METATRUST

Security Assessment for DSC

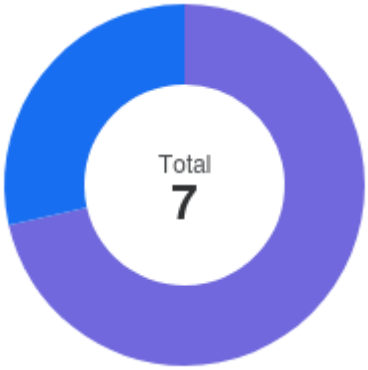
May 21, 2024

Executive Summary

Overview	
Project Name	DSC
Codebase URL	https://bscscan.com/address/0xa86a86b8acdc55812bec2971a2fc8a989455858c
Scan Engine	Security Analyzer
Scan Time	2024/05/21 08:00:00
Commit Id	-

Total	
Critical Issues	0
High risk Issues	0
Medium risk Issues	0
Low risk Issues	5
Informational Issues	2

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	0%	0
Low risk Issues	71%	5
Informational Issues	29%	2

Summary of Findings

MetaScan security assessment was performed on **May 21, 2024 08:00:00** on project **DSC** with the repository **bsc/0xa86a86b8acdc55812bec2971a2fc8a989455858c** 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 **7** vulnerabilities / security risks discovered during the scanning session, among which **5** low risk vulnerabilities, **2** informational issues.

ID	Description	Severity	Alleviation
MSA-001	Missing Zero Address Check	Low risk	Acknowledged
MSA-002	Initial Token Distribution	Low risk	Acknowledged
MSA-003	If a User Transfer Tokens to Himself/Herself	Low risk	Acknowledged
MSA-004	The Number of the Total Supply is Greater Than That of the Token Minted	Low risk	Mitigated
MSA-005	The `selectUpNodes` function may malfunction if there is a circle relation in the `fathers` array	Low risk	Acknowledged
MSA-006	Constant State Variable	Informational	Acknowledged
MSA-007	Missing Event Setter	Informational	Acknowledged

Findings

Low risk (5)

1. Missing Zero Address Check

Low risk Security Analyzer

The linked functions are lack of zero address check in important operation, which may cause some unexpected result.

File(s) Affected

0xa86a86b8acdc55812bec2971a2fc8a989455858c.bscscan.com-DSC/DSC.sol #16-18

```
16      constructor(string memory name, string memory symbol, address _to, address _owner) ERC20(name, symbo
17          setOwner(_owner, true);
18          _mint(_to, 5000000000e18);
```

Recommendation

Add check of zero address in important operations.

Alleviation Acknowledged

The team acknowledged this finding.

2. Initial Token Distribution

Low risk Security Analyzer

When deploying the contract, 100% of $DSC(the `5000000000e18` amount of $DSC) are minted to the `_to` address, which may incur the centralization risk.

File(s) Affected

0xa86a86b8acdc55812bec2971a2fc8a989455858c.bscscan.com-DSC/DSC.sol #16-18

```
16      constructor(string memory name, string memory symbol, address _to, address _owner) ERC20(name, symbo
17          setOwner(_owner, true);
18          _mint(_to, 5000000000e18);
```

Recommendation

Consider publishing the detailed tokenomics of the $DSC and applying multi-sign wallet to the token holders to mitigate the centralization risk.

Alleviation Acknowledged

The team acknowledged this finding.

3. If a User Transfer Tokens to Himself/Herself

Low risk Security Analyzer

The `_addUpNode` function ensures that both the sender and receiver have parent nodes assigned to them. If the sender does not have a parent, it sets the default parent node (if available), and it sets the sender as the parent of the receiver if the receiver does not have a parent.

But, there is case that a user transfer tokens to himself/herself, there he/she becomes his/her parent node, which may work unexpectly.

Or, a group of users make a circle relation, like Alice set a father node, Bob. Bob set a father node, Carol. Caral set a father node, Alice. So there is circle relation between them:

```
Alice -> Bob -> Carol -> Alice
```

File(s) Affected

0xa86a86b8acdc55812bec2971a2fc8a989455858c.bscscan.com-DSC/DSC.sol #34-46

```
34      function _addUpNode(address sender, address receiver) internal {
35          //   sender
36          if (fathers[sender] == address(0)) {
37              if (defaultUpNode != address(0)) { //       0
38                  fathers[sender] = defaultUpNode;
39              }
40          }
41
42          //   receiver       sender
43          if (fathers[receiver] == address(0)) {
44              fathers[receiver] = sender;
45          }
46      }
```

Recommendation

Consider checking the logic and redesigning the logic of cover the above cases accordingly.

Alleviation Acknowledged

The team acknowledged this finding.

4. The Number of the Total Supply is Greater Than That of the Token Minted

Low risk Security Analyzer

When deploying the contract, the **`5000000000e18`** $DSC are minted to the **`_to`** address.

```
_mint(_to, 5000000000e18);
```

But, the value that the **`totalSupply`** function gets is **`10000000000e18`**, which is greater than the actual tokens minted:

```
    function totalSupply() public view virtual override returns (uint256) {
        return 10000000000e18;
    }
```

Note that there is no public function to mint $DSC after the deployment, so the the number of the minted tokens is always less the number to total supply.

File(s) Affected

0xa86a86b8acdc55812bec2971a2fc8a989455858c.bscscan.com-DSC/DSC.sol #16-24

```
16      constructor(string memory name, string memory symbol, address _to, address _owner) ERC20(name, symbo
17          setOwner(_owner, true);
18          _mint(_to, 5000000000e18);
19          fathers[0x84d5BbD963764D2C2922326c7c9823DBDea75e2C] = address(this);
20      }
21
22      function totalSupply() public view virtual override returns (uint256) {
23          return 10000000000e18;
24      }
```

Recommendation

Making the numbers of minted token and the total supply the same.

Alleviation Mitigated

The team replied that the rest tokens will be minted on the other chain, this contract has no function to mint tokens after the deployment so the the number of the minted tokens on current chain is always less the number to total supply. The other chain's contract is still developing and will be published soon.

5. The `selectUpNodes` function may malfunction if there is a circle relation in the `fathers` array

Low risk Security Analyzer

The `selectUpNodes` function retrieves the parent nodes for the given address from and stores every third parent node in the nodes array, up to a maximum of `maxUpNodeNum / 3` nodes. The resulting array nodes contains the selected parent nodes for the given address.

It checks if "next" is not equal to address(0) and not equal to "defaultUpNode". If the condition is not met, the loop breaks, there is no further influence. If the condition is met and the index "i" is divisible by 3, it assigns the value of "next" to the "nodes" array at index "k" and increments "k" by 1. It then updates the value of "next" to the "fathers" mapping at the key "next".

As a result, if a user set his/her parent node to be himself/herself, the `selectUpNodes` function will return an array with all the element the same. Or if there is a circle relation in the `fathers` array, like, Alice has a father node, Bob. Bob has a father node, Carol. Caral has a father node, Alice. So there is circle relation between them:

```
Alice -> Bob -> Carol -> Alice
```

As a result, the `selectUpNodes` function still returns an array with all the element the same. So the return value of the `selectUpNodes` function could be manipulated.

File(s) Affected

0xa86a86b8acdc55812bec2971a2fc8a989455858c.bscscan.com-DSC/DSC.sol #48-60

```
48      function selectUpNodes(address from) external view returns (address[] memory nodes) {
49          nodes = new address[](maxUpNodeNum / 3);
50          address next = fathers[from];
51          uint k = 0;
52          for (uint i=0; i< maxUpNodeNum; i++) {
53              if (next != address(0) && next != defaultUpNode) {
54                  if (i % 3 == 0) {
55                      nodes[k++] = next;
56                  }
57                  next = fathers[next];
58              } else { break; }
59          }
60      }
```

Recommendation

Consider checking the implementation and redesigning the `selectUpNodes` function to cover the above circle relation cases.

Alleviation Acknowledged

The team responded that the document of the function `selectUpNodes` will be published to alert users the potential influcence.

Informational (2)

1. Constant State Variable

Informational Security Analyzer

Constant state variables should be declared constant to save gas.

File(s) Affected

0xa86a86b8acdc55812bec2971a2fc8a989455858c.bscscan.com-DSC/DSC.sol #10-10

```
10      uint public maxUpNodeNum = 30;
```

Recommendation

Add the `constant` attributes to state variables that never change.

Alleviation Acknowledged

The team acknowledged this finding.

2. Missing Event Setter

Informational Security Analyzer

Functions update key state variables are highly recommended to emit events

File(s) Affected

0xa86a86b8acdc55812bec2971a2fc8a989455858c.bscscan.com-DSC/DSC.sol #26-32

```
26      function changeCondition(uint newCondition) public onlyOwner {
27          amountCondition = newCondition;
28      }
29
30      function changeDefaultUpNode(address newDefaultUpNode) public onlyOwner {
31          defaultUpNode = newDefaultUpNode;
32      }
```

0xa86a86b8acdc55812bec2971a2fc8a989455858c.bscscan.com-DSC/DSC.sol #34-46

```
34      function _addUpNode(address sender, address receiver) internal {
35          //   sender
36          if (fathers[sender] == address(0)) {
37              if (defaultUpNode != address(0)) { //       0
38                  fathers[sender] = defaultUpNode;
39              }
40          }
41
42          //   receiver       sender
43          if (fathers[receiver] == address(0)) {
44              fathers[receiver] = sender;
45          }
46      }
```

Recommendation

Emit events for the functions update the key state variables.

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.