CERTIK

Preliminary Comments

PolkaEx

Sept 23rd, 2021

Table of Contents

Summary

This report has been prepared for PolkaEx to discover issues and vulnerabilities in the source code of the PolkaEx project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	PolkaEx
Description	A Liquidity Pool
Platform	Ethereum
Language	Solidity
Codebase	https://polkaex.io/solidity/Pkex_Solidity_2021_07_17.zip https://github.com/PolkaEX/PolkaEx/
Commit	48b0ce61eae92aab0335955f5a5b5ce9484c8df1

Audit Summary

Delivery Date	Sept 23, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	5	0	0	0	5	0
Medium	0	0	0	0	0	0
Minor	16	0	0	4	0	12
Informational	13	0	0	0	0	13
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
PLP	PrivateLunchpad.sol	d93d00ece2d8d455c7c9ff9497911c196589390db569585b7fffd663a14a53f2
PLE	PublicLunchpad.sol	e2ff890cb53b2fc4d10e88c8d1a982d7d7e5d7ee3f0cc80355603e16caaebd93
RPE	Reward.sol	6954841efc174c8a7ba3ba2326216d22daa2754b7a1bd6c29c965e33897e6806
PEP	claim_v5_pkex.sol	21c4fb7d8813b4bed55b8ecf3a5cd67bbc50efa52fbd46922fded96cb00f28b4
PCK	pkex-token.sol	27c84057ab6904ace8a479f575799b0cd5e2bf10b1d9fbd65e629934a8a87326

To bridge the trust gap between owner and users, the owner needs to express a sincere attitude regarding the consideration of the administrator team's anonymity.

The owner of `PkexTokenClaim` has the responsibility to notify users with the following capability:

- Add vestings by batches through `addMutiVestingInOneAddress()` and `addMutiVestingInMutiAddress()`
- Remove vestings through `removeVesting()`

The owner of `PrivateLaunchpad` has the responsibility to notify users with the following capability:

- Add user to a whitelist that only listed members may buy and withdraw tokens

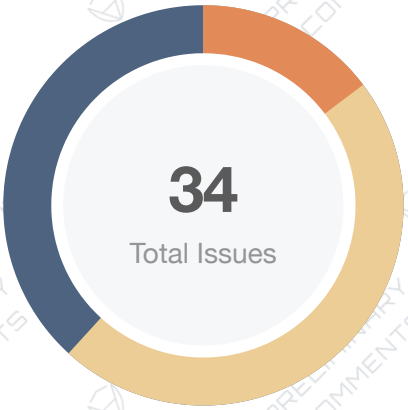
Pausers of `PrivateLaunchpad and` PublicLaunchpad` has the responsibility to notify users with the following capability:

- Pause/unpause the contract to disable/enable transactions

The reward distributor has the responsibility to notify users with the following capability:

- Add rewards and new pools through `notifyRewardAmound()`
- Transfer tokens other than stake or reward through `recoverERC20()`
- Set withdraw cooldown period through `setUnstakePeriod()`

Findings



Critical	0 (0.00%)
Major	5 (14.71%)
Medium	0 (0.00%)
Minor	16 (47.06%)
Informational	13 (38.24%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
PCK-01	Lack of Input Validation	Volatile Code	Minor	Resolved
PCK-02	Delegation Initialization	Logical Issue	Informational	Resolved
PCK-03	Unlocked and Inconsistent Compiler Version	Language Specific	Minor	Resolved
PEP-01	Inaccurate Error Message	Inconsistency	Informational	Resolved
PEP-02	Lack of Error Messages	Control Flow	Minor	Resolved
PEP-03	Unlocked and Inconsistent Compiler Version	Language Specific	Minor	Resolved
PEP-04	Logic Issues of `addVesting()`	Logical Issue	Major	Partially Resolved
PEP-05	Privileged Ownership	**Centralization / Privilege**	**Major**	Partially Resolved
PEP-06	Improper Usage of `public` and `external` Types	Gas Optimization	Informational	Resolved
PEP-07	Typo in Function Name	Language Specific	Informational	Resolved
PEP-08	Third Party Dependencies	Volatile Code	Minor	Acknowledged
PLE-01	Set `constant` to Variables	Gas Optimization	Informational	Resolved
PLE-02	Logic Issue of `buy()`	Logical Issue	Minor	Resolved
PLE-03	Lack of Input Validation	Volatile Code	Minor	Resolved

ID	Title	Category	Severity	Status
PLE-04	Typos in Contract Name	Coding Style	Informational	Resolved
PLE-05	Unlocked and Inconsistent Compiler Version	Language Specific	Minor	Resolved
PLE-06	Privileged Ownership	**Centralization / Privilege**	**Major**	Partially Resolved
PLE-07	Improper Usage of `public` and `external` Types	Gas Optimization	Informational	Resolved
PLE-08	Third Party Dependencies	Volatile Code	Minor	Acknowledged
PLP-01	Set `constant` to Variables	Gas Optimization	Informational	Resolved
PLP-02	Lack of Input Validation	Volatile Code	Minor	Resolved
PLP-03	Typos in Contract Name	Coding Style	Informational	Resolved
PLP-04	Unlocked and Inconsistent Compiler Version	Language Specific	Minor	Resolved
PLP-05	Privileged Ownership	**Centralization / Privilege**	**Major**	Partially Resolved
PLP-06	Improper Usage of `public` and `external` Types	Gas Optimization	Informational	Resolved
PLP-07	Typo in Function Name	Language Specific	Informational	Resolved
PLP-08	Third Party Dependencies	Volatile Code	Minor	Acknowledged
RPE-01	Privileged Ownership	**Centralization / Privilege**	**Major**	Partially Resolved
RPE-02	Lack of Input Validation	Volatile Code	Minor	Resolved
RPE-03	Missing Emit Events	Volatile Code	Informational	Resolved
RPE-04	Improper Usage of `public` and `external` Types	Gas Optimization	Informational	Resolved
RPE-05	Logical Issue of `getTotalRewarded`	Logical Issue	Minor	Resolved
RPE-06	Third Party Dependencies	Volatile Code	Minor	Acknowledged

ID	Title	Category	Severity	Status
RPE-07	Safetransfer for Reward	Logical Issue	Minor	Resolved

PCK-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	pkex-token.sol: 271	Resolved

Description

The assigned value to `account` should be verified as a non-zero value to prevent being mistakenly assigned as `address(0)` in the `constructor()` function.

Recommendation

We advise the client to check that the address is not zero in `constructor()` like as follows.

```
    require(account != address(0), "Zero address");
```

Alleviation

The client added a validator as we had suggested and resolved this issue.

PCK-02 | Delegation Initialization

Category	Severity	Location	Status
Logical Issue	Informational	pkex-token.sol: 271~274	Resolved

Description

Delegates are not assigned to any account in `constructor()`. We would like to enquire on how the voting system is initialized.

Alleviation

The client removed the vote functionalities and this is no longer an issue.

PCK-03 | Unlocked and Inconsistent Compiler Version

Category	Severity	Location	Status
Language Specific	Minor	pkex-token.sol: 18	Resolved

Description

The contract has unlocked and inconsistent compiler versions. An unlocked compiler version in the contract's source code permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

It is general practice to alternatively lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

The client locked the compiler versions to `0.7.0`.

PEP-01 | Inaccurate Error Message

Category	Severity	Location	Status
Inconsistency	Informational	claim_v5_pkex.sol: 259	Resolved

Description

The current error message may not give accurate feedback for contract construction as the token address is not bound to be matic tokens.

Recommendation

We advise the client to review their functionalities and change the require statements to make them consistent with function logic.

Alleviation

The client changed the error message to "Invalid PKEX token address".

PEP-02 | Lack of Error Messages

Category	Severity	Location	Status
Control Flow	Minor	claim_v5_pkex.sol	Resolved

Description

Most `require` statements in libraries in the linked file have the error message omitted.

Recommendation

We advise the client to add error messages to all `require` statements, as this pattern helps the user identify all the relevant procedural requirements.

Alleviation

The client added missing error messages to the validators in `PkexTokenClaim`.

PEP-03 | Unlocked and Inconsistent Compiler Version

Category	Severity	Location	Status
Language Specific	Minor	claim_v5_pkex.sol: 18	Resolved

Description

The contract has unlocked and inconsistent compiler versions. An unlocked compiler version in the contract's source code permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

It is general practice to alternatively lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

The client locked the compiler versions to `0.7.0`.

PEP-04 | Logic Issues of `addVesting()`

Category	Severity	Location	Status
Logical Issue	Major	claim_v5_pkex.sol: 336~338	Partially Resolved

Description

The function is declared `private` which contradicts its purpose. Besides, `Tokenamount` is not properly updated.

Recommendation

We advise the client to review its functionality and fix the implementation.

Alleviation

The client changed the visibility to `external` and added `Tokenamount` attribute in the function.

PEP-05 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	Major	claim_v5_pkex.sol	Partially Resolved

Description

To bridge the gap in trust between owner and users, the owner needs to express a sincere attitude regarding the considerations of the administrator team's anonymity. Owner of `PkexTokenClaim` has the responsibility to notify users with the following capability:

- Add vestings by batches through `addMutiVestingInOneAddress()` and `addMutiVestingInMutiAddress()`
- Remove vestings through `removeVesting()`

Recommendation

We advise the client to carefully oversee the manager account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, e.g. Multisignature wallets. Here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Client]: After doing a few IDOs this will be changed to a DAO in due time. More information would be available on our website.

PEP-06 | Improper Usage of `public` and `external` Types

Category	Severity	Location	Status
Gas Optimization	Informational	claim_v5_pkex.sol	Resolved

Description

`public` functions that are never called by the contract could be declared `external` . When the inputs are arrays external functions are more efficient than “public” functions.

Examples:

- `retrieveExcessTokens()`
- `mutiRelease()`
- `addMutiVestingInMutiAddress()`
- `addMutiVestingInOneAddress()`
- `removeVesting()`
- `myNextRelease()`
- `releaseAll()`
- `myVestings()`
- `token()`

Recommendation

We advise the client to use the `external` attribute for functions never called from the contract.

Alleviation

The client revised the code as we had suggested and resolved this issue.

PEP-07 | Typo in Function Name

Category	Severity	Location	Status
Language Specific	Informational	claim_v5_pkex.sol	Resolved

Description

"Multi" is misspelt as "Muti" in all functions involving batched transactions.

Recommendation

We advise the client to adopt the more accepted spelling in those function names.

Alleviation

The client fixed all the typos we mentioned.

PEP-08 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	claim_v5_pkex.sol	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party protocols, including:

- `pkexToken` that interacts with `PkexTokenClaim`
- `USDC` that interacts with `PrivateLunchpad` and `PublicLunchPad`
- `Token` that interacts with `PrivateLunchpad` and `PublicLunchPad`
- `rewardsToken` that interacts with `StakingRewardsV2`
- `stakingToken` that interacts with `StakingRewardsV2`

The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of PolkaEx requires interaction with the aforementioned protocols. We encourage the team to constantly monitor the status of 3rd parties to mitigate side effects when unexpected activities are observed.

PLE-01 | Set `constant` to Variables

Category	Severity	Location	Status
Gas Optimization	Informational	PrivateLunchpad.sol	Resolved

Description

Some variables are not changed throughout the smart contract and can be set as `constant`:

- `isWhite`
- `rate`
- `lockRate`
- `USDCap`
- `personCap`
- `personMinCap`
- `maxGasPrice`
- `TokenCap`

Recommendation

We advise the client to set them as constant variables and adopt the `UPPERCASE` naming convention.

Alleviation

The client set them as constants as we had suggested.

PLE-02 | Logic Issue of `buy()`

Category	Severity	Location	Status
Logical Issue	Minor	PublicLunchpad.sol: 719	Resolved

Description

According to the logic of the `buy()` function, the unlocked amount of `msg.sender` should not be `tokenLocked` but the difference of `tokenBought` and `tokenLocked`. Even though the result would be the same given the current `lockRate`, the current implementation is unsound and may introduce unnecessary rounding errors.

Recommendation

We advise the client to modify as follows:

```
719     userUnLocked[msg.sender] =
userUnLocked[msg.sender].add(tokenBought).sub(tokenLocked);
```

Alleviation

The client revised the code as we had suggested and resolved this issue.

PLE-03 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	PublicLunchpad.sol: 685~700	Resolved

Description

Input arguments of `constructor()` are not validated and could be volatile.

Recommendation

We advise the client to add a argument validator to check if the value of `_USDC`, `_Token`, `_fundWallet` and `TokenWallets` is set as address(0). Besides, consider add validator to ensure `startTime < endTime`.

Alleviation

The client added validators for all the arguments we had mentioned.

PLE-04 | Typos in Contract Name

Category	Severity	Location	Status
Coding Style	Informational	PublicLunchpad.sol: 731	Resolved

Description

The contract name `PirvateLunchpad` should be `PrivateLunchpad`. Besides, "LaunchPad" instead of "LunchPad" seems to be the correct word judging from website sources.

Recommendation

We recommend correcting this typo.

Alleviation

The client fixed all the typos we mentioned.

PLE-05 | Unlocked and Inconsistent Compiler Version

Category	Severity	Location	Status
Language Specific	Minor	PublicLunchpad.sol: 18, 3	Resolved

Description

The contract has unlocked and inconsistent compiler versions. An unlocked compiler version in the contract's source code permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

It is general practice to alternatively lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

The client locked the compiler versions to `0.7.0`.

PLE-06 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	Major	PublicLunchpad.sol	Partially Resolved

Description

To bridge the trust gap between owner and users, the owner needs to express a sincere attitude regarding the considerations of the administrator team's anonymity. The owner of `PrivateLunchpad` has the responsibility to notify users with the following capability:

- Add user to a whitelist that only listed members may buy and withdraw tokens

Pausers of `PrivateLunchpad` and `PublicLunchpad` has the responsibility to notify users with the following capability:

- Pause/unpause the contract to disable/enable transactions

Recommendation

We advise the client to carefully oversee the manager account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, e.g. Multisignature wallets. Here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Client]: After doing a few IDOs this will be changed to a DAO in due time. More information would be available on our website.

PLE-07 | Improper Usage of `public` and `external` Types

Category	Severity	Location	Status
Gas Optimization	Informational	PublicLunchpad.sol	Resolved

Description

`public` functions that are never called by the contract could be declared `external` . When the inputs are arrays external functions are more efficient than `public` functions.

Examples:

- `addMutiWhitelist()`
- `withdraw()`
- `buyToken()`

Recommendation

We advise the client to use the `external` attribute for functions never called from the contract.

Alleviation

The client changed the function visibilities as we had suggested and resolved this issue.

PLE-08 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	PublicLunchpad.sol	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party protocols, including:

- `pkexToken` that interacts with `PkexTokenClaim`
- `USDC` that interacts with `PrivateLunchpad` and `PublicLunchPad`
- `Token` that interacts with `PrivateLunchpad` and `PublicLunchPad`
- `rewardsToken` that interacts with `StakingRewardsV2`
- `stakingToken` that interacts with `StakingRewardsV2`

The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of PolkaEx requires interaction with the aforementioned protocols. We encourage the team to constantly monitor the status of 3rd parties to mitigate side effects when unexpected activities are observed.

PLP-01 | Set `constant` to Variables

Category	Severity	Location	Status
Gas Optimization	Informational	PublicLunchpad.sol	Resolved

Description

Some variables are not changed throughout the smart contract and can be set as `constant`:

- `isWhite`
- `rate`
- `lockRate`
- `USDCap`
- `personCap`
- `personMinCap`
- `maxGasPrice`
- `TokenCap`

Recommendation

We advise the client to set them as constant variables and adopt the `UPPERCASE` naming convention.

Alleviation

The client set them as constants as we had suggested.

PLP-02 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	PrivateLunchpad.sol: 776~791	Resolved

Description

Input arguments of `constructor()` are not validated and could be volatile.

Recommendation

We advise the client to add a argument validator to check if the value of `_USDC`, `_Token`, `_fundWallet` and `TokenWallets` is set as address(0). Besides, consider add validator to ensure `startTime < endTime`.

Alleviation

The client added validators for all the arguments we had mentioned.

PLP-03 | Typos in Contract Name

Category	Severity	Location	Status
Coding Style	Informational	PrivateLunchpad.sol: 731	Resolved

Description

The contract name `PirvateLunchpad` should be `PrivateLunchpad`. Besides, "LaunchPad" instead of "LunchPad" seems to be the correct word judging from website sources.

Recommendation

We recommend correcting this typo.

Alleviation

The client fixed all the typos we mentioned.

PLP-04 | Unlocked and Inconsistent Compiler Version

Category	Severity	Location	Status
Language Specific	Minor	PrivateLunchpad.sol: 18	Resolved

Description

The contract has unlocked and inconsistent compiler versions. An unlocked compiler version in the contract's source code permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

It is general practice to alternatively lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

The client locked the compiler versions to `0.7.0`.

PLP-05 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	Major	PrivateLunchpad.sol	Partially Resolved

Description

To bridge the trust gap between owner and users, the owner needs to express a sincere attitude regarding the considerations of the administrator team's anonymity. The owner of `PrivateLunchpad` has the responsibility to notify users with the following capability:

- Add user to a whitelist that only listed members may buy and withdraw tokens

Pausers of `PrivateLunchpad` and `PublicLunchpad` has the responsibility to notify users with the following capability:

- Pause/unpause the contract to disable/enable transactions

Recommendation

We advise the client to carefully oversee the manager account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, e.g. Multisignature wallets. Here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Client]: After doing a few IDOs this will be changed to a DAO in due time. More information would be available on our website.

PLP-06 | Improper Usage of `public` and `external` Types

Category	Severity	Location	Status
Gas Optimization	Informational	PrivateLunchpad.sol	Resolved

Description

`public` functions that are never called by the contract could be declared `external` . When the inputs are arrays external functions are more efficient than `public` functions.

Examples:

- `addMutiWhitelist()`
- `withdraw()`
- `buyToken()`

Recommendation

We advise the client to use the `external` attribute for functions never called from the contract.

Alleviation

The client changed the function visibilities as we had suggested and resolved this issue.

PLP-07 | Typo in Function Name

Category	Severity	Location	Status
Language Specific	Informational	PrivateLunchpad.sol	Resolved

Description

"Multi" is misspelt as "Muti" in all functions involving batched transactions.

Recommendation

We advise the client to adopt the more accepted spelling in those function names.

Alleviation

The client fixed all the typos we mentioned.

PLP-08 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	PrivateLunchpad.sol	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party protocols, including:

- `pkexToken` that interacts with `PkexTokenClaim`
- `USDC` that interacts with `PrivateLunchpad` and `PublicLunchPad`
- `Token` that interacts with `PrivateLunchpad` and `PublicLunchPad`
- `rewardsToken` that interacts with `StakingRewardsV2`
- `stakingToken` that interacts with `StakingRewardsV2`

The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of PolkaEx requires interaction with the aforementioned protocols. We encourage the team to constantly monitor the status of 3rd parties to mitigate side effects when unexpected activities are observed.

RPE-01 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	Major	Reward.sol	Partially Resolved

Description

To bridge the trust gap between owner and users, the owner needs to express a sincere attitude regarding the considerations of the administrator team's anonymity. The reward distributor has the responsibility to notify users with the following capability:

- Add rewards and new pools through `notifyRewardAmount()`
- Transfer tokens other than stake or reward through `recoverERC20()`
- Set withdraw cooldown period through `setUnstakePeriod()`

Recommendation

We advise the client to carefully oversee the manager account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, e.g. Multisignature wallets. Here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Client]: After doing a few IDOs this will be changed to a DAO in due time. More information would be available on our website.

RPE-02 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	Reward.sol: 863~873	Resolved

Description

The input address arguments are not validated as non-zero values.

Recommendation

We advise the client to add argument validators to check if any value of `_rewardsDistribution`, `_rewardsToken`, and `_stakingToken` is set as address(0).

Alleviation

The client added argument validators as we had suggested and resolved this issue.

RPE-03 | Missing Emit Events

Category	Severity	Location	Status
Volatile Code	Informational	Reward.sol: 1095~1097	Resolved

Description

The `setUnstakePeriod()` function affects the status of a sensitive variable and should be able to emit events as notifications to customers.

Recommendation

We advise the client to add an event for sensitive actions, and emit them in the function.

Alleviation

The client changed added the event as we had suggested and resolved this issue.

RPE-04 | Improper Usage of `public` and `external` Types

Category	Severity	Location	Status
Gas Optimization	Informational	Reward.sol	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays external functions are more efficient than “public” functions.

Examples:

- `setUnstakePeriod()`
- `myStakeInfo()`

Recommendation

We advise the client to use `external` attribute for functions never called from the contract.

Alleviation

The client revised these attributes and fixed this issue.

RPE-05 | Logical Issue of `getTotalRewarded`

Category	Severity	Location	Status
Logical Issue	Minor	Reward.sol: 952	Resolved

Description

When `pool.from > block.timestamp` for the current pool, the same condition will hold for all subsequent pools and the for loop should end.

Recommendation

We advise the client to `break` instead of `continue` the for loop.

Alleviation

The client revised the code as we had suggested and resolved this issue.

RPE-06 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	Reward.sol	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party protocols, including:

- `pkexToken` that interacts with `PkexTokenClaim`
- `USDC` that interacts with `PrivateLunchpad` and `PublicLunchPad`
- `Token` that interacts with `PrivateLunchpad` and `PublicLunchPad`
- `rewardsToken` that interacts with `StakingRewardsV2`
- `stakingToken` that interacts with `StakingRewardsV2`

The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of PolkaEx requires interaction with the aforementioned protocols. We encourage the team to constantly monitor the status of 3rd parties to mitigate side effects when unexpected activities are observed.

RPE-07 | Safetransfer for Reward

Category	Severity	Location	Status
Logical Issue	Minor	Reward.sol: 1011	Resolved

Description

The reward may be stuck in the contract if the user's claim exceeds the token balance available as the `safeTransfer()` would revert in that situation. The user is still entitled to whatever reward tokens are left in the contract and should be able to retrieve them.

Recommendation

We advise the client to reimplement the `safeTransfer()` function that allows the transfer of all reward that is left in the aforementioned situation.

Alleviation

The client reimplemented the the `safeTransfer()` function as we had suggested and resolved this issue.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.