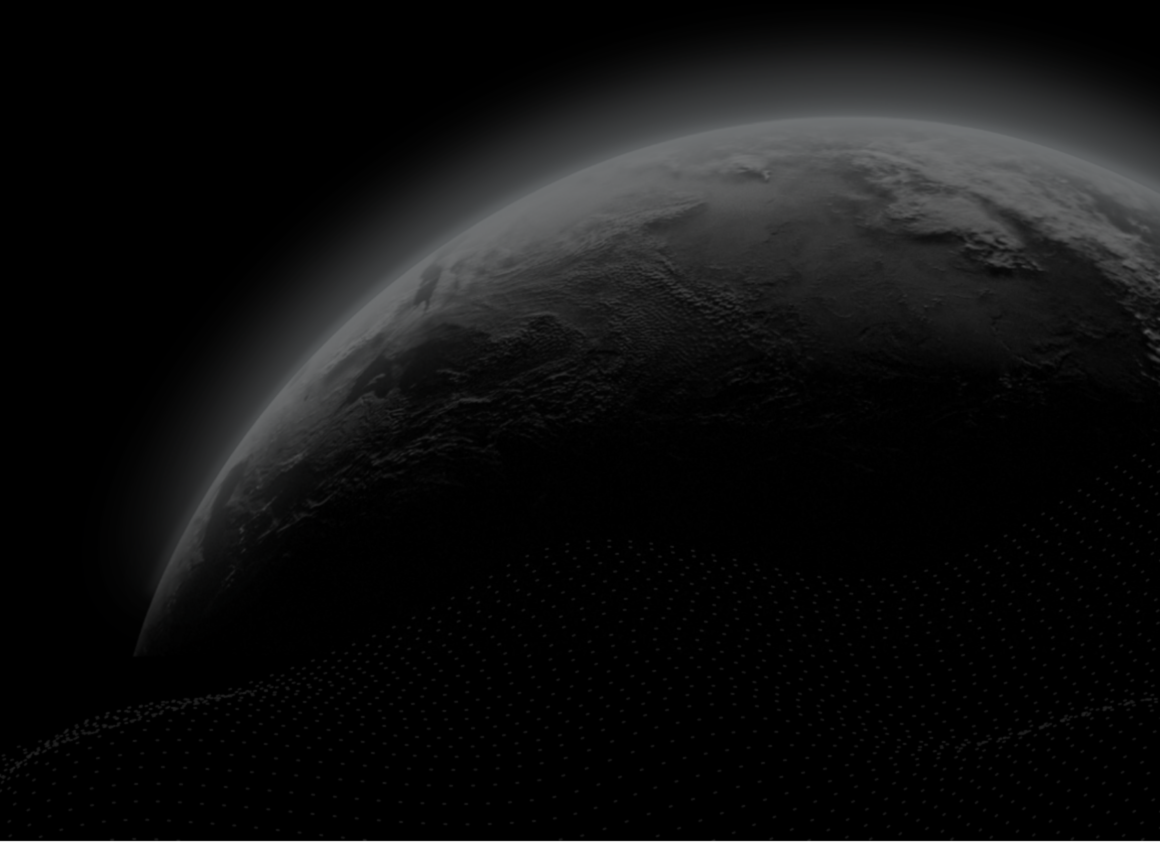
CERTIK

Security Assessment

Umbrella Network - phoenix

CertiK Assessed on Aug 17th, 2023



CertiK Assessed on Aug 17th, 2023

Umbrella Network - phoenix

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum (ETH)	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 08/17/2023	N/A

CODEBASE

https://github.com/umbrella-network/phoenix

View All in Codebase Page

COMMITS

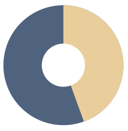
base: 29585531fd56f1265c8c138cd8efc67d10e95200

update1: 827d7c5be32332bbbb294d3de2f37fb91521bb48

update2: 5783a40481f812a071c34e7d8680cab66de70dde

View All in Codebase Page

Vulnerability Summary



9	8	0	0	1	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

🟥 0	Critical		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.
🟧 0	Major		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.
🟨 0	Medium		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.
🟦 5	Informational	4 Resolved, 1 Acknowledged	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.

TABLE OF CONTENTS | UMBRELLA NETWORK - PHOENIX

Summary

Executive Summary

Vulnerability Summary

Codebase

Audit Scope

Approach & Methods

Dependencies

Out Of Scope Dependencies

Assumptions

Recommendations

Findings

CON-03 : Missing Input Validation

UFC-02 : Missing Validation of `_priceDatas` data in function `update()`

UFC-03 : Use of `abi.encodePacked()`

UFC-04 : Missing Check For `v` And `s`

CFB-02 : Unused Custom Errors

SBP-01 : Hardcoded Addresses

SBP-02 : Unnecessary `address` Casting

UFC-05 : Missing Emit Events

UFC-06 : Function `reset()` Can Be Replayed

Optimizations

CON-01 : User-Defined Getters

CON-02 : Inherited Features are Unused

SBS-01 : Redundant Functions

Appendix

Disclaimer

CODEBASE | UMBRELLA NETWORK - PHOENIX

Repository

https://github.com/umbrella-network/phoenix

Commit

base: 29585531fd56f1265c8c138cd8efc67d10e95200

update1: 827d7c5be32332bbbb294d3de2f37fb91521bb48

update2: 5783a40481f812a071c34e7d8680cab66de70dde

update3: cdd3454c75699a746ef3ee818f2902fb37541e5c

Deployed Contracts

Linea Mainnet

UmbrellaFeeds: 0x455acbbC2c15c086978083968a69B2e7E4d38d34

UmbrellaFeedsReaderFactory: 0x150368e6bF2538B9Be8e5688F1D7457773C49463

StakingBankStaticProd: 0xda9a63d77406faa09d265413f4e128b54b5057e0

AUDIT SCOPE | UMBRELLA NETWORK - PHOENIX

5 files audited ● 3 files with Acknowledged findings ● 1 file with Partially Resolved findings ● 1 file without findings

ID	Repo	File	SHA256 Checksum
● UFC	umbrella-network/phoenix	contracts/onChainFeeds/UmbrellaFeeds.sol	eb1db13967134ef3f036799f5f8b8e80c83663d3bf7936c6346da8335d77d3d1
● SBP	umbrella-network/phoenix	contracts/stakingBankStatic/StakingBankStaticProd.sol	e8321ccba2d74ab3c513b7516a58a74bf31258272234164f95ae47e2b4c9dea9
● SBS	umbrella-network/phoenix	contracts/stakingBankStatic/StakingBankStatic.sol	865a4213a927a57bff57f59b129746e7857f22e3652fd2588e0b105e8392abb0
● UFR	umbrella-network/phoenix	contracts/onChainFeeds/UmbrellaFeedsReader.sol	b37c923fddf778d00b252f867525a24f204f9d38d9e5b3165d6c11866701a0ba
● UFF	umbrella-network/phoenix	contracts/onChainFeeds/UmbrellaFeedsReaderFactory.sol	e410d9c9cc037f0d81ae18307b4a6e7b90358696afe18616d47636bc84431035

APPROACH & METHODS | UMBRELLA NETWORK - PHOENIX

This report has been prepared for Umbrella Network to discover issues and vulnerabilities in the source code of the Umbrella Network - phoenix 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.

DEPENDENCIES | UMBRELLA NETWORK - PHOENIX

Out Of Scope Dependencies

The scope of the audit examines the security of a portion of the protocol's contracts. As such, the in-scope contracts are serving as an underlying entity to interact with out-of-scope dependencies within the protocol. The out-of-scope dependencies that the contracts interact with are:

- Contract `UmbrellaFeeds` has an out-of-scope dependency via interface `IRegistry` variable `REGISTRY` ;
- Contract `UmbrellaFeeds` relies on a proof-of-authority model for updating price feed information. In the case of the use of `StakingBankStaticProd` as `STAKING_BANK` , the validator addresses are hardcoded as follows. It is understood that this list of validators may change based on the mainnet the project is deployed on:
    - 0x977Ba523420110e230643B772Fe9cF955e11dA7B;
    - 0xe2422b23e52bc13ebA04d7FbB9F332Deb43360fB;
    - 0x57F404aD75e371c1A539589C1eFCA12e0C6980AD;
    - 0xD56C6A4f64E0bD70260472d1DB6Cf5825858CB0d;
    - 0x220230Eda8f50067Dd9e4729345dabCCe0C61542;
    - 0x93FdcAB283b0BcAc48157590af482E1CFd6af6aC;
    - 0xCd733E06B06083d52fC5867E8E3432aA5c103A38;
    - 0x42e210b110c6aa49CdfA7ceF1444Aa4719653111;
    - 0x501731c6a69803a53Ec6c3e12f293c247cE1092B;
    - 0x8bF9661F1b247522C75DD0FE84355aD2EfF27144;
    - 0x281754Ab58391A478B7aA4E7f39991CfB41118c4;
    - 0xB9C63a350A04d8BD245d18928a26EE036352dDd8;
    - 0x57A51D5BDcE188c2295fCA3b4687475a54E65A02;
    - 0x777FbA3666fa7747476a34577FcCC404b263E09F;
    - 0x2F85824B2B38F179E451988670935d315b5b9692;
    - 0xe868bE65C50b61E81A3fC5cB5A7916090B05eb2A;
    - 0xB12c5DFA8693a5890c4b5B9145E3CAE1502f17f0;
    - 0xe7129A4c7521452511249c26B018fEfbB10d108d;

The correct functioning of the contract is dependent on the timely accuracy of the `price` , `timestamp` , and `heartbeat` information signed by the requisite number of validators. The consideration of the accuracy of the updated price feed information is outside the scope of the audit;

The scope of the audit treats out-of-scope dependencies as black boxes and assumes their functional correctness.

Assumptions

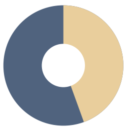
Within the scope of the audit, assumptions are made about the intended behavior of the protocol in order to inspect consequences based on those behaviors. Assumptions made within the scope of this audit include:

- The out-of-scope call to `REGISTRY.requireAndGetAddress("StakingBank")` in deployment of `UmbrellaFeeds` sets the `STAKING_BANK` with an instance of in-scope `StakingBankStaticProd` contract;
- Validator addresses only sign accurate information for updates to price feeds;
- All instances of `UmbrellaFeeds` and contracts have a common `address(REGISTRY)` contract address which is consistent with that set in `UmbrellaFeedsReaderFactory` ;
- All uses of `IUmbrellaFeeds` within the audited code refer to instances of in-scope contract `UmbrellaFeeds` ;
- All urls hardcoded into `StakingBankStaticProd` will be reachable by the time the contract is deployed;
- The decimal precision of each stored price in an instance of the `UmbrellaFeeds` contract is the same for every updated entry, consequently resulting in the need for one value `DECIMALS` to represent all exchange pairs in recorded in the contract.

Recommendations

We recommend all out-of-scope dependencies are carefully vetted to ensure they function as intended. Additionally, we recommend all assumptions about the behavior of the project are thoroughly reviewed and, if the assumptions do not match the intention of the protocol, documenting the intended behavior for review.

FINDINGS | UMBRELLA NETWORK - PHOENIX



9	0	0	0	4	5
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Umbrella Network - phoenix. Through this audit, we have uncovered 9 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
CON-03	Missing Input Validation	Volatile Code	Minor	● Resolved
UFC-02	Missing Validation Of `_priceDatas` Data In Function `update()`	Volatile Code	Minor	● Resolved
UFC-03	Use Of `abi.encodePacked()`	Volatile Code	Minor	● Resolved
UFC-04	Missing Check For `v` And `s`	Volatile Code, Logical Issue	Minor	● Resolved
CFB-02	Unused Custom Errors	Coding Issue	Informational	● Resolved
SBP-01	Hardcoded Addresses	Coding Style, Volatile Code	Informational	● Resolved
SBP-02	Unnecessary `address` Casting	Code Optimization	Informational	● Resolved
UFC-05	Missing Emit Events	Coding Style	Informational	● Acknowledged
UFC-06	Function `reset()` Can Be Replayed	Coding Style	Informational	● Resolved

CON-03 | MISSING INPUT VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/onChainFeeds/UmbrellaFeeds.sol (updated_base): 75~76; contracts/stakingBankStatic/StakingBankStaticProd.sol (updated_base): 28~29	● Resolved

Description

UmbrellaFeeds.sol

- In function `update()` , there is a missing check that the input `_priceKeys` and `_priceKeys` are the same length. There is a check that is commented out with the comment that the check is only used for pretty errors. However, if `_priceKeys` is a longer length than `_priceDatas` , there may be included data that is not properly updated, and the function will not revert;
- In function `update()` there is no check that `address(this)` is the `umbrellaFeeds` contract address listed in `REGISTRY` . This allows for the possibility of multiple `UmbrellaFeeds` contract instances reporting their separate information for a given key, where only one is listed in `REGISTRY` ;

StakingBankStaticProd.sol

- The constructor is missing a check that `_validatorsCount` matches the number of validators hardcoded into `StakingBankStaticProd` , which, at the time of the issue of the report, is 18. If the input value is different from the number of addresses hardcoded, then return values from functions such as `totalSupply()` , `getBalances()` , and `getNumberOfValidators()` will not accurately represent the contract;

Recommendation

We recommend adding the missing checks described above.

Alleviation

`[CertiK]` : The team made changes resolving the finding in commits

- 1ff50bb0bf5c8b9a851c0c1d8062196615681fa5
- 3056b6a8b9c1af74612fa2de382783ac1dfe2c07
- 5783a40481f812a071c34e7d8680cab66de70dde

- cdd3454c75699a746ef3ee818f2902fb37541e5c.

The `reset()` feature was replaced with a `destroy()` feature in order to prevent the simultaneous existence of two price feed contracts. The team notes that projects not using the registry smart contract to get the latest UmbrellaFeeds contract will have to include the fallback mechanism on their side. Since the function can be called by anyone if the proper conditions are not met, the remaining issue outlined above are considered resolved. It is noted that users can read the `DEPLOYED_AT` value in each `UmbrellaFeeds` contract and check if the contract is within 3 days of its deployment. If so, users can check that the instance address is that listed in the `REGISTRY` before interacting. If the current timestamp is more than 3 days past the `DEPLOYED_AT` value and the `UmbrellaFeeds` contract instance is not listed in the `REGISTRY`, this makes the contract eligible to be destroyed by anyone.

The understood pattern of calls for a new `UmbrellaPriceFeeds` contract is as follows:

1. Deploy new `UmbrellaFeeds` instance
2. Add the new instance to the `REGISTRY` within 3 days of deployment of the `UmbrellaFeeds` instance
3. Provide signed price feeds to be used in `update()` only after the contract instance is added to the `REGISTRY` contract
4. Once a new `UmbrellaFeeds` instance is to be used, the old contract address is replaced in the `REGISTRY` with the new contract instance, allowing `destroy()` to be called.

Any deviation from the sequence of calls outlined above may allow the contract to be self-destructed. The team states that validators are fetching all contract addresses from the registry and, as a result, they can not send updates to new contracts before they are added to the registry because they will not know its address.

While the functionality currently resolves the outlined issue, the use of `selfdestruct` is not recommended for long term use in a project. This is because the `SELFDESTRUCT` opcode has been deprecated for the EVM and its functionality will likely change in the near future.

Reference: https://eips.ethereum.org/EIPS/eip-6049

UFC-02 | MISSING VALIDATION OF `_priceDatas` DATA IN FUNCTION `update()`

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/onChainFeeds/UmbrellaFeeds.sol (updated_base): 87~88	● Resolved

Description

Function `update()` requires that each `_prices` mapping updated does not have that `data` is set to `DATA_RESET`. It is an implicit assumption of the design that if `data` is set to `DATA_RESET`, then the mapping was last updated by function `reset()`, where each of `heartbeat`, `timestamp` and `price` were updated as value 0. However, function `update()` will accept `priceData` inputs with `DATA_RESET` as the `data` value, as long as the information is properly signed. In that case, the values for `heartbeat`, `timestamp`, and `price` may be set to other values.

In particular, this may be an issue because the logic of the protocol checks for whether `timestamp` is nonzero, and if it is, this allows the bypass of checks that may otherwise trigger the check for an updated `UmbrellaFeeds` contract in the `REGISTRY`.

Recommendation

We recommend preventing the inclusion of `_priceDatas` inputs which include `data` set to the `DATA_RESET` value.

Alleviation

`[CertiK]`: The team resolved this finding in commits 5783a40481f812a071c34e7d8680cab66de70dde and cdd3454c75699a746ef3ee818f2902fb37541e5c.

They have done so by introducing a new `destroy()` feature and removing the logic pertaining to value `DATA_RESET`. The team notes that projects not using the registry smart contract to get the latest UmbrellaFeeds contract will have to include the fallback mechanism on their side.

The understood pattern of calls for a new `UmbrellaPriceFeeds` contract is as follows:

1. Deploy new `UmbrellaFeeds` instance
2. Add the new instance to the `REGISTRY` within 3 days of deployment of the `UmbrellaFeeds` instance
3. Provide signed price feeds to be used in `update()` only after the contract instance is added to the `REGISTRY` contract
4. Once a new `UmbrellaFeeds` instance is to be used, the old contract address is replaced in the `REGISTRY` with the new contract instance, allowing `destroy()` to be called.

Any deviation from the sequence of calls outlined above may allow the contract to be self-destructed. The team states that validators are fetching all contract addresses from the registry and, as a result, they can not send updates to new contracts before they are added to the registry because they will not know its address.

While the functionality currently resolves the outlined issue, the use of `selfdestruct` is not recommended for long term use in a project. This is because the `SELFDESTRUCT` opcode has been deprecated for the EVM and its functionality will likely change in the near future.

Reference: https://eips.ethereum.org/EIPS/eip-6049

UFC-03 | USE OF `abi.encodePacked()`

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/onChainFeeds/UmbrellaFeeds.sol (updated_base): 98~99	● Resolved

Description

Encoding through `abi.encodePacked()` with dynamically-sized inputs can cause hash collision in some instances, which may allow for an unintended validation of input. This type of encoding can be susceptible to malleability particularly when used with consecutive strings, bytes, or other dynamically-sized types.

While in this instance, only one dynamically sized parameter is user-provided, use of `abi.encode` should be considered for hashing messages instead.

Recommendation

We recommend using an alternative to `abi.encodePacked` such as `abi.encode` for signature verification methods.

Reference: https://swcregistry.io/docs/SWC-133

Alleviation

`[CertiK]` : The team made changes resolving the finding in commit a9330b4113a8576ebb231c34561d93d97250e711.

UFC-04 | MISSING CHECK FOR `v` AND `s`

Category	Severity	Location	Status
Volatile Code, Logical Issue	● Minor	contracts/onChainFeeds/UmbrellaFeeds.sol (updated_base): 259~260	● Resolved

Description

The following description is adapted from OpenZeppelin's ECDSA file:

EIP-2 still allows signature malleability for `ecrecover()`. Appendix F in the Ethereum Yellow paper, defines the valid range for s in (311): `0 < s < secp256k1n ÷ 2 + 1` and for the recovery identifier (312): `v ∈ {0,1}`. This should not be confused with the input for `ecrecover()` where `v ∈ {27,28}` (See doc). However, these values can be obtained by taking `27 + "recovery identifier"`, so that they will also yield a unique signature and are often the `v` values returned from signatures. (For example `web3.eth.accounts.sign()`)

If your library generates malleable signatures, such as `s`-values in the upper range, calculate a new `s`-value with `0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141 - s1` and flip `v` from `27` to `28` or vice versa. If your library also generates signatures with `0/1` for `v` instead of `27/28`, then add `27` to `v` so that `ecrecover()` accepts these signatures as well.

Recommendation

We recommend including a check such as that below, or considering the example in **`ECDSA.sol`** from the OpenZeppelin library.

```
require(uint256(s) <=
0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0, "ECDSA: invalid
signature 's' value");
require(uint8(v) == 27 || uint8(v) == 28, "ECDSA: invalid signature 'v' value");
```

Alleviation

`[CertiK]`: The team made changes resolving the finding in commit e5395109babab4acf97490926b703fa4e3ed35b4.

CFB-02 | UNUSED CUSTOM ERRORS

Category	Severity	Location	Status
Coding Issue	● Informational	contracts/onChainFeeds/UmbrellaFeeds.sol (updated_base): 44~45; contracts/onChainFeeds/UmbrellaFeedsReader.sol (updated_base): 22	● Resolved

Description

The smart contract contains one or more custom error definitions that are not used, which can lead to unnecessary complexity and reduced maintainability.

```
44      error ArraysDataDoNotMatch();
```

- `ArraysDataDoNotMatch` is declared but never used. It is noted that this custom error is commented out in function `update()` ;

```
22      error FeedNotExist();
```

- `FeedNotExist` is declared but never used.

Recommendation

We recommend ensuring that all necessary custom errors are used, and removing redundant custom errors.

Alleviation

`[CertiK]` : The team made changes resolving the finding in commit 50c1e6ac41b1f88d6c689f509244110e79f84326.

SBP-01 | HARDCODED ADDRESSES

Category	Severity	Location	Status
Coding Style, Volatile Code	● Informational	contracts/stakingBankStatic/StakingBankStaticProd.sol (updated_base): 10~26	● Resolved

Description

Contract `StakingBankStaticProd` hardcodes eighteen addresses which represent the validators for contract `UmbrellaFeeds` .

Recommendation

We recommend ensuring that all addresses are accurate and intended for use in signatures across all platforms in which this protocol will be implemented.

Alleviation

`[CertiK]` : The team confirms that the hardcoded addresses listed are accurate and to be used at the time of deployment on mainnet.

SBP-02 | UNNECESSARY `address` CASTING

Category	Severity	Location	Status
Code Optimization	● Informational	contracts/stakingBankStatic/StakingBankStaticProd.sol (updated_base): 10~26	● Resolved

Description

It is unnecessary to cast the hardcoded values for each validator as an `address`. The values are ready by default as addresses.

Recommendation

We recommend the removal of the `address` casting

Alleviation

`[CertiK]`: The team made changes resolving the finding in commit c32c0bb183c891193bbb8b43e7e4f8e839e317e3.

UFC-05 | MISSING EMIT EVENTS

Category	Severity	Location	Status
Coding Style	● Informational	contracts/onChainFeeds/UmbrellaFeeds.sol (updated_base): 70~71, 97~98	● Acknowledged

Description

Functions that update state variables should emit relevant events as notifications.

Recommendation

We recommend adding events for state-changing actions, and emitting them in their relevant functions.

Alleviation

`[CertiK]` : The team acknowledges the finding and states they opt not to make changes to the current version in order to optimize gas.

UFC-06 | FUNCTION `reset()` CAN BE REPLAYED

Category	Severity	Location	Status
Coding Style	● Informational	contracts/onChainFeeds/UmbrellaFeeds.sol (updated_base): 97~98	● Resolved

Description

Function `reset()` can be called multiple times using the same `_priceKeys` and `_signatures` input.

This finding is informational because once `reset()` is called for an entry in the `_prices` mapping, it cannot be updated to any other value through either `update()` or `reset()`. Calling `reset()` more than once resets the same values of `PriceData(DATA_RESET, 0, 0, 0)` for each entry in the input `_priceKeys`, causing no change to the state.

Recommendation

We recommend considering replay prevention in keeping with common security practices.

Alleviation

`[CertiK]` : The team removed the `reset()` functionality in commit 5783a40481f812a071c34e7d8680cab66de70dde.

While the functionality currently resolves the outlined issue, the use of `selfdestruct` is not recommended for long term use in a project. This is because the `SELFDESTRUCT` opcode has been deprecated for the EVM and its functionality will likely change in the near future.

Reference: https://eips.ethereum.org/EIPS/eip-6049

OPTIMIZATIONS | UMBRELLA NETWORK - PHOENIX

ID	Title	Category	Severity	Status
CON-01	User-Defined Getters	Gas Optimization, Code Optimization	Optimization	● Partially Resolved
CON-02	Inherited Features Are Unused	Gas Optimization, Code Optimization	Optimization	● Acknowledged
SBS-01	Redundant Functions	Gas Optimization, Code Optimization	Optimization	● Acknowledged

CON-01 | USER-DEFINED GETTERS

Category	Severity	Location	Status
Gas Optimization, Code Optimization	● Optimization	contracts/onChainFeeds/UmbrellaFeedsReader.sol (updated_base): 39~41, 44~46; contracts/stakingBankStatic/StakingBankStatic.sol (updated_base): 32~34, 65~67	● Partially Resolved

Description

The linked functions are equivalent to the compiler-generated getter functions for the respective variables.

Recommendation

We recommend relying on the use of the already public compiler-generated functions and removing the view functions.

For `totalSupply()` , we recommend making the constant `TOTAL_SUPPLY` internal, since function `totalSupply()` is an implementation on an inherited interface.

Alleviation

`[CertiK]` : The team made changes partially resolving the finding in commit 827d7c5be32332bbbb294d3de2f37fb91521bb48.

The team states they opt to leave the cited locations in `StakingBankStatic.sol` unaltered due to gas considerations.

CON-02 | INHERITED FEATURES ARE UNUSED

Category	Severity	Location	Status
Gas Optimization, Code Optimization	● Optimization	contracts/interfaces/IStakingBank.sol (updated_base): 6~7, 9~12, 14~17; contracts/stakingBankStatic/StakingBankStatic.sol (updated_base): 75~82; contracts/stakingBankStatic/StakingBankStaticProd.sol (updated_base): 6~7	● Acknowledged

Description

- Contracts `StakingBankStatic` and `StakingBankStaticProd` inherit the struct `Validator` from `IStakingBank` but do not use it.
- Contract `StakingBankStatic` implements functions `register()` and `unregister()` with empty logic.
- Contracts `StakingBankStatic` and `StakingBankStaticProd` inherit the events of interface `IERC20` from `IStakingBank` but do not use them. Additionally, both contracts inherit events declared in `IStakingBank` but do not use them.

Recommendation

We recommend considering the removal of inherited infrastructure which is unused in the contracts `StakingBankStaticProd` and `StakingBankStatic`.

Alleviation

`[CertiK]`: The team states that the use of interface `IStakingBank` is a necessity for backwards compatibility with with their old `StakingBank` contract. See below for more information.

`[UmbrellaNetwork]`: Contracts `StakingBankStatic` and `StakingBankStaticProd` inherit from `IStakingBank` because we need to be backward compatible with old `StakingBank` contract (already audited).

Not all features in the `IStakingBank` interface are used by the `StakingBankStaticProd` and `StakingBankStatic` contract but the interface cannot be split as it will force redeployment of the `StakingBank` contract.

SBS-01 | REDUNDANT FUNCTIONS

Category	Severity	Location	Status
Gas Optimization, Code Optimization	● Optimization	contracts/stakingBankStatic/StakingBankStatic.sol (updated_base): 19~20, 53~54, 60~61	● Acknowledged

Description

- Functions `balances()` and `balanceOf()` implement the same functionality. Function `balances()` can be removed in order to simplify the codebase.
- In `StakingBankStaticProd` , function `addresses()` is redundant, since each hardcoded address is public and labeled with its position in the `validators` list. The constant addresses can be made internal to avoid redundancy.

Recommendation

We recommend simplifying and optimizing the codebase by removing the redundant function logic.

Alleviation

`[CertiK]` : The team states that the functions are kept in for backwards compatibility.

APPENDIX | UMBRELLA NETWORK - PHOENIX

Finding Categories

Categories	Description
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.
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.

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

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.