Competitive Security Assessment

QnA3_ERC20

Apr 22nd, 2024

Secure3

secure3.io

Summary

This report is prepared for the project to identify vulnerabilities and issues in the smart contract source code. A group of NDA covered experienced security experts have participated in the Secure3's Audit Contest to find vulnerabilities and optimizations. Secure3 team has participated in the contest process as well to provide extra auditing coverage and scrutiny of the finding submissions.

The comprehensive examination and auditing scope includes:

• Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.

• Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.

• Using static analysis tools to analyze smart contracts against common known vulnerabilities patterns.

• Verify the code base is compliant with the most up-to-date industry standards and security best practices.

• Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in four severity levels: Critical, Medium, Low, Informational. For each of the findings, the report has included recommendations of fix or mitigation for security and best practices.

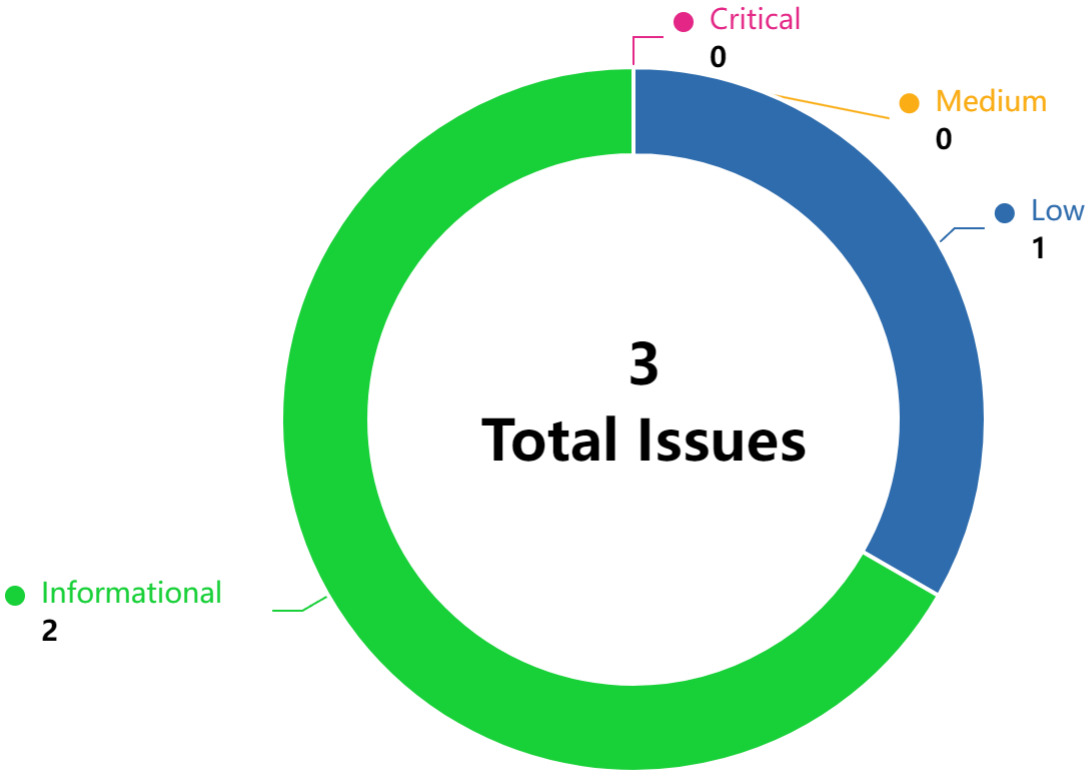
Overview

Project Name	QnA3_ERC20
Language	Solidity
Codebase	• QNA_ERC20.sol • audit version - 891e3b22d49992eb2b9a6cefb15cf8caa689f6f7 • final version - 891e3b22d49992eb2b9a6cefb15cf8caa689f6f7
Audit Methodology	• Audit Contest • Business Logic and Code Review • Privileged Roles Review • Static Analysis

Audit Scope

File	SHA256 Hash
QNA_ERC20.sol	891e3b22d49992eb2b9a6cefb15cf8caa689f6f7

Code Assessment Findings



ID	Name	Category	Severity	Client Response	Contributor
QAE-1	Use Ownable2Step Instead	Logical	Low	Acknowledged	0xCO2, xyzqwe123
QAE-2	Use The Latest Solidity Version	Language Specific	Informational	Acknowledged	0xCO2
QAE-3	Missing Zero Address Check on `_recipient` in Constructor	Logical	Informational	Acknowledged	toffee

QAE-1:Use Ownable2Step Instead

Category	Severity	Client Response	Contributor
Logical	Low	Acknowledged	0xCO2, xyzqwe123

Code Reference

- code/QNA_ERC20.sol#L4
- code/QNA_ERC20.sol#L9
- code/QNA_ERC20.sol#L18

```
4: import "@openzeppelin/contracts/access/Ownable.sol";
```

```
9: contract GPTToken is ERC20Burnable, ERC20Votes, ERC20Permit, Ownable {
```

```
18: Ownable(msg.sender)
```

Description

0xCO2: Ownable2Step is safer than Ownable for smart contracts because the owner cannot accidentally transfer smart contract ownership to a mistyped address. Rather than directly transferring to the new owner, the transfer only completes when the new owner accepts ownership.
Check the docs and the code here.
xyzqwe123: The contract QNA_ERC20.sol does not implement a 2-Step-Process for transferring ownership. So ownership of the contract can easily be lost when making a mistake when transferring ownership.

Recommendation

0xCO2: It is recommended to use `Ownable2Step.sol`.
xyzqwe123: Use the Ownable2Step variant of the Ownable contract to better safeguard against accidental transfers of access control.

Client Response

client response for 0xCO2: Acknowledged - There are currently no plans to transfer ownership
client response for xyzqwe123: Acknowledged - There are currently no plans to transfer ownership

QAE-2:Use The Latest Solidity Version

Category	Severity	Client Response	Contributor
Language Specific	Informational	Acknowledged	0xCO2

Code Reference

- code/QNA_ERC20.sol#L2

```
2: pragma solidity ^0.8.21;
```

Description

0xCO2: Developers should stay away from using floating and outdated pragma. Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation

0xCO2: It is recommended to lock a recent version of the Solidity compiler.

```
pragma solidity 0.8.25;
```

Client Response

client response for 0xCO2: Declined - The contract is currently running smoothly
Secure3: Acknowledged - Developers should stay away from using floating and outdated pragma. Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.
We need to ensure contract security, not just currently running smoothly.

QAE-3:Missing Zero Address Check on `_recipient` in Constructor

Category	Severity	Client Response	Contributor
Logical	Informational	Acknowledged	toffee

Code Reference

- code/QNA_ERC20.sol#L20

```
20: _mint(_recipient, maxSupply);
```

Description

toffee: As all the `maxSupply` is mint to `_recipient`, there is no validation on the `_recipient`

Recommendation

toffee: Add `require(_recipient != address(0), "invalid address")` in the constructor

Client Response

client response for toffee: Acknowledged - At present, the mint tokens are already in the _recipient wallet, so there's no risk involved.

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 Invoices, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Invoice. This report provided in connection with the services set forth in the Invoices shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Invoice. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Secure3's prior written consent in each instance.

This report is not an "endorsement" or "disapproval" of any particular project or team. This report is not an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Secure3 to perform a security assessment. This report does not provide any warranty or guarantee of free of bug of codes analyzed, nor do they provide any indication of the technologies, business model or legal compliancy.

This report should not be used in any way to make decisions around investment or involvement with any particular project. Instead, it represents an extensive assessing process intending to help our customers increase the quality of their code and high-level consistency of implementation and business model, while reducing the risk presented by cryptographic tokens and blockchain technology.

Secure3's position on the final decisions over blockchain technologies and corresponding associated transactions is that each company and individual are responsible for their own due diligence and continuous security.

The assessment services provided by Secure3 is subject to dependencies and under continuing development. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.