PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Dogira

24 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 Dogira on the Polygon and Ethereum networks. 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	Dogira
URL	https://dogira.net
Platform	Polygon, Ethereum
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
Dogira (Polygon)	0xdDa40cdfe4A0090f42Ff49f264A831402ADB801A	MATCH
Dogira (Ethereum)	0xD8C1232FcD219286E341271385bd70601503B3D7	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	3	3	-	-
Medium	3	3	-	-
Low	0	-	-	-
Informational	8	2	-	6
Total	**14**	**8**	**-**	**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 Dogira

ID	Severity	Summary	Status
01	HIGH	Owner can blacklist any address	RESOLVED
02	HIGH	`setUniswapRouter` could be used to turn the token into a honeypot	RESOLVED
03	HIGH	LP pair can be changed	RESOLVED
04	MEDIUM	`isIncludedInFees`, `isInBlacklist` and `removeFromBlacklist` may run out of gas	RESOLVED
05	MEDIUM	`swapTokens` function may fail and cause token to be a honeypot	RESOLVED
06	MEDIUM	Setting tax fee to zero is not possible if `taxFee` was previously set to a positive number	RESOLVED
07	INFO	Contract uses raw addition	ACKNOWLEDGED
08	INFO	`transferERC20` does not use SafeERC20	ACKNOWLEDGED
09	INFO	Minor correction to `setConvertMinimum`	ACKNOWLEDGED
10	INFO	Before-after method in `swapTokens` is unnecessary	ACKNOWLEDGED
11	INFO	Unnecessary `globalTradingEnabled` check in `transfer` function	RESOLVED
12	INFO	`routerWhitelist` mapping is `private`	RESOLVED
13	INFO	`excludeFromFee` has unnecessary setting to zero address	ACKNOWLEDGED
14	INFO	Lack of events for `takeFee` and `transferERC20`	ACKNOWLEDGED

2 Findings

2.1 DogiraToken

The Dogira token is an ERC20 token contract with the ability to blacklist and whitelist address, which may have implications on whether those address have the ability to transfer tokens. There is a long list of addresses blacklisted when this contract is initialized, which the client has stated are publicly-known frontrunners and flashbots. By default, the contract hard-codes the Quickswap Router address, and generates an LP pair with WMATIC, though both of these can be changed in the future. By default, there are no transfer fees but that can be set to a maximum value of 2%, and will be applicable to any addresses that have been added to the `includedInFees` array.

Additionally, it should be noted that the token contract has the option to sell 50,000 tokens for WMATIC if so desired (assuming there are sufficient tokens in the contract balance, which arises from the swap fee, if enabled), and the dev address will receive them. Additionally, the owner of the contract has the ability to take out tokens sent to the Dogira contract by mistake – note that no user funds are held in the token contract, so this function is safe.

2.1.1 Token Overview

Address	TBC
Token Supply	1,000,000,000 (one billion)
Decimal Places	9
Transfer Max Size	50,000,000 tokens (5%)
Transfer Min Size	No minimum
Transfer Fees	Up to 2% (currently 0%)
Pre-mints	1 billion tokens

2.1.2 Privileged Roles

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

- `includeInFee`
- `excludeFromFee`
- `setUniswapRouter`
- `requestPairSwitch`
- `setUniswapPairToPending`
- `revertToPreviousPair`
- `lockPairSwitching`
- `setBridgeAddress`
- `setTaxFeePercent`
- `setMaxTxPercent`
- `setConvertMinimum`
- `setSwapEnabled`
- `setGlobalTradingEnabled`
- `unlockBlacklist`
- `lockBlacklist`
- `addToWhitelist`
- `addToBlacklist`
- `removeFromBlacklist`

2.1.3 Issues & Recommendations

Issue #01	Owner can blacklist any address
Severity	HIGH SEVERITY
Description	The `addToBlacklist` function can be called by the owner to blacklist any address at any time, which essentially causes them to not be able to send or sell tokens. This makes the token a honeypot to that address as transferring native tokens would be impossible. Note that this function should never be called on relevant operational contracts such as the token, LP or Router, for example, which may result in all transfers being blocked until removed.
Recommendation	Consider removing this ability as it poses a significant risks towards users. Innocent users may be blacklisted without due cause, whether by accident, spite or malice. Should this function be considered absolutely necessary, consider setting the Owner behind a very long Timelock, ideally 2 days or more, which removes the ability to arbitrarily call this function on any user at any time. Consider also being transparent on when and who this function may be called. We realize that this function will probably be used to blacklist bots, though the risk it poses is nonetheless monumental.
Resolution	RESOLVED The client has explained that they will only ever call this function sparingly to blacklist addresses which have been identified as causing harm to the protocol, and given that there is a 3 day delay, users would be able to see the transaction being queued and react accordingly.

Issue #02	`setUniswapRouter` could be used to turn the token into a honeypot
Severity	HIGH SEVERITY
Description	The upgraded router contract could be a malicious contract that simply keeps the tokens sent to it, or even worse a contract that prevents selling transactions by always reverting.
Recommendation	Consider removing this function. It is very rare that any protocol should ever require changing the router, especially if they have already hard-coded and use an established, prominent router contract.
Resolution	RESOLVED The client has acknowledged and commits to ensuring that the Router used will never honeypot the token, and will transfer token ownership to a sufficiently long Timelock (more than 1 day).

Issue #03	LP pair can be changed
Severity	HIGH SEVERITY
Description	The `setUniswapPair` function currently gives the governance address the ability to change the address of the LP pair. We are unsure why there is this ability, as this should absolutely not be required in any circumstances.
Recommendation	Consider removing this function. If it is absolutely necessary, please do inform us of the reason as we are concerned for the ability of governance to change the LP pair address.
Resolution	RESOLVED The client has explained that they are still in discussions on which protocol's AMM they will ultimately settle on. Given this acceptable rationale and the 7 day delay, we shall mark this issue Resolved.

Issue #04	isIncludedInFees, isInBlacklist and removeFromBlacklist may run out of gas
Severity	MEDIUM SEVERITY
Description	As these use arrays, should the length become extremely large, querying these arrays could run out of gas. This may then cause the transfer function to fail. If they do not fail, then transfers would still be very gas-intensive and thus potentially costly.
Recommendation	Consider using mapping instead of arrays.
Resolution	RESOLVED The client has clarified that taxing certain addresses (as well as the other blacklist-related functions) will be used very sparingly, if at all. Additionally, this feature is only to be used in the very far future should there be a need for marketing or development expenses, and enabling taxes would require community approval via DAO.

Issue #05	swapTokens function may fail and cause token to be a honeypot
Severity	MEDIUM SEVERITY

Location	Lines 413-424
	`function swapTokens(uint256 contractTokenBalance) private lockTheSwap {`
	`    uint256 initialBalance = address(this).balance;`
	
	`    // swap tokens for ETH`
	`    uint256 toSwapForEth = contractTokenBalance;`
	`    swapTokensForEth(toSwapForEth);`
	
	`    // how much ETH did we just swap into?`
	`    uint256 fromSwap = address(this).balance.sub(initialBalance);`
	
	`    devWallet.transfer(fromSwap);`
	`}`

Description	Should the `devWallet` be set to a contract that does not contain the payable modifier, then `devWallet` would not be able to receive any ETH, and thus result in the token being turned into a honeypot.
Recommendation	The simplest solution would be to use `send` instead of `transfer`. A better solution would be to instead use `wETH` and `safeTransfer`, like so: `wETH.safeTransfer(devWallet, fromSwap);`
Resolution	RESOLVED The client has stated that that they will manually ensure that the `devWallet` is set to a valid EOA address, and that this issue should only ever potentially manifest if fees were enabled (currently disabled).

Issue #06	**Setting tax fee to zero is not possible if taxFee was previously set to a positive number**
Severity	MEDIUM SEVERITY
Location	Lines 345 - 354

```
function removeAllFee() private {
    if(_taxFee == 0) return;

    _previousTaxFee = _taxFee;
    _taxFee = 0;
}

function restoreAllFee() private {
    _taxFee = _previousTaxFee;
}
```

Lines 448 - 457

```
//this method is responsible for taking all fee, if takeFee
is true
function _tokenTransfer(address sender, address recipient,
uint256 amount,bool takeFee) private {
    if(!takeFee)
        removeAllFee();

     _transferStandard(sender, recipient, amount);

    if(!takeFee)
        restoreAllFee();
}
```

Description

Assuming that:

- `previousTaxFee` is 2%
- `takeFee == false`
- `setTaxFeePercent` has been called to set `_taxFee` to 0

When the `_tokenTransfer` function is called, `removeAllFee` would be triggered. The following series of events would thus occur:

1. Since `taxFee == 0` is `True`, `removeAllFee` would return early. previousTaxFee would still remain as 2%.
2. `_transferStandard` would pass without issue.
3. `restoreAllFee` would then set `_taxFee` to `_previousTaxFee`.

The end result of this is that the current `taxFee` is now 2% again.

Recommendation

Consider setting both previous and current tax fees to the new figure if `setTaxFeePercent` is called, like so:

```
function setTaxFeePercent(uint256 taxFee) external
onlyOwner() {
    require(taxFee <= 2, "Input number between 0 - 2");
    _taxFee = taxFee;
    _previousTaxFee = taxFee;

    emit TaxUpdated(taxFee);
}
```

Alternatively, if `taxFee == 0`, then returning early in `restoreAllFee` could also be considered (similar to the mechanism in `removeAllFee`).

Resolution

This issue should only ever potentially manifest if fees were enabled (currently disabled), and it can be corrected relatively quickly by manually calling `setTaxFeePercent` twice with a zero input.

Issue #07	Contract uses raw addition
Severity	INFORMATIONAL

Location	Line 251
	`pairSwitchUnlockTimestamp = now + 7 days;`
	
	Line 472
	`blacklistUnlockTimestamp = now + (_daysUntilUnlock * 60 * 60 * 24);`

Description	Although the risk of overflows is low, this risk still remains and may present itself in edge cases as the contract uses Solidity version ^0.6.12.
Recommendation	Consider using SafeMath's add rather than raw addition.
Resolution	ACKNOWLEDGED

Issue #08	`transferERC20` does not use SafeERC20
Severity	INFORMATIONAL

Location	Lines 542-543
	`function transferERC20(address tokenAddress, address ownerAddress, uint tokens) external onlyOwner returns (bool success) {`
	`    return IERC20(tokenAddress).transfer(ownerAddress, tokens);`
	`}`

Description	It is considered best practice to use OpenZeppelin's SafeERC20 for any token transfer operations.
Recommendation	Consider importing and using `safeTransfer` instead of `transfer` in the function.
Resolution	ACKNOWLEDGED

Issue #09	Minor correction to `setConvertMinimum`
Severity	INFORMATIONAL

Location	Lines 300-305
	`function setConvertMinimum(uint256 _numTokensSwap) external onlyOwner {`
	`    require(_numTokensSwap < 500000, "Minimum token conversion amount cannot exceed 500,000 tokens!");`
	`    require(_numTokensSwap > 500, "Minimum token conversion amount cannot be under 500 tokens!");`
	`    numTokensSwap = _numTokensSwap * 10**9;`
	`    emit MinTokensBeforeSwapUpdated(numTokensSwap);`
	`}`

Description	The comments mention that `numTokensSwap` should not exceed 500,000 tokens nor be set below 500. In its current state, setting to precisely those figures would not be possible due to the require statements.
Recommendation	Consider using <= and >= operators instead, to allow for the upper limit of 500,000 tokens and lower limit of 500 tokens to be precisely set.
Resolution	ACKNOWLEDGED

Issue #10	Before-after method in swapTokens is unnecessary
Severity	INFORMATIONAL

Location	Lines 414-421
	`uint256 initialBalance = address(this).balance;`
	
	`// swap tokens for ETH`
	`uint256 toSwapForEth = contractTokenBalance;`
	`swapTokensForEth(toSwapForEth);`
	
	`// how much ETH did we just swap into?`
	`uint256 fromSwap =`
	`address(this).balance.sub(initialBalance);`

Description	The before-after method allows for only the amounts of ETH received from `swapTokensForEth` to be transferred to the devWallet. However, as no user funds are actually held in the token contract, the before-after method looks to be redundant.
Recommendation	Consider removing this so as to reduce the length of the contract.
Resolution	ACKNOWLEDGED

Issue #11	**Unnecessary `globalTradingEnabled` check in `transfer` function**
Severity	INFORMATIONAL
Location	Line 366 `require(_msgSender() == owner() || globalTradingEnabled, "Trading has not yet been enabled.");`
Description	In the previous line, there is already a check for `globalTradingEnabled`: `if (!globalTradingEnabled && !_routerWhitelist[from] && !_routerWhitelist[to]) {` As such, this additional check on line 366 is unnecessary.
Recommendation	Consider removing `globalTradingEnabled` on line 366.
Resolution	RESOLVED The client has stated that this is an intended functionality.

Issue #12	**`routerWhitelist` mapping is `private`**
Severity	INFORMATIONAL
Description	The `routerWhitelist` mapping is private and there is currently no public function to allow for users to query which addresses have been whitelisted via the addToWhitelist function.
Recommendation	Consider having a public `isInWhitelist` function, or making the `routerWhitelist` mapping `public`.
Resolution	RESOLVED The client has stated that this is intended to only whitelist routers for liquidity provision.

Issue #13	`excludeFromFee` has unnecessary setting to zero address
Severity	INFORMATIONAL
Location	Line 231 `_includedInFees[_includedInFees.length - 1] = address(0x0);`
Description	Setting this to the zero address is not required since the next line simply just removes it.
Recommendation	Consider removing this line as it is unnecessary.
Resolution	ACKNOWLEDGED

Issue #14	Lack of events for `takeFee` and `transferERC20`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the above functions.
Resolution	ACKNOWLEDGED


PALADIN
BLOCKCHAIN SECURITY