SANDBLOCKS

CONSULTING

Author

Dr Maria G Vigliotti

Sandblocks Consulting

Distribution List

Senior Leadership Team
(Francesco Piras)

BrightNode

Introduction

Sandblocks Consulting (SC) was contracted by BrightNode Sagl (the Customer) to conduct Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between November 10th, 2022 – November 16th 2022.

Scope

The scope of the project is related to two smart-contracts (Blenda.sol and IdentityRegistry) located on the Bitbucket repo:
https://bitbucket.org/longwavestudio/unblended-contracts/src/main/ - commit `45712ffd12f424fcfe80f2d5e9a796ce6dc10c9a`.

SC has:

1. Reviewed the code manually and
2. Assess the main functionality of the contract, and
3. Derived the security requirements, and
4. Looked for smart contracts for commonly known security vulnerabilities.

Methodology	Vulnerabilities
Code Review	• Re-entrancy • Ownership Takeover • Timestamp Dependencies • Transaction-Ordering Dependencies • Style guide violation • Costly Loop • Unchecked external call • Unchecked math • Unsafe type inference
Functional Review	• Business Logics Review • Functionality Checks • Access Control & Authorization • User Balances manipulation • Kill-Switch Mechanism • Operation Trails & Event Generation

Summary of Findings

According to the assessment, the Customer's smart contract is secure

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution or to partial requirements and problems associated to actions taken off-chain
Style/Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

Audit Overview

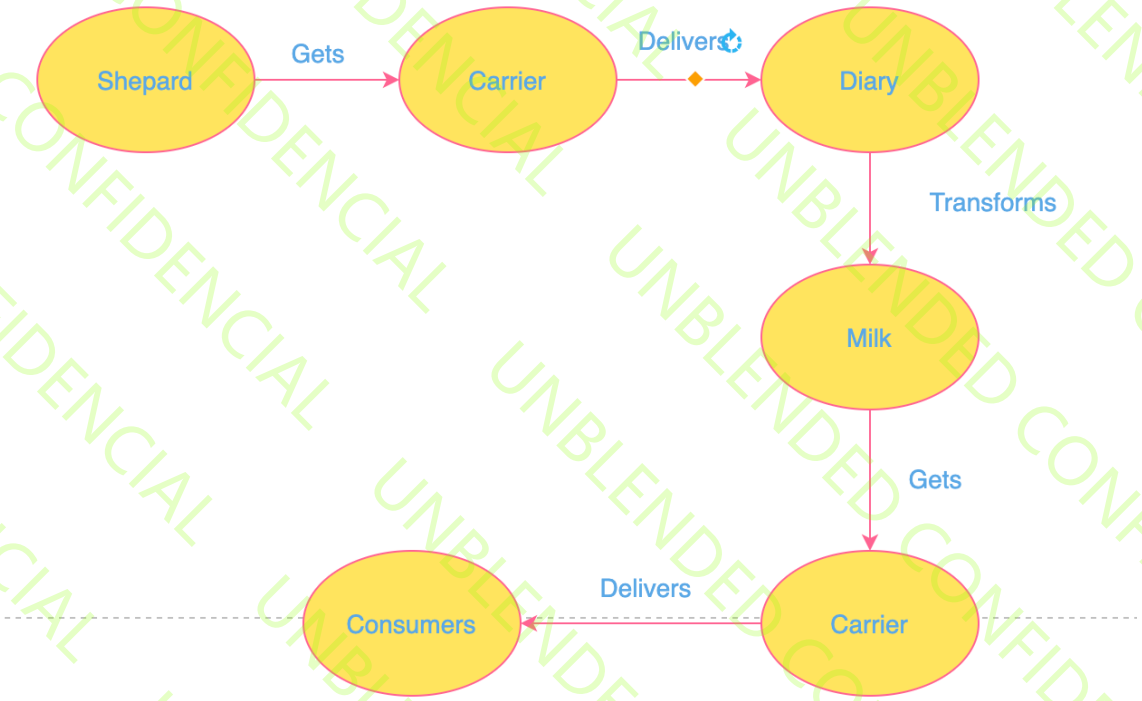
The code in the two smart contacts "IdentiyRegistry.sol" and "Blends.sol" implements the functionality of a diary supply chains as described in section "Functional and Security Requirements".

The code is readable, and elegant and reasonably commented. There are 57 tests, and covering calling most functions.

Summary of Functional requirements

From a high-level description, we have defined the functional requirements. The contracts "Blends.sol" and "IdentityRegistry.sol" are part of diary tracking solution. The solution should keep track of all the phases of transformations from milk to cheese up, and to some extent to the consumers.
For the cheese tracking the main roles are: **dairies, shepherds, carriers.**

To an abstract level, the supply chain relationship is depicted in the following diagram:



The trackable phases are in contract embodied in the concept of blend. Examples of blend could be: A shepherd milking a cow produces a blend record OR
A carrier moving milk from a place to another produces a blend record.

Each phase, products are moved from one party to another, and each phase is labelled as:
-**Accepted**
-Rejected
-Proposed
-ClosewithNote

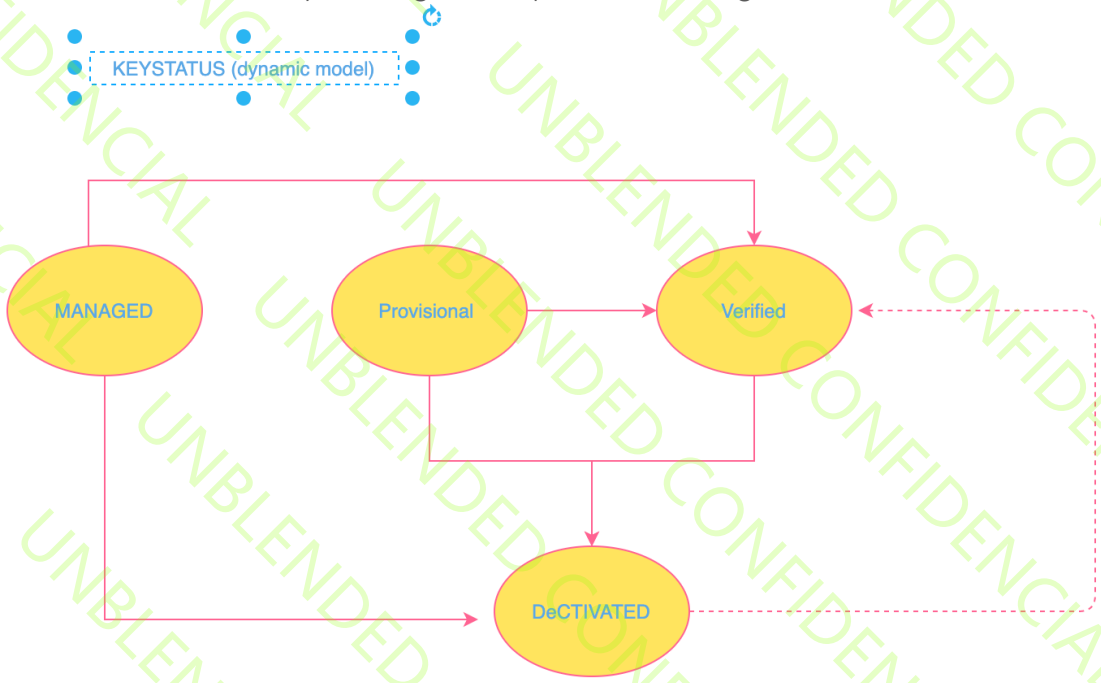
A proposed blend can be accepted, otherwise a blend can be Rejected or ClosewithNote
A blend record contains :
- an id (rid),
- a sender address,
- a signatory address - the actor's key approving the blend
- up to two ancestors – indicates who participate on the previous phase of the blend
- time when the record was created -
- a payload – record on the transaction
- the status of the record – it keeps track of the lifecycle of the blend

The Blends.sol is smart contract is meant to be proof that that data was inserted (integrity and timestamp) and it can be verified anytime later. A record of the number of blends is available to users, and blends are kept in a list.

The dairy has a privileged role for its own blends: it can hire and fire other actors.

The authorizations to write in the Blends.sol are regulated by identities. Each actor (role) in the transformation phase has ad identity are created and managed in the associated smart-contract IdentityRegistry.sol. The contract creates a trusted registry managed by a Trusted Party in charge of governing the whole ecosystem.

The Trusted Party is the only authority able to attest that an identity the attribute of an identify via the key. The identity attributes are:
-**Provisional**: means owned by some actors but not verified by the Trusted Party
- **Verified:** owned by the actors and verified by the Trusted Party
- **Managed**: owned by the Trusted Party on behalf of the actors
-**Deactivated**: an old key, no longer accepted for creating records



An identity will be attested to own one or more addresses.
Only the Trusted Party can write the IdentityRegistry, while anyone can query and check identities and check keys statuses.

The full ecosystem will have:
1 authority managing the identityRegistry (Unblended limited)
N dairies working with M shepherds, carriers and other

Identities notarized in the smart contract are opaque bytes32 id, they can be created off-chain and attested by the authority which attaches them with principal and extra attributes. All the attributes are stored as hashes in the contract, to verify the identity details one must gather actual off-chain data and check against the hashes. This is done to keep all data private yet provable and verifiable in case of dispute.

Security Requirements

1. Only the Third Party should be able to attest identities and modify the status of the keys.
2. A provisional key can become deactivated or verified
3. A deactivated key can become verified
4. A managed key can become verified
5. A managed key can become deactivated
6. Only the appointed signatory can accept or reject blends.

7. Only a Proposed blend can be Arccepted
8. Only diary can ForceClose a blend
9. Payloads can be updated only if it is not proposed
10. Only diaries can hire and fire carriers and shepherds
11. Shepherds cannot append a blend record

The security requirements above have been inferred from SC, however, these can be incomplete in the absence of a more formal description of the functional requirements.

We suggest for BrightNote to review such safety requirements. The results of our audit can be found below, and they assume the security requirements are the ones above.

Risk Level	Findings
Critical	**No critical severity issues were found.**
High	**No high-severity issues were found**
Medium	**No medium issues were found**
Low	Assuming the security requirements above are complete there are two main concerns: 1. What is the relationship between hiring and firing and the blends or the registry? The key fired carrier or sheperd should be become unverified? At the moment, there seems to be no consequences and this suggest some missing functional or security requirements. 2. As highlighted also by the devlopers in the tests, it is unclear, and at the moment cumbersome to create the initial relationship between the "uid" and the "key" for the diary, the shepards and the carriers. Who creates that? If if happeds off-chain there is the risk to undermine the value of the attestation of identity of the Trusted Party.
Style/Best Practice	Both smart contracts could be written in a more modular way.

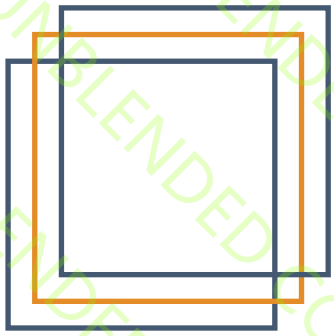
Conclusion

Smart contract within the scope has been audit in a manner that is proportional to the complexity of the code. SC has tested and manually reviews the code to ensure the functionality.

Disclaimer

The smart contract given for audit have been analysed in accordance with the best industry practices at the date of this report, in relation to security vulnerabilities and issues in smart contract source code. The Source Code compilation, deployment, and functionality (performing the intended functions).
The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.



Sandblocks Consulting
4 Christopher St
London, EC2A 2BS
United Kingdom
+44 (0) 7905320581
contact@sandblocksconsulting.co.uk