PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Pandora Swap

29 April 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 Pandora Swap on the Astar 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	Pandora Swap
URL	https://pandoraswapxyz.org/
Network	Astar
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
PandoraToken	0x8ea356004327E598729b4CE590eDC90428Dc6A89	MATCH
PandoraRouter02	0x0fd60f0B13F7d816aE2DF1B9a4B62a9d94FbCac5	MATCH
PandoraFactory	0x8D4f9b98FC21787382647BFCfC9ce75C08B50481	MATCH
Masterpandora	0x894d03D77b42bBeC83CEe221596ba17a83b995eC	MATCH
ProxyPandora	Proxy 0xFAD2dB84ec6b6496544FE9E34db2EBAee17eB691 Implementation 0xc05a719138Ca3d0F63b4636CbDD99f40E4953fdc	MATCH
Referral	0x1313bF6D51026CDaD577ac09D19A498aE044d37f	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	1	1	-	-
Medium	2	-	1	1
Low	9	1	1	7
Informational	27	1	2	24
Total	**39**	**3**	**4**	**32**

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 PandoraToken

ID	Severity	Summary	Status
01	LOW	The `mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef	✓ RESOLVED
02	LOW	`mint` function does not return `false` when `_mint` returns `false` even if the mint no longer mints tokens	ACKNOWLEDGED
03	INFO	Unused import: EnumerableSet	ACKNOWLEDGED
04	INFO	Governance functionality is broken	PARTIAL
05	INFO	`delegateBySig` can be frontrun and cause denial of service	PARTIAL
06	INFO	`mint` can be made external	ACKNOWLEDGED

1.3.2 PandoraRouter02

ID	Severity	Summary	Status
07	INFO	Swaps could revert if the swapFeeReward address is set to an incompatible contract or wallet	ACKNOWLEDGED
08	INFO	Phishing Issue: A malicious, hacked, frontend could adjust routes, tokens or to parameters to steal tokens when users make swaps (issue is present in Uniswap as well)	ACKNOWLEDGED
09	INFO	Various functions can be made external (issue is present in Uniswap as well)	ACKNOWLEDGED
10	INFO	Lack of event for `setSwapFeeReward`	ACKNOWLEDGED
11	INFO	The `addLiquidity` function does not properly support tokens with a fee on transfer (present in Uniswap as well)	ACKNOWLEDGED

1.3.3 PandoraFactory

ID	Severity	Summary	Status
12	LOW	Swap fees can be modified without any delay	ACKNOWLEDGED
13	INFO	Pairs without supply but with a partial reserve might crash the frontend if the user wants to swap on this pair (present in most frontends)	ACKNOWLEDGED
14	INFO	Lack of events for `setSwapFee`, `setFeeTo` and `setFeeToSetter`	ACKNOWLEDGED
15	INFO	`permit` can be frontrun and cause denial of service	✓ RESOLVED

1.3.4 Masterpandora

ID	Severity	Summary	Status
16	HIGH	Deposits do not support tokens with a fee on transfer	✓ RESOLVED
17	MEDIUM	Users could lose their previously deposited NFT if they redeposit in the same slot	PARTIAL
18	LOW	Users may lose rewards when the maximum supply is reached	PARTIAL
19	LOW	The `updateEmissionRate` function and NFT boosting mechanism have no maximum safeguard	ACKNOWLEDGED
20	LOW	The `updateEmissionRate` function could run out of gas	ACKNOWLEDGED
21	INFO	`msg.sender` is unnecessarily cast to `address(msg.sender)`	ACKNOWLEDGED
22	INFO	`BONUS_MULTIPLIER` is not actively used	ACKNOWLEDGED
23	INFO	Pool uses the contract balance to figure out the total deposits	ACKNOWLEDGED
24	INFO	Lack of validation	ACKNOWLEDGED
25	INFO	`pandora` and `startBlock` can be made immutable	ACKNOWLEDGED
26	INFO	SafeMath should be used within the `getBoost` function	ACKNOWLEDGED
27	INFO	`proxy` can be made constant	ACKNOWLEDGED
28	INFO	Various functions can be made external	ACKNOWLEDGED
29	INFO	Lack of events for various functions	ACKNOWLEDGED
30	INFO	Typographical errors	ACKNOWLEDGED

1.3.5 ProxyPandora

ID	Severity	Summary	Status
31	MEDIUM	Governance privilege: An upgradeable contract allows the contract to change the functionality at any point in time to a contract which potentially drains the tokens in this contract	ACKNOWLEDGED
32	LOW	Disabling the whitelist is highly discouraged as anyone can call safeMeerkatTransfer to take out all the Pandora tokens at that point	ACKNOWLEDGED
33	LOW	`_disable` is private	ACKNOWLEDGED
34	INFO	Typographical error	ACKNOWLEDGED
35	INFO	Unused functionality: `SafeERC20`	ACKNOWLEDGED
36	INFO	Lack of events for `drainBEP20Token`	ACKNOWLEDGED

1.3.6 Referral

ID	Severity	Summary	Status
37	LOW	The referral addresses for users without referral records can be arbitrarily set by operator	ACKNOWLEDGED
38	INFO	`getReferrer` can be made external	ACKNOWLEDGED
39	INFO	Unused import: `SafeERC20` and `IERC20`	ACKNOWLEDGED

2 Findings

2.1 PandoraToken

The `PandoraToken` represents the main governance token within the Pandora ecosystem. It is a trivial ERC-20 token implementation and can be minted by the owner of the contract, the `Masterpandora` contract.

The contract contains YAM-like delegation logic that presently does not work. This is not a big deal as Pandora presently does not use this logic and probably does not plan to use it. This malfunctioning code is present in most tokens (Sushi, JOE...) so should not be considered a red flag.

Most of the `PandoraToken` supply, 75,000,000 tokens (93%), is held in an out-of-scope vesting contract that unlocks over the course of 570 days to governance. The validity of this vesting contract was not checked by Paladin as it is presently out-of-scope.

2.1.1 Token Overview

Address	`0x8ea356004327E598729b4CE590eDC90428Dc6A89`
Token Supply	Unlimited
Decimal Places	18
Transfer Max Size	600,000,000
Transfer Min Size	None
Transfer Fees	None
Pre-mints	5,000,000

2.1.2 Privileged Functions

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

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

2.1.3 Issues & Recommendations

Issue #01	**The `mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef**
Severity	LOW SEVERITY
Description	The mint function allows the owner (contract deployer) to mint tokens before ownership is transferred to the Masterchef. This privilege could be used to mint a large number of tokens and potentially dump them on user-generated liquidity when the token contract after deployment but before ownership is set to the Masterchef contract. This risk is prevalent amongst less-reputable projects, and any pre-mints can be prominently seen on the Blockchain.
Recommendation	Consider being forthright if this `mint` function is to be used by letting your community know how much was minted, where the tokens are currently stored, if a vesting contract was used for token unlocking, and finally, the purpose of the mints.
Resolution	RESOLVED Ownership of the token has already been transferred to the MasterChef.

Issue #02	**`mint` function does not return `false` when `_mint` returns `false` even if the mint no longer mints tokens**
Severity	LOW SEVERITY
Description	The PandoraToken contract contains a maximum supply of 600,000,000 tokens. If this supply is ever about to be reached, tokens will stop being minted. However, the `mint` function which is callable by `Masterpandora` still returns `true` which could (and actually does) mislead the contracts that mint this token.
Recommendation	Consider using the result of `_mint` as the result of `mint`. This causes `mint` to return `false` once the supply is about to be reached.
Resolution	ACKNOWLEDGED

Issue #03	**Unused import: EnumerableSet**
Severity	INFORMATIONAL

Location	Lines 1135-1136
	`using EnumerableSet for EnumerableSet.AddressSet;`
	`EnumerableSet.AddressSet private _minters;`

Description	Libraries that are imported but not used within a contract could confuse third-party auditors. They also increase the contract length unnecessarily.
Recommendation	Consider removing the import to keep the contract short and simple. The BEP20 standard which is used here is generally considered to be a BSC standard. The client could also consider extending ERC20 instead of BEP20.
Resolution	ACKNOWLEDGED

Issue #04	Governance functionality is broken
Severity	INFORMATIONAL
Description	Although there is YAM-related delegation code in the token contract which is usually used for governance and voting, the delegation code can be abused as the delegates are not moved during transfers and burns. This allows for double spending attacks on the voting mechanism. It should be noted that this issue is present in pretty much every single farm out there including PancakeSwap and even SushiSwap but it does render this whole mechanism useless. Because of this reason, projects like SushiSwap and PancakeSwap all use snapshot.org nowadays.
Recommendation	The broken delegation-related code can be removed to reduce the size of the contract. If voting is ever desired, it can still be done through snapshot.org, used by many of the larger projects.
Resolution	PARTIALLY RESOLVED The client will use snapshot.org for governance.

Issue #05	`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 case it would have been executed by a contract that would have rewarded you for your delegation for example.
Recommendation	Similar to the broken governance functionality issue, the delegate logic can just be removed.
Resolution	PARTIALLY RESOLVED The client will use snapshot.org for governance.

Issue #06	`mint` can be made external
Severity	INFORMATIONAL
Description	Functions that are not used within the contract but only externally can be marked as such with the external keyword. 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 marking the functions mentioned above as external.
Resolution	ACKNOWLEDGED

2.2 PandoraRouter02

The Pandora AMM protocol is forked from Mad Meerkat Finance (MMF) on the Cronos network, a Uniswap V2 fork with nearly $1.5 billion in TVL at the time of writing.

The Pandora Router serves as an entry point for users to exchange tokens. The `PandoraRouter02` is responsible for determining the swap rate and allowing for user swaps to be done with safety checks. More specifically, the `PandoraRouter02` allows users to add liquidity, remove liquidity and swap tokens. Automated arbitrage bots might not use the router at all, as its main purpose is a safer user interface into the core protocol (The Factory and Pairs).

One interesting addition introduced by the original fork is in the internal `_swap` function of the router. If a `swapFeeReward` contract is set by the governance, this contract will be called on every swap, potentially for "trade mining" purposes, where users are rewarded for making swaps. At the time of writing, this address is zero and there is therefore there is no "trade mining" functionality. If such logic is ever added it would therefore not be covered by this audit.

2.2.1 Privileged Functions

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

- `setSwapFeeReward`
- `renounceOwnership`
- `transferOwnership`

2.2.2 Issues & Recommendations

Issue #07	**Swaps could revert if the swapFeeReward address is set to an incompatible contract or wallet**
Severity	INFORMATIONAL

Description	In case the `swapFeeReward` is modified by the Router's owner to a non-zero address, the following lines would be executed:
	
	Lines 774-776
	`if (swapFeeReward != address(0)) {`
	`    ISwapFeeReward(swapFeeReward).swap(msg.sender, input, output, amountOut);`
	`}`
	
	This logic is included to potentially include trade mining later on where users are rewarded with native tokens for making swaps.
	
	If this `swapFeeReward` address is set to an incompatible wallet or address, all `swap` transactions would revert.
	
	This issue is raised as informational as users can simply deploy a new router to swap through and remove their liquidity.

Recommendation	Consider adding a sanity check on `setSwapFeeReward` to call a `view` function and check with some confidence that the address is not accidentally not a swap fee contract. More idiomatically, one could implement EIP-165 on the `ISwapFeeReward` interface.
Resolution	ACKNOWLEDGED

Issue #08	**Phishing Issue: A malicious, hacked, frontend could adjust routes, tokens or `to` parameters to steal tokens when users make swaps (issue is present in Uniswap as well)**
Severity	INFORMATIONAL
Description	A malicious, e.g. compromised, frontend can easily mislead users in approving malicious transactions, even if the router matches the address described in this report. An trivial example of how this can be done is by changing the `to` parameter which indicates to whom tokens or liquidity has to be sent. Other ways to phish could include using malicious routes or tokens.
Recommendation	Consider carefully protecting the frontend and ideally having an unchangeable ipfs fallback implementation for it. Users should also verify that they are on the correct website when doing a swap.
Resolution	ACKNOWLEDGED

Issue #09	Various functions can be made external (issue is present in Uniswap as well)
Severity	INFORMATIONAL
Description	Functions that are not used within the contract but only externally can be marked as such with the external keyword. 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. - `setSwapFeeReward` - `quote` - `getAmountOut` - `getAmountIn` - `getAmountsOut` - `getAmountsIn`
Recommendation	Consider marking the functions above as `external`.
Resolution	ACKNOWLEDGED

Issue #10	Lack of event for `setSwapFeeReward`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Consider adding an event for the function.
Resolution	ACKNOWLEDGED

Issue #11	**The `addLiquidity` function does not properly support tokens with a fee on transfer (present in Uniswap as well)**
Severity	INFORMATIONAL
Description	The current `addLiquidity` function always assumes tokens are not reflective. If one adds liquidity with one of the tokens being a transfer-tax token, this results in tokens being wasted to the pair. You can read more about this issue here: https://github.com/Uniswap/v2-periphery/issues/106
Recommendation	Consider implementing the approach described within issue 106 of v2-periphery.
Resolution	ACKNOWLEDGED

2.3 PandoraFactory

The `PandoraFactory` contract is the core management contract for the Pandora AMM. It keeps track of all Pandora Pairs and allows users to create new ones. Any Pandora Pair created through the verified factory can be considered as verified as well, since the pair is deployed by the verified factory. Users can double check if a pair was deployed through the PandoraFactory by calling `getPair` with the two tokens.

The main change made compared to the standard Uniswap-V2 implementation is that Pandora pairs have a configurable swap fee. By default, the fee is set to 17 basis points (0.17%) but governance can configure it down to 1 basis point (0.01%) and all the way up to 1000 basis points (10%). LP holders receive 3/5th of these fees while 2/5th is sent to the fee address owned by governance.

As each LP contract can have its own swap fee, only the factory's `feeToSetter` can call the `setSwapFee` function.

Paladin has already taken the liberty to validate that the following pairs were in fact deployed by the factory:

- PANDORA/WASTR - `0x5e8a60839dC6F9C7595E0d9519d4bdB947cEb7A6`
- USDT/USDC - `0xcECbf254c22a5d5e5d75a215A4403A5B4dC1dA5A`
- USDC/WASTR - `0x3683d79a8Af26A56822C48a4eD1af80d51eB8399`
- USDC/PANDORA - `0x3ED26D25d047B0d01E181c1a0E955e00aac9A707`

2.3.1 Privileged Functions

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

- `setFeeTo`
- `setFeeToSetter`
- `setSwapFee`

The following privileged functions can be found in deployed pairs and are only callable by the factory:

- `setSwapFee`

2.3.2 Issues & Recommendations

Issue #12	Swap fees can be modified without any delay
Severity	LOW SEVERITY
Description	Although there is a maximum cap of 10% of swap fees, it is possible for the factory's `feeToSetter` address to modify the fees of any LP pair contract. At the point of this review, the following wallet is the `feeToSetter`: https://blockscout.com/astar/address/0xdf24c88016f9D8933fBC4D863B65B5c3dc87F048/transactions If the private key of this wallet is ever compromised, the hacker might raise the fees to the maximum of 10% and withdraw all received fees as hacking profit.
Recommendation	Consider setting a timelock with a reasonable duration as the `feeToSetter`. For further protection, consider setting a multisig as the owner of the timelock.
Resolution	ACKNOWLEDGED

Issue #13	**Pairs without supply but with a partial reserve might crash the frontend if the user wants to swap on this pair (present in most frontends)**
Severity	INFORMATIONAL
Description	A malicious DoS attack we have witnessed in practice is when a project wants to go live through a presale, people can instantiate the pair while there are no tokens yet. The malicious party will then send some of the counterparty token to this pair so it has a partial balance (e.g. 0.1 ASTR and 0 tokens). When `sync()` is then called, the pairs' reserves are updated to account for this balance. Due to a division by zero exception, many frontends cannot properly account for this state and will go through a blank page, preventing the original project from adding liquidity through the frontend.
Recommendation	Consider checking whether this is present in the frontend and adding a division by zero handler.
Resolution	ACKNOWLEDGED

Issue #14	**Lack of events for `setSwapFee`, `setFeeTo` and `setFeeToSetter`**
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for these functions.
Resolution	ACKNOWLEDGED

Issue #15	`permit` can be frontrun and cause denial of service
Severity	INFORMATIONAL
Description	Currently, if `permit` is executed twice, the second execution will be reverted. It is thus in theory possible for a bot to pick up permit transactions in the mempool and execute them before a contract can. The implications of this issue is that a bad actor could prevent a user from removing liquidity with a permit through the router. It is a denial-of-service attack which is present in all AMMs but which we have yet to witness being used since there is no profit from it.
Recommendation	Consider this issue if there are ever complaints by users that their `removeLiquidityWithPermit` transactions are failing. It could be the case that someone is using this vector against them. We do not recommend changing this behavior since it would cause a lot of extra work modifying the frontend to account for new permit behavior. This issue is also present in Uniswap after all.
Resolution	RESOLVED The client has indicated that they understand this issue and will monitor whether it ever presents itself.

2.4 Masterpandora

`Masterpandora` is a staking contract which allows the user to deposit various LP tokens and receive Pandora Tokens as a reward. It should be noted that, unlike most of other MasterChefs, it does not mint any token to the developers. Additionally, this contract does not have any deposit fees.

A user can set a referrer that will earn up to 10% of the user's rewards. This value is currently set at 1%.

The `Masterpandora` contract does not allow smart contracts to deposit into its pools, unless `whitelistAll` is enabled. The governance can however allow specific addresses to deposit into `Masterpandora`.

Finally, the contract interestingly differs from a traditional MasterChef in that users can stake up to three whitelisted NFTs per pool. Each of these NFTs will then boost the rewards of that pool for the user.

It should be noted that the NFTs that will be staked and the controller that returns the boost percentage are out of scope for this audit as they are currently not enabled.

2.4.1 Privileged Functions

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

- `add`
- `set`
- `updateEmissionRate`
- `setNftController`
- `setNftBoostRate`
- `setpandoraReferral`
- `flipWhitelistAll`
- `setReferralCommissionRate`
- `setWhitelist`
- `disableWhitelist`
- `renounceOwnership`
- `transferOwnership`

2.4.2 Issues & Recommendations

Issue #16	Deposits do not support tokens with a fee on transfer
Severity	HIGH SEVERITY
Description	Within the `deposit` function, there is no logic that supports tokens with a fee on transfer. Therefore, during a deposit, the Masterchef will receive less tokens than the user will get credited. This leads to an exploitation issue, where a malicious user can drain the whole pool, which results in absurd reward minting.
Recommendation	Consider adding logic for tokens with a fee on transfer: `uint256 balanceBefore = pool.lpToken.balanceOf(address(this)); pool.lpToken.safeTransferFrom(msg.sender, address(this), _amount); _amount = pool.lpToken.balanceOf(address(this)).sub(balanceBefore);`
Resolution	RESOLVED The client has indicated they will never add such tokens to the MasterChef.

Issue #17	Users could lose their previously deposited NFT if they redeposit in the same slot
Severity	MEDIUM SEVERITY
Description	Users can deposit up to three NFTs into a specific pool to boost the pools' earnings. Currently, `depositNFT` does not check if the user already has an NFT in the selected slot before adding it to that slot. This means that users who wrongly send 2 NFTs to the same slot will lose the first one that was sent into that slot.
Recommendation	Consider checking that no NFT is already stored within the requested deposit slot.
Resolution	PARTIALLY RESOLVED The client has indicated that they will carefully design their frontend to make sure this does not accidentally happen.

Issue #18	Users may lose rewards when the maximum supply is reached
Severity	LOW SEVERITY
Description	Currently the `updatePool` function does not check if the tokens were minted and still increments the `pool.accPandoraPerShare`. If the maximum supply is ever reached, tokens will stop being minted but `mint` will not revert. Although new tokens were not minted in this situation, the `safePandoraTransfer` function will keep transferring the previously minted tokens that are still in the contract, resulting in a loss for users that have not claimed their rewards yet.
Recommendation	Consider not incrementing the `pool.accPandoraPerShare` if the mint was not successful.
Resolution	PARTIALLY RESOLVED The client has indicated they will disable the emissions before the max supply is reached to completely avoid this issue.

Issue #19	**The `updateEmissionRate` function and NFT boosting mechanism have no maximum safeguard**
Severity	LOW SEVERITY
Description	The function to update rewards currently has no safeguard on its maximum value. Projects sometimes accidentally update their emission rate to a severely high number either by accident or with malicious intent. By having a maximum value, the code itself enforces the reward rate to always be within a reasonable range, preventing mistakes which cannot be reverted once they are made. This might furthermore boost investor confidence as they know that the governance cannot suddenly set the emission rate to a very high value. In addition, the `getBoost` function should also have a maximum safeguard as a malicious governance could change the boosting rate to a very high value and mint a large supply.

Recommendation	Consider adding a `MAX_EMISSION_RATE` variable and setting it to a reasonable value.
	
	`require(_pandoraPerBlock <= MAX_EMISSION_RATE, "Too high");`
	
	Consider capping the NFT boost to a maximum boost rate within the `getBoost` method.
	
	Lines 1518-1519
	`if (boost > MAX_BOOST_RATE)`
	`  boost = MAX_BOOST_RATE;`

Resolution	ACKNOWLEDGED

Issue #20	The `updateEmissionRate` function could run out of gas
Severity	LOW SEVERITY
Description	Currently, the `updateEmissionRate` could run out of gas because it calls `massUpdatePools`. Unlike the `add` and `set` functions, this call is not optional which could cause the issue.
Recommendation	Consider making the call to `massUpdatePools` optional similar to how this is done in `add` and `set`.
Resolution	ACKNOWLEDGED

Issue #21	`msg.sender` is unnecessarily cast to `address(msg.sender)`
Severity	INFORMATIONAL
Description	`msg.sender` is cast to `address(msg.sender)` throughout the contract when used with `pool.lpToken.safeTransfer()`. This is unnecessary.
Recommendation	Consider replacing all occurrences of `address(msg.sender)` with `msg.sender`.
Resolution	ACKNOWLEDGED

Issue #22	**`BONUS_MULTIPLIER` is not actively used**
Severity	INFORMATIONAL
Description	The constant variable `BONUS_MULTIPLIER` does not contain any extra information since it is constant and cannot be changed. This variable is therefore redundant and might mislead third-party reviewers into thinking that there is such a thing as bonus multipliers within the contract logic.
Recommendation	Consider removing the `BONUS_MULTIPLIER` variable.
Resolution	ACKNOWLEDGED

Issue #23	**Pool uses the contract balance to figure out the total deposits**
Severity	INFORMATIONAL
Description	As with pretty much all Masterchefs and staking contracts, the total number of tokens in the contract is used to determine the total number of deposits. This can cause dilution of rewards when people accidentally send tokens to the masterchef.
Recommendation	Consider adding an lpSupply variable to the PoolInfo that keeps track of the total deposits.
Resolution	ACKNOWLEDGED

Issue #24	Lack of validation
Severity	INFORMATIONAL
Description	The contract contains functions with parameters which are not properly validated. Having unvalidated parameters could allow the governance or users to provide variable values which are unexpected and incorrect. This could cause side-effects or exploits in other parts of the codebase. Consider validating the following function parameters. `constructor`: - `_pandoraPerBlock` should be capped - `_startBlock` should be in the future
Recommendation	Consider validating the function parameters mentioned above.
Resolution	ACKNOWLEDGED

Issue #25	`pandora` and `startBlock` can be made `immutable`
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the immutable keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
Recommendation	Consider making the variables explicitly `immutable`.
Resolution	ACKNOWLEDGED

Issue #26	SafeMath should be used within the `getBoost` function
Severity	INFORMATIONAL
Description	Within the `getBoost` function, boosts are summed up but not using SafeMath — this could overflow.
Recommendation	Consider using SafeMath.
Resolution	ACKNOWLEDGED

Issue #27	proxy can be made constant
Severity	INFORMATIONAL
Description	Variables that are never modified can be indicated as such with the constant keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
Recommendation	Consider making the aforementioned variables explicitly constant.
Resolution	ACKNOWLEDGED

Issue #28	Various functions can be made external
Severity	INFORMATIONAL
Description	Functions that are not used within the contract but only externally can be marked as such with the external keyword. 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. The following functions can be made external: - `getSlots` - `getTokenIds` - `add` - `set` - `depositNFT` - `withdrawNFT` - `deposit` - `withdraw` - `emergencyWithdraw` - `updateEmissionRate` - `setNftController` - `setNftBoostRate` - `setpandoraReferral` - `flipWhitelistAll` - `setReferralCommissionRate`
Recommendation	Consider marking the functions above as external.
Resolution	ACKNOWLEDGED

Issue #29	Lack of events for various functions
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications. The following functions lack an event: - `add` - `set` - `depositNFT` - `withdrawNFT` - `updatePool` - `setpandoraReferral` - `flipWhitelistAll` - `setReferralCommissionRate`
Recommendation	Add events for the above functions.
Resolution	ACKNOWLEDGED

Issue #30	Typographical errors
Severity	INFORMATIONAL

Description	The contract contains a number of typographic mistakes which we have consolidated below in a single issue in an effort to keep the report size reasonable.
	
	L1473
	`Proxypandora public proxy = Proxypandora(0x939fEE93Fd10925A64188dd21E70f9b2E82C0453);`
	
	Casting the address to `Proxypandora` is unnecessary.
	
	L1491
	`totalAllocPoint = 0;`
	
	Setting the `totalAllocPoint` to 0 is unnecessary as uint are already set at 0 by default within Solidity.
	
	L1498
	`require(poolExistence[_lpToken] == false, "nonDuplicated: duplicated");`
	
	The `== false` is unnecessary and increases gas cost. Consider using `!poolExistence[_lpToken]`.

Recommendation	Consider fixing the typographical errors.
Resolution	ACKNOWLEDGED

2.5 ProxyPandora

`ProxyPandora` is an upgradeable contract which stores the `PANDORA` tokens for user rewards. The governance can add whitelisted accounts which can withdraw these `PANDORA` tokens. The governance can also call `drainBEP20Token` to take out all tokens other than `PANDORA` sent to this contract by accident. This is not a user risk as no tokens should be in this contract.

Note that within the audited codebase, `PANDORA` is set to a non-existent test contract. The sole change in the deployed implementation is that this address was set to the actual Pandora token.

2.5.1 Privileged Functions

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

- `drainBEP20Token`
- `setWhitelist`
- `disableWhitelist`

The following privileged functions can called by any whitelisted account:

- `safeMeerkatTransfer`

2.5.2 Issues & Recommendations

Issue #31	**Governance privilege: An upgradeable contract allows the contract to change the functionality at any point in time to a contract which potentially drains the tokens in this contract**
Severity	MEDIUM SEVERITY
Description	`ProxyPandora` is an upgradeable proxy, which means that the implementation can be changed by the governance. At any point in time, the proxy admin can therefore add new functionality to take out all Pandora tokens in the contract. This is also possible by whitelisting their wallet.
Recommendation	Consider renouncing the `admin` privilege and ownership of the contract. If this governance control is desired, consider putting the contract behind both a timelock and multisig.
Resolution	ACKNOWLEDGED

Issue #32	**Disabling the whitelist is highly discouraged as anyone can call `safeMeerkatTransfer` to take out all the Pandora tokens at that point**
Severity	LOW SEVERITY
Description	The contract uses the OpenZeppelin `WhitelistUpgradeable` implementation to store a list of users that can withdraw . However, if the whitelist is ever disabled, this would mean that anyone can call the `safeMeerkatTransfer` function and drain the contract of Pandora tokens.
Recommendation	Consider upgrading ProxyPandora to only let `safeMeerkatTransfer` be called by the `Masterpandora`. One must be careful with upgrades as storage collisions should always be checked.
Resolution	ACKNOWLEDGED

Issue #33	**`_disable` is private**
Severity	LOW SEVERITY
Description	Important variables that third-parties might want to inspect should be marked as public so that these third-parties can easily inspect them through the explorer, web3 and derivative contracts.
Recommendation	Consider marking the variable as public.
Resolution	ACKNOWLEDGED

Issue #34	Typographical error
Severity	INFORMATIONAL
Description	The contract still mentions "meerkats" instead of "pandora tokens".
Recommendation	Consider fixing the typographical error.
Resolution	ACKNOWLEDGED

Issue #35	Unused functionality: SafeERC20
Severity	INFORMATIONAL
Location	Line 957 `using SafeERC20 for IERC20;`
Description	Files that are imported in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length unnecessarily.
Recommendation	Consider using `safeTransfer` instead of `transfer`, as `SafeERC20` is already included.
Resolution	ACKNOWLEDGED

Issue #36	Lack of events for `drainBEP20Token`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the function.
Resolution	ACKNOWLEDGED

2.6 Referral

`Referral` is a utility contract used by `Masterpandora` to store the users that referred other users. The main functionality is to let `Masterpandora` easily retrieve who a user was referred by when they harvest tokens.

This referrer will then earn the referral commission rate (presently set to 1% but changeable up to 10%) of the harvest value as a bonus.

2.6.1 Privileged Functions

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

- `updateOperator`
- `renounceOwnership`
- `transferOwnership`
- `recordReferral`

2.6.2 Issues & Recommendations

Issue #37	**The referral addresses for users without referral records can be arbitrarily set by operator**
Severity	LOW SEVERITY
Description	`recordReferral` is used by the operator role to record the referrer address of a referred user. As the `operator` role can be held by multiple addresses, the owner of `Referral` can add an address other than MasterChef into the role. This address can then set the referrer of user addresses to an arbitrary address, thus earning the commission fees of those users' harvests. This allows taking up to 10% of the user harvest value as a bonus.
Recommendation	Consider making the Masterchef the sole operator and renounce ownership afterwards. In this case, there is no way for the operators to arbitrarily update the referrals for users. Alternatively, remove the mapping and only use a single address as the variable for operator.
Resolution	ACKNOWLEDGED

Issue #38	`getReferrer` can be made `external`
Severity	INFORMATIONAL
Description	Functions that are not used within the contract but only externally can be marked as such with the `external` keyword. 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 marking the function as `external`.
Resolution	ACKNOWLEDGED

Issue #39	Unused import: SafeERC20 and IERC20
Severity	INFORMATIONAL
Location	Line 542 `using SafeERC20 for IERC20;`
Description	Files that are imported in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length unnecessarily.
Recommendation	Consider removing the import to keep the contract short and simple.
Resolution	ACKNOWLEDGED


PALADIN
BLOCKCHAIN SECURITY