PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Preliminary Report

For Aquarius Loan (Token only)

25 August 2022

paladinsec.co

info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Paladin is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Paladin may, at its discretion, claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Aquarius Loan's Ars token on the BitTorrent Chain. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	Aquarius Loan
URL	https://www.aquarius.loan/
Network	BitTorrent Chain
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`Ars`	`0xACaDD8eaC98F66a959E0DfaB66E3a548A6E57ce6`	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	0	-	-	-
Medium	0	-	-	-
Low	2	-	-	2
Informational	3	-	-	3
Total	**5**	**-**	**-**	**5**

Classification of Issues

Severity	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 Ars

ID	Severity	Summary	Status
01	LOW	Contract is susceptible to allowance front-running attacks	ACKNOWLEDGED
02	LOW	approve function has no spender address requirement	ACKNOWLEDGED
03	INFO	`delegate` and `delegateBySig` have a `return` statement that does not support any return values	ACKNOWLEDGED
04	INFO	Use `uint256` instead of `uint` for better interpretation of contract variables	ACKNOWLEDGED
05	INFO	`totalSupply` amount can be revised to increase readability further	ACKNOWLEDGED

2 Findings

2.1 Ars

`Ars` is an ERC20 token implementation that will be used as the protocol's governance token. Its expected purpose is to represent how much control an individual has over the protocol. Generally, one token is equal to one vote. Ars token holders will also be able to generate and vote on proposals via the GovernorAlpha contract.

The contract has the standard ERC20 functions to make it a transferable instrument with a few additions of governance functions for delegates. Upon deployment, its initial supply is preminted via the constructor at an address the deployer will input.

Additionally, the token contract does not use any privileged function or libraries. It is a straightforward token contract with no transfer restrictions or fees.

2.1.1 Token Overview

Address	`0xACaDD8eaC98F66a959E0DfaB66E3a548A6E57ce6`
Token Supply	1,000,000,000 (one billion)
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	None

2.1.2 Issues & Recommendations

Issue #01	Contract is susceptible to allowance front-running attacks
Severity	LOW SEVERITY
Description	The contract follows the old ERC-20 interface which allows users to use the `approve` and `transferFrom` functions to grant allowance and spend tokens on behalf of the owner. In the old ERC-20 interface, the owner can only revoke the allowance granted by submitting another `approve` transaction. This allows the spender to exploit the previous allowance by using the `transferFrom` function and submit it with a higher gas fee which allows the transaction to front-run other transactions. The spender would then be able to transfer the previously allowed amount if the frontrunning transaction is successful, and submit another transaction with the newly approved amount afterward if it is greater than zero.
Recommendation	Consider applying the latest ERC-20 standard on allowances from OpenZeppelin to mitigate such front-running risks.
Resolution	ACKNOWLEDGED

Issue #02	`approve` function has no spender address requirement
Severity	LOW SEVERITY
Description	The `approve` function allows the spender address to be zero. Approving the zero address will have no effect as this address has no specific owner.
Recommendation	Consider adding a requirement for the spender address to not equal zero.
Resolution	ACKNOWLEDGED

Issue #03	**`delegate` and `delegateBySig` have a `return` statement that does not support any return values**
Severity	INFORMATIONAL

Location	Lines 148-149
	`function delegate(address delegatee) public {`
	`    return _delegate(msg.sender, delegatee);`
	
	Lines 161, 169
	`function delegateBySig(address delegatee, uint nonce, uint expiry, uint8 v, bytes32 r, bytes32 s) public {`
	`    ...`
	`    return _delegate(signatory, delegatee);`

Description	The above statements intend to return a value upon execution, however, the functions that these statements are located at does not explicitly support value returns nor does the internal function being called is returning any value.
Recommendation	Consider removing the `return` variable and proceed instead with internally calling the `_delegate` function.
Resolution	ACKNOWLEDGED

Issue #04	**Use `uint256` instead of `uint` for better interpretation of contract variables**
Severity	INFORMATIONAL
Description	It is recommended to remain consistent throughout the project and only use `uint256`. Being consistent shows third-party validators that the code has been carefully thought through.
Recommendation	Consider using `uint256` throughout the contract.
Resolution	ACKNOWLEDGED

Issue #05	**`totalSupply` amount can be revised to increase readability further**
Severity	INFORMATIONAL
Location	Line 15 `uint public constant totalSupply = 10000000e18; // 10 million Ars`
Description	Number values can use an underscore (`_`) as a thousand separator to increase readability. Example: `1_000 ether`, `10_000 ether`, `100_000 ether`.
Recommendation	Consider revising the values to the method as mentioned above.
Resolution	ACKNOWLEDGED


PALADIN
BLOCKCHAIN SECURITY