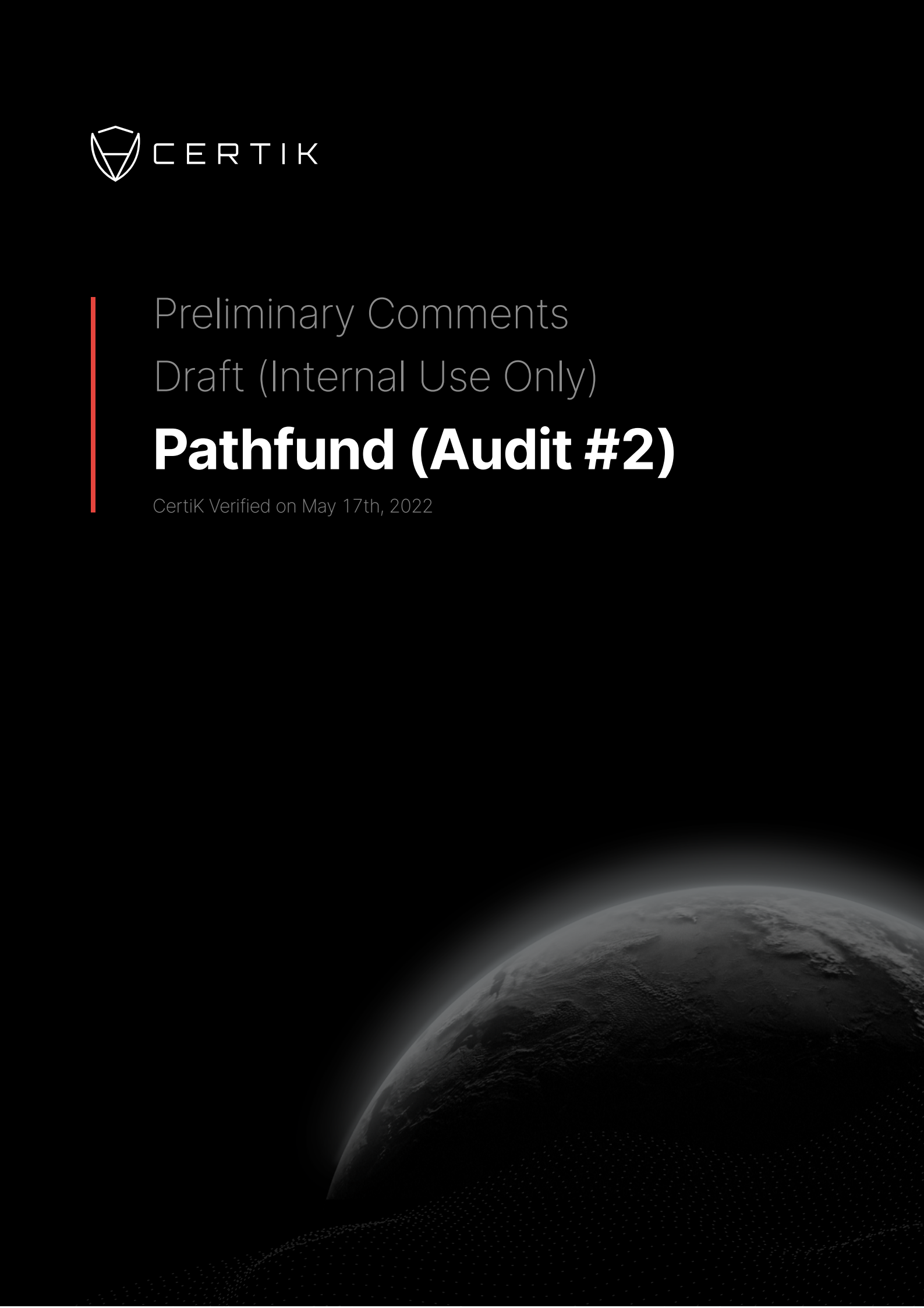
CERTIK

Preliminary Comments
Draft (Internal Use Only)

Pathfund (Audit #2)

CertiK Verified on May 17th, 2022



CertiK Certified

CertiK Verified on May 17th, 2022

Pathfund (Audit #2)

These preliminary comments were prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
ERC-20, Staking	Ethereum	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 05/17/2022	N/A

CODEBASE

https://github.com/CertiKProject/certik-audit-projects/tree/8433e6b36644add8e32a4fa8ee9a86dcd2eb4729/contracts

...View All

COMMITS

8433e6b36644add8e32a4fa8ee9a86dcd2eb4729

...View All

Vulnerability Summary

18	16	0	0	2	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

2	Critical	2 Resolved	Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
3	Major	1 Resolved, 2 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
2	Medium	2 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
4	Minor	4 Resolved	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
7	Informational	7 Resolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.
0	Discussion		The impact of the issue is yet to be determined, hence requires further clarifications from the project team.

TABLE OF CONTENTS | PATHFUND (AUDIT #2)

CODEBASE | PATHFUND (AUDIT #2)

Repository

https://github.com/CertiKProject/certik-audit-projects/tree/8433e6b36644add8e32a4fa8ee9a86dcd2eb4729/contracts

Commit

8433e6b36644add8e32a4fa8ee9a86dcd2eb4729

AUDIT SCOPE | PATHFUND (AUDIT #2)

0 files audited

ID	File	SHA256 Checksum

APPROACH & METHODS | PATHFUND (AUDIT #2)

This report has been prepared for Pathfund to discover issues and vulnerabilities in the source code of the Pathfund (Audit #2) project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis 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:

- Testing the smart contracts against both common and uncommon attack vectors;
- 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.

FINDINGS | PATHFUND (AUDIT #2)

18	2	3	2	4	7	0
Total Findings	Critical	Major	Medium	Minor	Informational	Discussion

This report has been prepared to discover issues and vulnerabilities for Pathfund (Audit #2). Through this audit, we have uncovered 18 issues ranging from different severity levels. Utilizing Static Analysis techniques to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
LFC-01	**Centralization Risks In LaunchFactory.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
LFC-02	Potential Reentrancy Attack (Not Involving Ether)	Volatile Code	Medium	Resolved
LFC-03	Usage Of `transfer()` For Sending BNB	Volatile Code	Medium	Resolved
SCC-01	All Funds Will Be Blocked Forever	Volatile Code	Critical	Resolved
SCC-02	A Reverting Fallback Function Will Lock Up All Payouts	Volatile Code	Critical	Resolved
SCC-03	**Centralization Risks In LaunchSpawnedC.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
SCC-04	Off-By-One Error	Logical Issue	Major	Resolved
SCC-05	Contract Size Exceeds Limit	Language Specific	Minor	Resolved
SCC-06	Missing Zero Address Validation In `constructor()`	Volatile Code	Minor	Resolved
SCC-07	Hidden Assumption For `pairz`	Logical Issue	Minor	Resolved
SCC-08	Unchecked Value Of ERC-20 `transfer()` / `transferFrom()` Call	Volatile Code	Minor	Resolved

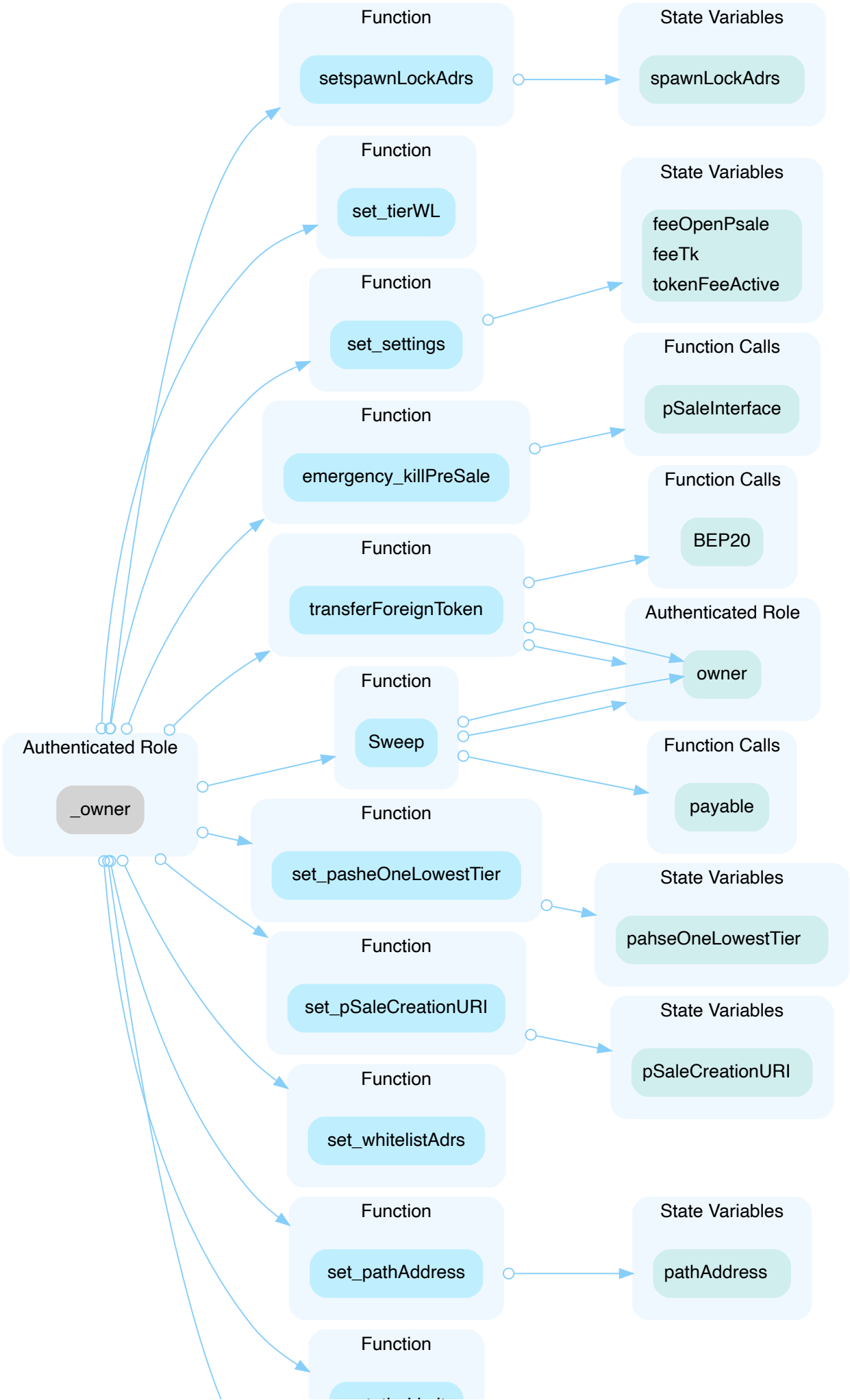
ID	Title	Category	Severity	Status
CKP-01	Missing Library Methods	Language Specific	Informational	Resolved
CKP-02	Function Visibility Optimization	Gas Optimization	Informational	Resolved
CKP-03	Missing Emit Events	Coding Style	Informational	Resolved
LFC-04	Unused Functions	Gas Optimization	Informational	Resolved
SCC-09	Variable Could Be Declared As `constant`	Language Specific	Informational	Resolved
SCC-10	Variables Never Used Can Be Removed	Gas Optimization	Informational	Resolved
SCC-11	Missing Error Messages	Coding Style	Informational	Resolved

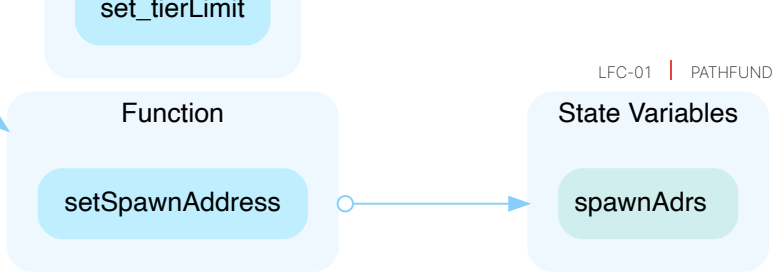
LFC-01 | CENTRALIZATION RISKS IN LAUNCHFACTORY.SOL

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/LaunchFactory.sol (Basw): 85, 153, 161, 168, 175, 178, 182, 186, 189, 195, 199, 203**	**Acknowledged**

Description

In the contract `PathLaunchpad_factory`, the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and kill all presales at once via the function `emergency_killPresale()`. Also, `_owner` can collect all BNB via the function `Sweep()`.


Function
setspawnLockAdrs
State Variables
spawnLockAdrs
Function
set_tierWL
State Variables
feeOpenPsale
feeTk
tokenFeeActive
Function
set_settings
Function Calls
pSaleInterface
Function
emergency_killPreSale
Function Calls
BEP20
Function
transferForeignToken
Authenticated Role
owner
Function
Sweep
Function Calls
payable
Authenticated Role
_owner
Function
set_pasheOneLowestTier
State Variables
pahseOneLowestTier
Function
set_pSaleCreationURI
State Variables
pSaleCreationURI
Function
set_whitelistAdrs
Function
set_pathAddress
State Variables
pathAddress
Function



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR

- Remove the risky functionality.

Alleviation

[PathFund] - Issue acknowledged. The ownership will be handled through multi-sig wallets to prevent a single point of failure due to the private key being compromised.

[CertiK] - The `PathFund` team acknowledged the finding and will make the owner's account multi-sig.

LFC-02 | POTENTIAL REENTRANCY ATTACK (NOT INVOLVING ETHER)

Category	Severity	Location	Status
Volatile Code	Medium	contracts/LaunchFactory.sol (Basw): 64, 66	Resolved

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

External call(s)

File: contracts/LaunchFactory.sol (Line 64, Function `PathLaunchpad_factory.createPsale`)

```
            require(BEP20(addressInfo[0]).transferFrom(msg.sender, address(pool),
tokWfee));
```

File: contracts/LaunchFactory.sol (Line 66, Function `PathLaunchpad_factory.createPsale`)

```
            require(BEP20(addressInfo[0]).transferFrom(msg.sender, address(pool),
tokNoFee));
```

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

[PathFund] - Fixed

[CertiK] - The `PathFund` team acknowledged the finding and applied Checks-Effects-Interactions Pattern to mitigate the reentrancy issue.

The changes can be seen in commit hash 676460e46131c67dee3ce91f5f4879545099c2c9 at line 65 ~ 68 of launchFactory.sol

LFC-03 | USAGE OF `transfer()` FOR SENDING BNB

Category	Severity	Location	Status
Volatile Code	Medium	contracts/LaunchFactory.sol (Basw): 163~164	Resolved

Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring native tokens as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

Recommendation

We advise that the linked `.transfer()` call is substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

Alleviation

[PathFund] - Fixed

[CertiK] - The `PathFund` team acknowledged the finding and fixed the source code by using `.call()` for transferring ETH, instead of `.transfer()`.

The changes can be seen in commit hash 676460e46131c67dee3ce91f5f4879545099c2c9 at line 169 of launchFactory.sol

SCC-01 | ALL FUNDS WILL BE BLOCKED FOREVER

Category	Severity	Location	Status
Volatile Code	Critical	contracts/launchSpawnedC.sol (Basw): 188~194	Resolved

Description

The linked for loop is an infinite loop.
A transaction will run out of gas, and all funds will be blocked forever.

The issue is that `claimSentCounter` is updated at the end of the block.

Recommendation

We recommend implementing a queuing mechanism to allow investors to initiate the withdrawal on their own using a ‘pull-over-push pattern.
Ignore a failed transfer and leave the responsibility to users to receive them properly.
If the number of participants is too big, a transaction can run out of gas, and all funds will be blocked forever.

Alleviation

[PathFund] - Fixed. Removed the whole process/airdrop part since the initial problem wasn't clear enough as we had a limit to 75 transfers per transaction. The new process resolves this issue any any others that may have occurred.

[CertiK] - The `PathFund` team acknowledged the finding and fixed the infinite loop and Block Gas Limit issue.

The changes can be seen in commit hash b62010687ea4e0d699d20d676637b92a500c0658 at line 182 ~ 193 of launchSpawnedC.sol

SCC-02 A REVERTING FALLBACK FUNCTION WILL LOCK UP ALL PAYOUTS

Category	Severity	Location	Status
Volatile Code	Critical	contracts/launchSpawnedC.sol (Basw): 341~355	Resolved

Description

The emergency_killPreSale(uint) function processes a list of transfers. If any of the recipients of a BNB transfer is a smart contract that reverts, then the entire payout will fail.

Recommendation

We recommend implementing a queuing mechanism to allow users to initiate the withdrawal on their own using a 'pull-over-push pattern.'

Alleviation

[PathFund] - Removed the process part of the function, users can get their funds back with the dedicated function since the presale can't be finalized if killed.

[CertiK] - The `PathFund` team acknowledged the finding and fixed the issue.

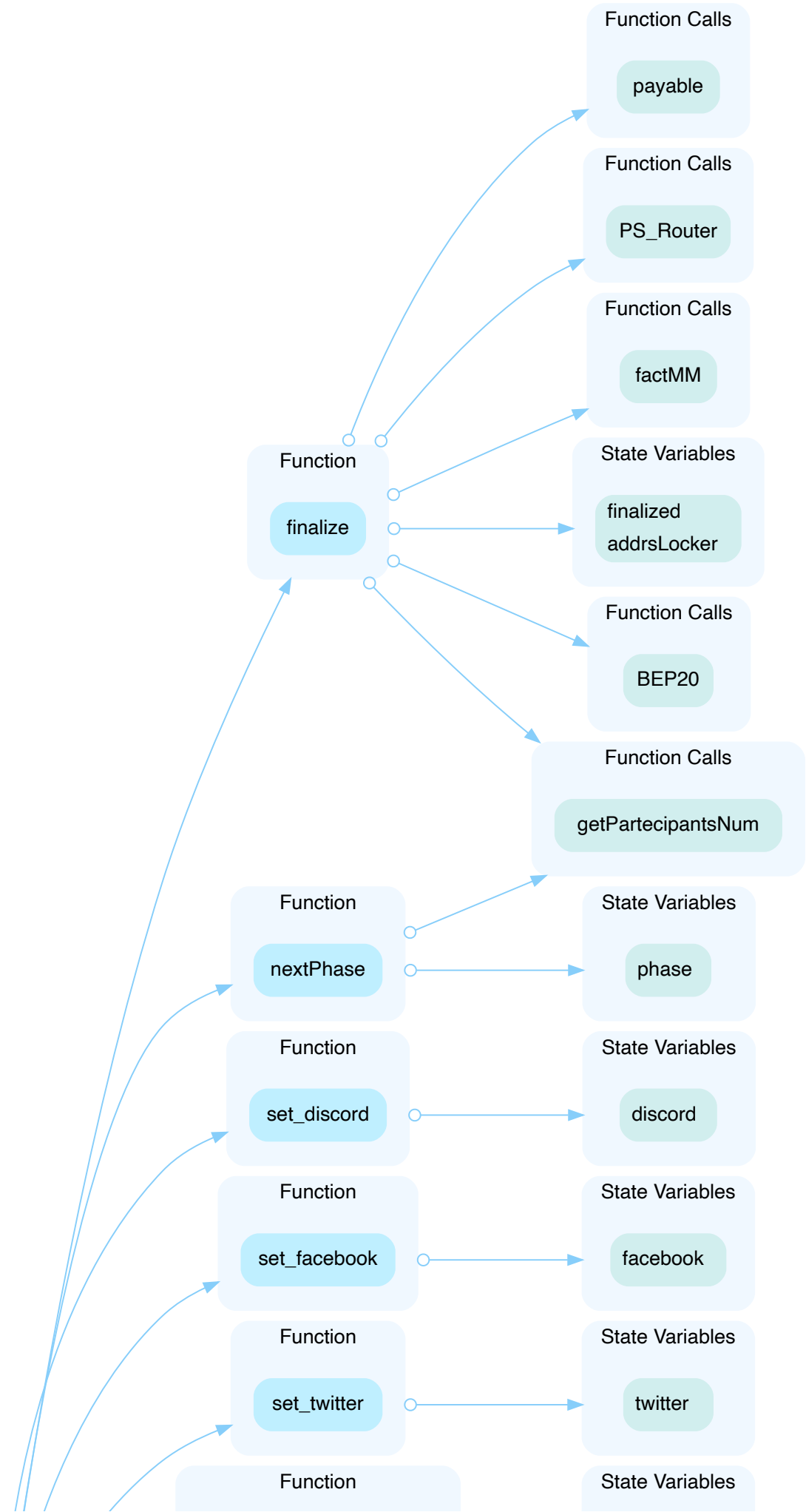
The changes can be seen in commit hash b62010687ea4e0d699d20d676637b92a500c0658 at line 314 ~ 317 of launchSpawnedC.sol

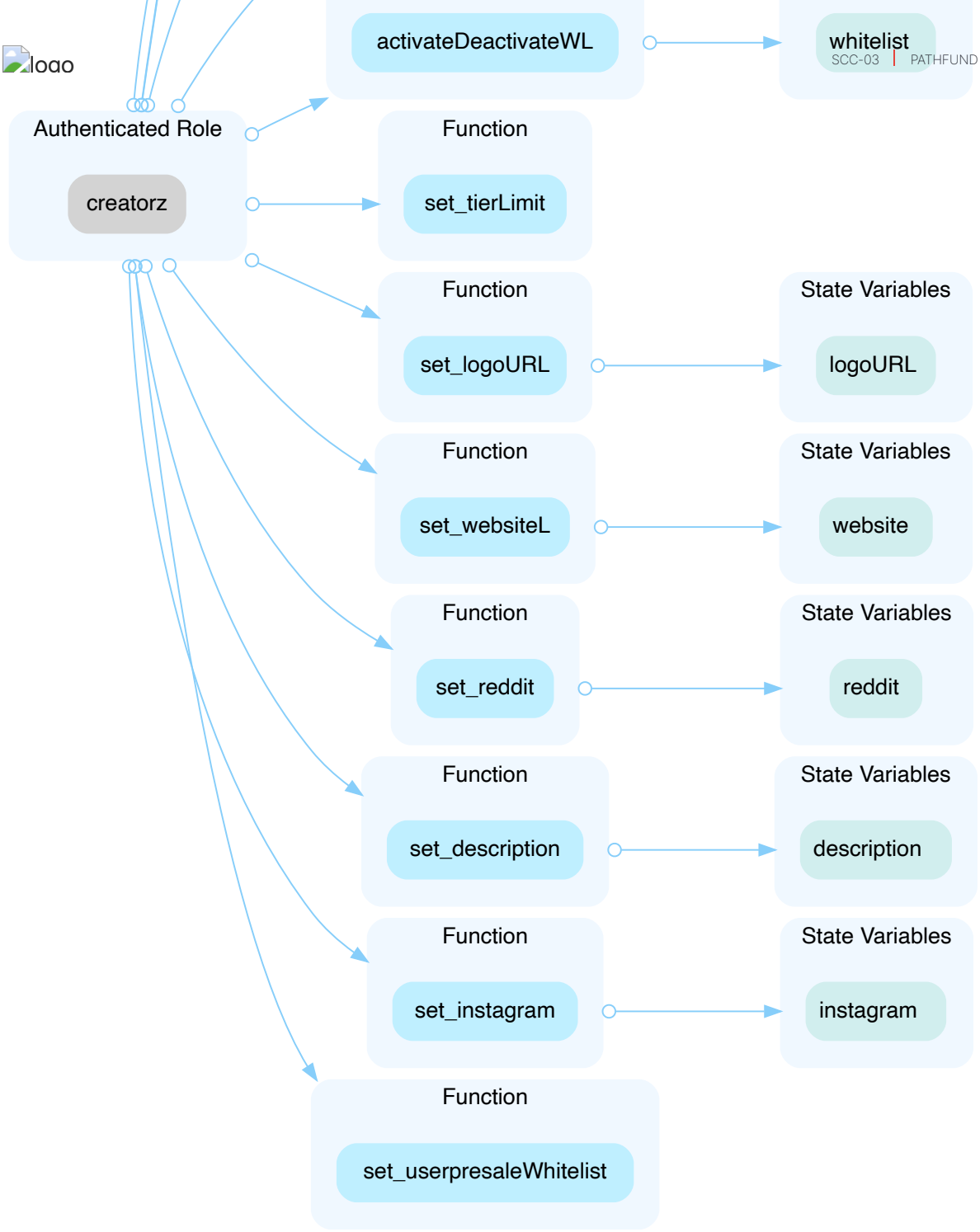
SCC-03 | CENTRALIZATION RISKS IN LAUNCHSPAWNEDC.SOL

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/launchSpawnedC.sol (Basw): 123, 177, 227, 285, 297, 304, 308, 311, 314, 317, 320, 323, 326, 329**	**Acknowledged**

Description

In the contract `PATHlaunchpad` the role `creatorz` has authority over the functions shown in the diagram below. Any compromise to the `creatorz` account may allow the hacker to take advantage of this authority and collect all the funds via the function `finalize()`.


Function Calls
payable
Function Calls
PS_Router
Function Calls
factMM
Function
finalize
State Variables
finalized
addrsLocker
Function Calls
BEP20
Function Calls
getPartecipantsNum
Function
nextPhase
State Variables
phase
Function
set_discord
State Variables
discord
Function
set_facebook
State Variables
facebook
Function
set_twitter
State Variables
twitter
Function
State Variables



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

SCC-04 | OFF-BY-ONE ERROR

Category	Severity	Location	Status
Logical Issue	Major	contracts/launchSpawnedC.sol (Basw): 186	Resolved

Description

An off-by-one error or off-by-one bug (known by acronyms OBOE, OBO, OB1 and OBOB) is a logic error involving the discrete equivalent of a boundary condition. For more information, please check this link.

Recommendation

We recommend replacing the linked line with the following line of code:

```
require(claimSentCounter < tPart);
```

Alleviation

[PathFund] - Fixed

[CertiK] - The `PathFund` team acknowledged the finding and fixed the source code.

The changes can be seen in commit hash 676460e46131c67dee3ce91f5f4879545099c2c9 at line 184 of launchSpawnedC.sol

SCC-05 | CONTRACT SIZE EXCEEDS LIMIT

Category	Severity	Location	Status
Language Specific	Minor	contracts/launchSpawnedC.sol (Basw): 10~11	Resolved

Description

The contract code size is 33487 bytes and exceeds 24576 bytes (a limit introduced in Spurious Dragon), so this contract may not be deployable on the mainnet. For more information, please check this link.

Recommendation

We recommend separating your contracts or using libraries or proxy systems. For more information, please check this link.

Alleviation

[Pathfund] - The contract is split in 5 parts to avoid this problem, after the fixes had to reduce the compile optimizer to 50 to avoid the size issue.

[CertiK] - With the usage of an optimizer the contracts can be deployed. This resolves the issue.

SCC-06 | MISSING ZERO ADDRESS VALIDATION IN `constructor()`

Category	Severity	Location	Status
Volatile Code	Minor	contracts/launchSpawnedC.sol (Basw): 80~82	Resolved

Description

In `constructor()` the assigned value to `token`, `pair`, and `router` should be verified as non-zero values to prevent being mistakenly assigned as `address(0)`.

Recommendation

We recommend checking that these addresses are not zero with requires statements.

Alleviation

[Pathfund] - The requirement to check if token, pair, and router are not address(0) is done on the factory contract inside the function before the creation of this(launchSpawnedC.sol) contract.

[CertiK] - The implicit check does exist. This resolves the issue.

SCC-07 | HIDDEN ASSUMPTION FOR `pairz`

Category	Severity	Location	Status
Logical Issue	Minor	contracts/launchSpawnedC.sol (Basw): 240	Resolved

Description

There is a hidden assumption that the variable `pairz` is always a non-zero address but it is not the case.
Function getPair(address, address) returns the address of the pair for tokenA and tokenB, if it has been created, else address(0) (0x0000000000000000000000000000000000000000).
If the pair has not been created yet, this function will never succeed.

Recommendation

We recommend checking if the pair has been created or not before adding liquidity.

Alleviation

[PathFund] - Fixed

[CertiK] - The `PathFund` team acknowledged the finding and fixed the source code.

The changes can be seen in commit hash 676460e46131c67dee3ce91f5f4879545099c2c9 at line 293 ~241 of launchSpawnedC.sol

SCC-08 | UNCHECKED VALUE OF ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	Minor	contracts/launchSpawnedC.sol (Basw): 353~354	Resolved

Description

"The linked `transfer()` / `transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

Recommendation

"As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

====

It is recommended to use SafeERC20 or make sure that the value returned from 'transferFrom()' is checked."

Alleviation

[PathFund] - Fixed

[CertiK] - The `PathFund` team acknowledged the finding and fixed the source code.

The changes can be seen in commit hash 676460e46131c67dee3ce91f5f4879545099c2c9 at line 354 of launchSpawnedC.sol

CKP-01 | MISSING LIBRARY METHODS

Category	Severity	Location	Status
Language Specific	Informational	contracts/LaunchFactory.sol (Basw): 7~8; contracts/launchSpawnedC.sol (Basw): 20~21	Resolved

Description

We should add the library methods before declaring a set state variable.

Recommendation

We recommend adding the following line of code:

```
using EnumerableSet for EnumerableSet.UintSet;
```

For more information, please check this link

(https://docs.openzeppelin.com/contracts/3.x/api/utils#EnumerableSet).

Alleviation

[PathFund] - Fixed

[CertiK] - The `PathFund` team acknowledged the finding and fixed the source code.

The first change can be seen in commit hash 676460e46131c67dee3ce91f5f4879545099c2c9 at line 10 of launchSpawnedC.sol

The second change can be seen in commit hash 676460e46131c67dee3ce91f5f4879545099c2c9 at line 6 of launchFactory.sol

CKP-02 | FUNCTION VISIBILITY OPTIMIZATION

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/LaunchFactory.sol (Basw): 76, 85, 153, 161, 168, 175, 178, 182, 186, 189, 195, 199, 203; contracts/launchSpawnedC.sol (Basw): 123, 156, 183, 208, 227, 297, 304, 308, 311, 314, 317, 320, 323, 326, 329	Resolved

Description

The linked function is declared as `public` and is not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

Recommendation

We advise that the functions' visibility specifiers are set to `external`, optimizing the gas cost of the function.

Alleviation

[PathFund] - Fixed

[CertiK] - The `PathFund` team resolved the finding by following the recommendation above.

The changes can be seen in commit hash 676460e46131c67dee3ce91f5f4879545099c2c9 of launchSpawnedC.sol

The changes can also be seen in commit hash 676460e46131c67dee3ce91f5f4879545099c2c9 of launchFactory.sol

CKP-03 | MISSING EMIT EVENTS

Category	Severity	Location	Status
Coding Style	Informational	contracts/LaunchFactory.sol (Basw): 175, 187~188; contracts/launchSpawnedC.sol (Basw): 309, 312, 315, 318, 321, 324, 327, 330; contracts/semiproxy.sol (Basw): 37, 41; contracts/spawnLock.sol (Basw): 25	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

Alleviation

[PathFund] - Added the missing emits on the launchapadFactory.sol, can't add them in launchSpawnedC.sol due to contract size limit.

[CertiK] - The `PathFund` team acknowledged the finding and fixed the source code.

The changes can be seen in commit hash 676460e46131c67dee3ce91f5f4879545099c2c9 of launchFactory.sol.

LFC-04 | UNUSED FUNCTIONS

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/LaunchFactory.sol (Basw): 81~84	Resolved

Description

The linked function is supposed to be called by the contract launchSpawnedC.sol but is never used.

Recommendation

We advise adding a function in the contract launchSpawnedC.sol so that finalized IDs can be removed.

Alleviation

[PathFund] - It is used at line 242 of launchSpawnedC.sol

[CertiK] - An external call is made to the linked function. This resolves the issue.

SCC-09 | VARIABLE COULD BE DECLARED AS `constant`

Category	Severity	Location	Status
Language Specific	Informational	contracts/launchSpawnedC.sol (Basw): 12, 14, 37	Resolved

Description

Variables `bnbA` and `dead` could be declared as `constant` since these state variables are never to be changed.

Recommendation

We recommend declaring these variables as `constant`.

```
address constant public bnbA = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
address constant public dead = 0x000000000000000000000000000000000000dEaD;
unit constant public precisionNum = 10**30;
```

Alleviation

[PathFund] - Fixed

[CertiK] - The `PathFund` team acknowledged the finding and fixed the source code.

The changes can be seen in commit hash 676460e46131c67dee3ce91f5f4879545099c2c9 at line 11, 13, and 35 of launchSpawnedC.sol

SCC-10 | VARIABLES NEVER USED CAN BE REMOVED

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/launchSpawnedC.sol (Basw): 15~16	Resolved

Description

The variable `pSaleCreationURI` is never used.

Recommendation

We advise reviewing the unused variable. If it is not need then we recommend removing unused variables. Otherwise please update the team the need of the linked variable.

Alleviation

[PathFund] - Fixed

[CertiK] - The `PathFund` team acknowledged the finding and fixed the source code.

The changes can be seen in commit hash 676460e46131c67dee3ce91f5f4879545099c2c9 at line 15 of launchSpawnedC.sol

SCC-11 | MISSING ERROR MESSAGES

Category	Severity	Location	Status
Coding Style	Informational	contracts/launchSpawnedC.sol (Basw): 151, 208	Resolved

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise adding an error message for this **require** statement.

Alleviation

[PathFund] - Added some more error messages, unfortunately, we're limited by the contract size, we can't add error messages just in the .call and .transfer requires.

[CertiK] - The `PathFund` team acknowledged the finding and fixed the infinite loop and Block Gas Limit issue.

The changes can be seen in commit hash b62010687ea4e0d699d20d676637b92a500c0658 in launchSpawnedC.sol.

APPENDIX | PATHFUND (AUDIT #2)

Finding Categories

Categories	Description
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.
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.

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 | CERTIK

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.

CertiK | Securing the Web3 World

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.