Competitive Security Assessment

ZkBase

Jan 2nd, 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

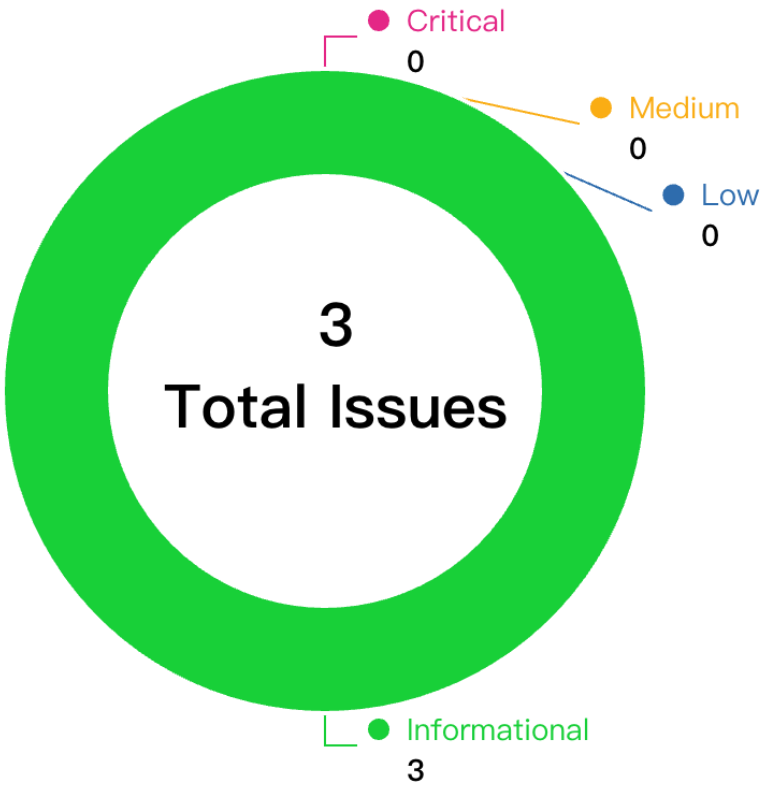
Project Detail

Project Name	ZkBase
Platform & Language	Solidity
Codebase	• https://github.com/l2labs/zkb-contract • audit commit - 0d1bd10b3bdea9fa60fe7b2438a71975a76b25b7 • final commit - 0d1bd10b3bdea9fa60fe7b2438a71975a76b25b7
Audit Methodology	• Audit Contest • Business Logic and Code Review • Privileged Roles Review • Static Analysis

Audit Scope

File	SHA256 Hash
src/ZKBConverter.sol	f3c68023be38f7e70094b4bdae0ec11519cdf2ef19aed8b9e033d53999224d22
src/ZKB.sol	27826da9073e7d113890fa98cfde4278fff3e7918b354233623234d2c1d8e96c

Code Assessment Findings



ID	Name	Category	Severity	Client Response	Contributor
ZKB-1	ZKB mint should do parameter validation	Logical	Informational	Acknowledged	ethprinter, toffee
ZKB-2	ZKBConverter assumes same decimal for ZKS_TOKEN and ZKB_TOKEN	Logical	Informational	Acknowledged	toffee
ZKB-3	Withdraw ETH function only	Logical	Informational	Acknowledged	zigzag

ZKB-1:ZKB mint should do parameter validation

Category	Severity	Client Response	Contributor
Logical	Informational	Acknowledged	ethprinter, toffee

Code Reference

- code/src/ZKB.sol#L15
- code/src/ZKB.sol#L17

```
15:function mint(address to, uint256 amount) public onlyOwner {

17:_mint(to, amount);
```

Description

ethprinter : In contract `ZKB`::`mint()` function, it doesn't check the input params `to` is not a zero address, since it can only be minted once, so it could cause unexpected results.
toffee : as the mint function can only be called once, better to do validation on `address to` and `uint256 amount`

Recommendation

ethprinter : add check `require(to != address(0), "Invalid address");`
toffee : do validation

```
    function mint(address to, uint256 amount) public onlyOwner {
      require(!_minted, "ZKB: already minted");
+     require(to != address(0), "ZKB to error");
+     require(amount > 0, "ZKB amount error");
```

Client Response

Acknowledged. This validation has been done by openzepplin's ERC20 lib: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/a72c9561b9c200bac87f14ffd43a8c719fd6fa5a/contracts/token/ERC20/ERC20.sol#L227-L229

ZKB-2:ZKBConverter assumes same decimal for ZKS_TOKEN and ZKB_TOKEN

Category	Severity	Client Response	Contributor
Logical	Informational	Acknowledged	toffee

Code Reference

- code/src/ZKBConverter.sol#L28-L34

```
28:function convert(uint256 amount) public {
29:    // transfer ZKS to black hole
30:    ZKS_TOKEN.safeTransferFrom(msg.sender, BLACK_HOLE, amount);
31:    // transfer ZKB to user
32:    ZKB_TOKEN.safeTransfer(msg.sender, amount);
33:    // emit event
34:    emit Converted(msg.sender, amount);
```

Description

toffee : the `ZKBConverter::convert` assumes the same decimal of ZKB (18) and ZKS (?), as I do not know the deployed address of `ZKS`, do not sure if a scaling needs to be applied to the `amount` for the newly ZKB_TOKEN transfer.

Recommendation

toffee : Confirm ZKB and ZKS has the same default 18 decimal, or apply a scaling factor in the amount

Client Response

Acknowledged. The ZKS contract already deployed with 18 decimal: https://etherscan.io/token/0xe4815AE53B124e7263F08dcDBBB757d41Ed658c6. The ZKB contract inherit openzepplin's ERC20 and doesn't override the decimal. It will be 18 too. So we confirm that the decimals of ZKB and ZKS are the same.

ZKB-3:Withdraw ETH function only

Category	Severity	Client Response	Contributor
Logical	Informational	Acknowledged	zigzag

Code Reference

- code/src/ZKBConverter.sol#L40

```
40:(bool result,) = payable(msg.sender).call{value: amount}("");
```

Description

zigzag : Under the existing contract code, the ETH balance of `ZKBConverter` can only be increased through contract `self-destruct`, but the probability of this happening is extremely low.
So i guess:

- Missing complete deposit ETH function
  or
- The withdraw code is redundancy

Recommendation

zigzag : - Add complete deposit ETH function
or

- Remove the withdraw code

Client Response

Acknowledged. The withdraw of ETH is redundancy while the ERC20's is not.

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.