GA GUARDIAN

Magna

Magna Staking

Security Assessment

June 18th, 2025

Summary

Audit Firm Guardian

Prepared By Parker Stevens, Zdravko Hristov, Michael Lett

Client Firm Magna

Final Report Date June 18, 2025

Audit Summary

Magna engaged Guardian to review the security of their Magna fixed and dynamic staking. From the 2nd of June to the 4th of June, a team of 3 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Confidence Ranking

Given the lack of critical issues detected and minimal code changes following the main review, Guardian assigns a Confidence Ranking of 5 to the protocol. Guardian advises the protocol to consider periodic review with future changes. For detailed understanding of the Guardian Confidence Ranking, please see the rubric on the following page.

🔗 Blockchain network: **Ethereum, Base, Optimism, Polygon, Arbitrum, BNB Chain**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

📊 Code coverage & PoC test suite: https://github.com/GuardianOrg/staking-pocmagna-staking-team1, https://github.com/GuardianOrg/staking-pocmagna-staking-fuzz

Guardian Confidence Ranking

Confidence Ranking	Definition and Recommendation	Risk Profile
5: Very High Confidence	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. **Recommendation:** Code is highly secure at time of audit. Low risk of latent critical issues.	0 High/Critical findings and few Low/Medium severity findings.
4: High Confidence	Code is clean, well-structured, and adheres to best practices. Only Low or Medium-severity issues were discovered. Design patterns are sound, and test coverage is reasonable. Small changes, such as modifying rounding logic, may introduce new vulnerabilities and should be carefully reviewed. **Recommendation:** Suitable for deployment after remediations; consider periodic review with changes.	0 High/Critical findings. Varied Low/Medium severity findings.
3: Moderate Confidence	Medium-severity and occasional High-severity issues found. Code is functional, but there are concerning areas (e.g., weak modularity, risky patterns). No critical design flaws, though some patterns could lead to issues in edge cases. **Recommendation:** Address issues thoroughly and consider a targeted follow-up audit depending on code changes.	1 High finding and ≥ 3 Medium. Varied Low severity findings.
2: Low Confidence	Code shows frequent emergence of Critical/High vulnerabilities (~2/week). Audit revealed recurring anti-patterns, weak test coverage, or unclear logic. These characteristics suggest a high likelihood of latent issues. **Recommendation:** Post-audit development and a second audit cycle are strongly advised.	2-4 High/Critical findings per engagement week.
1: Very Low Confidence	Code has systemic issues. Multiple High/Critical findings (≥5/week), poor security posture, and design flaws that introduce compounding risks. Safety cannot be assured. **Recommendation:** Halt deployment and seek a comprehensive re-audit after substantial refactoring.	≥5 High/Critical findings and overall systemic flaws.

Table of Contents

Project Overview

Project Summary

Project Name	Magna
Language	Solidity
Codebase	https://github.com/magna-eng/staking-poc
Commit(s)	Initial commit(s): 9ec365cb0f30bdde6617fce6616d1208ba931e91 Final commit: 9ec365cb0f30bdde6617fce6616d1208ba931e91

Audit Summary

Delivery Date	June 18, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
High	1	0	0	0	0	1
Medium	3	0	0	0	0	3
Low	16	0	0	3	0	13
Info	0	0	0	0	0	0

Audit Scope & Methodology

Scope and details:
Contract, source, total, comment staking-poc/src/dynamicApy/DynamicStakingFactory.sol,21,36,9
staking-poc/src/dynamicApy/DynamicStaking.sol,281,363,6
staking-poc/src/fixedApy/FixedStakingFactory.sol,21,26,0
staking-poc/src/fixedApy/FixedStaking.sol,292,464,100
source count: {total: 889, source: 615, comment: 115, single: 56, block: 59, mixed: 5, empty: 171, todo: 0, blockEmpty: 7, commentToSourceRatio: 0.18699186991869918}

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	🔴 Critical	🟠 High	🟡 Medium
Likelihood: *Medium*	🟠 High	🟡 Medium	🟢 Low
Likelihood: *Low*	🟡 Medium	🟢 Low	🟢 Low

Impact

High Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.

Medium A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.

Low Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.

Medium An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.

Low Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

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. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Invariants Assessed

During Guardian's review of Magna, fuzz-testing was performed on the protocol's main functionalities. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

Throughout the engagement the following invariants were assessed for a total of 10,000,000+ runs with a prepared fuzzing suite.

ID	Description	Tested	Passed	Remediation	Run Count
GLOB-01	Contract balances match internal accounting	✅	✅	✅	10M+
GLOB-02	Total supply should not increase unexpectedly	✅	✅	✅	10M+
GLOB-03	Contract has sufficient tokens to cover obligations	✅	✅	✅	10M+
GLOB-04	Compounding/interest math correctness (fixed staking)	✅	✅	✅	10M+
GLOB-05	Reward calculation accounts for decimal differences	✅	✅	✅	10M+
ERR-01	Failed operations don't corrupt state	✅	✅	✅	10M+
ERR-02	State remains consistent after failures	✅	✅	✅	10M+
STAKE-01	Sum of individual stakes equals total staked	✅	✅	✅	10M+
STAKE-02	Mathematical correctness of reward formulas	✅	✅	✅	10M+
STAKE-03	Proper handling of time-based logic	✅	✅	✅	10M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
ACCESS-01	Only authorized actors can call restricted functions	✅	✅	✅	10M+
ACCESS-02	Role assignments remain consistent	✅	✅	✅	10M+
ECON-01	No value created or destroyed unexpectedly	✅	✅	✅	10M+
ECON-02	Correct fee calculations and transfers	✅	✅	✅	10M+
ECON-03	Fair and accurate reward distribution	✅	✅	✅	10M+
DYN-01	Pending stakes/unstakes/claims are properly tracked	✅	✅	✅	10M+
DYN-02	APY calculations based on total staked and rewards	✅	✅	✅	10M+
DYN-03	Campaign periods and timing constraints	✅	✅	✅	10M+
DYN-04	Unstake and claim fees properly collected	✅	✅	✅	10M+
FIXED-01	Valid state transitions only	✅	✅	✅	10M+
FIXED-02	Minimum lock periods and boost calculations	✅	✅	✅	10M+
FIXED-03	Maximum staking limits respected	✅	✅	✅	10M+
FIXED-04	Consistent fixed APY with time-based compounding	✅	✅	✅	10M+

Findings & Resolutions

ID	Title	Category	Severity	Status
H-01	FixedStaking Doesn't Consider Token Decimals	Unexpected Behavior	High	Resolved
M-01	User Can Achieve Max Boost	Unexpected Behavior	Medium	Resolved
M-02	Underflow On Interest Percentage Calculation	Math	Medium	Resolved
M-03	Fee Discount Via Multicall	Logical Error	Medium	Resolved
L-01	Inaccurate getClaimable() Results	Unexpected Behavior	Low	Resolved
L-02	Pool Capacity Validation Can Be Bypassed	Validation	Low	Resolved
L-03	User Stakes Can Be Spammed	DoS	Low	Resolved
L-04	poolCapacity Update Can Be Frontrun	Informational	Low	Acknowledged
L-05	withdrawOnBehalf() Marked As Payable	Informational	Low	Resolved
L-06	Wrong Error Usage	Validation	Low	Resolved
L-07	State Is Broken After defund	Informational	Low	Resolved
L-08	Campaign Change Can Cause Zero Reward Rate	Rewards	Low	Resolved
L-09	0 Transfer Reverts	Best Practices	Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-10	Contract Can Be Created	Informational	Low	Resolved
L-11	Loss Of Interest Because Of Precision Loss	Rounding	Low	Acknowledged
L-12	Insufficient ConstructorParams Validation	Validation	Low	Acknowledged
L-13	Misused Variable tokenAmountToRescue	Informational	Low	Resolved
L-14	withdrawOnBehalf() Always Withdraws For Msg.sender	Informational	Low	Resolved
L-15	Pending Rewards Can Not Be Claimed	Informational	Low	Resolved
L-16	No Max Lock Duration	Best Practices	Low	Resolved
L-17	Typos And Nitpicks	Best Practices	Low	

H-01 | FixedStaking Doesn't Consider Token Decimals

Category	Severity	Location	Status
Unexpected Behavior	High	FixedStaking.sol: 435	Resolved

Description PoC

`FixedStaking` can be deployed by anyone using any two tokens as `stakeToken` and `rewardToken`. The earned amount of rewards are calculated in `calculateRewardAmountFutureDate()` as

```
FixedPointMathLib.mulDiv(stakeAmount, (scaledInterestRate - WAD), WAD)
```

This means the result ends up being in `stake` token decimals even though it's used in the context of rewards. This mistake leads to unexpected results depending on what decimals the tokens have and how many rewards are currently available.

For example, if the `stakeToken` has 18 decimals, but the `rewardToken` has 6, in most cases the earned amount will be stored in `pendingRewards` and users won't be able to claim it or if they do, they will earn a lot more than they have to.

The other case is also possible - 6 decimals for `stakeToken` and 18 decimals for `rewardToken` - then users will receive almost nothing compared to what they are entitled to.

Recommendation

Store the two tokens' decimals by loading them in the constructor and modify `calculateRewardAmountFutureDate()` to return the result in `rewards` decimals.

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

M-01 | User Can Achieve Max Boost

Category	Severity	Location	Status
Unexpected Behavior	Medium	FixedStaking.sol: 303-307	Resolved

Description

Users can initiate last minute deposits prior to the admin updating the stake state. The user is able to enter a large `lockDuration` .

They will immediately be able to claim the rewards when the stake state changes to `SKIP_LOCKUP_ALLOWED` and the large lock duration will apply the `maxBoost` to the user's rewards, even though their funds were only locked up for a short time.

Recommendation

Ensure that the unlock time can not be later than the compoundEndDate to avoid last second locks.

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

M-02 | Underflow On Interest Percentage Calculation

Category	Severity	Location	Status
Math	Medium	FixedStaking.sol: 475	Resolved

Description PoC

FixedStaking.calculateInterestPercentageFutureDate() calculates elapsedTime by taking the maximum between (0, currentTimestamp - startDate).

The problem is that both currentTimestamp and startDate are unsigned integers. Because of this, when startDate > currentTimestamp, the result won't be 0, but the transaction with revert with an underflow instead.

This can happen when a user has staked after compoundingEndDate. The user will lose their staked funds as a result.

Recommendation

```
• uint256 elapsedTime = FixedPointMathLib.max(0, currentTimestamp - startDate);
+ uint256 elapsedTime = currentTimestamp > startDate • currentTimestamp -
startDate :  0;
```

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

M-03 | Fee Discount Via Multicall

Category	Severity	Location	Status
Logical Error	Medium	DynamicStaking.sol	Resolved

Description

DynamicStaking.unstake() and DynamicStaking.claim() require that msg.value = unstakeFee and msg.value = claimFee respectively.

However, the Multicall contract is inherited which allows users to execute multiple unstake() and claim() calls in the same transaction.

As a result, the msg.value will always be the same value for the different calls and users can execute the function many times, but pay only once.

If claimFee = unstakeFee, they can even batch a call to both claim() and unstake() and pay only once.

Recommendation

An easy solution is to add a check to the constructor that if the fees are positive, they must not be equal. However, this solution limits the flexibility of the contract.

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

L-01 | Inaccurate getClaimable() Results

Category	Severity	Location	Status
Unexpected Behavior	Low	FixedStaking.sol: 377-409	Resolved

Description

`FixedStaking.getClaimable()` returns the amount of rewards earned by a given active stake or all active stakes.

However, it doesn't check if the state is `FORCEFULLY_TERMINATED`. If it is, the actual claimable amount is 0, but they will return a positive amount.

Recommendation

Return 0 if `state = State.FORCEFULLY_TERMINATED`.

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

L-02 | Pool Capacity Validation Can Be Bypassed

Category	Severity	Location	Status
Validation	Low	FixedStaking.sol: 151	Resolved

Description

The constructor of `FixedStaking` ensures that `poolCapacity > 0`, but this check is not present in the `changePoolCapacity()` function.

Calling that function right after deployment bypassed the check in the constructor and the value can be set to 0.

Recommendation

Consider implementing the check in `changePoolCapacity()` as well.

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

L-03 | User Stakes Can Be Spammed

Category	Severity	Location	Status
DoS	Low	FixedStaking.sol: 260-276	Resolved

Description

`FixedStaking.stake()` allows staking on behalf of any user without any restrictions on the staked amount besides it's positive.

This allows anyone to spam the stakes of another user by depositing 1 wei multiple times and in result because `getClaimable()` and `getClaimableIncludingPending()` revert with OOG when looping through the stakes. It can also cause problems for any integrators using `getNumberOfStakes()`.

Recommendation

Consider requiring minimum stake amount if `initiator = onBehalfOf`

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

L-04 | poolCapacity Update Can Be Frontrun

Category	Severity	Location	Status
Informational	Low	FixedStaking.sol	Acknowledged

Description

The `changePoolCapacity()` function in `FixedStaking` allows the admin to change the capacity of the pool at any given point in time.

They can use it to change the capacity to a lower value, but anyone can frontrun their transaction and stake before the changes have taken place.

For example:
• `totalStaked = 1000`
• `poolCapacity = 1500`
• the admin submits a transaction to change `poolCapacity` to 1200
• A staker frontruns them (or their transaction just ends up earlier in the block) and stakes up to 1500
• `poolCapacity` is set to `1200`, but `totalStaked = 1500`

Recommendation

This is a common problem encountered with `ERC20.approve()` as well. You can document this behavior and be careful when you change the limit.

Resolution

Magna Team: Acknowledged.

L-05 | withdrawOnBehalf() Marked As Payable

Category	Severity	Location	Status
Informational	Low	DynamicStaking.sol: 225-230	Resolved

Description

The function `withdrawOnBehalf()` is marked as `payable` but does not require a fee for its execution.

Recommendation

Consider removing the `payable` keyword to avoid confusion.

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

L-06 | Wrong Error Usage

Category	Severity	Location	Status
Validation	Low	FixedStaking.sol: 264	Resolved

Description

`FixedStaking.stake()` incorrectly reverts with the `ZeroAddressPassed()` error if the stake is using a lock less than the minimum allowed one.

Recommendation

It should revert with `MinimumLockDurationViolated()` instead.

```
- require(lockDuration >= minimumLockDuration, ZeroAddressPassed());
+ require(lockDuration >= minimumLockDuration, MinimumLockDurationViolated());
```

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

L-07 | State Is Broken After defund

Category	Severity	Location	Status
Informational	Low	DynamicStaking.sol	Resolved

Description

Once `DynamicStaking.defundContractBalance()` is called, all of the tokens in the contract - stakes + unstakes + rewards are send to a recipient address, but `totalStaked` and entries of the user stakes are not modified at all, which breaks the accounting mechanism of the contract.

Recommendation

Consider pausing the contract when defunding so users won't interact with it anymore.

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

L-08 | Campaign Change Can Cause Zero Reward Rate

Category	Severity	Location	Status
Rewards	Low	DynamicStaking.sol	Resolved

Description

DynamicStaking.modifyCampaign() doesn't check the length of the reward period - its end must only be in the future.

If the reward period is too long, it can cause the reward rate to become too low, resulting in 0 rewards for stakers because of rounding issues.

Recommendation

Consider checking that reward rate is at least greater than 0 at the end of modifyCampaign.

```
require(rewardRemaining * WAD / (periodFinish - lastUpdateTime) > 0);
```

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

L-09 | 0 Transfer Reverts

Category	Severity	Location	Status
Best Practices	Low	DynamicStaking.sol	Resolved

Description

`DynamicStaking.modifyCampaign()` transfers the token unconditionally in the `else` statement, even if the amount is 0. This may result in reverts for some tokens that fail on 0 transfer.

It may be a desired use case to call `modfyCampaign()` with the same amount as the contract holds if for example it couldn't be claimed during the last distribution.

Recommendation

Consider transferring the tokens only if amount is not 0.

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

L-10 | Contract Can Be Created

Category	Severity	Location	Status
Informational	Low	FixedStaking.sol: 114	Resolved

Description

The SUPER_ADMIN role is the only role allowed to handle the following processes:

```
Super admin manage other roles like admin can initiate all state transitions
can rescue reward tokens at any time if it was allowed during the creation of
the pool can rescue staked tokens if it was allowed during the creation of the
pool
```

However, the constructor only checks that there is at least 1 admin role or 1 super admin role, but does not enforce that there is at least 1 super admin role. This will disallow any of the above processes from happening.

```
require((params.superAdmins.length + params.admins.length > 0)
```

Recommendation

Ensure that there is at least one super admin role assigned in the constructor.

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

L-11 | Loss Of Interest Because Of Precision Loss

Category	Severity	Location	Status
Rounding	Low	FixedStaking.sol: 476	Acknowledged

Description

FixedStaking.calculateInterestPercentageFutureDate() computes the interest percentage the user is eligible to receive when they unstake their tokens. This percentage depends on how many periods have passed

`periodsElapsed = (elapsedTime) / compoundingPeriodLength`

Because there is no scaling applied to these values, unstaking even a second earlier will result in a loss of interest for the user for 1 entire period.

In addition, if the state was transition to TERMINATED or the block.timestamp is beyond compoundingEndDate, then lastValidTimestamp will be used to calculate timeElapsed and the user won't be able to time their unstake transaction even if they are aware of the issue.

The contract configuration is permissionless, this means a period can be 7 days, a year, 4 years, etc... Losing 1 whole period of rewards is a substantial amount in these cases.

Recommendation

```
function calculateInterestPercentageFutureDate(uint256 startDate, uint256 futureDate)

      public view virtual returns (uint256 interestRate) {uint256 lastValidTimestamp =
terminationTimestamp = 0 • FixedPointMathLib.min(futureDate, terminationTimestamp) :
futureDate; uint256 currentTimestamp = FixedPointMathLib.min(lastValidTimestamp,
compoundingEndDate); uint256 elapsedTime = FixedPointMathLib.max(0, currentTimestamp -
startDate); • uint256 periodsElapsed = elapsedTime / compoundingPeriodLength; + uint256
periodsElapsed = elapsedTime * WAD / compoundingPeriodLength; • uint256 allowedPeriodsElapsed
= maxCompoundingPeriods = 0 • FixedPointMathLib.min(periodsElapsed, maxCompoundingPeriods) :
periodsElapsed; + uint256 allowedPeriodsElapsed = maxCompoundingPeriods = 0 •
FixedPointMathLib.min(periodsElapsed, maxCompoundingPeriods * WAD) : periodsElapsed; • int256
interest = FixedPointMathLib.powWad(int256(WAD + interestRatePerPeriod),
int256(allowedPeriodsElapsed * WAD)); + int256 interest = FixedPointMathLib.powWad(int256(WAD
+ interestRatePerPeriod), int256(allowedPeriodsElapsed));
interestRate = uint256(interest);}
```

Resolution

Magna Team: Acknowledged.

L-12 | Insufficient ConstructorParams Validation

Category	Severity	Location	Status
Validation	Low	FixedStaking.sol	Acknowledged

Description

A lot of the fields passed in `IFixedParams.ConstructorParams` are not sufficiently validated:

• `compoundingPeriodLength` is not capped to a reasonable value. If it's set to a very large value like `type(uint256).max`, every time `periodElapsed` which is calculated as `elapsedTime / compoundingPeriodLength` will end up being 0 and users will never receive rewards.

• `interestRatePerPeriod` should be a value in a reasonable range. If it's not it's possible to grief users by exploiting `WAD + interestRatePerPeriod > type(uint256).max` and making `calculateInterestPercentageFutureDate()` always revert. It should also be in reasonable ranges because if it's too big, there will never be enough rewards to be paid out.

• `withdrawRewardsDeadline` must not be lower than a given value chosen by the protocol because now it can be set to 0 which allows the admin to transition the state from `INITIALIZED` all the way to `REWARD_RESCUABLE` in a single block and withdraw all the rewards.

• `boostYOffset` and `boostdYdX` should also be in reasonable ranges because `boostYOffset + boostdYdX * extraLockSeconds` should not surpass `type(uint256).max`

Recommendation

Implement the proposed validations in the `FixedStaking` constructor.

Resolution

Magna Team: Acknowledged.

L-13 | Misused Variable tokenAmountToRescue

Category	Severity	Location	Status
Informational	Low	FixedStaking.sol: 205	Resolved

Description

In the `emergencyRescueStakeTokens` function, there is the following check:

```
if (tokenAmountToRescue > 0)

{totalStaked = rescuedTokenAmount; stakeToken.safeTransfer(rescuedTokenReceiver, rescuedTokenAmount);}
```

However, this should always evaluate to `true` because `tokenAmountToRescue` is a function parameter. It would not make sense to call this function with a zero value.

Recommendation

Use `rescuedTokenAmount` in the if clause instead.

Resolution

Magna Team: Resolved in commit 74b3f78a8917a425b0893a5ad09d5ef93d158d1b.

L-14 | withdrawOnBehalf() Always Withdraws For Msg.sender

Category	Severity	Location	Status
Informational	Low	DynamicStaking.sol: 225-244	Resolved

Description

DynamicStaking.withdrawOnBehalf() indicates that users are able to withdraw on behalf of another user, however, the function always withdraws from msg.sender.

Recommendation

Consider renaming this function to withdraw() to follow similar conventions in the rest of the codebase.

Resolution

Magna Team: Resolved.

L-15 | Pending Rewards Can Not Be Claimed

Category	Severity	Location	Status
Informational	Low	FixedStaking.sol: 347-359	Resolved

Description

The `unstakeAndClaim()` function allow users to receive their rewards to another address, however, `claimPendingRewards()` does not. The rewards are always transferred to `msg.sender`.

Recommendation

Consider adding a parameter for the user to specify which address should receive the rewards.

Resolution

Magna Team: Resolved.

L-16 | No Max Lock Duration

Category	Severity	Location	Status
Best Practices	Low	FixedStaking.sol: 260	Resolved

Description

Users are able to create locks of any duration larger than the minimum lock duration. This may cause issues if a user initiates a deposit with an enormously large lock duration by accident.

Recommendation

Impose a strict max lock duration, e.g. 4 years.

Resolution

Magna Team: Resolved.

Disclaimer

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 Guardian 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. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’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 Guardian 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.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian

Founded in 2022 by DeFi experts, Guardian is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits