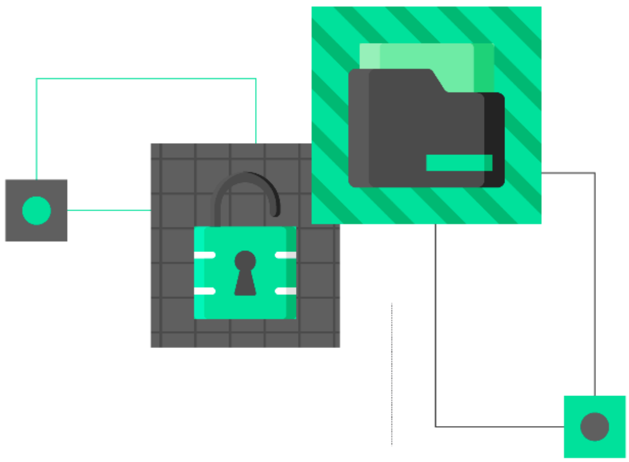
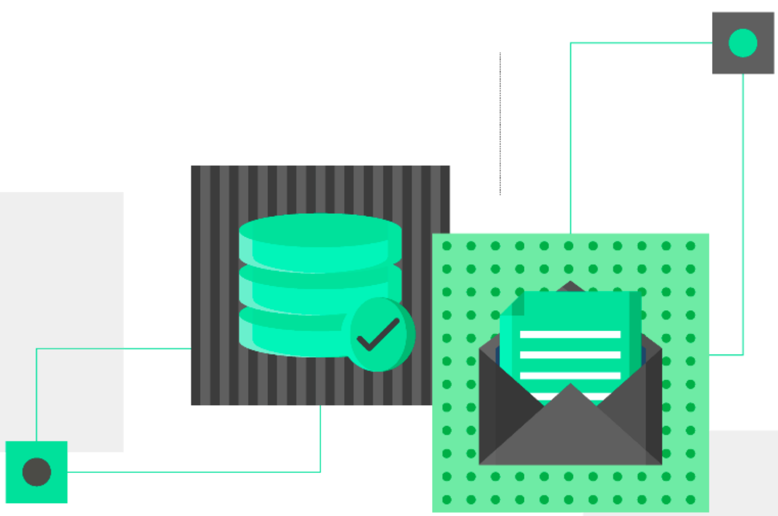
Mantisec Labs



Smart Contract Audit

`SearchStaking.sol`
Presearch



July 2025

Mantisec Labs

Contents

Disclaimer	3
Audit Process & Methodology	4
Audit Purpose	5
Contract Details	5
Security Level Reference	6
Findings	7
Additional Details	14
Concluding Remarks	15

Mantisec Labs

Disclaimer

This disclaimer is to inform you that the report you are reading has been prepared by Mantisec Labs for informational purposes only and should not be considered as investment advice. It is important to conduct your own independent investigation before making any decisions based on the information contained in the report. The report is provided "as is" without any warranties or conditions of any kind and Mantisec Labs excludes all representations, warranties, conditions, and other terms. Additionally, Mantisec Labs assumes no liability or responsibility for any kind of loss or damage that may result from the use of this report.

It is important to note that the analysis in the report is limited to the security of the smart contracts only and no applications or operations were reviewed. The report contains proprietary information, and Mantisec Labs holds the copyright to the text, images, photographs, and other content. If you choose to share or use any part of the report, you must provide a direct link to the original document and credit Mantisec Labs as the author.

By reading this report, you are accepting the terms of this disclaimer. If you do not agree with these terms, it is advisable to discontinue reading the report and delete any copies in your possession.

Mantisec Labs

Audit Process & Methodology

The Mantisec Labs team carried out a thorough audit for the project, starting with an in-depth analysis of code design patterns. This initial step ensured the smart contract's architecture was well-structured and securely integrated with third-party smart contracts and libraries.Also, our team conducted a thorough line-by-line inspection of the smart contract,seeking out potential issues such as Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, among others.

During the Unit testing phase, we assessed the functions authored by the developer to ascertain their precise functionality. Our Automated Testing procedures leveraged proprietary tools designed in-house to spot vulnerabilities and security flaws within the Smart Contract. The code was subjected to an in-depth audit administered by an independent team of auditors, encompassing the following critical aspects:

- Scrutiny of the smart contract's structural analysis to verify its integrity.
- Extensive automated testing of the contract
- A manual line-by-line Code review, undertaken with the aim of evaluating, analyzing, and identifying potential security risks.
- An evaluation of the contract's intended behavior, encompassing a review of provided documentation to ensure the contract conformed to expectations.
- Rigorous verification of storage layout in upgradeable contracts.
- An integral component of the audit procedure involved the identification and recommendation of enhanced gas optimization techniques for the contract

Audit Purpose

Mantisec Labs was hired by the Presearch team to review their smart contract.
This audit was conducted in July 2025.

The main reasons for this review were:

- To find any possible security issues in the smart contract.
- To carefully check the logic behind the given smart contract.

This report provides valuable information for assessing the level of risk associated with this smart contract and offers suggestions on how to improve its security by addressing any identified issues.

Contract Details

Project Name	Presearch
Contract links	`SearchStaking.sol`
Language	Solidity
Type	ERC20

Mantisec Labs

Security Level Reference

Each problem identified in this report has been categorized into one of the following severity levels:

- **Critical**: Vulnerabilities that present an immediate and serious threat to system or data integrity, demanding urgent action.
- **High**: Significant risks that have the potential to cause major security breaches or loss of functionality.
- **Medium**: Issues that moderately affect system performance or security and require timely resolution.
- **Low**: Low-risk concerns primarily related to optimization and code quality, with minimal direct impact on system security.
- **Informational**: Observations or recommendations that do not pose any direct risk but provide insights for potential improvements or best practices.

Severity	Score
Critical	4-5
High	3-4
Medium	2-3
Low	1-2
Informational	0-1

Findings Overview

Contract Names:

- `SearchStaking.sol`

Critical	High	Medium	Low	Informational
0	0	3	2	0

Issue	Severity	Fix Date
M01- Missing Emergency Pause Mechanism	**Medium**(2)	10-07-2025
M02- Signature Hash Not Fully Typed	**Medium**(2)	10-07-2025
M03- No Domain Separation in Signatures	**Medium**(2)	10-07-2025
L01- Unbounded batchId String Length	**Low**(1)	10-07-2025
L02- Unsafe Token Transfers Without SafeERC20	**Low**(1)	10-07-2025

Findings Details

`SearchStaking.sol`

M01- Missing Emergency Pause Mechanism

Severity: Medium
Impact: Inability to halt staking or migrations during a critical incident
Affected Functions: `stake()`, `unstake()`, `migrateSearchStake()`

Status: Patched

Explanation:

The contract lacks an emergency pause mechanism to temporarily disable core functions such as staking, unstaking, or stake migrations. If a vulnerability is discovered (e.g. signature replay, compromised hot wallet, authorization key leakage), there is no on-chain way to mitigate the threat or protect user funds in real-time.

This absence poses a significant operational and security risk, especially for a system that depends on off-chain authorization and may interact with external signers, queues, or user interfaces.

Suggested Solution:

Introduce a `paused` state variable controlled by the admin and enforce a `whenNotPaused` modifier on all critical external entrypoints:

Add `pause()` and `unpause()` admin functions

Add `whenNotPaused` modifier to:

- `stake(...)`
- `unstake(...)`
- `migrateSearchStake(...)`

Emit `Paused()` and `Unpaused()` events

Add `ContractPaused()` custom error for gas-efficient revert handling

M02- Signature Hash Not Fully Typed (abi.encodePacked Used)

Severity: Medium
Impact: Risk of hash collisions or signature ambiguity
Affected Functions: _verifyStakeSignature(), _verifyMigrateSignature()

Status: Patched

Description:

This is unsafe when used with encodePacked. This is because encodePacked does not take into consideration the individual lengths and just puts them together, and thus can lead to hash collisions with different inputs.

As an example, the hash of encodePacked([a,b],[c]) is identical to the hash of encodePacked([a],[b,c]). This also extends to bytes objects, which are also variable length.

While such collisions may not be exploitable in every case, they weaken signature integrity and introduce unnecessary risk in critical authorization flows. Consider using abi.encode() instead.

Recommendation:

Replace abi.encodePacked(...) with abi.encode(...) to ensure type-safety and canonical encoding:
bytes32 messageHash = keccak256(abi.encode(msg.sender, amount, deadline, nonce));

This eliminates ambiguity, aligns with best practices for ECDSA-based signing, and strengthens off-chain authorization robustness.

M03- No Domain Separation in Signatures

Severity: Medium
Impact: Signature replay across networks or contracts
Affected: _verifyStakeSignature, _verifyMigrateSignature

Status: Patched

Description:

The contract uses abi.encodePacked(...) and the legacy "\x19Ethereum Signed Message" format to validate off-chain signatures. While functional, this method lacks domain separation, meaning:

- Signatures can be reused on other contracts
- Or on other chains (e.g., Goerli → Mainnet)

This opens the door to replay attacks, where a valid signature on one network or contract could be reused in a different context.

Recommendation:

Upgrade to EIP-712 typed structured data signing, which:

- Ties the signature to the chain, contract, and version
- Prevents reuse across contexts

This change is especially important for contracts handling off-chain authorization like staking or token migration.

Mantisec Labs

L01- Unbounded batchId String Length

Severity: Low
Impact: Increased gas cost and potential for denial-of-service (DoS) scenarios

Status: Patched

Description:

The batchId string used in migrateSearchStake() is not length-restricted. An attacker could submit excessively long strings, increasing gas usage or bloating off-chain logs and storage, potentially leading to DoS-like issues in edge cases.

Recommendation:

Enforce a maximum length on batchId (e.g., 32 or 64 characters) using bytes(batchId).length. This ensures predictable gas costs and prevents abuse.

L02- Unsafe Token Transfers Without SafeERC20

Severity: Low
Impact: Silent transfer failures may go undetected
Affected: stake(), unstake(), recoverTokens(), migrateSearchStake()

Status: Patched

Description:

The contract guards token transfers with try/catch.

This only detects failures that revert. Some ERC-20 tokens, however, signal failure by returning false instead of reverting. When that happens, try/catch sees the call as "successful," and the contract can proceed with incorrect assumptions about balances.

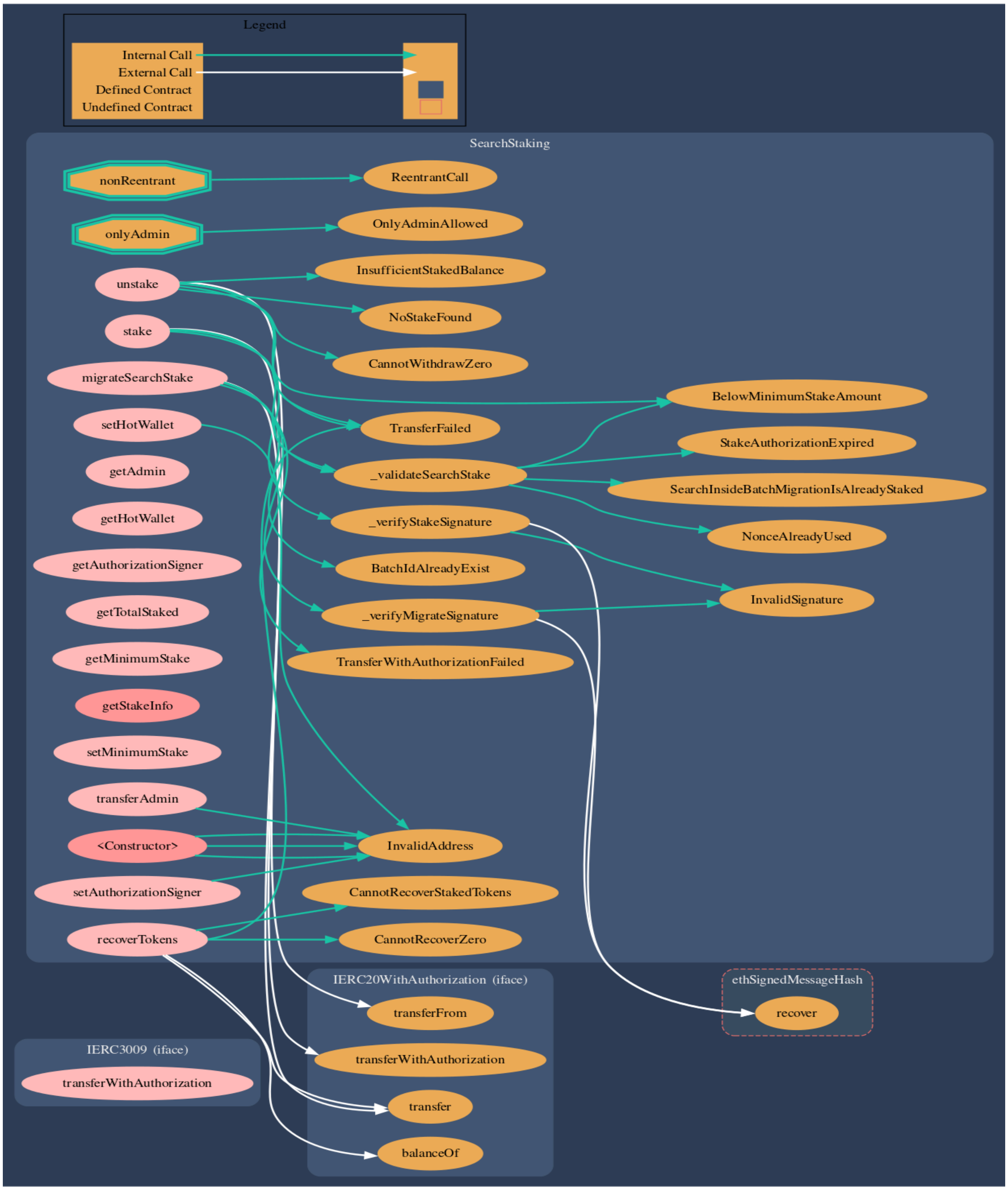
Recommendation:

Replace raw transfer/transferFrom calls with OpenZeppelin's SafeERC20 helpers:

```
import { SafeERC20 } from "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";

using SafeERC20 for IERC20WithAuthorization;

STAKING_TOKEN.safeTransfer(to, amount);

STAKING_TOKEN.safeTransferFrom(from, to, amount);
```

SafeERC20 reverts if the token call either reverts or returns false, guaranteeing reliable failure detection across all ERC-20 implementations.

Additional Details:



Concluding Remarks

To wrap it up, this audit has given us a good look at the contract's security and functionality.

Our auditors confirmed that all the issues are now resolved by the developers.