PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For PolyWantsACracker

17 July 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 PolyWantsACracker. 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	PolyWantsACracker
URL	https://polywantsacracker.farm/
Platform	Polygon
Language	Solidity

1.2 Contracts Assessed

The final contracts below have been provided to Paladin in a zip archive after resolving the issues raised in the preliminary report. We have verified that the final deployed contracts match the ones in the zip archive.

Name	Contract	Live Code Match
LithToken	0xfE1a200637464FBC9B60Bc7AeCb9b86c0E1d486E	MATCH
MasterChefV2	0xfcD73006121333C92D770662745146338E419556	MATCH
PLithToken	0xfD30189bD6de5503bB1db60cf1136123EdEA837A	MATCH
LithRedeem	0xCcA55FAF3BF71dba92694877CB09c577A226aEaF	MATCH
Timelock	0x6a8af1dbFdb32dAc39BF8A386c03cC8857a107a8	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	1	1	-	-
Medium	1	1	-	-
Low	5	5	-	-
Informational	20	20	-	-
Total	**27**	**27**	**0**	**0**

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 LithToken

ID	Severity	Summary	Status
01	INFORMATIONAL	Gas inefficiency: SafeMath is redundant in Solidity v0.8.0	RESOLVED
02	INFORMATIONAL	`delegateBySig` can be frontrun and cause denial of service	RESOLVED
03	INFORMATIONAL	`console.sol` is imported but not used	RESOLVED

1.3.2 Masterchef

ID	Severity	Summary	Status
04	HIGH	Severely excessive rewards issue when a token with a transfer tax is added	RESOLVED
05	MEDIUM	Setting `feeAddress` to the zero address will break most functionality	RESOLVED
06	LOW	`add` function is unnecessarily complex	RESOLVED
07	INFORMATIONAL	Pools use the contract balance to figure out the total deposits	RESOLVED
08	INFORMATIONAL	Gas inefficiency: SafeMath is redundant in Solidity v0.8.0	RESOLVED
09	INFORMATIONAL	`add`, `set`, `withdraw`, `emergencyWithdraw`, `setStartBlock` and `setFeeAddress` can be made `external`	RESOLVED
10	INFORMATIONAL	Lack of events for `add` and `set`	RESOLVED
11	INFORMATIONAL	Adding pools requires the `msg.sender` to have approved the relevant token and have a small amount of them in their wallet	RESOLVED

1.3.3 PLithToken

ID	Severity	Summary	Status
12	LOW	Transaction does not revert when `purchaseAmount` is zero	RESOLVED
13	LOW	Sale price can be set to any amount	RESOLVED
14	LOW	Purchase does not revert if a low-level matic transfer fails	RESOLVED
15	INFORMATIONAL	Unnecessary `return` statement after purchase logic	RESOLVED
16	INFORMATIONAL	Assertion that cannot fail is made with the `require` keyword	RESOLVED
17	INFORMATIONAL	`msg.sender` is unnecessarily cast to an address	RESOLVED
18	INFORMATIONAL	`msg.value` is unnecessarily cast to an `uint256`	RESOLVED
19	INFORMATIONAL	Gas inefficiency: SafeMath is redundant in Solidity v0.8.0	RESOLVED
20	INFORMATIONAL	Lack of events for `setSalePriceE35`, `setStartBlock`, and `buyPLith`	RESOLVED
21	INFORMATIONAL	`setSalePriceE35`, `setStartBlock`, and `buyPLith` can be made `external`	RESOLVED

1.3.4 LithReedem

ID	Severity	Summary	Status
22	LOW	Surplus LITH tokens, and by extension any tokens sent to this contract, cannot be withdrawn.	RESOLVED
23	INFORMATIONAL	Presale LITH tokens remain stuck in the contract	RESOLVED
24	INFORMATIONAL	Constructor does not have parameter safeguards	RESOLVED
25	INFORMATIONAL	Gas inefficiency: SafeMath is redundant in Solidity v0.8.0	RESOLVED
26	INFORMATIONAL	`setStartBlock` and `swapPLithForLith` can be made external	RESOLVED
27	INFORMATIONAL	Lack of events for `setStartBlock` and `swapPLithForLith`	RESOLVED

1.3.5 Timelock

No issues found.

2 Findings

2.1 LithToken

The LithToken is a simple token with basic governance functionality. It allows for LITHIUM tokens to be minted when the `mint` function is called by the Owner prior to ownership being transferred to the Masterchef. This can be used to pre-mint tokens for various uses including injection of initial liquidity, token presale, airdrops, and others. After ownership is transferred to the Masterchef, only the Masterchef will be able to mint tokens.

At the time of this report, 37,500 tokens were pre-minted to 0x3a1D1114269d7a786C154FE5278bF5b1e3e20d31 (an EOA).

2.1.1 Token Overview

This address below is the contract that Paladin has audited. Once we have received the final deployed contracts, we will update the address below.

Address	0xfE1a200637464FBC9B60Bc7AeCb9b86c0E1d486E
Token Supply	100,000 (one hundred thousand)
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	None

2.1.2 Privileged Roles

The ownership of the token contract has been transferred to the Masterchef. The following `onlyOwner` functions can be called by the Masterchef:

- `mint`

2.1.3 Issues & Recommendations

Issue #01	Gas inefficiency: SafeMath is redundant in solidity v0.8.0
Severity	INFORMATIONAL
Description	Overflow checks have been added to solidity v0.8.0 (https://docs.soliditylang.org/en/breaking/080-breaking-changes.html). The usage of SafeMath is thus redundant when contracts are compiled in this version and will result in unnecessary gas wasted.
Recommendation(s)	Consider replacing the `ERC20` implementation with a v0.8.0 version that does not use SafeMath.
Resolution	RESOLVED The client has moved to a v0.8.0 implementation of `ERC20`.

Issue #02	`delegateBySig` can be frontrun and cause denial of service
Severity	INFORMATIONAL
Description	Currently if `delegateBySig` is executed twice, the second execution will be reverted. It is thus in theory possible for a bot to pick up `delegateBySig` transactions in the mempool and execute them before a contract can. The issue with this is that the rest of said contract functionality would be lost as well. This could be a problem in the case it would have been executed by a contract that would have rewarded you for your delegation for example.
Recommendation(s)	Consider adding the desired message sender in the struct hash and requiring this desired sender to be equal to `msg.sender`. This reduces the problem to having the message sender be able to frontrun you which is okay if it is a reviewed contract.
Resolution	RESOLVED The client has removed all boilerplate governance functionality including this as they will not need it.

Issue #03	`console.sol` is imported but not used
Severity	INFORMATIONAL
Description	Within the `BEP20` library, `console.sol` is imported but never used.
Recommendation(s)	Consider removing this dependency to make third-party reviews easier.
Resolution	RESOLVED The client has omitted this library from their deployment.

2.2 Masterchef

The Masterchef is a fork of Goose Finance's MasterchefV2. Deposit fees are limited to 4.01% and token emissions halt when 100,000 tokens have been minted. Finally, the `startBlock` can be moved as long as farming has not started.

2.2.1 Privileged Roles

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

- `add`
- `set`
- `setStartBlock`
- `setFeeAddress`

2.2.2 Issues & Recommendations

Issue #04	Severely excessive rewards issue when a token with a transfer tax is added
Severity	HIGH SEVERITY
Description	When tokens with a transfer tax are added to the pools, this will result in significant excessive rewards for the pool. Due to the way the Masterchef handles the reward mechanism, rewards can be heavily inflated when the balance of the Masterchef no longer matches that of user deposits. This happens for example with transfer tax tokens. This flaw of the Masterchef has recently been exploited on a significant number of projects, in which the native tokens went to $0 afterwards in all cases. This issue was also present in SushiSwap (the original Masterchef). Since the Masterchef was designed hold only LP tokens and not other types of tokens, it was not a problem in SushiSwap but has become a problem to projects who have started forking it for usage with less standard tokens.

Recommendation(s)	Consider using the current standard of handling deposits, which is based on how Uniswap handles transfer fees:
	
	`uint256 balanceBefore = pool.lpToken.balanceOf(address(this));`
	`pool.lpToken.transferFrom(msg.sender, address(this), _amount);`
	`_amount = pool.lpToken.balanceOf(address(this)).sub(balanceBefore);`
	
	Note that by using this method, you can also add the specific transfer tax logic for the native token.

Resolution	RESOLVED The recommended before-after pattern has been implemented.

Issue #05	Setting `feeAddress` to the zero address will break most functionality
Severity	MEDIUM SEVERITY
Description	Within the token contract, minting or transferring tokens to the zero address will revert the transaction. Deposits and withdrawals will break if the `feeAddress` is ever set to the zero address. Harvesting will fail as well.
Recommendation(s)	To prevent this from ever happening by accident and to limit governance risks, consider adding a requirement like `require(feeAddress != address(0), "!nonzero");` to the configuration function.
Comments	RESOLVED The recommended safety check has been added.

Issue #06	`add` function is unnecessarily complex.
Severity	LOW SEVERITY
Description	Simplification of the bottom half which attempts to accurately account for emission rewards.
Recommendation(s)	Consider replacing this code with the following line that achieves the same purpose, as well as the corresponding comments to explain its purpose. `deposit(poolInfo.length.sub(1), 100);`
Resolution	RESOLVED The client has removed this functionality completely and reverted to a standard `add` function. The reasoning behind the previous function was that their minting system would mint slightly less tokens than their hard-cap. Since they want to provide their investors with safe and reviewable code, they've decided to keep things simple and remove this additional functionality. Removing this code however led to the introduction of a secondary issue where the client could accidentally add a token address which is actually not a token (eg. a simple wallet). Doing this would break the `updatePool` functionality. This was resolved as well by calling the `balanceOf` function on the token in the `add` function. This essentially ensures that at the point of addition, the address is a token address with the required function working. This secondary issue is thus resolved as well.

Issue #07	Pools use the contract balance to figure out the total deposits
Severity	INFORMATIONAL
Description	As with pretty much all masterchefs, the total number of tokens in the Masterchef contract is used to determine the total number of deposits. This can cause a dilution of rewards when people accidentally send tokens to the Masterchef. More severely, because the native token is constantly minted, this will cause severe dilution on the native token pool.
Recommendation(s)	Consider adding an `lpSupply` variable to the `PoolInfo` that keeps track of the total deposits.
Resolution	RESOLVED The recommended solution of using `lpSupply` has been implemented.

Issue #08	Gas inefficiency: SafeMath is redundant in Solidity v0.8.0
Severity	INFORMATIONAL
Description	Overflow checks have been added to Solidity v0.8.0 (https://docs.soliditylang.org/en/breaking/080-breaking-changes.html). The usage of SafeMath is thus redundant when contracts are compiled in this version and will result in unnecessary gas wasted.
Recommendation(s)	Consider replacing the `ERC20` implementation with a v0.8.0 version that does not use SafeMath.
Resolution	RESOLVED The client has removed their SafeMath dependency and has replaced the SafeMath functions with standard arithmetic.

Issue #09	`add`, `set`, `withdraw`, `emergencyWithdraw`, `setStartBlock` and `setFeeAddress` can be made `external`
Severity	INFORMATIONAL
Description	The `add`, `set`, `withdraw`, `emergencyWithdraw`, `setStartBlock` and `setFeeAddress` 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(s)	Consider making these functions `external`.
Comments	RESOLVED

Issue #10	Lack of events for `add` and `set`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation(s)	Add events for `add` and `set`.
Resolution	RESOLVED

Issue #11	Adding pools requires the `msg.sender` to have approved the relevant token and have a small amount of them in their wallet
Severity	INFORMATIONAL
Description	When adding a pool, make sure that the sender actually has a small amount of tokens in their wallet and has given the Masterchef approval on the token. If the owner is a timelock, the `add()` function thus has to be preceded with an `approve()` function on the token.
Recommendation(s)	Consider this requirement when adding new pools.
Comments	RESOLVED This issue is no longer relevant as the code has been removed.

2.3 PLithToken

The PLithToken contract is a presale contract, where 1 MATIC = 0.1666 PLith tokens at the current rate, which is subject to change up to 4 hours before the start of the presale as they are intending to price their presale tokens at $5, subject to the volatility of MATIC's prices. One account can purchase up to 600 tokens.

2.3.1 Privileged Roles

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

- `setStartBlock`
- `setSalePriceE35`

2.3.2 Issues & Recommendations

Issue #12	Transaction does not revert when `purchaseAmount` is zero
Severity	LOW SEVERITY
Description	When the purchase amount is zero, the whole transferred amount will be refunded to the user. This causes unnecessary gas to be wasted and might even confuse users as there is no error message.
Recommendation(s)	Consider adding `require(plithPurchaseAmount > 0, "reason")` as soon as this value is finalized.
Comments	RESOLVED

Issue #13	Sale price can be set to any amount
Severity	LOW SEVERITY
Description	There are no upper or lower bounds on prices beyond the initial price of 1 MATIC = 0.1666 tokens

Recommendation(s)	Consider adding sanity checks, which has the benefit of letting investors know of the minimum or maximum possible token prices.
	`require(_newSalePriceE35 >= 3 * (10 ** 33));`
	`require(_newSalePriceE35 <= 6 * (10 ** 34));`

	RESOLVED The sale price must now be between 0.02 and 0.4 PLith tokens per MATIC.

Issue #14	Purchase does not revert if a low-level matic transfer fails
Severity	LOW SEVERITY
Description	In case a solidity low-level call fails, the parent transaction will not fail. Instead, the contract needs to manually handle the return type of low-level calls. This could result in either the presale participants not receiving their refund or the `feeAddress` not receiving the presale funds.
Recommendation(s)	Consider handling the `return` value of all low-level calls.
Resolution	RESOLVED The low level call `return` value is now taken into consideration.

Issue #15	Unnecessary `return` statement after purchase logic
Severity	INFORMATIONAL
Description	After the purchase logic is finished, a `return` statement appears. This statement is unnecessary as no code is ever executed after this.
Recommendation(s)	Consider removing the unnecessary `return` statement at the end of the purchase logic.
Resolution	RESOLVED The `return` statement in question has been removed.

Issue #16	Assertion that cannot fail is made with the `require` keyword
Severity	INFORMATIONAL
Description	When assertions are not meant to ever fail, it is recommended to use the keyword `assert` instead of `require`.

Recommendation(s)	Consider replacing
	`require(plithPurchaseAmount <= IBEP20(address(this)).balanceOf(address(this)),`
	with
	`assert(plithPurchaseAmount <= IBEP20(address(this)).balanceOf(address(this)),`

Resolution	RESOLVED The requirement statements have been replaced with the syntactically more correct assert statements.

Issue #17	`msg.sender` is unnecessarily cast to an address
Severity	INFORMATIONAL
Description	In solidity, the `msg.sender` literal is an address and thus does not need to be cast to an address manually.
Recommendation(s)	Consider replacing `address(msg.sender)` with `msg.sender`.
Resolution	RESOLVED All occurrences have been replaced.

Issue #18	`msg.value` is unnecessarily cast to an `uint256`
Severity	INFORMATIONAL
Description	In Solidity, the `msg.value` literal is an `uint256` and thus does not need to be cast to an `uint256` manually.
Recommendation(s)	Consider replacing `uint256(msg.value)` with `msg.value`.
Resolution	RESOLVED All occurrences have been replaced.

Issue #19	Gas inefficiency: SafeMath is redundant in Solidity v0.8.0
Severity	INFORMATIONAL
Description	Overflow checks have been added to Solidity v0.8.0 (https://docs.soliditylang.org/en/breaking/080-breaking-changes.html). The usage of SafeMath is thus redundant when contracts are compiled in this version and will result in unnecessary gas wasted.
Recommendation(s)	Consider replacing the `ERC20` implementation with a v0.8.0 version that does not use SafeMath.
Resolution	RESOLVED The SafeMath dependency has been removed and replaced with standard arithmetic, which is safe against overflows and underflows in Solidity v0.8.0 and up.

Issue #20	Lack of events for `setSalePriceE35`, `setStartBlock`, and `buyPLith`
Severity	INFORMATIONAL
Description	`setSalePriceE35`, `setStartBlock`, and `buyPLith` should emit events as notifications.
Recommendation(s)	Add events for these functions.
Resolution	RESOLVED

Issue #21	`setSalePriceE35`, `setStartBlock`, and `buyPLith` can be made external
Severity	INFORMATIONAL
Description	`setSalePriceE35`, `setStartBlock`, and `buyPLith` 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 (https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices).
Recommendation(s)	Consider making these functions `external`.
Resolution	RESOLVED

2.4 LithRedeem

LithRedeem allows for the swapping of the presale PLITH tokens for the native LITH tokens at a 1:1 rate. Note that this requires LITH to be present in the LithRedeem contract.

2.4.1 Privileged Roles

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

- `setStartBlock`

2.4.2 Issues & Recommendations

Issue #22	Surplus LITH tokens, and by extension any tokens sent to this contract, cannot be withdrawn
Severity	LOW SEVERITY
Description	Any excess LITH tokens are permanently stuck in this contract.
Recommendation(s)	Consider adding the following to allow for excess LITH tokens to be withdrawn to the Owner's address. This function should only be callable after sufficient time has passed, for example, by requiring that at least 1 week has passed after `startBlock`.

```
// Withdraw excess Lith tokens. EMERGENCY ONLY.

function emergencyLithTokensWithdraw(uint256 _amount) public onlyOwner {
    require(_amount < IBEP20(lithAddress).balanceOf(address(this)), 'not enough
tokens');
    IBEP20(lithAddress).safeTransfer(address(msg.sender), _amount);
}
```

A function that simply sends these tokens to the burn address might be more desirable.

Resolution	RESOLVED The unclaimed LITH tokens are now sent to the burn address.

Issue #23	Presale LITH tokens remain stuck in the contract
Severity	INFORMATIONAL
Description	When presale tokens are swapped for the actual token, these old tokens remain stuck in the contract which could mislead investors into thinking the presale supply is still large.
Recommendation(s)	Consider transferring these tokens directly to the dead address instead of the contract itself.
Resolution	RESOLVED The presale LITH tokens are now sent to the burn address.

Issue #24	Constructor does not have parameter safeguards
Severity	INFORMATIONAL
Description	Checks to ensure that PLITH and LITH are not the same token.
Recommendation(s)	Consider adding the following check to the constructor : `require(_lithAddress != _plithAddress)`
Resolution	RESOLVED The recommended check has been implemented.

Issue #25	Gas inefficiency: SafeMath is redundant in Solidity v0.8.0
Severity	INFORMATIONAL
Description	Overflow checks have been added to Solidity v0.8.0 (https://docs.soliditylang.org/en/breaking/080-breaking-changes.html). The usage of SafeMath is thus redundant when contracts are compiled in this version and will result in unnecessary gas wasted.
Recommendation(s)	Consider replacing the `ERC20` implementation with a v0.8.0 version that does not use SafeMath.
Resolution	RESOLVED SafeMath has been removed as a dependency.

Issue #26	`setStartBlock` and `swapPLithForLith` can be made external
Severity	INFORMATIONAL
Description	The `setStartBlock` and `swapPLithForLith` 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(s)	Consider making these functions `external`.
Resolution	RESOLVED

Issue #27	Lack of events for `setStartBlock` and `swapPLithForLith`
Severity	INFORMATIONAL
Description	`setStartBlock` and `swapPLithForLith` should emit events as notifications.
Recommendation(s)	Add events for `setStartBlock` and `swapPLithForLith`.
Resolution	RESOLVED

2.5 Timelock

The timelock is a standard clone of Compound Finance's timelock.

2.5.1 Issues & Recommendations

No issues found.


PALADIN
BLOCKCHAIN SECURITY