



# SECURITY ASSESSMENT ClusterYield Contract



June 27, 2025

Audit Status: Fail

CFG Ninja Verified on June 27, 2025



## ClusterYield

### Executive Summary

TYPES

DeFi

ECOSYSTEM

BNBCHAIN

LANGUAGE

Solidity

### Timeline



**Audit Request**  
2025-06-24



**Onboarding Process**  
2025-06-24

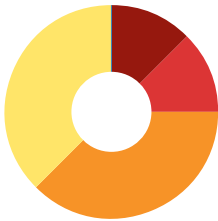


**Audit Preview**  
2025-06-24



**Audit Release**  
2025-06-24

### Vulnerability Summary



8

Total Findings

0

Resolved

8

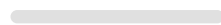
Pending

8

Unresolved

#### 1 Critical

0 Resolved, 1 Pending



Critical risks are the most severe and can have a significant impact on the smart contracts functionality, security, or the entire system. These vulnerabilities can lead to the loss of user funds, unauthorized access, or complete system compromise.

#### 1 High

0 Resolved, 1 Pending



High-risk vulnerabilities have the potential to cause significant harm to the smart contract or the system. While not as severe as critical risks, they can still result in financial losses, data breaches, or denial of service attacks.

#### 3 Medium

0 Resolved, 3 Pending



Medium-risk vulnerabilities pose a moderate level of risk to the smart contracts security and functionality. They may not have an immediate and severe impact but can still lead to potential issues if exploited. These risks should be addressed to ensure the contracts overall security.

#### 3 Low

0 Resolved, 3 Pending



Low-risk vulnerabilities have a minimal impact on the smart contracts security and functionality. They may not pose a significant threat, but it is still advisable to address them to maintain a robust security posture.

---

## 0 Informational

---

Informational risks are not actual vulnerabilities but provide useful information about potential improvements or best practices. These findings may include suggestions for code optimizations, documentation enhancements, or other non-critical areas for improvement.

---

# PROJECT OVERVIEW | ClusterYield.

## Token Summary

Parameter	Result
Address	0xa3512bd47d64fda7ad9160a259f1cf95e35d0f61
Name	ClusterYield
Token Tracker	ClusterYield (ClusterYield)
Decimals	0
Supply	0
Platform	BNBCHAIN
Compiler	v0.8.20+commit.a1b79de6
Contract Name	ClusterYield
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	<a href="https://bscscan.com/address/0xa3512bd47d64fda7ad9160a259f1cf95e35d0f61#code">https://bscscan.com/ address/0xa3512bd47d64fda7ad9160a259f1cf95e35d0f61#code</a>

## Privileged Functions (onlyOwner)






Please Note if the contract is Renounced none of this functions can be executed.

Function Name	Parameters	Visibility
addClusterYieldPlan	_lockPeriod	external
updateClusterYieldPlan	bool _isActive	external
emergencyWithdraw	_token	external
emergencyWithdraw ETH	_amount	external
transferToBot	_amount	external
receiveProfits	_amount	external
setTradingBot	_botAddress	external
addPlan	_dailyReturn	external
updatePlan	_planId	external
updatePlatformFee	_newFee	external

## TECHNICAL FINDINGS | ClusterYield.



Smart contract security audits classify risks into several categories: Critical, High, Medium, Low, and Informational. These classifications help assess the severity and potential impact of vulnerabilities found in smart contracts.

### Classification of Risk

Severity	Description
 Critical	Critical risks are the most severe and can have a significant impact on the smart contracts functionality, security, or the entire system. These vulnerabilities can lead to the loss of user funds, unauthorized access, or complete system compromise.
 High	High-risk vulnerabilities have the potential to cause significant harm to the smart contract or the system. While not as severe as critical risks, they can still result in financial losses, data breaches, or denial of service attacks.
 Medium	Medium-risk vulnerabilities pose a moderate level of risk to the smart contracts security and functionality. They may not have an immediate and severe impact but can still lead to potential issues if exploited. These risks should be addressed to ensure the contracts overall security.
 Low	Low-risk vulnerabilities have a minimal impact on the smart contracts security and functionality. They may not pose a significant threat, but it is still advisable to address them to maintain a robust security posture.
 Informational	Informational risks are not actual vulnerabilities but provide useful information about potential improvements or best practices. These findings may include suggestions for code optimizations, documentation enhancements, or other non-critical areas for improvement.

By categorizing risks into these classifications, smart contract security audits can prioritize the resolution of critical and high-risk vulnerabilities to ensure the contract's overall security and protect user funds and data.

## ClusterYield-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	 Low	ClusterYield.sol: L: 582 C: 9	 Detected

### Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes. The linked code does not create an event for the transfer.

### Recommendation


Emit an event for critical parameter changes. It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

### Mitigation

### References:

Understanding Events in Smart Contracts

## ClusterYield-19 | Centralization Privileges of ClusterYield.

Category	Severity	Location	Status
Ownership	<span style="color: orange;">●</span> Medium	ClusterYield.sol: L: 734 C: 14	 Detected

### Description

In a smart contract, the concept of "onlyOwner" functions refers to certain functions that can only be executed by the owner or creator of the contract. These functions are typically designed to perform critical actions or modify sensitive data within the contract. By restricting access to these functions, the contract owner maintains control and ensures the integrity and security of the contract.

Function Name	Parameters	Visibility
addClusterYieldPlan	_lockPeriod	external
updateClusterYieldPlan	bool _isActive	external
emergencyWithdraw	_token	external
emergencyWithdrawETH	_amount	external
transferToBot	_amount	external
receiveProfits	_amount	external
setTradingBot	_botAddress	external
addPlan	_dailyReturn	external
updatePlan	_planId	external
updatePlatformFee	_newFee	external

### Recommendation



Inheriting from Ownable and calling its constructor on yours ensures that the address deploying your contract is registered as the owner. The `onlyOwner` modifier makes a function revert if not called by the address registered as the owner. It is important that deployer or owner secure the credentials that has owner privilege to ensure the security of the project.



## Mitigation

### References:

[Guide to Ownership and Access Control in Solidity](#)

[Writing Clean Code for Solidity: Best Practices for Solidity Development](#)

## ClusterYield-20 | Potential Overflow in Referral Levels Calculation.

Category	Severity	Location	Status
Logic	 Low	ClusterYield.sol: L: 433 C: 9	 Detected

### Description

The referral levels calculation in `getUserReferralLevels` could lead to unexpected behavior if not properly checked..

### Recommendation

Add checks to prevent overflow and ensure calculations are within expected bounds..

### Mitigation

### References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

## ClusterYield-21 | Lack of Input Validation for setTradingBot.

Category	Severity	Location	Status
Security	Medium	ClusterYield.sol: L: 582 C: 9	Detected

### Description

The setTradingBot function allows setting any address as the trading bot without additional validation..

### Recommendation


Implement checks to ensure the address is a valid contract..

### Mitigation

#### References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

## ClusterYield-22 | Insufficient Balance Check Before Transfers.

Category	Severity	Location	Status
Security	<span>●</span> High	ClusterYield.sol: L: 276 C: 9	 Detected

### Description

Functions like `claimRewards` and `withdrawCapital` check balance after calculating rewards, which might lead to issues if balance changes between checks and transfers..

### Recommendation



Ensure balance checks are done immediately before transfers..

### Mitigation

#### References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

## ClusterYield-23 | Unverified Trading Bot Address.

Category	Severity	Location	Status
Security	 Critical	ClusterYield.sol: L: 612 C: 9	 Detected

### Description

The function assumes the trading bot address is legitimate without verification..

### Recommendation



Implement checks to verify the trading bot's authenticity..

### Mitigation

### References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

## ClusterYield-24 | Missing Fallback Function.

Category	Severity	Location	Status
Best Practice	 Low	ClusterYield.sol: N/A	 Detected

### Description

The contract lacks a fallback function, which could lead to accidental Ether transfers being lost..

### Recommendation



Implement a fallback function to handle unexpected Ether transfers..

### Mitigation

#### References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

## ClusterYield-25 | No Limit on Daily Returns and Lock Periods.

Category	Severity	Location	Status
Business Logic	 Medium	ClusterYield.sol: SecureAdminProxy(addClusterYieldPlan (Line 142),updateClusterYieldPlan (Line 161))	 Detected

### Description

The contract allows unlimited daily returns and lock periods, which could be misused..

### Recommendation

Implement reasonable limits for daily returns and lock periods..






### Mitigation

### References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

## FINDINGS

In this document, we present the findings and results of the smart contract security audit. The identified vulnerabilities, weaknesses, and potential risks are outlined, along with recommendations for mitigating these issues. It is crucial for the team to address these findings promptly to enhance the security and trustworthiness of the smart contract code.

Severity	Found	Pending	Resolved
 Critical	1	1	0
 High	1	1	0
 Medium	2	3	0
 Low	2	3	0
 Informational	3	0	0
Total	9	8	0

In a smart contract, a technical finding summary refers to a compilation of identified issues or vulnerabilities discovered during a security audit. These findings can range from coding errors and logical flaws to potential security risks. It is crucial for the project owner to thoroughly review each identified item and take necessary actions to resolve them. By carefully examining the technical finding summary, the project owner can gain insights into the weaknesses or potential threats present in the smart contract. They should prioritize addressing these issues promptly to mitigate any risks associated with the contract's security. Neglecting to address any identified item in the security audit can expose the smart contract to significant risks. Unresolved vulnerabilities can be exploited by malicious actors, potentially leading to financial losses, data breaches, or other detrimental consequences. To ensure the integrity and security of the smart contract, the project owner should engage in a comprehensive review process. This involves understanding the nature and severity of each identified item, consulting with experts if needed, and implementing appropriate fixes or enhancements. Regularly updating and maintaining the smart contract's codebase is also essential to address any emerging security concerns. By diligently reviewing and resolving all identified items in the technical finding summary, the project owner can significantly reduce the risks associated with the smart contract and enhance its overall security posture.



## SOCIAL MEDIA CHECKS | ClusterYield.

Social Media	URL	Result
Website	<a href="https://clusteryield.com/">https://clusteryield.com/</a>	Fail
Telegram	<a href="https://t.me/clusteryield">https://t.me/clusteryield</a>	Pass
Twitter	<a href="https://twitter.com/clusteryield">https://twitter.com/clusteryield</a>	Fail
Facebook		N/A
Reddit	N/A	N/A
Instagram	N/A	N/A
CoinGecko		Fail
Github	<a href="https://github.com/clusteryield">https://github.com/clusteryield</a>	Error 404
CMC		Fail
Email		Contact
Other		N/A

From a security assessment standpoint, inspecting a project's social media presence is essential. It enables the evaluation of the project's reputation, credibility, and trustworthiness within the community. By analyzing the content shared, engagement levels, and the response to any security-related incidents, one can assess the project's commitment to security practices and its ability to handle potential threats.

### Social Media Information Notes:

**Auditor Notes:** Website needs a bit of improvement.

**Project Owner Notes:**

## Assessment Results

## Final Audit Score ClusterYield.

Review	Score
Security Score	75
Auditor Score	75

Our security assessment or audit score system for the smart contract and project follows a comprehensive evaluation process to ensure the highest level of security. The system assigns a score based on various security parameters and benchmarks, with a passing score set at 80 out of a total attainable score of 100. The assessment process includes a thorough review of the smart contracts codebase, architecture, and design principles. It examines potential vulnerabilities, such as code bugs, logical flaws, and potential attack vectors. The evaluation also considers the adherence to best practices and industry standards for secure coding. Additionally, the system assesses the projects overall security measures, including infrastructure security, data protection, and access controls. It evaluates the implementation of encryption, authentication mechanisms, and secure communication protocols. To achieve a passing score, the smart contract and project must attain a minimum of 80 points out of the total attainable score of 100. This ensures that the system has undergone a rigorous security assessment and meets the required standards for secure operation.



**AUDIT  
FAILED**

## Important Notes for ClusterYield

- ClusterYield.sol Contract.
- Security:
  - The contract employs ReentrancyGuard to prevent reentrancy attacks, which is a strong security measure.
  - No admin recovery functions are present, reducing the risk of unauthorized fund recovery.
  - Input validation for critical functions like setTradingBot is insufficient; consider adding checks to ensure the address is a valid contract.
  - Balance checks should be performed immediately before transfers to prevent race conditions.
- Functionality:
  - The contract supports multiple investment plans with varying returns and lock periods, providing flexibility to users.
  - A 17-level referral system incentivizes user growth but should be monitored for potential abuse.
  - Integration with an AI trading bot for additional profit generation is a unique feature, but ensure the bot's address is secure and verified.
- Code Quality:
  - The code is well-documented with comments explaining the purpose of functions and structures.
  - Consider adding more robust input validation and event emissions for critical state changes.
  - The lack of a fallback function could lead to accidental Ether transfers being lost.
- Best Practices:
  - The contract generally adheres to best practices, but improvements can be made by implementing a fallback function and ensuring all critical changes emit events.

- Regular audits and updates are recommended to keep up with evolving security standards and best practices.
- Recommendations:
- Implement input validation for addresses and critical parameters.
- Ensure balance checks are conducted immediately before any fund transfers.
- Add event emissions for all critical state changes to improve transparency.
- Consider implementing a fallback function to handle unexpected Ether transfers.
- These notes provide a comprehensive overview of the contract's current state and areas for improvement. Addressing these points will enhance the contract's security, functionality, and overall quality.
- SecureAdminProxy.sol:
- General Overview:
- The contract serves as a secure admin proxy for managing the ClusterYield contract.
- It focuses on fee management, plan administration, and emergency withdrawals with strict security measures.
- Strengths:
- Security Measures: Utilizes ReentrancyGuard to prevent reentrancy attacks.
- Ownership Restrictions: Permanently disables critical functions like ownership transfer to ensure security.
- Transparency: Provides functions to list disabled and allowed admin functions.
- Potential Issues:
- Lack of Plan ID Tracking:
- Events for adding/updating plans do not track plan IDs.
- Recommendation: Implement plan ID tracking.

- Missing Access Control for Fee Update:
- The old fee is not retrieved or emitted when updating the platform fee.
- Recommendation: Retrieve and emit the old fee in events.
- Potential Reentrancy in forwardFees:
- Ensure state changes occur before external calls.
- Recommendation: Verify the order of operations in forwardFees.
- No Limit on Daily Returns and Lock Periods:
- Unlimited values could be misused.
- Recommendation: Implement reasonable limits.
- Lack of Event for Fee Updates:
- The updateClusterYieldPlatformFee function lacks event emission for old fee values.
- Recommendation: Emit events with both old and new fee values.
- Recommendations:
- Enhance Event Emission: Improve transparency by emitting comprehensive events.
- Implement Limits: Introduce limits on daily returns and lock periods for better control.
- State Management: Ensure state changes precede external calls to prevent reentrancy issues.
- Conclusion:
- The SecureAdminProxy contract is well-structured with strong security features. Addressing the noted issues will further enhance its robustness and reliability.



**AUDIT  
FAILED**

## Appendix

### Finding Categories

#### 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.

#### 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.

#### Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different requirements on the input variables than a setter function.

#### Coding Best Practices

ERC 20 Coding Standards are a set of rules that each developer should follow to ensure the code meets a set of criteria and is readable by all the developers.

## Disclaimer

The purpose of this disclaimer is to outline the responsibilities and limitations of the security assessment and smart contract audit conducted by Bladepool/CFG NINJA. By engaging our services, the project owner acknowledges and agrees to the following terms:

1. Limitation of Liability: Bladepool/CFG NINJA shall not be held liable for any damages, losses, or expenses incurred as a result of any contract malfunctions, vulnerabilities, or exploits discovered during the security assessment and smart contract audit. The project owner assumes full responsibility for any consequences arising from the use or implementation of the audited smart contract. 2. No Guarantee of Absolute Security: While Bladepool/CFG NINJA employs industry-standard practices and methodologies to identify potential security risks, it is important to note that no security assessment or smart contract audit can provide an absolute guarantee of security. The project owner acknowledges that there may still be unknown vulnerabilities or risks that are beyond the scope of our assessment. 3. Transfer of Responsibility: By engaging our services, the project owner agrees to assume full responsibility for addressing and mitigating any identified vulnerabilities or risks discovered during the security assessment and smart contract audit. It is the project owner's sole responsibility to ensure the proper implementation of necessary security measures and to address any identified issues promptly. 4. Compliance with Applicable Laws and Regulations: The project owner acknowledges and agrees to comply with all applicable laws, regulations, and industry standards related to the use and implementation of smart contracts. Bladepool/CFG NINJA shall not be held responsible for any non-compliance by the project owner. 5. Third-Party Services: The security assessment and smart contract audit conducted by Bladepool/CFG NINJA may involve the use of third-party tools, services, or technologies. While we exercise due diligence in selecting and utilizing these resources, we cannot be held liable for any issues or damages arising from the use of such third-party services. 6. Confidentiality: Bladepool/CFG NINJA maintains strict confidentiality regarding all information and data obtained during the security assessment and smart contract audit. However, we cannot guarantee the security of data transmitted over the internet or through any other means. 7. Not a Financial Advice: Bladepool/CFG NINJA please note that the information provided in the security assessment or audit should not be considered as financial advice. It is always recommended to consult with a financial professional or do thorough research before making any investment decisions.

By engaging our services, the project owner acknowledges and accepts these terms and releases Bladepool/CFG NINJA from any liability, claims, or damages arising from the security assessment and smart contract audit. It is recommended that the project owner consult legal counsel before entering into any agreement or contract.

