PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Fury Labs Protocol

24 January 2022

paladinsec.co

info@paladinsec.co

Table of Contents

Table of Contents 2

Disclaimer 3

1 Overview 4

1.1 Summary 4

1.2 Contracts Assessed 4

1.3 Findings Summary 5

1.3.1 FuryToken 6

1.3.2 MasterChef 6

1.3.3 FeesProcessor 7

1.3.4 Timelock 7

2 Findings 8

2.1 FuryToken 8

2.1.1 Token Overview 8

2.1.2 Privileges 9

2.1.3 Issues & Recommendations 10

2.2 MasterChef 12

2.2.1 Privileges 13

2.2.2 Issues & Recommendations 14

2.3 FeesProcessor 19

2.3.1 Privileges 19

2.3.2 Issues & Recommendations 20

2.4 Timelock 23

2.4.1 Issues & Recommendations 23

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 Fury Labs on the Fantom Opera network. 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	Fury Labs
URL	https://furylabsfinance.com/
Platform	Fantom Opera
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`FuryToken`	`0xB1822A7ee73DD7de6Eda328A0681f8E1779CC4B6`	MATCH
`MasterChef`	`0x23e2DA1657C2b552185d7AF485d6f4825f68200a`	MATCH
`FeesProcessor`	`0x7fa56e721737ba4f44fd58678932d4e153dbe209`	MATCH
`Timelock`	`0xFd15bA6686aD0d3b65d30258c3c9e0a623236827`	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	0	-	-	-
Medium	2	1	-	1
Low	7	4	-	3
Informational	7	4	-	3
Total	**16**	**9**	**-**	**7**

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 FuryToken

ID	Severity	Summary	Status
01	MEDIUM	Masterchef needs to be whitelisted in `_excludedFromAntiWhale` or large harvest amounts will revert	RESOLVED
02	LOW	Inconsistency between usage of `_msgSender()` and `msg.sender`	RESOLVED
03	LOW	`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef	RESOLVED
04	INFO	`BURN_ADDRESS` is an unused state variable	RESOLVED

1.3.2 MasterChef

ID	Severity	Summary	Status
05	LOW	`updateEmissionRate` will run out of gas if there are too many pools	RESOLVED
06	LOW	`furyReward` used for `accFuryPerShare` calculation is not updated if there is a change in token amount minted	RESOLVED
07	LOW	Initial reward emission set in constructor can be higher than `MAX_EMISSION_RATE`	ACKNOWLEDGED
08	INFO	Unnecessary casting of `msg.sender` into `address`	RESOLVED
09	INFO	`startTime` can be set to a time before deployment time in the constructor	ACKNOWLEDGED
10	INFO	`pendingFury` will show inaccurate pending harvests on the dapp frontend if the pending rewards causes `totalSupply` to be exceed `MAX_SUPPLY`	RESOLVED

1.3.3 FeesProcessor

ID	Severity	Summary	Status
11	MEDIUM	add function still allows manual input of `tokenA` and `tokenB`	ACKNOWLEDGED
12	LOW	`convertToUSDT` will fail if there is no USDT pair for the token	ACKNOWLEDGED
13	LOW	`withdrawFunds` and `feesProcessing` can run out of gas if there are too many entries in `lpInfo`	ACKNOWLEDGED
14	INFO	The line `USDT.approve(address(spookyswapRouter), tokenAmount);` has results that is not affecting the functions in the contract	ACKNOWLEDGED
15	INFO	Use `safeIncreaseAllowance` instead of `entryToken.approve`	ACKNOWLEDGED
16	INFO	Comments reference a different project in the contract	RESOLVED

1.3.4 Timelock

No issues found.

2 Findings

2.1 FuryToken

The Fury token is an ERC-20 token which will be used as the main reward token for the Masterchef. The contract allows for Fury tokens to be minted when the `mint` function is called by the owner of the contract, which at the time of deployment would be the Fury team. Ownership of the contract should be transferred to the Masterchef after deployment. There is a maximum supply of 800,000 Fury tokens, enforced by the Masterchef contract.

There is a maximum amount per transaction. If a transfer greater than this amount is made, and the sender or recipient is not in `_excludedFromAntiWhale`, the transaction will revert and not be successfully completed. The minimum capped for this is 1% of the current supply.

2.1.1 Token Overview

Address	0xB1822A7ee73DD7de6Eda328A0681f8E1779CC4B6
Token Supply	800,000
Decimal Places	18
Transfer Max Size	3% (Lower cap of 1 % of current supply)
Transfer Min Size	None
Transfer Fees	None
Pre-mints	36,700

2.1.2 Privileges

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

- `mint`
- `transferOwnership`
- `renounceOwnership`

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

- `updateMaxUserTransferAmountRate`
- `updateExclusionFromAntiWhale`
- `transferOperator`

2.1.3 Issues & Recommendations

Issue #01	**Masterchef needs to be whitelisted in `_excludedFromAntiWhale` or large harvest amounts will revert**
Severity	MEDIUM SEVERITY
Description	If Masterchef is not whitelisted in `_excludedFromAntiWhale` and a harvest amount which exceeds `maxTransferAmount` is done, the transaction will revert. This would result in the user being unable to make normal deposits or withdrawals from the Masterchef.
Recommendation	Ensure that the Masterchef contract address is whitelist in `_excludedFromAntiWhale`.
Resolution	RESOLVED The Masterchef has been whitelisted in `_excludedFromAntiWhale`.

Issue #02	**Inconsistency between usage of `_msgSender()` and `msg.sender`**
Severity	LOW SEVERITY
Description	There are instances in the code where `msg.sender` is used, and others where `_msgSender()` is used. This inconsistency could cause some issues as `_msgSender()` returns `msg.sender` for regular transactions, but for meta transactions, it can be used to return the end user instead of the relayer.
Recommendation	Consider using `_msgSender()` to replace all instances of `msg.sender`.
Resolution	RESOLVED `_msgSender` is used to replace all instances of `msg.sender`.

Issue #03	**`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef**
Severity	LOW SEVERITY
Description	This function could be used to pre-mint tokens before ownership is transferred to the Masterchef contract. This could have a significant impact on the tokenomics of the project.
Recommendation	Consider being forthright if this mint function has been used by letting your community know how much was minted, where they are currently stored, if a vesting contract was used for token unlocking, and finally the purpose of the mints.
Resolution	RESOLVED 36,700 tokens were pre-minted and ownership has been transferred to the Masterchef.

Issue #04	**`BURN_ADDRESS` is an unused state variable**
Severity	INFORMATIONAL
Description	Although declared and defined as a constant, the `BURN_ADDRESS` is never used anywhere else in the contract.
Recommendation	Consider removing the `BURN_ADDRESS` state variable.
Resolution	RESOLVED The unused state variable has been removed.

2.2 MasterChef

The Fury Masterchef is a fork of Goose Finance's Masterchef. A notable feature of forking the latter is the removal of the `migrator` function from Sushiswap, which can be used maliciously to steal tokens staked in the Masterchef. The deposit fee is limited to a maximum of 4%.

Emissions have been modified to be per second instead of per block. The Masterchef has a check that ensures that only a maximum of 800,000 Fury tokens are minted. The maximum emission rate that can be set is 10 tokens per second.

2.2.1 Privileges

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

- `add`
- `set`
- `updateEmissionRate`
- `transferOwnership`
- `renounceOwnership`

The following functions can be called by the DevAddr:

- `setDevAddr`

The following functions can be called by the FeeAddr:

- `setFeeAddr`

2.2.2 Issues & Recommendations

Issue #05	**`updateEmissionRate` will run out of gas if there are too many pools**
Severity	LOW SEVERITY
Description	`updateEmissionRate` calls `massUpdatePools` to update all the pools, and is unlike `add` and `set` which have an option to skip the `massUpdatePools` call. Thus, `updateEmissionRate` might not be able to successfully execute if there are too many pools which result in `massUpdatePools` running out of gas.
Recommendation	Add a bool `_withUpdate` parameter to allow skipping of `massUpdatePools` if necessary. However, it should only be skipped if the number of pools causes the gas limit to exceed.
Resolution	RESOLVED A `_withUpdate` parameter has been added to allow skipping `massUpdatePools` if necessary. In general, it should not be skipped.

Issue #06	**furyReward used for accFuryPerShare calculation is not updated if there is a change in token amount minted**
Severity	LOW SEVERITY

Location	Line 1389~
	`if(fury.totalSupply().add(furyReward) <= MAX_SUPPLY) {`
	`  fury.mint(address(this), furyReward);`
	`} else if(fury.totalSupply() < MAX_SUPPLY) {`
	`  fury.mint(address(this), MAX_SUPPLY.sub(fury.totalSupply()));`
	`}`

Description	The above code in `updatePool` checks that the sum of total supply and `furyReward` will not exceed `MAX_SUPPLY`, and adjusts it to mint less if it does. However, the `furyReward` amount used for calculating the `accFuryPerShare` and the actual minted amount are not the same. This can cause shares that give more rewards per share that the Masterchef contract actually has.

Recommendation	The `furyReward` should be updated to the amount actually minted.
	`if(fury.totalSupply().add(furyReward) <= MAX_SUPPLY) {`
	`  fury.mint(address(this), furyReward);`
	`} else if(fury.totalSupply() < MAX_SUPPLY) {`
	`  furyReward = MAX_SUPPLY.sub(fury.totalSupply());`
	`  fury.mint(address(this), furyReward);`
	`} else {`
	`  furyReward = 0;`
	`}`

Resolution	RESOLVED The recommended change has been made.

Issue #07	Initial reward emission set in constructor can be higher than `MAX_EMISSION_RATE`
Severity	LOW SEVERITY
Description	Although the `updateEmissionRate` function ensures that the emission does not exceed `MAX_EMISSION_RATE`, it is not done in the constructor.
Recommendation	Consider adding the same check for the emission to be less than the maximum emission rate in the constructor.
Resolution	ACKNOWLEDGED

Issue #08	Unnecessary casting of `msg.sender` into `address`
Severity	INFORMATIONAL
Location	`(Example) Line 1446` `pool.lpToken.safeTransfer(address(msg.sender), _amount);`
Description	There are a number of instances where `msg.sender` is cast into an `address`, when it already is an `address` datatype.
Recommendation	Remove the casting of `msg.sender` as an `address`.
Resolution	RESOLVED `msg.sender` is no longer casted as an `address`.

Issue #09	**`startTime` can be set to a time before deployment time in the constructor**
Severity	INFORMATIONAL
Description	There is a lack of check that the `startTime` has to be equal to or after the deployment time.
Recommendation	Consider adding a check in the constructor to ensure that the `startTime` is set at or after the deployment time.
Resolution	ACKNOWLEDGED

Issue #10	**pendingFury will show inaccurate pending harvests on the dApp frontend if the pending rewards causes totalSupply to be exceed MAX_SUPPLY**
Severity	INFORMATIONAL
Description	pendingFury does not check if the pending rewards will cause the total supply to exceed the MAX_SUPPLY. This can cause inaccurate pending harvests to be shown towards the end of token emissions.

Recommendation	Consider factoring in the MAX_SUPPLY, and set the pending reward to be the difference between MAX_SUPPLY and totalSupply if the pending reward causes totalSupply to exceed MAX_SUPPLY.
	`uint256 furyReward = multiplier.mul(furyPerSecond).mul(pool.allocPoint).div(totalAllocPoint);`
	`if (fury.totalSupply().add(furyReward) > MAX_SUPPLY) { furyReward = MAX_SUPPLY.sub(fury.totalSupply()); }`
	`accFuryPerShare = accFuryPerShare.add(furyReward.mul(1e18).div(pool.lpSupply));`

Resolution	RESOLVED The recommended change has been made.

2.3 FeesProcessor

The FeeProcessor contract manages a portion of the deposit fees transferred from the Masterchef contract. The contract has the ability to add addresses into an array for tracking, can assist in breaking down LP tokens, convert tokens to USDT, and fund withdrawal.

2.3.1 Privileges

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

- `transferOwnership`
- `renounceOwnership`
- `mint`
- `withdrawFunds`

The following functions can be called by the operator:

- `add`
- `feesProcessing`

2.3.2 Issues & Recommendations

Issue #11	add function still allows manual input of `tokenA` and `tokenB`
Severity	MEDIUM SEVERITY
Description	add function still allows the manual input of `tokenA` and `tokenB` which could potentially break the swaps.
Recommendation	Consider pulling from the LP contract directly by calling the corresponding functions for each token in the LP to minimize inadvertent address inputs.
Resolution	ACKNOWLEDGED

Issue #12	`convertToUSDT` will fail if there is no USDT pair for the token
Severity	LOW SEVERITY
Description	Currently, the function only gets a path where a specific token is converted into USDT. Should the pair have no liquidity with USDT, the function will revert.
Recommendation	Consider inserting a backup path in case the USDT path will not push through due to insufficient liquidity.
Resolution	ACKNOWLEDGED

Issue #13	**`withdrawFunds` and `feesProcessing` can run out of gas if there are too many entries in `lpInfo`**
Severity	LOW SEVERITY
Description	As `withdrawFunds` and `feesProcessing` cycle through all the LP information that is added into the contract, it is possible that the caller can run out of gas if there are too many pools and the function might not be able to successfully execute.
Recommendation	Consider adding in a limiter to avoid running out of gas or the ability to manually input which pool IDs can be processed to continue processing in case the said functions continuously runs out of gas when calling.
Resolution	ACKNOWLEDGED

Issue #14	**The line `USDT.approve(address(spookyswapRouter), tokenAmount);` has results that are not affecting the functions in the contract**
Severity	INFORMATIONAL
Description	The function `convertToUSDT` still calls for an approval of USDT to be spent. Since the `entryToken` is the one to be swapped/spent, USDT approval is not necessary.
Recommendation	Consider removing the line to minimize unintended results.
Resolution	ACKNOWLEDGED

Issue #15	Use `safeIncreaseAllowance` instead of `entryToken.approve`
Severity	INFORMATIONAL
Description	The contract currently calls `approve` every time after processing of fees. Changing an allowance with this method could bring about a risk of using both old and new allowance due to frontrunning transactions.
Recommendation	After the initial approvals, consider using `safeIncreaseAllowance` or `safeDecreaseAllowance` when adjusting for the spending limits instead of setting a new approval for the tokens.
Resolution	

Issue #16	Comments reference a different project in the contract
Severity	INFORMATIONAL
Description	Several comments point towards AuroraSwap as the router.
Recommendation	Consider renaming or aligning the comments to the intended one to avoid confusion.
Resolution	RESOLVED Comments now refer to SpookySwap as the router.

2.4 Timelock

The Timelock contract is a clean fork of Compound Finance's timelock. This is the most common contract used in DeFi to time lock governance access and is thus compatible with most third-party tools. This contract should be the owner of the Masterchef contract to time delay making sensitive changes such as adding a new pool, or changing the allocation for an existing pool.

Parameter	Value	Description
Delay	24 hours	The `delay` indicates the time the administrator has to wait after queuing a transaction to execute it.
Minimum Delay	6 hours	The `minDelay` indicates the lowest value that the `delay` can minimally be set. Sometimes, projects will queue a transaction that sets the `delay` to zero with the hope that nobody notices it. However, because of the minimum delay parameter, the value of `delay` can never be lower than that of the `minDelay` value. Note that the administrator could still queue a transaction to simply transfer the ownership back to their own account so it is still important to inspect every transaction carefully.
Grace Period	14 days	After the delay has expired after queueing a transaction, the administrator can only execute it within the grace period. This is to prevent them from hiding a malicious transaction among much earlier transactions, hoping that it goes unnoticed or buried, which can be executed in the future.

2.4.1 Issues & Recommendations

No issues found.


PALADIN
BLOCKCHAIN SECURITY