METATRUST

Security Assessment for

HF RealX

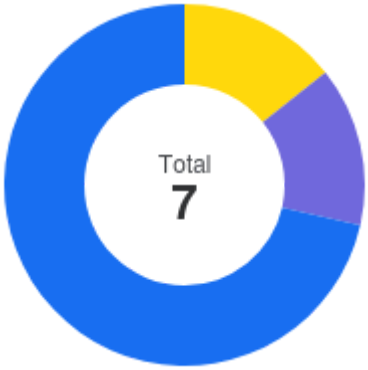
December 04, 2024

Executive Summary

Overview	
Project Name	HF RealX
Codebase URL	https://bscscan.com/address/0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2#code
Scan Engine	Security Analyzer
Scan Time	2024/12/04 08:00:00
Commit Id	-

Total	
Critical Issues	0
High risk Issues	0
Medium risk Issues	1
Low risk Issues	1
Informational Issues	5

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	14%	1
Low risk Issues	14%	1
Informational Issues	71%	5

Summary of Findings

MetaScan security assessment was performed on **December 04, 2024 08:00:00** on project **HF RealX** with the repository **bsc/0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2** 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 **1** medium risk vulnerabilities, **1** low risk vulnerabilities, **5** informational issues.

ID	Description	Severity	Alleviation
MSA-001	Premature Restake May Cause Loss of Stake Information and Rewards	Medium risk	Mitigated
MSA-002	Property `stakeType` Never Used	Low risk	Acknowledged
MSA-003	Uninitialized Local Variables	Informational	Acknowledged
MSA-004	Split `require` statements using `&&`	Informational	Acknowledged
MSA-005	Unhandled Return Value	Informational	Acknowledged
MSA-006	Improving The Clarity And Detail of The Error Messages	Informational	Acknowledged
MSA-007	External Calls Inside `for` Loops	Informational	Acknowledged

Findings

Medium risk (1)

1. Premature Restake May Cause Loss of Stake Information and Rewards

Medium risk | Security Analyzer

The `restake()` function in the `StakeHF` contract allows users to continue staking on an existing stake by updating the stake information stored at a specific index. However, this operation directly modifies the `StakeInfo` data for the given index, effectively replacing the previous stake details with the new ones.

This poses a risk that if a user calls the `restake()` function prematurely, before the system has calculated any rewards or performed other operations relying on the previous stake information, the historical data will be overwritten. As a result, any rewards or calculations based on the old stake information will be lost, and the user may not receive the correct rewards for their past stake.

Since the audit scope does not include the calculation of staking rewards and the logic for handling staking tokens, we hope the development team can provide additional clarification regarding this finding.

File(s) Affected

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #56-77

```
56      function restake(uint256 id, uint256 amount, uint256 stakeDays, uint256 stakeType) external {
57          require(stakeDays <= 10000, 'd');
58          StakeInfo storage s = users[msg.sender][id];
59          require(block.timestamp > s.stakeTime + s.stakeDays*86400, 't');
60
61          uint256 c = s.stakeAmount;
62          uint256 isR = s.isRedeem;
63          if(isR == 1) {
64              c_erc20.transferFrom(msg.sender, address(this), amount);
65          }else if(amount < c) {
66              c_erc20.transfer(msg.sender, c-amount);
67          }else if(amount > c) {
68              c_erc20.transferFrom(msg.sender, address(this), amount-c);
69          }
70
71          s.isRedeem = 0;
72          s.stakeType = uint16(stakeType);
73          s.stakeTime = uint32(block.timestamp);
74          s.stakeDays = uint32(stakeDays);
75          s.stakeAmount = uint160(amount);
76          emit Restake(msg.sender, id, amount, stakeType, stakeDays, block.timestamp);
77      }
```

Recommendation

We recommend that the client carefully review the implementation of the `restake()` function to ensure alignment with the intended design. Before allowing a `restake` to overwrite existing data, consider implementing a mechanism to confirm that the current stake data has been properly processed (e.g., rewards have been distributed or calculations are complete).

Alleviation Mitigated

[HF RealX, 12/04/2024]: This function does not affect users' staking rewards, ensuring that there are no issues or disruptions to their earnings.
[Metatrust, 12/04/2024]: The development team has assured that this function will not negatively impact users' staking. The staking reward mechanism is not within the scope of this audit, we encourage users to gather more information about the staking reward mechanism through public communities or the official website before participating in staking.

Low risk (1)

1. Property `stakeType` Never Used

Low risk | Security Analyzer

The `StakeInfo` struct contains a property `stakeType`, but this property is never used in the contract. Additionally, there is no validation in place to ensure that the value assigned to `stakeType` is valid.

Storing unnecessary data increases gas costs and can result in unused storage in the contract, reducing efficiency. And, without validation, there is a risk of assigning invalid or unexpected values to `stakeType`, potentially leading to undefined behavior.

File(s) Affected

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #23-23

```
23          uint16 stakeType;
```

Recommendation

If the stakeType is not used, we recommend removing it from the StakeInfo struct to optimize storage and reduce unnecessary complexity. If the stakeType is required, we recommend implementing validation to ensure it contains valid values before assignment.

Alleviation Acknowledged

[HF RealX, 12/02/2024]: The development team acknowledged this issue and decided not to make any changes.

Informational (5)

1. Uninitialized Local Variables

Informational | Security Analyzer

A local variable is either never initialized or is initialized only under certain conditions, while the variable will be used regardless of its initialization. As a result, the default zero value is used, which is not desired.

File(s) Affected

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #43-43

```
43          for(uint256 i; i < len; ++i) {
```

Recommendation

We recommend Initializing the local variable to a reasonable value. Explicitly setting it to zero if it is meant to be initialized to zero.

Alleviation Acknowledged

[HF RealX, 12/02/2024]: The development team acknowledged this issue and decided not to make any changes.

2. Split `require` statements using `&&`

Informational | Security Analyzer

Two or more statements connected by `&&` in `require` cost more gas than using separate `require` statements. For example, replacing with two separate `require` statements can save 16 gas. The same goes for `assert`.

File(s) Affected

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #50-50

```
50          require(s.isRedeem == 0 && block.timestamp > s.stakeTime + s.stakeDays*86400, 't');
```

Recommendation

We recommend replacing it with two or more separate `require` statements.

Alleviation Acknowledged

[HF RealX, 12/02/2024]: The development team acknowledged this issue and decided not to make any changes.

3. Unhandled Return Value

Informational Security Analyzer

The `IERC20` interface's `transfer()` and `transferFrom()` methods return a bool type value to indicate the success or failure of the token transfer operation. However, in the contract, the return values of these methods are not checked when the methods are invoked.

File(s) Affected

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #36-36

```
36          c_erc20.transferFrom(msg.sender, address(this), amount);
```

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #52-52

```
52          c_erc20.transfer(msg.sender, s.stakeAmount);
```

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #64-64

```
64              c_erc20.transferFrom(msg.sender, address(this), amount);
```

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #66-66

```
66              c_erc20.transfer(msg.sender, c-amount);
```

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #68-68

```
68              c_erc20.transferFrom(msg.sender, address(this), amount-c);
```

Recommendation

We recommend handling the return value of `transfer()` and `transferFrom()` to ensure the success of the token transfer operation.

Alleviation Acknowledged

[HF RealX, 12/02/2024]: The development team acknowledged this issue and decided not to make any changes.

4. Improving The Clarity And Detail of The Error Messages

Informational Security Analyzer

Although the require statements in the contract include error messages, the current messages are overly simple (e.g., "d" and "t"), which makes it difficult for users or developers to understand the root cause of the errors. For example, these messages fail to explain why a certain condition is not met or what corrective action the user should take. Developers and end-users may struggle to debug or correct their actions due to unclear error messages, they may spend additional time deciphering the intent of the messages.

File(s) Affected

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #35-35

```
35          require(stakeDays <= 10000, 'd');
```

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #50-50

```
50          require(s.isRedeem == 0 && block.timestamp > s.stakeTime + s.stakeDays*86400, 't');
```

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #57-57

```
57          require(stakeDays <= 10000, 'd');
```

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #59-59

```
59          require(block.timestamp > s.stakeTime + s.stakeDays*86400, 't');
```

Recommendation

We recommend improving the clarity and detail of the error messages to enhance readability and usability. Ensure they provide meaningful descriptions that accurately reflect the underlying issue.

Alleviation Acknowledged

[HF RealX, 12/02/2024]: The development team acknowledged this issue and decided not to make any changes.

5. External Calls Inside `for` Loops

Informational Security Analyzer

Consider limiting the number of iterations `for` loops that contain external calls to avoid an out-of-gas error.

File(s) Affected

0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol #41-46

```
41      function multiRedeem() external {
42          uint256 len = users[msg.sender].length;
43          for(uint256 i; i < len; ++i) {
44              redeem(i);
45          }
46      }
```

Alleviation Acknowledged

[HF RealX, 12/02/2024]: The development team acknowledged this issue and decided not to make any changes.

Audit Scope

File	SHA256	File Path
0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol	ffe4f081cdb7c39c17a0dcfe3fce7965474cab03dadbb77f7898e0f4819a09b5	/0×8AE3C263E111D1F1E2fAF6ddc729c0eFAc27c4f2.bscscan.com-StakeHF.sol

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.