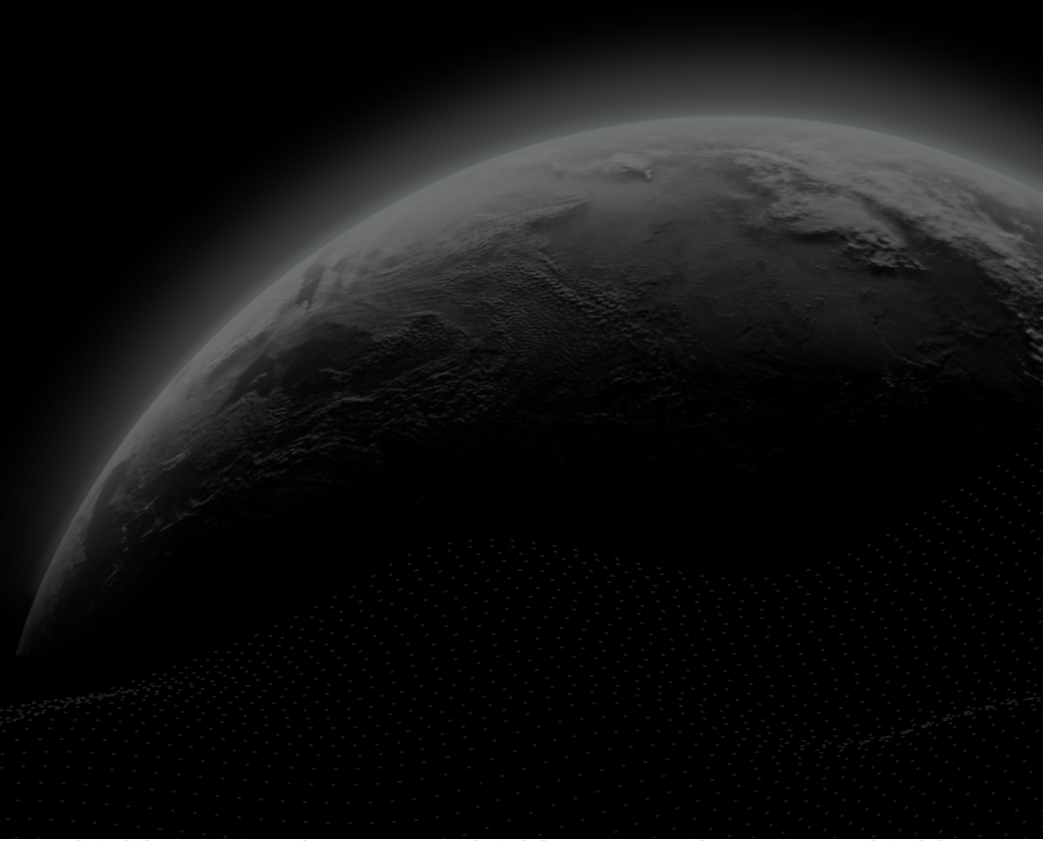
CERTIK

Preliminary Comments Draft (Internal Use Only)

PS.io

CertiK Verified on Sept 2nd, 2022



CertiK Verified on Sept 2nd, 2022

PS.io

These preliminary comments were prepared by CertiK, the leader in Web3.0 security.

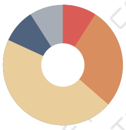
Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 09/02/2022	N/A

CODEBASE
https://github.com/project-seed-io/contracts-solidity
...View All

COMMITS
63d2f3128c592bc7d13b20c41436e0beb8b85f71
...View All

Vulnerability Summary



Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved
11	0	0	0	0	0	11

🟥	1	Critical	1 Unresolved	Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧	3	Major	3 Unresolved	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨	0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	5	Minor	5 Unresolved	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦	1	Informational	1 Unresolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.
⬜	1	Discussion	1 Unresolved	The impact of the issue is yet to be determined, hence requires further clarifications from the project team.

TABLE OF CONTENTS | PS.IO

CODEBASE | PS.IO

Repository

https://github.com/project-seed-io/contracts-solidity

Commit

63d2f3128c592bc7d13b20c41436e0beb8b85f71

AUDIT SCOPE | PS.IO

7 files audited ● 6 files with Unresolved findings ○ 1 file without findings

ID	File	SHA256 Checksum
● SLB	bridge-created by nonceblox/ShillLock.sol	4620089043e68797c80dd4481d2180eac3351f9f0638c1179f55a84aceadddf1
● NFE	nftengine-created by nonceblox/NFTEngine.sol	f5552259f5aef48f16a5d25f499838cb784aede23b80c91416cabd1c64fa23cf
● SHL	shill erc20 -created by nonceblox/Shill.sol	c46e041142db22d2a002e44110ced7a0d0fa931f7ec8bba09aec98ef519d6f11
● CSU	staking-modified by ps/extensions/CappableStaking.sol	f886c89af68d56b1cc22f192d33f47ead01c5af2c773ac9b22f1ba1b318fc874
● STN	staking-modified by ps/Staking.sol	6943a093d2fee7c900b1580d3d0554ce158e5a91118e87bd83b0af13214ad220
● APB	utils/AccessProtected.sol	b5fe59a3f38361cc5795d030f275793f1e10e7b85067cc1c3cf3db2a664daeda
○ ISU	staking-modified by ps/interfaces/IStaking.sol	a9549f27fcfe059c43dd9f1c22e571f6b24c9ed5a15a48a023acbff038072bab

APPROACH & METHODS | PS.IO

This report has been prepared for PS.io to discover issues and vulnerabilities in the source code of the PS.io project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

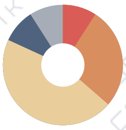
The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

FINDINGS | PS.IO



11 Total Findings

1 Critical

3 Major

0 Medium

5 Minor

1 Informational

1 Discussion

This report has been prepared to discover issues and vulnerabilities for PS.io. Through this audit, we have uncovered 11 issues ranging from different severity levels. Utilizing Static Analysis techniques to complement rigorous manual code reviews, we discovered the following findings:

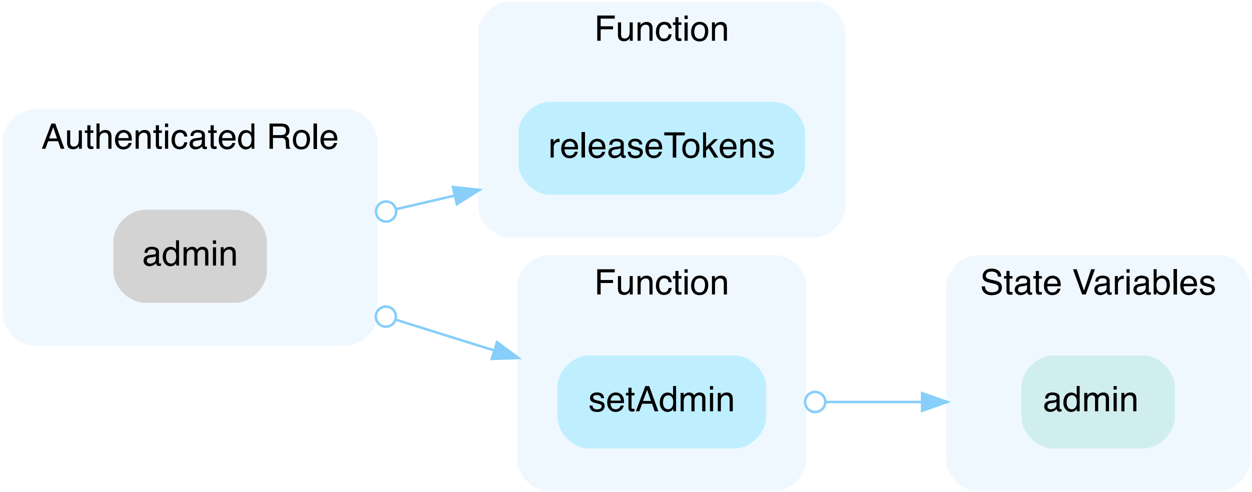
ID	Title	Category	Severity	Status
CON-01	**Centralization Related Risks**	**Centralization / Privilege**	**Major**	● **Unresolved**
CON-02	Third Party Dependency	Volatile Code	Minor	● Unresolved
CON-03	Lack Of Zero Address Validation	Volatile Code	Minor	● Unresolved
CON-04	Missing Input Validation	Volatile Code	Minor	● Unresolved
SHL-01	**Initial Token Distribution**	**Centralization / Privilege**	**Major**	● **Unresolved**
STN-01	Critical State Variable Not Updated	Logical Issue	Critical	● Unresolved
STN-02	Potential Reentrancy Attack (Sending Ether Or Tokens)	Volatile Code	Major	● Unresolved
STN-03	Unchecked ERC-20 `transfer()` / `transferFrom()` Call	Volatile Code	Minor	● Unresolved
STN-04	Function `loadReward()` Has No Access Control	Control Flow	Minor	● Unresolved
STN-07	`if` Condition Will Never Be Met	Logical Issue	Discussion	● Unresolved
STN-06	Incompatibility With Deflationary Tokens	Logical Issue	Informational	● Unresolved

CON-01 | CENTRALIZATION RELATED RISKS

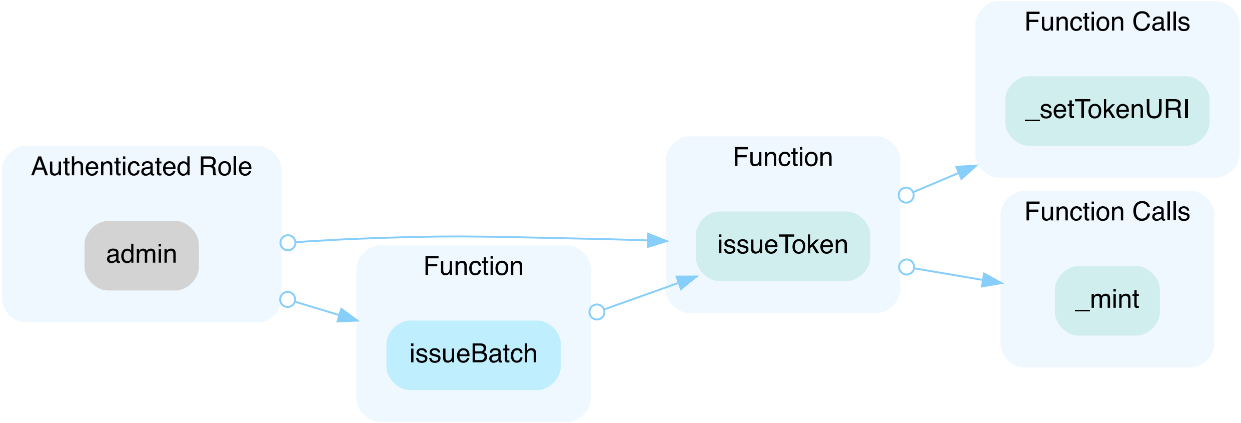
Category	Severity	Location	Status
Centralization / Privilege	● Major	bridge-created by nonceblox/ShillLock.sol: 33, 38; nftengine-created by nonceblox/NFTEngine.sol: 26, 42; staking-modified by ps/Staking.sol: 64, 70, 74; staking-modified by ps/extensions/CappableStaking.sol: 32; utils/AccessProtected.sol: 18	● Pending

Description

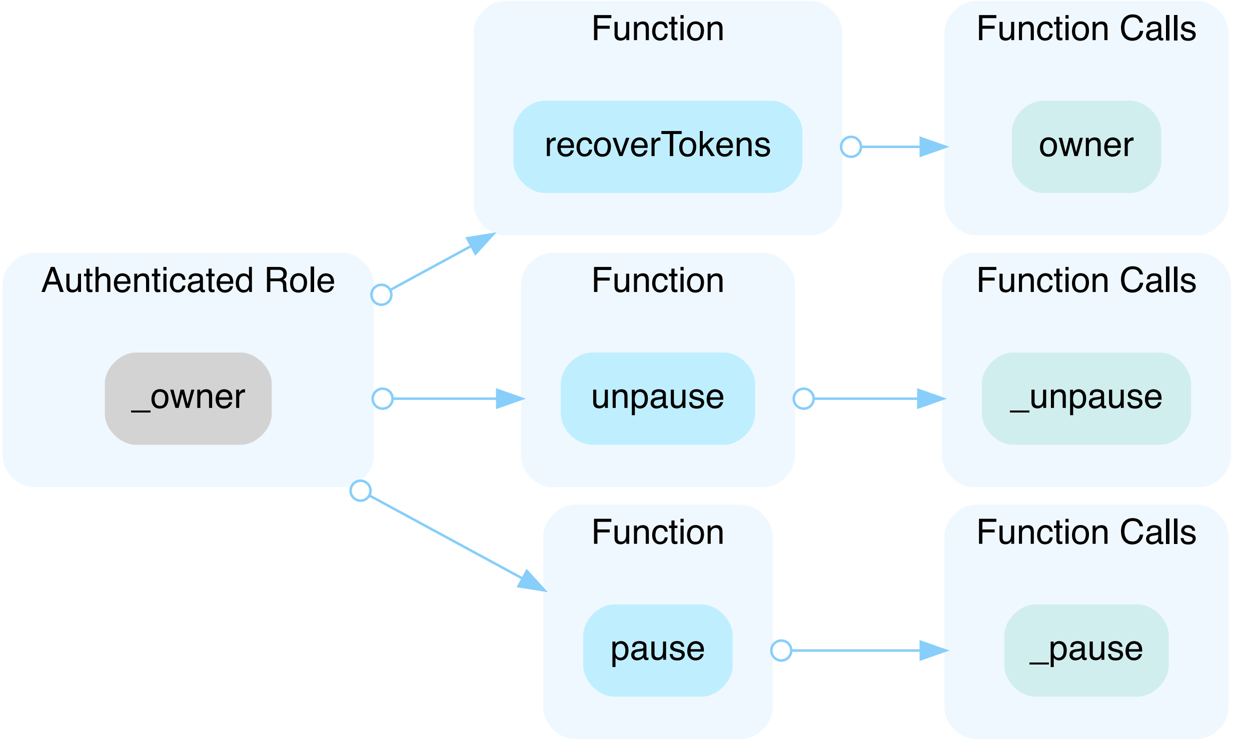
In the contract `ShillLock` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` account may allow the hacker to take advantage of this authority and drain all tokens by calling function `releaseTokens()`.



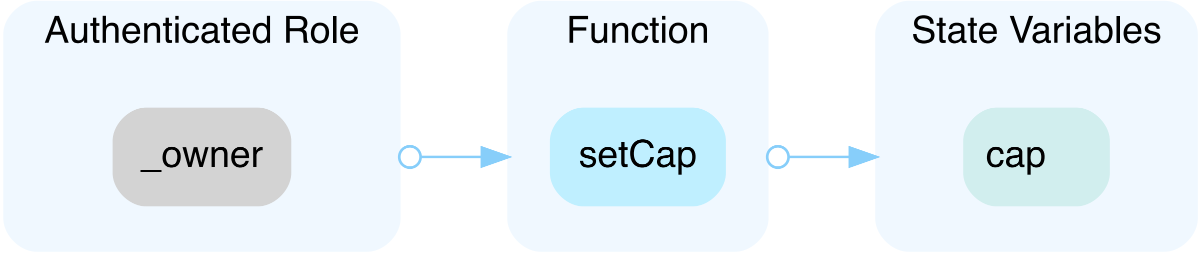
In the contract `NFTEngine` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` account may allow the hacker to take advantage of this authority and mint NFTs to attacker-controlled addresses by calling function `issueToken()` or `issueBatch()`.



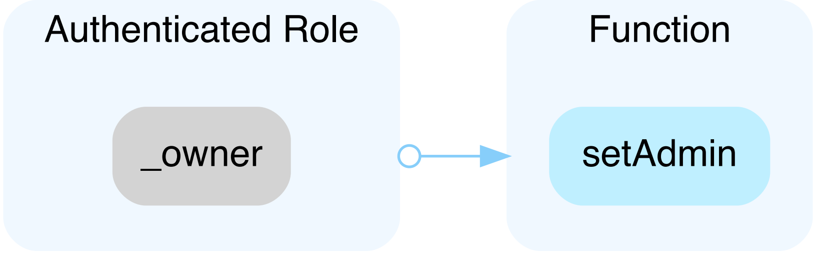
In the contract `Staking` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and drain all tokens from the contract, as well as pause the main utility of the contract, keeping users from claiming rewards or unstaking their tokens.



In the contract `CappableStaking` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and change the cap amount, making it difficult for users to call `stake()`.



In the contract `AccessProtected` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set an attacker-controlled address as an admin or revoke original admins by calling function `setAdmin()`.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

CON-02 | THIRD PARTY DEPENDENCY

Category	Severity	Location	Status
Volatile Code	● Minor	bridge-created by nonceblox/ShillLock.sol: 11; nftengine-created by nonceblox/NFTEngine.sol: 26~27; staking-modified by ps/Staking.sol: 13, 14	● Pending

Description

The contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

```
11      IERC20 public immutable token;
```

- The contract `ShillLock` interacts with third party contract with `IERC20` interface via `token`.

```
13      address public override stakingToken;
```

- The contract `Staking` interacts with third party contract with `IERC20` interface via `stakingToken`.

```
14      address public override rewardToken;
```

- The contract `Staking` interacts with third party contract with `IERC20` interface via `rewardToken`.

```
26        string memory hash
```

- The contract `NFTEngine` has functions `issueToken()` and `issueBatch()` which rely on a third party to provide input `hash` that is unique for the purposes of minting ERC721 tokens.

Recommendation

We understand that the business logic requires interaction with the third parties. We encourage the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

CON-03 | LACK OF ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	bridge-created by nonceblox/ShillLock.sol: 19~20, 33~34; utils/AccessProtected.sol: 18~19, 29~30	● Pending

Description

The cited functions lack important checks that the input addresses are never `address(0)` .

Recommendation

We recommend the client add a line to each function requiring that each address input is not `address(0)` .

CON-04 | MISSING INPUT VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	bridge-created by nonceblox/ShillLock.sol: 28~29, 33~34; staking-modified by ps/Staking.sol: 101~102; utils/AccessProtected.sol: 18~19	● Pending

Description

In contract `ShillLock` :

- The function `lockTokens()` is missing input validation that `_amount` is nonzero.
- The function `releaseTokens()` is missing input validation that `_amount` is nonzero.

In contract `AccessProtected` :

- The function `setAdmin()` is missing input validation that bool `enabled` is not already set as the output for mapping `_admins[admin]` .

In contract `Staking` :

- The function `unstake()` is missing input validation that `_amount` is non-zero.

Recommendation

We recommend adding in the validation checks specified above to prevent unexpected errors.

SHL-01 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization / Privilege	● Major	shill erc20 -created by nonceblox/Shill.sol: 8	● Pending

Description

Tokens are minted to the contract owner when deploying the contract. This is a centralization risk as the contract owner can distribute tokens without obtaining the consensus of the community.

Recommendation

We recommend the team provide an outline regarding the initial token distribution process. Additionally, we recommend making efforts to restrict the access of the token deployer's private key.

STN-01 | CRITICAL STATE VARIABLE NOT UPDATED

Category	Severity	Location	Status
Logical Issue	● Critical	staking-modified by ps/Staking.sol: 102~103, 110~111, 116~117, 122~123, 127	● Pending

Description

In function `claimReward()`, the stakeholder information is loaded into temporary variable `stakeholder` in *memory*, and is never used to update the storage information for the stakeholder. As such, when `stakeholder.timestamp` is updated to be `block.timestamp` in function `claimReward()`, it does not update the state, it only updates the temporary memory variable. Function `unstake()` has a similar issue.

With this vulnerability, attackers may drain almost all reward tokens and lock all users' staking tokens by calling `claimReward()` multiple times after `block.timestamp` is larger than `stopTime`.

As a result, other users are unable to receive staking tokens by calling function `unstake()` since the reward token amount may be insufficient and the transaction may revert.

Recommendation

We recommend updating the state variable `stakeholder.timestamp` for the user to avoid such an exploit.

STN-02 | POTENTIAL REENTRANCY ATTACK (SENDING ETHER OR TOKENS)

Category	Severity	Location	Status
Volatile Code	● Major	staking-modified by ps/Staking.sol: 106, 110~112, 116~117, 126~128, 170	● Pending

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

External call(s)

```
106                 _withdrawReward(msg.sender);
```

- This function call executes the following external call(s).
- In `Staking._withdrawReward`,
    - `IERC20(rewardToken).transfer(_to,reward)`

State variables written after the call(s)

```
110             stakeholder.staked -= _amount;
111             stakeholder.stakedRatio -= stakedRatio;
```

```
116           stakeholder.timestamp = 0;
```

External call(s)

```
126                 _withdrawReward(msg.sender);
```

- This function call executes the following external call(s).
- In `Staking._withdrawReward`,
    - `IERC20(rewardToken).transfer(_to,reward)`

State variables written after the call(s)

```
        stakeholder.timestamp = block.timestamp;

```

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

STN-03 | UNCHECKED ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	● Minor	staking-modified by ps/Staking.sol: 60, 66, 96, 165, 170	● Pending

Description

The return value of the transfer()/transferFrom() call is not checked.

```
60          IERC20(rewardToken).transferFrom(msg.sender, address(this),
rewardAmount);
```

```
66          _token.transfer(owner(), balance);
```

```
96          IERC20(stakingToken).transferFrom(msg.sender, address(this), _amount);
```

```
165         IERC20(stakingToken).transfer(_to, _amount);
```

```
170         IERC20(rewardToken).transfer(_to, reward);
```

Recommendation

Since some ERC-20 tokens return no values and others return a `bool` value, they should be handled with care. We recommend using the OpenZeppelin's `SafeERC20.sol` implementation to interact with the `transfer()` and `transferFrom()` functions of external ERC-20 tokens. The OpenZeppelin implementation checks for the existence of a return value and reverts if `false` is returned, making it compatible with all ERC-20 token implementations.

STN-04 | FUNCTION `loadReward()` HAS NO ACCESS CONTROL

Category	Severity	Location	Status
Control Flow	● Minor	staking-modified by ps/Staking.sol: 59~60	● Pending

Description

The function `loadReward()` can be called by anyone. This function transfers the `rewardAmount` of `rewardToken` from the `msg.sender` to the contract. While this action can be performed by any user on their own and the user calling the function would need to approve the contract to transfer the specified amount of tokens, it is in the user's best interest to keep this function protected.

Recommendation

We recommend adding in the modifier `onlyOwner` from the `Ownable` inheritance.

STN-07 | `if` CONDITION WILL NEVER BE MET

Category	Severity	Location	Status
Logical Issue	● Discussion	staking-modified by ps/Staking.sol: 88~90	● Pending

Description

The condition,

```
if (stakeholder.timestamp == 0)
```

will never be met, because in the previous line `stakeholder.timestamp` is set to `block.timestamp`. Therefore, the `if` block logic will never be executed.

Recommendation

We recommend the client reviews the code in the cited lines and revises the function based upon their intention.

STN-06 | INCOMPATIBILITY WITH DEFLATIONARY TOKENS

Category	Severity	Location	Status
Logical Issue	● Informational	staking-modified by ps/Staking.sol: 85, 96	● Pending

Description

When transferring deflationary ERC20 tokens, the input amount may not be equal to the received amount due to the charged transaction fee. For example, if a user sends 100 deflationary tokens (with a 10% transaction fee), only 90 tokens actually arrived to the contract. However, a failure to discount such fees may allow the same user to withdraw 100 tokens from the contract, which causes the contract to lose 10 tokens in such a transaction.

```
96          IERC20(stakingToken).transferFrom(msg.sender, address(this), _amount);
```

- Transferring tokens by `_amount`.

```
85          stakeholder.staked += _amount;
```

- The `_amount` appears to be used for bookkeeping purposes without compensating the potential transfer fees.

Recommendation

We recommend the client review whether a deflationary token will be used as the `stakingToken` and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

OPTIMIZATIONS | PS.IO

ID	Title	Category	Severity	Status
CON-05	Improper Usage Of `public` And `external` Type	Gas Optimization	Optimization	● Unresolved
STN-05	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	● Unresolved

CON-05 | IMPROPER USAGE OF `public` AND `external` TYPE

Category	Severity	Location	Status
Gas Optimization	● Optimization	nftengine-created by nonceblox/NFTEngine.sol: 42~43; staking-modified by ps/Staking.sol: 70, 74, 78, 101, 122, 138, 142, 153, 160; staking-modified by ps/extensions/CappableStaking.sol: 28, 32; utils/AccessProtected.sol: 29	● Pending

Description

`public` functions that are never called by the contract could be declared as `external`.

Recommendation

We recommend the client use the external attribute for public functions that are never called within the contract.

STN-05 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	● Optimization	staking-modified by ps/Staking.sol: 13, 14, 15, 16, 17, 18, 19	● Pending

Description

The linked variables assigned in the constructor can be declared as `immutable` . Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. An advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as immutable. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

APPENDIX | PS.IO

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Control Flow	Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

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 Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. 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 CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. 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.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.