



---

**SMART CONTRACT  
AUDIT**



**ARC MARKET**  
International change

**ArcMarket**


## Conclusion

The project team of ArcMarket (ARC) Token has applied for security auditing of the Smart Contract (0xB410274055B91162c8623537DD4343046ECA9a85)

After detailed test process and examination performed by our expert team, we declare that Smart Contract is **FAILED** the security audit

## Summary

### Audit Summary

<b>Audit Result</b>	 Failed
<b>KYC Verification</b>	NA
<b>Token Name</b>	ArcMarket Token
<b>Token Symbol</b>	ARC
<b>Contract Address</b>	0xB410274055B91162c8623537DD4343046ECA9a85
<b>Contract Link</b>	<a href="https://bscscan.com/token/0xB410274055B91162c8623537DD4343046ECA9a85">https://bscscan.com/token/0xB410274055B91162c8623537DD4343046ECA9a85</a>
<b>Testnet Address Link</b>	<a href="https://testnet.bscscan.com/address/0x6d0c0dea16c05e199ac979a6954d4f46c8128135">https://testnet.bscscan.com/address/0x6d0c0dea16c05e199ac979a6954d4f46c8128135</a>
<b>Ownership Status</b>	Ownership Renounced
<b>Current Fee</b>	No trading fee



The audit of the provided contract has uncovered critical issues, resulting in its failure. Notably, the withdrawTokens function lacks a proper unlock time check, allowing premature withdrawals, and the lockTokens function is flawed due to an ownership check limiting token locking. A severe token transfer flaw in withdrawTokens leads to tokens getting stuck, and the fee mechanism is improperly implemented, allowing fee-less unlocks. The withdrawFees function becomes ineffective post-owner-renouncement, and the lockTokens function restricts locking to the native ARC token. These cumulative issues compromise the contract's security and functionality, necessitating immediate corrective actions for a successful audit.



## Summary of Owner's Privileges

<b>Privilege Description</b>	<b>INFO</b>	<b>LOW</b>	<b>MEDIUM</b>	<b>HIGH</b>
<i>setBnbFee</i>	✓			
<i>WithdrawFees</i>	✓			
<i>renounceOwnership</i>	✓			
<i>transferOwnership</i>	✓			

\* Privileges are listed as informational since contract is renounced

## Summary of Findings

<b>Description</b>	<b>INFO</b>	<b>LOW</b>	<b>MEDIUM</b>	<b>HIGH</b>
<i>withdrawTokens Function Unlock Time Check</i>				!
<i>lockTokens Function Ownership Check</i>				!
<i>Token Transfer Flaw in withdrawTokens Function</i>				!
<i>Fee Mechanism Implementation</i>			!	
<i>withdrawFees Function and Contract Renouncement</i>			!	
<i>lockTokens Function Implementation</i>		!		





## 1- withdrawTokens Function Unlock Time Check

### Description:

The withdrawTokens function does not properly enforce the unlock time, allowing token withdrawals at any time. This exposes a critical vulnerability as users can potentially withdraw locked tokens before the designated unlock time has elapsed.

### Recommendation:

Implement a check at the beginning of the withdrawTokens function to ensure that the unlock time has passed before allowing token withdrawals. This enhancement will prevent unauthorized early withdrawals.

### Code Suggestion:

```
function withdrawTokens(uint256 _id) external payable {
    require(block.timestamp >= lockedToken[_id].unlockTime, "Tokens are still
locked");

    // ... (rest of the function remains unchanged)
}
```





## 2- lockTokens Function Ownership Check

### Description:

The lockTokens function allows any address to lock tokens, regardless of ownership. This poses a serious security risk, as it enables unauthorized users to lock tokens that do not belong to them.

### Recommendation:

Add an ownership check within the lockTokens function to ensure that only the rightful owner of the tokens can initiate the locking process. This will mitigate the risk of unauthorized token locking.

### Code Suggestion:

```
function lockTokens(
    address _tokenAddress,
    uint256 _amount,
    uint256 _unlockTime
) external returns (uint256 _id) {
    require(msg.sender == owner(), "Only the token owner can lock tokens");

    // ... (rest of the function remains unchanged)
}
```





### 3- Token Transfer Flaw in withdrawTokens Function

#### Description:

The withdrawTokens function suffers from a critical flaw in the token transfer logic. Instead of transferring the locked tokens to the intended recipient, the function erroneously transfers an equivalent amount of tokens from the address that initiated the withdrawal. This results in the locked tokens becoming stuck within the contract, while the balance of the address initiating the withdrawal is decreased by the same amount.

The identified flaw in the withdrawTokens function, where the transfer logic results in tokens being stuck within the contract and the incorrect deduction from the initiator's address, poses a severe risk to the contract's functionality. Immediate corrective measures are essential to rectify this issue and prevent potential token loss. This critical vulnerability significantly impacts the success of the audit, necessitating urgent attention and corrective actions to fortify the contract's token transfer mechanism.

#### Recommendation:

Revise the withdrawTokens function to ensure that it correctly transfers the locked tokens from the contract to the intended recipient's address, as intended. Verify that the token transfer logic is accurate and update the relevant code to rectify this critical issue. Thoroughly test the modified function to guarantee the proper flow of token transfers, preventing the loss of tokens and ensuring the integrity of the withdrawal process.





#### 4- Fee Mechanism Implementation

**Description:**

The fee mechanism in the contract is not correctly implemented, allowing users to unlock tokens without paying the specified fee. The contract lacks the necessary logic to enforce and collect fees when users attempt to unlock tokens.

**Recommendation:**

Implement a robust fee mechanism in the `withdrawTokens` function to ensure that users cannot unlock tokens without paying the required fee. This may involve modifying the fee calculation and collection logic within the function.





## 5- withdrawFees Function and Contract Renouncement

### Description:

The withdrawFees function becomes ineffective when the contract is renounced. If the contract owner renounces ownership, the withdrawFees function loses its purpose, as there is no designated owner to initiate fee withdrawals.

### Recommendation:

Consider updating the contract to handle fee withdrawals in a more decentralized manner, even if ownership is renounced. This could involve a multisig or governance mechanism for fee withdrawal to ensure continued functionality in a decentralized context.







## 6- lockTokens Function Implementation

### Description:

The lockTokens function is not implemented correctly. In the `_tokenAddress` parameter, the function incorrectly accepts a wallet address that holds the ARC native token. This limitation restricts the locking of tokens to only the native ARC token.

### Recommendation:

Revise the lockTokens function to allow the locking of any ERC-20 token, not just the native ARC token. Modify the ownership check and authorization mechanism to permit the locking of tokens from different addresses.



## Table of Contents

Conclusion .....	1
Summary .....	1
Audit Summary.....	1
Summary of Owner's Privileges.....	2
Summary of Findings.....	2
1- withdrawTokens Function Unlock Time Check .....	3
2- lockTokens Function Ownership Check.....	4
3- Token Transfer Flaw in withdrawTokens Function .....	5
4- Fee Mechanism Implementation .....	6
5- withdrawFees Function and Contract Renouncement .....	7
6- lockTokens Function Implementation .....	8
Report Data .....	10
Project Info .....	10
OVERVIEW .....	11
Auditing Approach and Applied Methodologies .....	11
Security.....	11
Sound Architecture.....	11
Code Correctness and Quality.....	11
Risk Classification .....	12
Disclaimer .....	13



## Report Data

This report has been prepared by Cryptocrat experts based on detailed examination of ArcMarket Token Smart Contract on November 19, 2023.

Audit process performed carefully using Static Analysis and Manual review Techniques as well as Automated test procedures.

The auditing process focuses to the following considerations with collaboration of an expert team

- Functionality test of the Smart Contract to determine if proper logic has been followed throughout the whole process.
- Manually detailed examination of the code line by line by experts.
- Live test by multiple clients using Testnet.
- Analysing failure preparations to check how the Smart Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analysing the security of the on-chain data.

## Project Info

<b>Contract Name</b>	<b>ArcMarket (ARC)</b>
<b>Contract</b>	0xB410274055B91162c8623537DD4343046ECA9a85
<b>Link to Contract</b>	<a href="https://bscscan.com/token/0xB410274055B91162c8623537DD4343046ECA9a85">https://bscscan.com/token/0xB410274055B91162c8623537DD4343046ECA9a85</a>
<b>Token Type</b>	BEP20
<b>Platform</b>	Binance Smart Chain
<b>Language</b>	Solidity
<b>Project Web Site</b>	<a href="https://arcmarket.capital/">https://arcmarket.capital/</a>
<b>Twitter</b>	TBA
<b>Telegram Group</b>	TBA



## OVERVIEW

This Audit Report mainly focuses on overall security of the smart contract. Cryptocrat team scanned the contract and assessed overall system architecture and the smart contract codebase against vulnerabilities, exploitations, hacks, and back-doors to ensure its reliability and correctness.

### Auditing Approach and Applied Methodologies

Cryptocrat team has performed rigorous test procedures of the project

- Code design patterns analysis in which smart contract architecture is reviewed to ensure it is structured according to industry standards and safe use of third-party smart contracts and libraries.
- Line-by-line inspection of the Smart Contract to find any potential vulnerability like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.
- Unit testing Phase, we coded/conducted custom unit tests written for each function in the contract to verify that each function works as expected.
- Automated Test performed with our in-house developed tools to identify vulnerabilities and security flaws of the Smart Contract.

The focus of the audit was to verify that the Smart Contract System is secure, resilient, and working according to the specifications. The audit activities can be grouped in the following three categories:

### Security

Identifying security related issues within each contract and the system of contract.

### Sound Architecture

Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

### Code Correctness and Quality

A full review of the contract source code. The primary areas of focus include:

- Accuracy
- Readability
- Sections of code with high complexity
- Quantity and quality of test coverage



## Risk Classification

SEVERITY	EXPLANATION
INFORMATIONAL	Informational risks are classified as low-impact issues that do not pose an immediate threat to the security or functionality of the smart contract. These risks typically include suggestions for code optimization, improvements in documentation, or best practices that can enhance the overall quality and maintainability of the contract. While not critical, addressing these informational risks is recommended to further strengthen the contract's security posture.
LOW	Low-risk vulnerabilities are minor issues that may have limited impact on the contract's security. These risks are typically related to non-critical code flaws or deviations from best practices that could potentially be exploited under certain circumstances. While the impact is relatively low, it is still advisable to address these vulnerabilities to reduce any potential security risks and ensure the contract operates as intended.
MEDIUM	Medium-risk vulnerabilities pose a moderate level of risk to the security and functionality of the smart contract. These risks may include code vulnerabilities that could potentially be exploited, but with certain constraints or prerequisites. Addressing medium-risk vulnerabilities is crucial to prevent potential security breaches or unintended behaviour that could impact the contract or its users.
HIGH	High-risk vulnerabilities are critical issues that pose significant threats to the security and functionality of the smart contract. These risks typically involve severe code vulnerabilities that can be exploited to manipulate or compromise the contract's behavior, resulting in financial loss or unauthorized access. Immediate attention and remediation of high-risk vulnerabilities are necessary to ensure the contract's integrity and protect the funds and assets associated with it.

It is important to note that risk classification may vary based on the specific audit methodology or framework used, and the assigned risk level should be interpreted in the context of the smart contract being audited.





## Disclaimer

This document has been prepared by Cryptocrat solely for the use of the investors to whom it is addressed and for no other purpose. The information contained in this report is based on an analysis of the smart contract code itself. This report is not a prospectus or offering document, and it does not constitute an offer to sell or a solicitation of an offer to buy any securities or other financial instruments. The report should not be considered as investment, legal, tax, or other advice.

Cryptocrat makes no representation or warranty, express or implied, regarding the accuracy or completeness of the information contained in this report. Cryptocrat shall not be liable for any loss arising from the use of or reliance on this report. It is important to note that the forward-looking statements made in this report are subject to various risks and uncertainties that could cause actual results to differ materially from those expressed or implied. Cryptocrat does not undertake any obligation to update or revise any forward-looking statements, whether as a result of new information, future events, or otherwise.

Investors are advised to conduct their own thorough analysis and seek independent professional advice before making any investment decisions. The information provided in this report should be considered in the context of the specific smart contract and its associated risks.

