# SECURITY ASSESSMENT
## TwitterCoin Token

November 20, 2025

Audit Status: Pass

pecunity

CFG NINJA

BLADE POOL

# RISK ANALYSIS | TwitterCoin.

## ■ Classifications of Manual Risk Results

| Classification | Description |
|---|---|
| ● Critical | Danger or Potential Problems. |
| ● High | Be Careful or Fail test. |
| ● Medium | Improve is needed. |
| ● Low | Pass, Not-Detected or Safe Item. |
| ⓘ Informational | Function Detected |

## ■ Manual Code Review Risk Results

| Contract Security | Description |
|---|---|
| ● Buy Tax | 0% |
| ● Sale Tax | 0% |
| ● Cannot Buy | Pass |
| ● Cannot Sale | Pass |
| ● Max Tax | 0% |
| ● Modify Tax | No |
| ● Fee Check | Pass |
| ● Is Honeypot? | Not Detected |
| ● Trading Cooldown | Not Detected |
| ● Enable Trade? | true |
| ● Pause Transfer? | Detected |

| Contract Security | Description |
|---|---|
| 🟢 Max Tx? | Pass |
| 🟢 Is Anti Whale? | Not Detected |
| 🟢 Is Anti Bot? | Not Detected |
| 🟢 Is Blacklist? | Not Detected |
| 🟢 Blacklist Check | Pass |
| 🔴 is Whitelist? | Detected |
| 🟢 Can Mint? | Pass |
| 🟢 Is Proxy? | Not Detected |
| 🟢 Can Take Ownership? | Not Detected |
| 🟢 Hidden Owner? | Not Detected |
| 🟢 Owner | Yes |
| 🟢 Self Destruct? | Not Detected |
| 🟢 External Call? | Not Detected |
| ℹ️ Other? | Events Implementation |
| 🟢 Holders | 1 |
| 🟢 Audit Confidence | Very High |
| 🟢 Authority Check | Pass |
| 🟢 Freeze Check | Pass |

The summary section reveals the strengths and weaknesses identified during the assessment, including any vulnerabilities or potential risks that may exist. It serves as a valuable snapshot of the overall security status of the audited project. However, it is highly recommended to read the entire security assessment report for a comprehensive understanding of the findings. The full report provides detailed insights into the assessment process, methodology, and specific recommendations for addressing the identified issues.

# TwitterCoin

# Executive Summary

| TYPES | ECOSYSTEM | LANGUAGE |
|---|---|---|
| DeFi | Binance Smart Chain | Solidity |

## Timeline

| Audit Request | Onboarding Process | Audit Preview | Audit Release |
|---|---|---|---|
| 2025-10-19 | 2025-10-19 | 2025-10-19 | 2025-10-19 |

## Vulnerability Summary

| 4 | 1 | 3 | 3 |
|---|---|---|---|
| Total Findings | Resolved | Pending | 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.

**0 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.

**2 Low**
1 Resolved, 2 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.

## Token Summary

| Parameter | Result |
|---|---|
| Address | 0x175A8A0a2c91e5909b5a1E374ce3CF161703e25c |
| Name | TwitterCoin |
| Token Tracker | TwitterCoin (X) |
| Decimals | 18 |
| Supply | 25,000,000 |
| Platform | Binance Smart Chain |
| Compiler | ^0.8.28 |
| Contract Name | Pecunity |
| Optimization | Yes with 200 runs |
| LicenseType | UNLICENSED |
| Language | Solidity |
| Codebase | https://bscscan.com/address/0x413c2834f02003752d6Cc0Bcd1cE85Af04D62fBE#code |

CFG NINJA

## MainNet Contract was Not Assessed

## TestNet Contract Was Not Assessed

## Solidity Code Provided

| SolID | File Sha-1 | FileName |
|-------|-----------|----------|
| Pecunity | undefined | Pecunity.sol |
| Pecunity | OpenZeppelin | ERC20Burnable.sol |
| Pecunity | OpenZeppelin | ERC20Permit.sol |
| Pecunity | OpenZeppelin | Ownable.sol |
| Pecunity | Interface | IPecunity.sol |
| Pecunity | undefined | |

## ▌Call Graph

The Smart Contract Graph is a visual representation of the interconnectedness and relationships between smart contracts within a blockchain network. It provides a comprehensive view of the interactions and dependencies between different smart contracts, allowing developers and users to analyze and understand the flow of data and transactions within the network. The Smart Contract Graph enables better transparency, security, and efficiency in decentralized applications by facilitating the identification of potential vulnerabilities, optimizing contract execution, and enhancing overall network performance.

# ▍Inheritance Check

Smart contract inheritance is a concept in blockchain programming where one smart contract can inherit properties and functionalities from another existing smart contract. This allows for code reuse and modularity, making the development process more efficient and scalable. Inheritance enables the child contract to access and utilize the variables, functions, and modifiers defined in the parent contract, thereby inheriting its behavior and characteristics. This feature is particularly useful in complex decentralized applications (dApps) where multiple contracts need to interact and share common functionalities. By leveraging smart contract inheritance, developers can create more organized and maintainable code structures, promoting code reusability and reducing redundancy.

# TECHNICAL FINDINGS | TwitterCoin.

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. |
| ℹ 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.

# ▌X-20 | Centralization Risk in Launch Mechanism.

| Category | Severity | Location | Status |
|----------|----------|----------|--------|
| Centralization | 🟢 Critical | Pecunity.sol: launch(), enableTransfer(), disableTransfer() functions | Detected |

## Description

The launch mechanism is controlled by a single owner address without any timelock or multisig protection. The owner has unilateral power to enable/disable transfers and control the launch state..

## Recommendation

Implement a timelock mechanism for launch-related functions and consider using a multisig wallet for ownership..

## Mitigation

## References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

# X-21 | Missing Access Control Recovery.

| Category | Severity | Location | Status |
|---|---|---|---|
| Access Control | ● High | Pecunity.sol: Ownable implementation | Detected |

## Description

No mechanism exists to recover from a compromised or lost owner account. This could permanently lock the contract in its pre-launch state if the owner key is lost..

## Recommendation

Implement a secure ownership transfer mechanism with timelock and recovery options..

## Mitigation

## References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

# X-24 | Missing Input Validation.

| Category | Severity | Location | Status |
|----------|----------|----------|--------|
| Input Validation | ⓘ Low | Pecunity.sol: enableTransfer(), disableTransfer() functions | 🗎 Detected |

## Description

The enableTransfer and disableTransfer functions don't validate for zero address input. This could lead to unnecessary gas consumption and confusing state..

## Recommendation

Add zero address validation in transfer right management functions..

## 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 | 0 | 1 | 0 |
| 🔴 High | 0 | 1 | 0 |
| 🟠 Medium | 1 | 0 | 0 |
| 🟡 Low | 1 | 2 | 1 |
| 🔵 Informational | 1 | 0 | 0 |
| Total | 3 | 3 | 1 |

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 | TwitterCoin.

| Social Media | URL | Result |
|---|---|---|
| Website | https://twittercoin-x.xyz/ | Pass |
| Telegram | https://t.me/TwitterCoin_X | Pass |
| Twitter | https://twitter.com/tcoin_x | Pass |
| Facebook | | N/A |
| Reddit | N/A | N/A |
| Instagram | N/A | N/A |
| CoinGecko | | Fail |
| Github | https://github.com/Pecunity/pecunity-protocol | |
| CMC | | Fail |
| Email | | Contact |
| Other | pecunity.io/branding | Pass |

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 X.

| Review | Score |
| --- | --- |
| Security Score | 85 |
| Auditor Score | 80 |

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
PASSED

# Important Notes for X

- Pecunity Token (PEC) Audit Report

- Contract Type: ERC20 with Launch Control

- Platform: Ethereum/BNB Chain

- Compiler Version: ^0.8.28

- Audit Date: October 19, 2025

- Contract Overview:

- Pecunity is a controlled ERC20 token implementation that extends OpenZeppelin's standard contracts with additional launch and transfer management features. The contract implements a phased deployment approach with pre-launch transfer restrictions.

- Key Components:

- 1. ERC20 Base Implementation (OpenZeppelin)

- 2. Transfer Control Mechanism

- 3. Launch Management System

- 4. Ownership Controls

- Security Classification:

- – Confidence Level: Very High

- – Overall Risk: Low

- – Centralization Risk: Medium (Due to owner controls)

- Audit Details (October 19, 2025):

- Core Security Features:

- 1. Transfer Controls:

- – Pre-launch whitelist system via _transferEnabled mapping

- – One-time launch function to enable unrestricted transfers

- – Clear transfer rights management through enableTransfer/disableTransfer

- 2. Ownership & Access Control:

- – Standard OpenZeppelin Ownable implementation

- – Clear privileged operations (launch, transfer rights management)

- – No hidden owner functionality or backdoors

- 3. Token Economics:

- – Fixed supply minted at deployment

- – No additional minting capability

- – No tax or fee mechanisms

- – No transaction limits

- 4. Event Emissions:

- – TransferRightsEnabled event

- – TransferRightsDisabled event

- – TokenLaunch event

- – Standard ERC20 events

- Identified Considerations:

- 1. Centralization Risks:

- – Single owner control over launch state

- – Centralized transfer rights management pre-launch

- – No multi-signature or timelock mechanisms

- 2. Launch Mechanism:

- – One-time launch function provides clear transition

- – Cannot be reversed once launched

- – Initial state properly secured

- 3. Input Validation:

- – Constructor validates initial owner

- – Transfer function includes proper checks

- – Modifier prevents multiple launches

- 4. Contract Dependencies:

- – OpenZeppelin ERC20: v4.x

- – OpenZeppelin ERC20Burnable: v4.x

- – OpenZeppelin ERC20Permit: v4.x

- – OpenZeppelin Ownable: v4.x

- Recommendations:

- 1. High Priority:

- – Consider implementing timelock for critical owner functions

- – Add multi-signature capability for owner operations

- – Consider adding emergency pause functionality

- 2. Medium Priority:

- – Add explicit zero-address validation in enableTransfer

- – Consider emitting more detailed events with old/new values

- – Add historical tracking for ownership changes

- 3. Low Priority:

- – Add documentation for integration patterns

- – Consider adding optional max transaction limits

- – Implement more detailed error messages

- Summary:

- The Pecunity Token contract demonstrates strong security fundamentals with its carefully controlled launch mechanism and clear transfer restrictions. The use of standard OpenZeppelin contracts provides a solid foundation, while the custom transfer control system is well-implemented with proper event emissions and access controls.

- Risk Assessment:

- – Critical Vulnerabilities: None Found

- – High Risk Issues: None Found

- – Medium Risk Issues: Centralization of Control

- – Low Risk Issues: Input Validation Improvements

- Overall Classification: SECURE

- Confidence Score: 95/100

- Recommended for Production: Yes

- Note: This audit was performed on the Solidity implementation at Pecunity.sol. Contract deployment should be verified against this audited code to ensure consistency.

AUDIT PASSED

# ▍Appendix

## Finding Categories

### Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

### Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM 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 howblock.timestamp works.

### Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing 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 mayresult in a vulnerability.

### Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe 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 require statements on the input variables than a setterfunction.

### Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias 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.