommendation

Recommend emitting events, for all the essential state variables that are possible to be changed during runtime.

Alleviation

[EarnX Team]

Acknowledged as ownership to contract is renounced and deployed contract cannot be updated, decided to retain the code base unchanged.

EXE-08 | Function and variable naming doesn't match the operating environment

Category	Severity	Location	Status
Coding Style	● Informational	EarnX.sol	ⓘ Acknowledged

Description

The EarnX contract uses Pancakeswap for swapping and add liquidity to Pancakeswap pool, but naming it Uniswap. Function swapTokensForEth(uint256 tokenAmount) swaps EarnX token for BNB instead of ETH.

Recommendation

Change "Uniswap" and "ETH" to "Pancakeswap" and "BNB" in the contract respectively to match the operating environment and avoid confusion.

Alleviation

[EarnX Team]

Acknowledged as ownership to contract is renounced and deployed contract cannot be updated, decided to retain the code base unchanged.

EXE-09 | Typos in the contract

Category	Severity	Location	Status
Coding Style	● Informational	EarnX.sol: 763, 956	✓ Resolved

Description

There are several typos in the code and comments.

1. In the following code snippet, `tokensIntoLiquidity` should be `tokensIntoLiquidity`.

```
1 event SwapAndLiquify(  
2     uint256 tokensSwapped,  
3     uint256 ethReceived,  
4     uint256 tokensIntoLiquidity  
5 );
```

2. `recieve` should be `receive` and `swaping` should be `swapping` in the line of comment `//to
recieve ETH from uniswapV2Router when swaping`.

Recommendation

We recommend correcting all typos in the contract.

Alleviation

The update has been applied to EarnX.sol.

[<https://github.com/YearnClassic/EarnX/commit/8f050c86bdd4a18a789659db378585605f9e8ecc#diff-32502555365e03ac659fb4417d59c756d6f4d78fe4d3ae271bfdd3729fa7826a>]

EXE-10 | Potential sandwich attack

Category	Severity	Location	Status
Logical Issue	● Informational	EarnX.sol	① Acknowledged

Description

Potential sandwich attacks could happen if calling

`uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens` and
`uniswapV2Router.addLiquidityETH` without setting restrictions on slippage.

For example, when we want to make a transaction of swapping 100 AToken for 1 ETH, an attacker could raise the price of ETH by adding AToken into the pool before the transaction so we might only get 0.1 ETH. After the transaction, the attacker would be able to withdraw more than he deposited because the total value of the pool increases by 0.9 ETH.

Recommendation

We recommend using Oracle to get an estimation of prices and setting minimum amounts based on the prices when calling the aforementioned functions.

Alleviation

[EarnX Team]

Acknowledged as ownership to contract is renounced and deployed contract cannot be updated, decided to retain the code base unchanged.

EXE-11 | Centralized risk in `addLiquidity`

Category	Severity	Location	Status
Centralization / Privilege	● Major	EarnX.sol: 1142~1149	ⓘ Acknowledged

Description

```
1 // add the liquidity
2 uniswapV2Router.addLiquidityETH{value: ethAmount}(
3     address(this),
4     tokenAmount,
5     0, // slippage is unavoidable
6     0, // slippage is unavoidable
7     owner(),
8     block.timestamp
9 );
```

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the `to` address specified as `owner()` for acquiring the generated LP tokens. As a result, over time the `_owner` address will accumulate a significant portion of LP tokens. If the `_owner` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

We advise the `to` address of the `uniswapV2Router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the `_owner` account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

[EarnX Team]

Acknowledged as ownership to contract is renounced and deployed contract cannot be updated, decided to retain the code base unchanged.

EXE-12 | Leftover tokens cause by the unoptimized way to perform the one-sided supply.

Category	Severity	Location	Status
Mathematical Operations	● Minor	EarnX.sol: 1095~1117	ⓘ Acknowledged

Description

The `swapAndLiquify` function converts half of the BNB to the target token. The other half of the BNB tokens and part of the converted tokens are deposited into the corresponding pool on PancakeSwap as liquidity.

A small amount of swapped tokens will be leftover in the contract for a `swapAndLiquify` function call with a large amount of BNB as the input. This is because the price of target tokens increases after swapping the first half of BNB into the target token, and the other half of BNB can pair with less than the converted amount of tokens when adding liquidity.

Recommendation

The article(<https://blog.alphafinance.io/onesideduniswap/>) from alpha finance explains the optimal way to perform the one-sided supply.

Note that the leftover token amount is negligible if the input BNB is a small number, or the liquidity pool has sufficient reserves. The team can decide if the optimization is necessary dependent on the contract use cases.

Alleviation

[EarnX Team]

Acknowledged as ownership to contract is renounced and deployed contract cannot be updated, decided to retain the code base unchanged.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Mathematical Operations

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

Logical Issue

Logical Issue findings 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.

Coding Style

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

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux `"sha256sum"` command against the target file.

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.

