METATRUST

Security Assessment for

Debox V

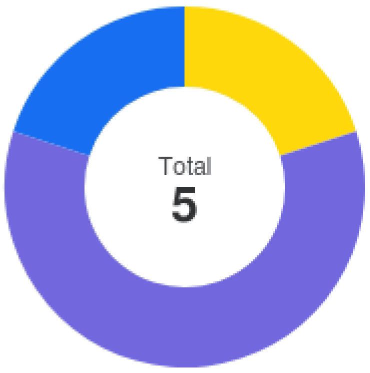
May 09, 2024

Executive Summary

Overview	
Project Name	Debox V
Codebase URL	https://github.com/debox-pro/
Scan Engine	Security Analyzer
Scan Time	2024/05/09 08:00:00
Commit Id	628e3233900ab9cc2b0d45b63b81f107d59c91ac

Total	
Critical Issues	0
High risk Issues	0
Medium risk Issues	1
Low risk Issues	3
Informational Issues	1

Critical Issues	The issue can cause large economic losses, large-scale data disorder, loss of control of authority management, failure of key functions, or indirectly affect the correct operation of other smart contracts interacting with it.
High Risk Issues	The issue puts a large number of users' sensitive information at risk or is reasonably likely to lead to catastrophic impacts on clients' reputations or serious financial implications for clients and users.
Medium Risk Issues	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk Issues	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational Issue	The issue does not pose an immediate risk but is relevant to security best practices or Defence in Depth.



Critical Issues	0%	**0**
High risk Issues	0%	**0**
Medium risk Issues	20%	**1**
Low risk Issues	60%	**3**
Informational Issues	20%	**1**

Summary of Findings

MetaScan security assessment was performed on **May 09, 2024 08:00:00** on project **Debox V** with the repository on branch **default branch**. The assessment was carried out by scanning the project's codebase using the scan engine **Security Analyzer**. There are in total **5** vulnerabilities / security risks discovered during the scanning session, among which **1** medium risk vulnerabilities, **3** low risk vulnerabilities, **1** informational issues.

ID	Description	Severity	Alleviation
MSA-001	Interval and releaseTimes could be set by users	Medium risk	Acknowledged
MSA-002	Initial token distribution	Low risk	Acknowledged
MSA-003	The releaseTimes lacks the upper boundary	Low risk	Fixed
MSA-004	The balanceOf lacks incurring all the releasable amount	Low risk	Fixed
MSA-005	The actual lock amount	Informational	Acknowledged

Findings

Medium risk (1)

1. Interval and releaseTimes could be set by users

Medium risk   Security Analyzer

Users can set the interval as small as 1 hour and set the releaseTimes as small as 1 to get a quick release.

File(s) Affected

DBXLockup.sol #84-90

```
84    function lock(address beneficiary, uint256 lockAmount, uint256 interval, uint256 releaseTimes) extern
85      require(canLock[beneficiary], "DBXLockup: not allowed to lock");
86      require(locked[beneficiary].length <= 16, "DBXLockup: lock limit reached"); // only allow 16 locks
87      require(lockAmount >= 10000 ether, "DBXLockup: lock amount too low"); // safety check
88      require(interval >= 1 hours, "DBXLockup: interval too short");
89      require(releaseTimes >= 1, "DBXLockup: release times too low");
90      require(interval <= 365 days, "DBXLockup: interval too long");
```

Alleviation Acknowledged

The team acknowledged this finding.

Low risk (3)

1. Initial token distribution

Low risk   Security Analyzer

In the contract DBXToken contract, during the deployment on Ethereum,

- 5,000,000,000 $DBX will be allocated to the 2/2 multi-signature wallet `0x2745F97f501087caF8eA740854Cfcac011fb34C3`,
- 500,000,000 $DBX will be allocated to the 2/2 multi-signature wallet 0×5b1AfdB8C23569484773aF7bD4c98Af9ee7599D9,
- 4,000,000,000 $DBX will be allocated to the deployer.

File(s) Affected

DBXToken.sol #22-28

```
22    constructor() ERC20Permit("DeboxToken") ERC20("DeboxToken", "DBX") {
23      _mint(0x2745F97f501087caF8eA740854Cfcac011fb34C3, 5.5e9 ether); // 5.5 billion
24      _mint(0x5b1AfdB8C23569484773aF7bD4c98Af9ee7599D9, 0.5e9 ether);
25      _mint(msg.sender, 4e9 ether);
26      // safety check
27      require(totalSupply() == 10_000_000_000 ether, "incorrect total supply"); // 10 billion
28    }
```

Recommendation

Consider posting the detailed tokenomics of the $DBX to mitigate the centralization risk of token distribution.

Alleviation Acknowledged

The team acknowledged this finding.

2. The releaseTimes lacks the upper boundary

Low risk   Security Analyzer

The releaseTimes could be an extremely large value if a user set it by mistake, which would result in a kind of token-lock forever, due to the releaseTimes contributing to the oneReleaseAmount that stands for the amount a user can release per release:

```
uint256 oneReleaseAmount = lockAmount / releaseTimes;
```

E.g., if the contract is deployed on the Ethereum, and oneReleaseAmount is 1 wei, users probably would not start a transaction to unlock 1 wei token due to there is no benefit, which results in the tokens being locked in the contract forever.

File(s) Affected

DBXLockup.sol #84-90

```
function lock(address beneficiary, uint256 lockAmount, uint256 interval, uint256 releaseTimes) extern
  require(canLock[beneficiary], "DBXLockup: not allowed to lock");
  require(locked[beneficiary].length <= 16, "DBXLockup: lock limit reached"); // only allow 16 locks
  require(lockAmount >= 10000 ether, "DBXLockup: lock amount too low"); // safety check
  require(interval >= 1 hours, "DBXLockup: interval too short");
  require(releaseTimes >= 1, "DBXLockup: release times too low");
  require(interval <= 365 days, "DBXLockup: interval too long");
```

Recommendation

Consider adding an upper boundary check on the releaseTimes, like

```
require(releaseTimes <=0)
```

or

```
require(releaseTimes * interval <= 10 * 365 days)
```

Alleviation Fixed

The team addressed this finding, in commit 628e323.

3. The balanceOf lacks incurring all the releasable amount

Low risk   Security Analyzer

1. The `balanceOf`: This balanceOf function iterates through all items in the locked mapping for a given beneficiary. It accumulates two sums: total, which represents the total locked amount for the beneficiary, and releaseable, which represents the amount ready to be released.
2. The `_calculate`: This function is called within the loop in balanceOf for each locked item. It checks if the current timestamp is greater than or equal to item.nextReleaseAt. If so, it computes the releaseable amount as item.oneReleaseAmount. If item.oneReleaseAmount is greater than item.lockAmount, it adjusts releaseable to be item.lockAmount.
3. Concern Point Regarding Multiple Periods Missed: The issue of accumulating releaseable amounts for multiple missed periods requires computing how many release periods have passed since item.nextReleaseAt and multiplying item.oneReleaseAmount by this number. However, the provided _calculate function does not perform this calculation. It effectively considers only one release period, regardless of how much time has elapsed since item.nextReleaseAt.

File(s) Affected

DBXLockup.sol #56-62

```
function balanceOf(address beneficiary) public view returns (uint256 total, uint256 releaseable) {
  for (uint256 i = 0; i < locked[beneficiary].length; i++) {
    Lock storage item = locked[beneficiary][i];
    total += item.lockAmount;
    releaseable += _calculate(item);
  }
}
```

DBXLockup.sol #147-152

```
function _calculate(Lock storage item) private view returns (uint256 releaseable) {
  if (item.lockAmount > 0 && item.nextReleaseAt <= block.timestamp) {
    releaseable = item.oneReleaseAmount;
    if (releaseable > item.lockAmount) releaseable = item.lockAmount;
  }
}
```

Recommendation

Consider taking multiple available release periods into account, and accumulating the `releaseable` amounts for multiple releasable periods in the implementation of the `_calculate` function.

Alleviation Fixed

The team addressed this finding, in the commit 628e323.

Informational (1)

1. The actual lock amount

Informational   Security Analyzer

The lock amount is intended to be oneReleaseAmount * releaseTimes, rather than the lockAmount, though the fraction token is able to be released when calculating the releasable amount:

```
function _calculate(Lock storage item) private view returns (uint256 releaseable) {
  if (item.lockAmount > 0 && item.nextReleaseAt <= block.timestamp) {
    releaseable = item.oneReleaseAmount;
    if (releaseable > item.lockAmount) releaseable = item.lockAmount;
  }
}
```

File(s) Affected

DBXLockup.sol #84-96

```
function lock(address beneficiary, uint256 lockAmount, uint256 interval, uint256 releaseTimes) exter
  require(canLock[beneficiary], "DBXLockup: not allowed to lock");
  require(locked[beneficiary].length <= 16, "DBXLockup: lock limit reached"); // only allow 16 locks
  require(lockAmount >= 10000 ether, "DBXLockup: lock amount too low"); // safety check
  require(interval >= 1 hours, "DBXLockup: interval too short");
  require(releaseTimes >= 1, "DBXLockup: release times too low");
  require(interval <= 365 days, "DBXLockup: interval too long");

  uint256 oneReleaseAmount = lockAmount / releaseTimes;
  require(oneReleaseAmount > 0, "DBXLockup: release amount too low");

  // transfer
  dbx.safeTransferFrom(msg.sender, address(this), lockAmount);
```

Alleviation Acknowledged

The team acknowledged this finding.

Disclaimer

This report is governed by the stipulations (including but not limited to service descriptions, confidentiality, disclaimers, and liability limitations) outlined in the Services Agreement, or as detailed in the scope of services and terms provided to you, the Customer or Company, within the context of the Agreement. The Company is permitted to use this report only as allowed under the terms of the Agreement. Without explicit written permission from MetaTrust, this report must not be shared, disclosed, referenced, or depended upon by any third parties, nor should copies be distributed to anyone other than the Company.

It is important to clarify that this report neither endorses nor disapproves any specific project or team. It should not be viewed as a reflection of the economic value or potential of any product or asset developed by teams or projects engaging MetaTrust for security evaluations. This report does not guarantee that the technology assessed is completely free of bugs, nor does it comment on the business practices, models, or legal compliance of the technology's creators.

This report is not intended to serve as investment advice or a tool for investment decisions related to any project. It represents a thorough assessment process aimed at enhancing code quality and mitigating risks inherent in cryptographic tokens and blockchain technology. Blockchain and cryptographic assets inherently carry ongoing risks. MetaTrust's role is to support companies and individuals in their security diligence and to reduce risks associated with the use of emerging and evolving technologies. However, MetaTrust does not guarantee the security or functionality of the technologies it evaluates.

MetaTrust's assessment services are contingent on various dependencies and are continuously evolving. Accessing or using these services, including reports and materials, is at your own risk, on an as-is and as-available basis. Cryptographic tokens are novel technologies with inherent technical risks and uncertainties. The assessment reports may contain inaccuracies, such as false positives or negatives, and unpredictable outcomes. The services may rely on multiple third-party layers.

All services, labels, assessment reports, work products, and other materials, or any results from their use, are provided "as is" and "as available," with all faults and defects, without any warranty. MetaTrust expressly disclaims all warranties, whether express, implied, statutory, or otherwise, including but not limited to warranties of merchantability, fitness for a particular purpose, title, non-infringement, and any warranties arising from course of dealing, usage, or trade practice. MetaTrust does not guarantee that the services, reports, or materials will meet specific requirements, be error-free, or be compatible with other software, systems, or services.

Neither MetaTrust nor its agents make any representations or warranties regarding the accuracy, reliability, or currency of any content provided through the services. MetaTrust is not liable for any content inaccuracies, personal injuries, property damages, or any loss resulting from the use of the services, reports, or materials.

Third-party materials are provided "as is," and any warranty concerning them is strictly between the Customer and the third-party owner or distributor. The services, reports, and materials are intended solely for the Customer and should not be relied upon by others or shared without MetaTrust's consent. No third party or representative thereof shall have any rights or claims against MetaTrust regarding these services, reports, or materials.

The provisions and warranties of MetaTrust in this agreement are exclusively for the Customer's benefit. No third party has any rights or claims against MetaTrust regarding these provisions or warranties. For clarity, the services, including any assessment reports or materials, should not be used as financial, tax, legal, regulatory, or other forms of advice.