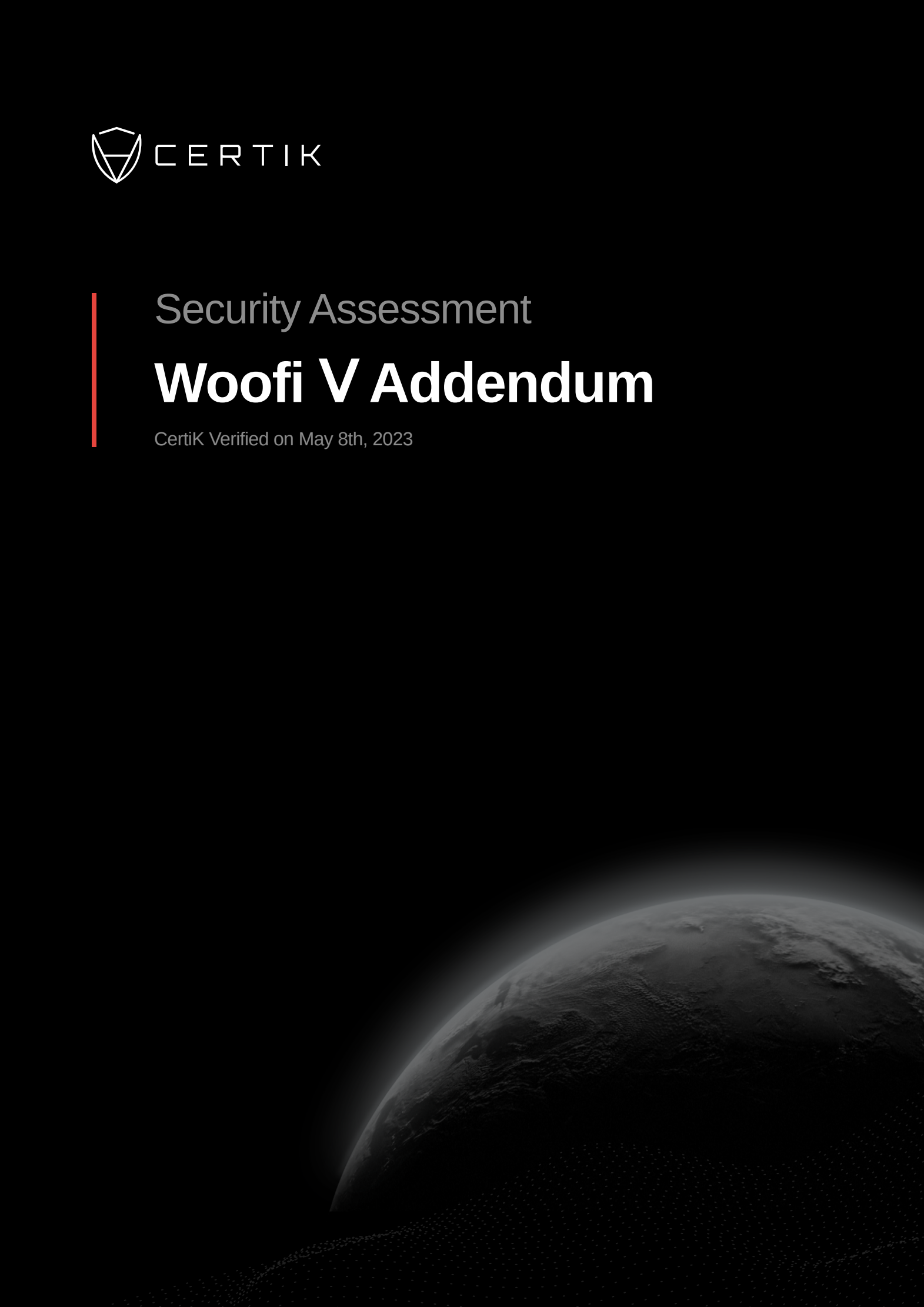
CERTIK

Security Assessment

Woofi V Addendum

CertiK Verified on May 8th, 2023



CertiK Verified on May 8th, 2023

Woofi V Addendum

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum (ETH)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 05/08/2023	N/A

CODEBASE

https://github.com/woonetwork/WooStakingV2

...View All

COMMITS

56c5e5e2fb9b05d25974dc129ccd9a08f932e088

ce04fbc7a67934c8f3e1158795984dfcca13e34a

...View All

Vulnerability Summary

9	8	0	0	1	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

0	Critical		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.
1	Major	1 Acknowledged	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.
1	Medium	1 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
3	Minor	3 Resolved	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.
4	Informational	4 Resolved	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.

TABLE OF CONTENTS | WOOFI V ADDENDUM

CODEBASE | WOOFI V ADDENDUM

Repository

https://github.com/woonetwork/WooStakingV2

Commit

56c5e5e2fb9b05d25974dc129ccd9a08f932e088
ce04fbc7a67934c8f3e1158795984dfcca13e34a

AUDIT SCOPE | WOOFI V ADDENDUM

3 files audited ● 3 files with Resolved findings

ID	Repo	Commit	File	SHA256 Checksum
● MRS	woonetwork/WooStakingV2	56c5e5e	contracts/rewarders/MpRewarder.sol	b23114820595fef06eb3d6f6c13c208a01aa22bf448ab50ff3feec1c81a21ed2
● RBS	woonetwork/WooStakingV2	56c5e5e	contracts/rewarders/RewardBooster.sol	4892ad8545c4fc1630c383b968d2efeb7e2ebc2fa6f2bc3434c115f4646ca26e
● SRS	woonetwork/WooStakingV2	56c5e5e	contracts/rewarders/SimpleRewarder.sol	899a33e86d3784d324c90fffa8c356dd315288d6ee6b7303a8955f269990eaee

APPROACH & METHODS | WOOFI V ADDENDUM

This report has been prepared for Woofi to discover issues and vulnerabilities in the source code of the Woofi V Addendum 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.

DEPENDENCIES | WOOFI V ADDENDUM

Third Party Dependencies

The protocol is serving as the underlying entity to interact with third party protocols. The third parties that the contracts interact with are:

- ERC20 Tokens

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.

Out Of Scope Dependencies

The protocol is serving as the underlying entity to interact with out-of-scope dependencies. The out-of-scope dependencies that the contracts interact with are:

- WooStakingCompounder

The scope of the audit treats out-of-scope dependencies as black boxes and assumes their functional correctness.

Furthermore this audit was conducted independently of the audit completed for the `BaseAdminOperation`, `WooStakingController`, `WooStakingManager`, and `WooStakingProxy`. The interactions between these contracts and `MpRewarder`, `RewardBooster`, `SimpleRewarder` are not in scope of either audit.

Recommendations

We recommend constantly monitoring the third parties involved to mitigate any side effects that may occur when unexpected changes are introduced. Additionally, we recommend all out-of-scope dependencies are carefully vetted to ensure they function as intended.

FINDINGS | WOOFI V ADDENDUM

9	0	1	1	3	4
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Woofi V Addendum. Through this audit, we have uncovered 9 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
GLOBAL-01	**Centralization Related Risks**	**Centralization / Privilege**	**Major**	**Acknowledged**
MRS-04	Admin Can Claim MP To Another User	Logical Issue	Medium	Resolved
REW-01	Missing Zero Address Validation	Volatile Code	Minor	Resolved
REW-02	Does Not Revoke Previous Authority	Logical Issue	Minor	Resolved
WSV-01	Potential Reentrancy	Volatile Code	Minor	Resolved
MRS-01	Time Units Can Be Used Directly	Language Specific	Informational	Resolved
MRS-02	`rewardToken` Unused In `MpRewarder`	Logical Issue	Informational	Resolved
REW-03	Usage Of Magic Numbers	Coding Style	Informational	Resolved
REW-04	Missing Checks	Logical Issue	Informational	Resolved

GLOBAL-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization / Privilege	**Major**		**Acknowledged**

Description

MpRewarder

In the contract `MpRewarder` the role `isAdmin` has authority over the following functions:

- `claim()`
- `setStakingManager()`
- `setRewardRate()`
- `setBooster()`

Any compromise to the `isAdmin` role may allow the hacker to take advantage of this authority and do the following:

- claim rewards for the user to another address;
- change the staking manager or booster to a malicious contract allowing them to gain more rewards or clear the rewards of other users;
- change the reward rate;

RewardBooster

In the contract `RewardBooster` the role `isAdmin` has authority over the following functions:

- `setUserRatios()`
- `setMPRewarder()`
- `setAutoCompounder()`
- `setVolumeBR()`
- `setTvlBR()`
- `setAutoCompoundBR()`

Any compromise to the `isAdmin` role may allow the hacker to take advantage of this authority change the reward boost of any user to whatever value they wish.

SimpleRewarder

In the contract `SimpleRewarder` the role `isAdmin` has authority over the following functions:

- `claim()`
- `setStakingManager()`
- `setRewardPerBlock()`

Any compromise to the `isAdmin` role may allow the hacker to take advantage of this authority and do the following:

- claim rewards for the user to another address;
- change the staking manager to a malicious contract allowing them to gain more rewards or clear the rewards of other users;
- change the reward rate per block;

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 recommend carefully managing 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., multi-signature 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.

MRS-04 | ADMIN CAN CLAIM MP TO ANOTHER USER

Category	Severity	Location	Status
Logical Issue	Medium	contracts/rewarders/MpRewarder.sol: 100~111	Resolved

Description

When `Woo` is unstaked a proportional amount of `MP` tokens are to be burned from the user, however, an admin can claim a users `MP` to another address allowing them to keep their `MP` in another address. This allows the user to unstake their `Woo`, but still have the same amout of `MP` held by another address.

Scenario

Assume that `UserA` has `addressA` which has a pending reward of 100 `MP` and that an admin calls `claim(addressA, addressB)`.

- Then the staking manager adds the reward amount of 100 `MP` for `addressB`.
- `UserA` then unstakes all `Woo`, but the `MP` for `addressB` will not be burned.

This shows how `MP` tokens may not be burned when all `Woo` that was staked to generate them is burned.

Recommendation

We recommend removing this function from the contract `MpRewarder` as the user whose rewards are claimed should always be the same user who receives the MP to ensure that when unstaking the correct proportional amount of `MP` is burned.

Alleviation

`[CertiK]`: The client changed the modifier so it can only be called by the staking manager, as it should only be used during auto compounding, in commit: ce04fbc7a67934c8f3e1158795984dfcca13e34a.

REW-01 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	contracts/rewarders/MpRewarder.sol: 64~65, 169, 182; contracts/rewarders/RewardBooster.sol: 64~65, 84, 89; contracts/rewarders/SimpleRewarder.sol: 60~61, 156	Resolved

Description

Addresses should be checked before assignment or external call to make sure they are not zero addresses.

The following functions do not verify the input `address` is not the zero address:

MpRewarder

- In the `constructor()`, the input `_rewardToken` and `stakingManager`.
- In the function `setStakingManager()`, the input `_manager`.
- In the function `setBooster()`, the input `_booster`.

RewardBooster

- In the `constructor()`, the input `_mpRewarder` and `_compounder`.
- In the function `setMPRewarder()`, the input `_rewarder`.
- In the function `setAutoCompounder(), the input` _compounder`.

SimpleRewarder

- In the `constructor()`, the input `_rewardToken` and `_stakingManager`.
- In the function `setStakingManager()`, the input `_manager`.

Recommendation

We recommend adding a zero-check for the passed-in address value to prevent unexpected errors. If these may want to be set to the zero address on deployment, then the checks do not need to be included in the `constructor()`.

Alleviation

`[CertiK]`: The client stated this is not needed for admin only methods as they will verify the inputs before calling.

REW-02 | DOES NOT REVOKE PREVIOUS AUTHORITY

Category	Severity	Location	Status
Logical Issue	Minor	contracts/rewarders/MpRewarder.sol: 168~172; contracts/rewarders/SimpleRewarder.sol: 155~159	Resolved

Description

If `setStakingManager()` is called to change the `stakingManager`, the new `stakingManager` is set as an admin. However, the old `stakingManager` is not removed as an admin. If the `stakingManager` is changed due to a compromise in the old `stakingManager` it should be ensured that they are removed as an Admin.

Recommendation

We recommend ensuring the old `stakingManager` is removed as an admin when `setStakingManager()` is called.

Alleviation

`[CertiK]`: The client made the recommended changes in commit: ce04fbc7a67934c8f3e1158795984dfcca13e34a.

WSV-01 | POTENTIAL REENTRANCY

Category	Severity	Location	Status
Volatile Code	Minor	MpRewarder.sol (ec47a037987ebedbc5ac05a65e5e60c0db6b41cd): 86, 87, 88; SimpleRewarder.sol (ec47a037987ebedbc5ac05a65e5e60c0db6b41cd): 155, 156, 157	Resolved

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.

Considering the reentrancy is only possible if certain tokens are used or an implementation is changed we mark this as minor

External call(s)

```
86          stakingManager.addMP(_to, rewardAmount);
```

State variables written after the call(s)

```
87          rewardClaimable[_user] = 0;
```

```
88          totalRewardClaimable -= rewardAmount;
```

With the current implementation of `addMP` in the contract `stakingManager` this is not possible. However, if the implementation changes to make an external call it may be possible to reenter the contract `MpRewarder` and call `updateReward()` or `updateRewardForUser()`.

External call(s)

```
155          TransferHelper.safeTransfer(rewardToken, _to, rewardAmount);
```

State variables written after the call(s)

```
157          rewardClaimable[_user] = 0;
```

```
156          totalRewardClaimable -= rewardAmount;
```

In this case if the `rewardToken` implements hooks, then it may be possible to reenter the contract `SimpleRewarder` to call `updateReward()` or `updateRewardForUser()`.

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or to ensure that any future implementation of `stakingManager` does not make an external call allowing reentrancy in `addMP()` and that no `rewardToken` that uses hooks allowing reentrancy is allowed.

Alleviation

`[CertiK]`: The client moved the external calls so that they are made after the state variable updates in commit: ce04fbc7a67934c8f3e1158795984dfcca13e34a.

MRS-01 | TIME UNITS CAN BE USED DIRECTLY

Category	Severity	Location	Status
Language Specific	Informational	contracts/rewarders/MpRewarder.sol: 83, 93, 122, 134	Resolved

Description

Suffixes like seconds, minutes, hours, and days can be used to specify units of time where seconds are the base unit and units are considered naively in the following way:

- `1` == `1 seconds`
- `1 minutes` == `60 seconds`
- `1 hours` == `60 minutes`
- `1 days` == `24 hours`

This can increase the readability of the code.

Recommendation

We recommend using `365 days` instead of `31536000` to improve readability.

Alleviation

`[CertiK]`: The client made the recommended changes in commit: ce04fbc7a67934c8f3e1158795984dfcca13e34a.

MRS-02 | `rewardToken` UNUSED IN `MpRewarder`

Category	Severity	Location	Status
Logical Issue	Informational	contracts/rewarders/MpRewarder.sol: 49, 63~64	Resolved

Description

The contract `MpRewarder` inherits `IRewarder` and thus must implement a `rewardToken()` function. However, this contract never uses the reward token and is designed to reward `MP` as opposed to a token.

Recommendation

We recommend creating a `MpRewarder` interface and removing the `rewardToken` from the contract to avoid any potential confusion.

Alleviation

`[CertiK]`: The client removed `rewardToken` from the contract `MPRewarder` in commit: ce04fbc7a67934c8f3e1158795984dfcca13e34a.

REW-03 | USAGE OF MAGIC NUMBERS

Category	Severity	Location	Status
Coding Style	Informational	contracts/rewarders/MpRewarder.sol: 83, 84, 87, 93, 122, 123, 134, 135, 138, 148; contracts/rewarders/SimpleRewarder.sol: 79, 82, 117, 129, 132, 142	Resolved

Description

The magic number `10000` is used as a denominator for the reward rate in the contract `MpRewarder`. Similarly the magic number `1e18` is used in the contract `MpRewarder` and `SimpleRewarder` for precision.

Recommendation

We recommend declaring descriptive constants and using them in place of the magic numbers to improve the codes maintainability and readability.

Alleviation

`[CertiK]`: The client added clarifying comments in commit: ce04fbc7a67934c8f3e1158795984dfcca13e34a.

REW-04 | MISSING CHECKS

Category	Severity	Location	Status
Logical Issue	Informational	contracts/rewarders/MpRewarder.sol: 176; contracts/rewarders/RewardBooster.sol: 93~106	Resolved

Description

In the contract `RewardBooster`, the functions `setVolumeBR()`, `setTvlBR()`, and `setAutoCompoundBR()` are all used to change the respective boost ratio. However, the input is never checked to ensure that it will boost the rewards. We recommend checking that the input is greater than or equal to `base` to ensure that the boost is always greater than or equal to 1.

In the contract `MpRewarder`, the function `setRewardRate()` does not have an upper or lower bound. We recommend setting reasonable upper and lower bounds for the reward rates and checking against them.

Recommendation

We recommend adding the checks mentioned above.

Alleviation

`[CertiK]`: The client stated there are no checks needed for the admin-only methods as they will ensure proper values are input.

OPTIMIZATIONS | WOOFI V ADDENDUM

ID	Title	Category	Severity	Status
MRS-03	Divide By Multiplication Instead Of Double Division	Gas Optimization	Optimization	Resolved

MRS-03 DIVIDE BY MULTIPLICATION INSTEAD OF DOUBLE DIVISION

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/rewarders/MpRewarder.sol: 83, 93, 122, 134	Resolved

Description

When calculating `rewards` the following calculation is made:

```
uint256 rewards = ((block.timestamp - lastRewardTs) * _totalWeight * rewardRate) /
10000 / 31536000;
```

However if instead of the dividing by 10000 and then dividing by 31536000, if it is divided by $(1000 * 31536000)$ it can reduce gas costs. This will slightly reduce the deployment size saving on deployment gas costs and save a small amount of gas on function calls.

Recommendation

We recommend multiplying the values and dividing by that value as opposed to dividing by one then the other to save gas.

Alleviation

`[CertiK]` : The client made the recommended changes in commit: ce04fbc7a67934c8f3e1158795984dfcca13e34a.

APPENDIX | WOOFI V ADDENDUM

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.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Language Specific	Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

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.