



Security Assessment

Beyond Finance

Apr 13th, 2021



Summary

This report has been prepared for Beyond Finance 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.

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	Beyond Finance
Description	The smart contract is designed for Beyond Finance's Liquidity Generation Event
Platform	Ethereum
Language	Solidity
Codebase	Zip Folder
Commits	beyond-finance

Audit Summary

Delivery Date	Apr 13, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	11
● Critical	0
● Major	2
● Minor	0
● Informational	9
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
CKP	byn210413_1.sol	7297060da73fefed0378d4e631515cf78010ccc60771af5c4db26100103c8324
BYN	byn210413_3.sol	0406fcbd674a8085227305c2249e800d421319b6573e01ed59f3076e0a27e2b7

Centralized Issue

- The owner address is allowed to operate the user accounts by blocking or locking address through `manageAddress.sol`.

Findings



Critical	0 (0.00%)
Major	2 (18.18%)
Minor	0 (0.00%)
Informational	9 (81.82%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
CKP-01	Missing Error Message	Logical Issue	● Informational	⌚ Partially Resolved
CKP-02	Code Simplification	Gas Optimization	● Informational	✓ Resolved
CKP-03	Use Require Instead of Assert	Language Specific	● Informational	⚠ Pending
CKP-04	Code Simplification	Gas Optimization	● Informational	✓ Resolved
CKP-05	Redundant Settings Before Deleting	Language Specific	● Informational	✓ Resolved
CKP-06	Conditions Should Be Merged	Language Specific	● Informational	✓ Resolved
CKP-07	SafeMath Not In Used	Mathematical Operations	● Major	✓ Resolved
CKP-08	Check Input Validation <code>_percent</code>	Logical Issue	● Informational	⚠ Pending
CKP-09	Unused Return Variable Name	Language Specific	● Informational	⚠ Pending
CKP-10	Incorrect Event Emit	Logical Issue	● Major	✓ Resolved
CKP-11	Proper Usage of <code>public</code> and <code>external</code> Type	Gas Optimization	● Informational	⚠ Pending

CKP-01 | Missing Error Message

Category	Severity	Location	Status
Logical Issue	● Informational	byn210413_1.sol: 7, 10, 15, 18, 87, 92, 103, 107, 136, 140, 146, 170, 176, 184, 204, 265, 266, 267, 268, 293, 294, 295	⌚ Partially Resolved

Description

Error messages are helpful for users to identify the reasons of transaction reverts. Error messages in the aforementioned lines are missing.

Recommendation

We recommend adding error messages to calls of `require` or `revert`.

Alleviation

[Beyond Team]: The issues are addressed and reflected in the latest file `byn210413_3.sol`

CKP-02 | Code Simplification

Category	Severity	Location	Status
Gas Optimization	● Informational	byn210413_1.sol: 296~312	🟢 Resolved

Description

The aforementioned line can be simplify the logic for gas optimization.

Recommendation

We advice the code can be simplified as following:

```
296 if(lockTimeAddress[msg.sender]){
297     lockPermitBalance[msg.sender] = lockPermitBalance[msg.sender].sub(_value);
298 }
299 balanceOf[msg.sender] = balanceOf[msg.sender].sub(_value);
300 balanceOf[_to] = balanceOf[_to].add(_value);
301 emit Transfer(msg.sender, _to, _value);
302 return true;
```

Alleviation

[Beyond Team]: The issues are addressed and reflected in the latest file `byn210413_3.sol`

CKP-03 | Use Require Instead of Assert

Category	Severity	Location	Status
Language Specific	● Informational	byn210413_1.sol: 67	ⓘ Pending

Description

`assert` should only be used when the condition should never happen.

Recommendation

We recommend using `require` instead of `assert` at the aforementioned line.

CKP-04 | Code Simplification

Category	Severity	Location	Status
Gas Optimization	● Informational	byn210413_1.sol: 270~288	🟢 Resolved

Description

The aforementioned line can be simplify the logic for gas optimization.

Recommendation

We advice the code can be simplified as following:

```
270 if(lockTimeAddress[msg.sender]){
271     lockPermitBalance[_from] = lockPermitBalance[_from].sub(_value);
272 }
273 balanceOf[_from] = balanceOf[_from].sub(_value);
274 balanceOf[_to] = balanceOf[_to].add(_value);
275 allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
276 emit Transfer(_from, _to, _value);
277 return true;
```

Alleviation

[Beyond Team]: The issues are addressed and reflected in the latest file `byn210413_3.sol`

CKP-05 | Redundant Settings Before Deleting

Category	Severity	Location	Status
Language Specific	● Informational	byn210413_1.sol: 121~128	👍 Resolved

Description

It is not necessary to set variables to 0 or `false` before deleting them.

Recommendation

We recommend removing line 121 to 124.

Alleviation

[Beyond Team]: The issues are addressed and reflected in the latest file `byn210413_3.sol`

CKP-06 | Conditions Should Be Merged

Category	Severity	Location	Status
Language Specific	● Informational	byn210413_1.sol: 142~144	👍 Resolved

Description

The conditions at the aforementioned lines could be merged to simplify the implementation.

Recommendation

We recommend changing the aforementioned lines to

```
if(idx != 0 && lockTime[_address][idx - 1] >= _time) {  
    ...  
}
```

Alleviation

[Beyond Team]: The issues are addressed and reflected in the latest file `byn210413_3.sol`

CKP-07 | SafeMath Not In Used

Category	Severity	Location	Status
Mathematical Operations	● Major	byn210413_1.sol: 161, 205, 206, 190	✓ Resolved

Description

Integer overflow and underflow in operations in the aforementioned lines are not checked.

Recommendation

We recommend applying library `SafeMath` for integer operations at the aforementioned lines.

Alleviation

[Beyond Team]: The issues are addressed and reflected in the latest file `byn210413_3.sol`

CKP-08 | Check Input Validation `_percent`

Category	Severity	Location	Status
Logical Issue	● Informational	byn210413_1.sol: 132	ⓘ Pending

Description

According to the design, `_percent` should never be greater than 100.

Recommendation

We recommend add checks for `_percent` in order to avoid human errors.

CKP-09 | Unused Return Variable Name

Category	Severity	Location	Status
Language Specific	● Informational	byn210413_1.sol: 202, 221	ⓘ Pending

Description

The return variable names at the aforementioned lines are never used and thus could be removed.

Recommendation

We recommend removing the return variable names at the aforementioned lines.

CKP-10 | Incorrect Event Emit

Category	Severity	Location	Status
Logical Issue	● Major	byn210413_1.sol: 285	✓ Resolved

Description

The token is transferred from `_from` to `_to`, so the first argument of the event `Transfer` should be `_from` rather than `msg.sender`.

Recommendation

We recommend changing `msg.sender` in event `Transfer` to `_from`.

Alleviation

[Beyond Team]: The issues are addressed and reflected in the latest file `byn210413_3.sol`

CKP-11 | Proper Usage of `public` and `external` Type

Category	Severity	Location	Status
Gas Optimization	● Informational	byn210413_1.sol: 81, 85, 90, 95, 99, 115, 152, 180	ⓘ Pending

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays `external` functions are more efficient than "public" functions.

Examples:

- `add_allowedAddress()`
- `delete_allowedAddress()`
- `delete_blockedAddress()`
- `add_timeAddress()`
- `delete_timeAddress()`
- `refresh_lockPermitBalance()`

Recommendation

We advise the client to consider using the `external` attribute for functions never called from the contract.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings 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 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.

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 but rather 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.

