Competitive Security Assessment

Bondings_Update

Apr 22th, 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.

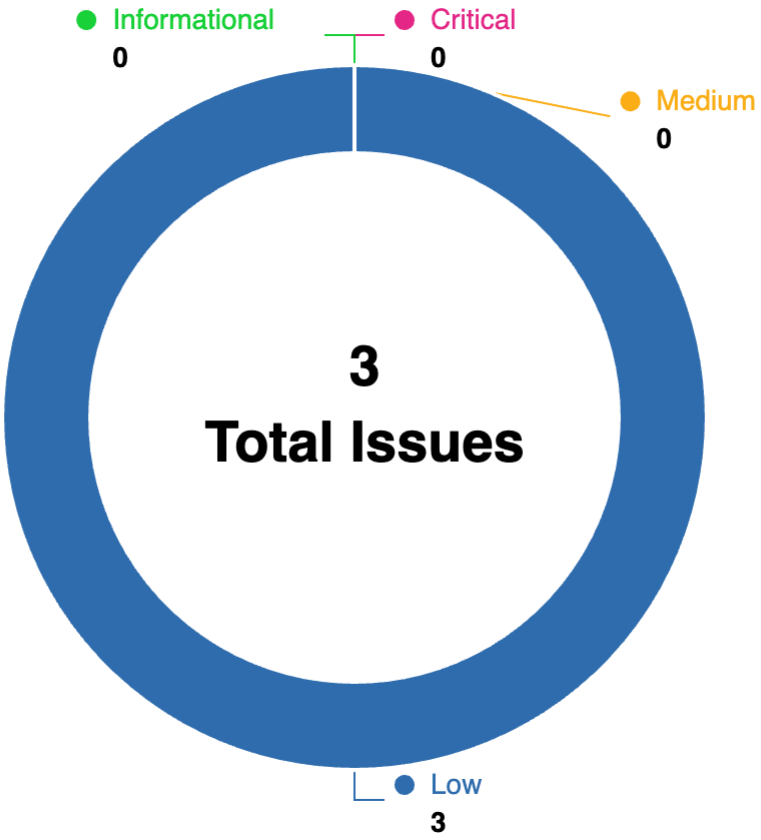
Overview

Project Name	Bondings_Update
Language	Solidity
Codebase	• https://github.com/bondings/bondings-contract • audit version - 6aad6e6fb472e79c9e1b659aca77aad8a7249b85 • final version - 773f25d3c050dc9c420bdd351de04aac96a0865c
Audit Methodology	• Audit Contest • Business Logic and Code Review • Privileged Roles Review • Static Analysis

Audit Scope

File	SHA256 Hash
./bondings-contract/contracts/BondingsCore.sol	442addfddfa72a644a49f625de9e5b2fb2013924c8b2b8187019a68371594126
./bondings-contract/contracts/interface/interface.sol	ee2a2a24d572661b15da8d60f518a914cc19e5945fe2ad53b457ad7095a1bb72

Code Assessment Findings



ID	Name	Category	Severity	Client Response	Contributor
BON-1	should use `safeTransfer` instead of `transfer`	Logical	Low	Fixed	toffee
BON-2	Missing zero address check in multiple functions	Logical	Low	Fixed	ethprinter, toffee
BON-3	Inconsistent `BondingsTotalShare` logic in different functions	Logical	Low	Acknowledged	ethprinter

BON-1:should use `safeTransfer` instead of `transfer`

Category	Severity	Client Response	Contributor
Logical	Low	Fixed	toffee

Code Reference

- code/bondings-contract/contracts/BondingsCore.sol#L162
- code/bondings-contract/contracts/BondingsCore.sol#L195-L197

```
162: IERC20(unitTokenAddress).transfer(protocolFeeDestination, fee);
```

```
195: IERC20(unitTokenAddress).transfer(user, priceAfterFee);
196:         if (fee > 0)
197:             IERC20(unitTokenAddress).transfer(protocolFeeDestination, fee);
```

Description

toffee: the return value of **`IERC20(unitTokenAddress).transfer()`** is not checked, and it could be failure and result the wrong internal update.

Recommendation

toffee: check the return value of the **`transfer`** call or use **`safeTransfer`** library

Client Response

toffee: client response for toffee: Fixed. Fix at:

- https://github.com/bondings/bondings-contract/commit/773f25d3c050dc9c420bdd351de04aac96a0865c#diff-c1fd6debce6f6582bfeaf7c28a32a027485fb747c96cf34f5e209f9385a3ada5R162
- https://github.com/bondings/bondings-contract/commit/773f25d3c050dc9c420bdd351de04aac96a0865c#diff-c1fd6debce6f6582bfeaf7c28a32a027485fb747c96cf34f5e209f9385a3ada5R197

BON-2:Missing zero address check in multiple functions

Category	Severity	Client Response	Contributor
Logical	Low	Fixed	ethprinter, toffee

Code Reference

- code/bondings-contract/contracts/BondingsCore.sol#L62
- code/bondings-contract/contracts/BondingsCore.sol#L226
- code/bondings-contract/contracts/BondingsCore.sol#L271
- code/bondings-contract/contracts/BondingsCore.sol#272

```
62: address unitTokenAddress_, address protocolFeeDestination_
```

```
226: userShare[bondingsId][to] += share;
```

```
271: function setProtocolFeeDestination(address newProtocolFeeDestination) public onlyOwner {
```

```
272: address oldProtocolFeeDestination = protocolFeeDestination;
```

Description

ethprinter: The code misses zero address check in many functions, lacking of validating for a zero address, it could result in the function performing unexpected or erroneous behavior, such as transferring shares to a nonexistent or set rotocolFeeDestination to a invalid Ethereum address.
toffee: `**protocolFeeDestination**` is used for receiving selling fees, and there is lack of zero address check. the risk is that if the `**newProtocolFeeDestination**` is zero the fee will be sent to zero and equilavent to burn

Recommendation

ethprinter:

```
function setProtocolFeeDestination(address newProtocolFeeDestination) public onlyOwner {
require(newProtocolFeeDestination != address(0), "Invalid address");
        address oldProtocolFeeDestination = protocolFeeDestination;
        protocolFeeDestination = newProtocolFeeDestination;
        emit AdminSetParam(
            "protocolFeeDestination",
            bytes32(uint256(uint160(oldProtocolFeeDestination))),
            bytes32(uint256(uint160(newProtocolFeeDestination)))
        );
    }
```

toffee: add `require(newProtocolFeeDestination != address(0), "zero address") in the L272

Client Response

ethprinter: client response for ethprinter: Fixed. Fixed at:

- https://github.com/bondings/bondings-contract/commit/773f25d3c050dc9c420bdd351de04aac96a0865c#diff-c1fd6debce6f6582bfeaf7c28a32a027485fb747c96cf34f5e209f9385a3ada5R225
- https://github.com/bondings/bondings-contract/commit/773f25d3c050dc9c420bdd351de04aac96a0865c#diff-c1fd6debce6f6582bfeaf7c28a32a027485fb747c96cf34f5e209f9385a3ada5R225

toffee: client response for toffee: Fixed. Fixed at:

- https://github.com/bondings/bondings-contract/commit/773f25d3c050dc9c420bdd351de04aac96a0865c#diff-c1fd6debce6f6582bfeaf7c28a32a027485fb747c96cf34f5e209f9385a3ada5R225
- https://github.com/bondings/bondings-contract/commit/773f25d3c050dc9c420bdd351de04aac96a0865c#diff-c1fd6debce6f6582bfeaf7c28a32a027485fb747c96cf34f5e209f9385a3ada5R225

BON-3:Inconsistent `BondingsTotalShare` logic in different functions

Category	Severity	Client Response	Contributor
Logical	Low	Acknowledged	ethprinter

Code Reference

- code/bondings-contract/contracts/BondingsCore.sol#L137

```
137: uint256 totalShare = bondingsTotalShare[bondingsId];
```

Description

ethprinter: When a new bonding is launched, its share should be zero, it seems that the developer has considered that problem in the **`getBondingsTotalShare()`** function, which returns **`bondingsTotalShare[bondingsId] - 1`**, however, the code doesn't consider this when using totalshare to do the actual calculation(in line 137 for example).

Recommendation

ethprinter: In line 122, **`bondingsTotalShare[bondingsId] = 1`**, every bonding's share is set to 1 after being launched, which is strange because that share doesn't belong to anyone.Should consider give that 1 share to the launcher and returns **`bondingsTotalShare[bondingsId]`** in **`getBondingsTotalShare()`** to maintain the consistence.

Client Response

ethprinter: client response for ethprinter: Acknowledged. This inconsistent logic is designed on purpose for the requirement "the first (0th) share's price is 1, not 0". See the annotation at https://github.com/bondings/bondings-contract/commit/773f25d3c050dc9c420bdd351de04aac96a0865c#diff-c1fd6debce6f6582bfeaf7c28a32a027485fb747c96cf34f5e209f9385a3ada5R112

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.