PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For ADAFlect

03 September 2021

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.

1 Overview

This report has been prepared for the ADAFlect token on the Binance Smart Chain (BSC). 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	ADAFlect
URL	https://adaflect.com/
Platform	Binance Smart Chain (BSC)
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`Adaflect.sol`	`0xd6ca9451bba47e26706a701ae05be45a712d4b1b`	MATCH
`ADAFLECTDividendTracker`	`0x3d44a95E83b3399c5c39EbE4ea6a9B611Dd0af53` *unverified contract, but it is deployed by the verified contract above so the code must match what we have audited.	MATCH
`DividendPayingToken.sol`	Dependency of the ADAFLECTDividendTracker contract	MATCH
`IterableMapping.sol`	Dependency of the ADAFLECTDividendTracker contract	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	3	3	-	-
Medium	2	2	-	
Low	5	4	-	1
Informational	12	6	1	5
Total	**22**	**15**	**1**	**6**

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 with 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 ADAFlect

ID	Severity	Summary	Status
01	HIGH	Gov Privilege: updateUniswapV2Router could be used to revert transactions, siphon fees or turn the token into a honeypot	RESOLVED
02	HIGH	Gov Privilege: Fees are freely adjustable up to over 100%	RESOLVED
03	HIGH	Gov Privilege: Owner can blacklist wallets until `disableBlacklist` is called	RESOLVED
04	MEDIUM	Gov Privilege: Setting the marketing wallet to the zero address turns the token into a honeypot	RESOLVED
05	MEDIUM	Gov Privilege: Owner can update the dividend tracker to siphon all dividends and potentially block sell transactions	RESOLVED
06	LOW	Accounts can no longer be whitelisted after disableBlacklist is called	RESOLVED
07	LOW	`addLiquidity` is called even if the `eth` amount is zero	RESOLVED
08	LOW	Token contract could build up dust BNB due to `addLiquidity`	ACKNOWLEDGED
09	INFO	Generated liquidity is transferred	ACKNOWLEDGED
10	INFO	`deadWallet`, `ADA` and `swapTokensAtReward` can be made `constant`	RESOLVED
11	INFO	`isExcludedFromFees` and `dividendTokenBalanceOf` can be made `external`	PARTIAL
12	INFO	`ProcessedDividendTracker` event should be moved to the `dividendTracker` contract	ACKNOWLEDGED

1.3.2 ADAFLECTDividendTracker

ID	Severity	Summary	Status
13	INFO	Wrongful usage of `require` instead of `revert`	ACKNOWLEDGED
14	INFO	Token symbol exceeds 11 characters which makes adding it to MetaMask more cumbersome	RESOLVED
15	INFO	`process` and `getAccountAtIndex` can be made `external`	RESOLVED
16	INFO	`processesUntilEndOfArray` is a misnomer	RESOLVED
17	INFO	Lack of event for `process` function	ACKNOWLEDGED
18	INFO	Owner can give themself a dividend receiving position before ownership is transferred	RESOLVED

1.3.3 DividendPayingToken

ID	Severity	Summary	Status
19	LOW	The success check in distributeADADividends is insufficient since ADA transfers will revert instead of returning false which could cause transfers to be blocked if there is ever insufficient ADA in the contract	RESOLVED
20	LOW	`distributeADADividends` does not verify that enough ADA has been deposited into the contract, which could block transfers and withdrawals	RESOLVED
21	INFO	Owner can give themselves a dividend-receiving position before ownership is transferred	RESOLVED
22	INFO	Wrongful usage of `require` instead of `revert`	ACKNOWLEDGED

1.3.4 IterableMapping

No issues found.

2 Findings

2.1 ADAFlect

The Adaflect token is a dividend distributing token which imposes a tax of 15% on every transfer that is used for the following purposes:

- Selling for ADA to distribute to holders (initially 8% tax)
- Selling for liquidity generations (initially 2% tax)
- Selling for marketing budget/project ownership (initially 5%)

The transfer tax and distribution can be freely changed by the owner until `disableFeeChanging` is called. Furthermore, the owner can blacklist wallets until `disableBlacklist` is called.

The PancakeSwap V2 pair is excluded from receiving dividends and collected fees are sold off for distribution on every 2 million tokens collected. The owner can exclude addresses from paying the transfer tax. Initially, the owner and the marketing wallets are excluded from said fees.

2.1.1 Token Overview

Address	0xd6ca9451bba47e26706a701ae05be45a712d4b1b
Token Supply	100,000,000,000 (One hundred billion) (no minting)
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	15%

2.1.2 Privileged Roles

The following functions can be called by the owner of the contract:

- `updateDividendTracker`
- `updateUniswapV2Router`
- `excludeFromFees`
- `excludeMultipleAccountsFromFees`
- `setMarketingWallet`
- `setADARewardsFee`
- `setLiquidityFee`
- `setMarketingFee`
- `setAutomatedMarketMakerPair`
- `blacklistAddress`
- `disableBlacklist`
- `disableFeeChanging`
- `disableWalletChanging`
- `updateGasForProcessing`
- `updateClaimWait`
- `excludeFromDividends`

2.1.3 Issues & Recommendations

Issue #01	**Gov Privilege: `updateUniswapV2Router` could be used to revert transactions, siphon fees or turn the token into a honeypot**
Severity	HIGH SEVERITY
Description	The owner can update the router that generates liquidity to an address or contract of choice. This contract could be a malicious contract that simply keeps the tokens sent to it and thus siphons all deposit fees. Furthermore, this contract could be used to revert sell transactions turning the token into a honeypot.
Recommendation	Consider removing this function. If this is not possible, consider using an Operator account that is behind a significantly longer timelock so investors can reasonably see this change coming and inspect the new router.
Resolution	RESOLVED The function has been removed.

Issue #02	Gov Privilege: Fees are freely adjustable up to over 100%
Severity	HIGH SEVERITY
Description	The owner of the contract can set the individual fees to any variable. This might deter investors as they could be wary that these fees might one day be set to 100% to force transfers to go to the contract owner.

Recommendation	Consider adding an explicit cap to the total fee on every fee adjustment function. The example below requires the total fee to be less than 20%.
	
	`totalFees = ADARewardsFee.add(liquidityFee).add(marketingFee);`
	`require(totalFees <= 20, "too high");`
	
	This issue will also be marked as resolved once `disableFeeChanging` is called.

Resolution	RESOLVED The fee for ADA reward generation now has a cap of 10%, the fee for liquidity generation now has a cap of 4% and the fee for the owner/marketing now has a cap of 5%.

Issue #03	**Gov Privilege: Owner can blacklist wallets until `disableBlacklist` is called**
Severity	HIGH SEVERITY
Description	Currently, the owner can blacklist wallets from purchasing or selling the coins they hold. This might deter investors from purchasing the coin. Note that once `disableBlacklist` is called, accounts can no longer be blacklisted.
Recommendation	Consider either removing the blacklist functionality, clearly indicating why it is present in both the contract or documentation, or putting the Adaflect token behind a significantly long timelock. This issue will also be marked as resolved once `disableBlacklist` is called.
Resolution	RESOLVED The owner has moved the `mint` function to a one time callable `prepForLaunch` function. After it is called, it won't be able to be called again. Once it is called, the owner has only a 1 hour period to ban addresses and the team has communicated that it will be exclusively used to nuke bots. They have included a disclosure of this in their contract and plan to include it in their documentation as well so as to not mislead users with automated tooling. After the 1 hour period has passed, nobody can be banned and only unbanning is still possible.

Issue #04	Gov Privilege: Setting the marketing wallet to the zero address turns the token into a honeypot
Severity	MEDIUM SEVERITY
Description	The owner can set the marketing address to zero, which has the side-effect that a part of the fees will be sent to the zero address. However, these transfers will revert since ADA does not allow transfers to the zero address. Thus, if the governance ever sets the marketing wallet to the zero address, they would effectively block all sell transactions, turning the token into a honeypot since `swapAndSendToFee` is not called on purchases.
Recommendation	Consider requiring the marketing wallet to be non-zero. `require(wallet != address(0), "zero!");`
Resolution	RESOLVED The recommended check has been added, along with a check to ensure the wallet is not the dead address.

Issue #05	Gov Privilege: Owner can update the dividend tracker to siphon all dividends and potentially block sell transactions
Severity	MEDIUM SEVERITY
Description	Currently the owner of the Adaflect token can freely upgrade to a new underlying dividend tracker. If this is done to a malicious tracker, it could block sell transactions (through `swapAndSendDividends`) and siphon the ADA dividends to the owner instead of distributing them. This privilege could have a negative impact on investor confidence.
Recommendation	Consider removing the `updateDividendTracker` function if there is no use of upgradeability. Otherwise, consider putting the contract behind a significantly long timelock.
Resolution	RESOLVED The `updateDividendTracker` function has been removed.

Issue #06	**Accounts can no longer be whitelisted after `disableBlacklist` is called**
Severity	LOW SEVERITY
Description	After the `disableBlacklist` function is called, the blacklist function can no longer execute. This however also means that no previously blacklisted accounts can be whitelisted anymore.
Recommendation	Consider only reverting the blacklist function if an actual blacklist call has been made, and to still allow whitelisting wallets again after blacklisting has been disabled.
Resolution	RESOLVED Although this behavior is still present, it is no longer relevant as blacklisting is is disabled through a secondary mechanism after an hour, we see no use in disabling the blacklisting manually anymore. Furthermore, the scondary automated mechanism still allows whitelisting after the 1 hour period has passed.

Issue #07	**`addLiquidity` is called even if the `eth` amount is zero**
Severity	LOW SEVERITY
Description	In theory, `addLiquidity` could be called even without any `eth`. As PancakeSwap reverts transactions with zero amounts, this would thus revert all transactions. This is unlikely as the only case this would happen is if there is not yet any liquidity in the LP pair.
Recommendation	Consider adding an if-statement in the `addLiquidity` function to only call PancakeSwap if the `eth` amount is non-zero.
Resolution	RESOLVED

Issue #08	Token contract could build up dust BNB due to `addLiquidity`
Severity	LOW SEVERITY
Description	Due to the liquidity generating mechanism, some dust BNB which is not withdrawable may accumulate in the contract over time.
Recommendation	Consider adding an `onlyOwner convertBNBAndDistribute` function to use this dust to buy ADA for distribution. This aligns well with the project tokenomics.
Resolution	ACKNOWLEDGED The client has indicated that this amount will be so minimal that it's not worth the extra code which could make third-party reviewing more difficult.

Issue #09	Generated liquidity is transferred
Severity	INFORMATIONAL
Description	Currently the generated liquidity is sent to the zero address instead of to the dead wallet. This is not an issue but if the code is ever forked to use a different AMM than pancakeswap, this could cause reversion as many tokens do not allow transfers to the zero address.
Recommendation	Consider keeping this in mind when moving to new exchanges.
Resolution	ACKNOWLEDGED

Issue #10	`deadWallet`, `ADA` and `swapTokensAtReward` can be made `constant`
Severity	INFORMATIONAL
Description	Variables that are never changed throughout the contract can be marked as such using the `constant` keyword. This allows for the compiler to optimize them and reduce the gas cost involved with using these variables.
Recommendation	Consider marking the aforementioned variables as `constant`.
Resolution	RESOLVED

Issue #11	`isExcludedFromFees` and `dividendTokenBalanceOf` can be made `external`
Severity	INFORMATIONAL
Description	The above functions functions can be changed from `public` to `external`. Apart from being a best practice when the function is not used within the contract, this can lead to a lower gas usage in certain cases.
Recommendation	Consider making these functions `external`.
Resolution	PARTIALLY RESOLVED `dividendTokenBalanceOf` has been made `external`.

Issue #12	**`ProcessedDividendTracker` event should be moved to the `dividendTracker` contract**
Severity	INFORMATIONAL
Description	The process function on the `dividendTracker` can be called by anyone – this means that the `ProcessedDividendTracker` event on the Adaflect contract might not be always emitted when progress is made since people can call the underlying contract directly.
Recommendation	Consider emitting the event on the `dividendTracker` contract instead of on the Adaflect contract. Otherwise, `process` could be made `onlyOwner` to ensure it can only be called by Adaflect, which might be even better since by doing so, the event can be centralised in the Adaflect contract.
Resolution	ACKNOWLEDGED

2.2 ADAFLECTDividendTracker

The ADAFLECTDividendTracker contract extends the DividendPayingToken contract to automate the distribution of ADA to token holders. A minimum of 200,000 tokens is necessary for users to be eligible for a dividend. Users are only eligible for automatic distribution every hour (can be adjusted up to 24 hours); however, they can still claim their ADA distributions manually using the claim function.

2.2.1 Privileged Roles

The Adaflect contract should eventually be the owner of the ADAFLECTDividendTracker contract. The following functions can be called by the owner of the contract:

- `setBalance`
- `processAccount`

2.2.2 Issues & Recommendations

Issue #13	Wrongful usage of `require` instead of `revert`
Severity	INFORMATIONAL

Location	Line 550
	`require(false, "ADAFlect_Dividend_Tracker: No transfers allowed");`
	
	Line 554
	`require(false, "ADAFlect_Dividend_Tracker: withdrawDividend disabled. Use the 'claim' function on the main Adaflect contract.");`

Description	To make sure that people cannot manually transfer the token, a `require(false)` statement is added to the transfer override function. This will always revert said function. However, for this behavior, Solidity recommends using the `revert();` keyword instead, since `require` is meant to actually require things to be true.
Recommendation	Consider using `revert("reason");` instead.
Resolution	ACKNOWLEDGED

Issue #14	**Token symbol exceeds 11 characters which makes adding it to MetaMask more cumbersome**
Severity	INFORMATIONAL
Description	Although the ERC-20 metadata standard does not specify a maximum length for a token symbol, MetaMask does not allow the length to exceed 11 characters. Adding any token to MetaMask with a symbol that is over 11 characters will require the user to manually adjust the symbol, which could be considered bad user experience.
Recommendation	Consider whether it is possible to remove a single letter from the symbol string to make it compliant with MetaMask without user intervention.
Resolution	RESOLVED The token is now called ADAFlectDvt

Issue #15	**`process` and `getAccountAtIndex` can be made external**
Severity	INFORMATIONAL
Description	The above functions can be changed from `public` to `external`. Apart from being a best practice when the function is not used within the contract, this can lead to a lower gas usage in certain cases.
Recommendation	Consider making these functions `external`.
Resolution	RESOLVED

Issue #16	`processesUntilEndOfArray` is a misnomer
Severity	INFORMATIONAL
Location	Lines 605-607 `uint256 processesUntilEndOfArray = tokenHoldersMap.keys.length > lastProcessedIndex ? tokenHoldersMap.keys.length.sub(lastProcessedIndex) : 0;`
Description	The variable `processesUntilEndOfArray` is erraneously named since it actually keeps track of the processes until the beginning of the array. Take for example an array of 3 holders and we are already at the last index, index 2. The variable will in this case indicate that there is still 1 process to go.
Recommendation	Since the business logic actually matches the purpose of this variable, the variable should simply be renamed to `processesUntilbeginOfArray`.
Resolution	RESOLVED

Issue #17	Lack of event for `process` function
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications. In this case we believe it might be valuable to emit an event for a whole batch process call that indicates from and to which index the processing occurred.
Recommendation	Add an event for the above function.
Resolution	ACKNOWLEDGED

Issue #18	**Owner can give themself a dividend receiving position before ownership is transferred**
Severity	INFORMATIONAL
Description	The owner of the DividendPayingToken can manually give shares to accounts to receive a share of the dividends. Usually this is the Adaflect contract, however, Adaflect contains a function `updateDividendTracker` which allows the token to be moved to a new token which could then have premined balances. Premined balances would result in the owner taking a part of the ADA dividends as long as they have the balance.
Recommendation	Consider removing the `updateDividendTracker` function or putting it behind an extremely long timelock for this issue to be marked as resolved.
Resolution	RESOLVED The Adaflect token can no longer set a new dividend tracker and the initial one is generated in the constructor so cannot have balances.

2.3 DividendPayingToken

The DividendPayingToken is a token contract that allows for the distribution of ADA dividends to the token holders. Dividends need to be sent to it by the contract owner which should be the Adaflect contract.

2.3.1 Privileged Roles

The Adaflect contract should eventually be the owner of the ADAFLECTDividendTracker contract. The following functions can be called by the owner of the contract:

- `distributeADADividends`

2.3.2 Issues & Recommendations

Issue #19	**The success check in `distributeADADividends` is insufficient since ADA transfers will revert instead of returning false which could cause transfers to be blocked if there is ever insufficient ADA in the contract**
Severity	LOW SEVERITY
Location	Line 81 `bool success = IERC20(ADA).transfer(user, _withdrawableDividend);`
Description	When `distributeADADividends` is called by the owner of the contract to distribute the dividends, this does not explicitly ensure that enough ADA was actually deposited into the contract. This issue has been lowered to low severity since the Adaflect contract should own the tracker, forcing this logic to always be correct.
Recommendation	Consider adding try-catch logic to the transfer call to also handle the failure case when the transfer does not succeed and reverts. This way, token transfers are not blocked due to there being insufficient ADA in the contract.
Resolution	RESOLVED A try-catch has been used – it should be noted that the `bool` result has not been handled but in this case it doesn't matter since ADA always reverts on failure.

Issue #20	**`distributeADADividends` does not verify that enough ADA has been deposited into the contract, which could block transfers and withdrawals**
Severity	LOW SEVERITY
Location	Line 55 `function distributeADADividends(uint256 amount) public onlyOwner{`
Description	When `distributeADADividends` is called by the owner of the contract to distribute the dividends, this does not explicitly ensure that enough ADA was actually deposited into the contract.
Recommendation	Consider either verifying that enough ADA was added to the contract by, for example, pulling it in, or consider the recommendations from the previous issue to ensure that transfers do not fail if the ADA transfer is unsuccesful.
Resolution	RESOLVED Since the Adaflect token now must own the DividendPayingToken, this issue is no longer relevant. It might still cause issues if this code is ever forked to distribute a transfer-tax token.

Issue #21	**Owner can give themselves a dividend-receiving position before ownership is transferred**
Severity	INFORMATIONAL
Description	The owner of the DividendPayingToken can manually give shares to accounts to receive a share of the dividends. Usually, this is the Adaflect contract, however, Adaflect contains a function `updateDividendTracker` which allows the token to be moved to a new token which could then have premined balances. Premined balances would result in the owner taking a part of the ADA dividends as long as they have the balance.
Recommendation	Consider removing the `updateDividendTracker` function or putting it behind an extremely long timeline, this would mark this issue as resolved.
Resolution	RESOLVED The Adaflect token can no longer set a new dividend tracker and the initial one is generated in the constructor and so cannot have balances.

Issue #22	Wrongful usage of `require` instead of `revert`
Severity	INFORMATIONAL
Location	Line 133 `require(false);`
Description	To make sure that people cannot manually transfer the token, a `require(false)` statement is added to the transfer override function. This will always revert said function. However, for this behavior, Solidity recommends using the `revert();` keyword instead, since `require` is meant to actually require things to be true.
Recommendation	Consider using `revert();` instead.
Resolution	ACKNOWLEDGED

2.4 IterableMapping

The IterableMapping library is a dependency which allows to create Solidity key-value mappings which are iterable as well.

2.4.1 Issues & Recommendations

No issues found.


PALADIN
BLOCKCHAIN SECURITY