Covalent

Security Assessment

October 6th, 2020

For :
Ganesh Swami @ Covalent
ganesh@covalenthq.com

By :
Angelos Apostolidis @ CertiK
angelos.apostolidis@certik.org

Disclaimer

CertiK reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are 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 review.

CertiK Reports do 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.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing 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.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

What isn’t a CertiK report?

- A statement about the overall bug free or vulnerability free nature of a piece of source code or any modules, technologies or code it interacts with.
- Guarantee or warranty of any sort regarding the intended functionality or security of any or all technology referenced in the report.
- An endorsement or disapproval of any company, team or technology.

Summaries

Project Summary

Project Name	Covalent
Description	ERC-20 Token with Permit mechanism and a Vesting contract
Platform	Ethereum, Solidity
Codebase	GitHub Repository

Audit Summary

Delivery Date	Oct. 02, 2020
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	1
Timeline	Sep. 20th, 2020 - Oct. 6th 2020

Vulnerability Summary

Total Issues	4
Total Critical	0
Total Major	0
Total Minor	0
Total Informational	4

Findings

ID	Title	Type	Severity
CQT-01	Address Restriction	Volatile Code	Informational
CQTV-01	Redundant Variable Initialization	Optimization	Informational
CQTV-02	Redundant Utilization of `SafeMath`	Optimization	Informational
CQTV-03	`struct` Optimization	Optimization	Informational

CQT-01: Address Restriction

Type	Severity	Location
Volatile Code	Informational	CovalentQueryToken.sol L27-L29

Description:

The linked function provides the `owner` with unlimited potential, making the contract very centralized.

Recommendation:

We advise the team to add the following `require` statement before L28:

```
require(token != destination, "error message");
```

Alleviations:

The team opted to consider our references and added a `require` statement, as recommended. The team also commented that the `owner` is used for Rescue function and that nothing else can be or should be possible with this privilege. The `owner` is assumed to be governances controlled multi-sig to rescue accidentally sent funds.

CQTV-01: Redundant Variable Initialization

Type	Severity	Location
Optimization	Informational	CovalentQueryTokenVesting.sol L20, L21

Description:

When declaring variables without an initial value, they are assigned the specific data type's default value. Hence, the initialization of `uint256` to zero is redundant.

Recommendation:

We advise the team to remove the redundant assignments to the linked variables.

Alleviations:

The team opted to consider our references and removed the initialization to zero for the linked variables.

CQTV-02: Redundant Utilization of `SafeMath`

Type	Severity	Location
Optimization	Informational	CovalentQueryTokenVesting.sol L161

Description:

The variable `vestingId` should never overflow, as the variable is only incremented by one through a restricted function.

Recommendation:

We advise the team to use simple Math operations instead of the SafeMath library for this operation.

Alleviations:

No alleviations were applied, as the gas saved by this change is minimal.

CQTV-03: `struct` Optimization

Type	Severity	Location
Optimization	Informational	CovalentQueryTokenVesting.sol L29-L34

Description:

Every `struct` withholds the member information in 256-bit blocks. So, its members' data types should be as optimized as possible to reserve as little space possible. The member `releaseTime` of the `Vesting` struct contains a timestamp, and Unix timestamps can even be represented with a 64-bit variable type. This way, only two 256-bit blocks will be reserved for every `stuct` instantiation.

Recommendation:

We advise the team to change the `Vesting` struct composition to:

```
struct Vesting {
    uint256 amount;
    uint64 releaseTime;
    address beneficiary;
    bool released;
}
```

Alleviations:

The case was situational, and no alleviations were applied, as the team refers to the gas cost as no issue due to the small interaction points.