



METATRUST

# Security Assessment for **Hope-oracle**

July 28, 2023






## Executive Summary

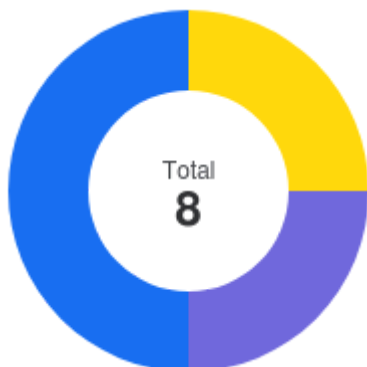
Overview			
Project Name	Hope-oracle		
Codebase URL	<a href="https://github.com/Light-Ecosystem/hope-oracle/tree/audit">https://github.com/Light-Ecosystem/hope-oracle/tree/audit</a>		
Scan Engine	Security Analyzer		
Scan Time	2023/07/28 15:32:44		
Commit Id	18ff6bb3a4bb46c6f544a74397c1235fd67bcd67		






  

Total	
Critical Issues	0
High risk Issues	0
Medium risk Issues	2
Low risk Issues	2
Informational Issues	4

Critical Issues		The issue can cause large economic losses, large-scale data disorder, loss of control of authority management, failure of key functions, or indirectly affect the correct operation of other smart contracts interacting with it.
High Risk Issues		The issue puts a large number of users' sensitive information at risk or is reasonably likely to lead to catastrophic impacts on clients' reputations or serious financial implications for clients and users.
Medium Risk Issues		The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk Issues		The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational Issue		The issue does not pose an immediate risk but is relevant to security best practices or Defence in Depth.



	Critical Issues	0%	0
	High risk Issues	0%	0
	Medium risk Issues	25%	2
	Low risk Issues	25%	2
	Informational Issues	50%	4

## Summary of Findings

MetaScan security assessment was performed on **July 28, 2023 15:32:44** on project **Hope-oracle** with the repository **hope\_oracle\_Jun\_16** on branch **-**. The assessment was carried out by scanning the project's codebase using the scan engine **Security Analyzer**. There are in total **8** vulnerabilities / security risks discovered during the scanning session, among which **0** critical vulnerabilities, **0** high risk vulnerabilities, **2** medium risk vulnerabilities, **2** low risk vulnerabilities, **4** informational issues.

ID	Description	Severity	Alleviation
MSA-001	Capped price for <b>HOPE</b> price	Medium risk	Acknowledged
MSA-002	Constant price is returned for the <b>BASE_CURRENCY</b>	Medium risk	Acknowledged
MSA-003	Lack of checking the <b>failoverActive</b>	Low risk	Acknowledged
MSA-004	Lack of checking the timestamp for a price	Low risk	Acknowledged
MSA-005	Zero address check	Informational	Acknowledged
MSA-006	Lack of set cap value for <b>heartbeat</b>	Informational	Acknowledged
MSA-007	Gas optimization	Informational	Acknowledged
MSA-008	Unused storage variable	Informational	Fixed

## Findings

### Critical (0)



No Critical vulnerabilities found here

### High risk (0)

No High risk vulnerabilities found here

### Medium risk (2)

#### 1. Capped price for HOPE price

 Medium risk Security Analyzer

In the `latestAnswer` function, the price of HOPE will always be less than or equal to `PRICE_SCALE`:

```
function latestAnswer() external view override returns (uint256) {
    ...
    if (hopePrice >= PRICE_SCALE) return PRICE_SCALE;
    return hopePrice;
}
```

It means that the highest price of **HOPE** from the `HOPEPriceFeed` is `PRICE_SCALE`, which may differ from the actual price and then leaves the chance for arbitrageurs to get the profit from the ecosystems:

#### File(s) Affected

contracts/HOPEPriceFeed/HOPEPriceFeed.sol #52-70

```
52  function latestAnswer() external view override returns (uint256) {
53      uint256 hopeSupply = getHOPETotalSupply();
54      uint256 reserveTotalValue;
55      uint256 hopePrice;
56
57      unchecked {
58          for (uint256 i = 0; i < reserveTokens.length; i++) {
59              TokenConfig memory config = reserveTokenConfigs[reserveTokens[i]];
60              uint256 reserveInToken = _calculateReserveAmount(hopeSupply, config);
61              uint256 reserveValueInToken = _calculateReserveValue(reserveInToken, config);
62              reserveTotalValue += reserveValueInToken;
63          }
64
65          hopePrice = reserveTotalValue / hopeSupply;
66      }
67
68      if (hopePrice >= PRICE_SCALE) return PRICE_SCALE;
69      return hopePrice;
70  }
```


#### Recommendation

Consider checking the design of the capped price for **HOPE**, monitoring their prices, and preventing malicious arbitrage with the help of governance strategies.

**Alleviation** Acknowledged

[HOPE]: HOPE's price discovery is only a temporary data source for our ecosystem, the intrinsic value of HOPE is calculated from the reserve as its price. HOPE is a stablecoin, when the price of the reserve token doubles, HOPE is anchored to \$1, at which point we will change the data source of HOPE.

## 2. Constant price is returned for the **BASE\_CURRENCY**

 Medium risk Security Analyzer

The price for the **BASE\_CURRENCY** from the `getAssetPrice` function is always **BASE\_CURRENCY\_UNIT**:

```
function getAssetPrice(address asset) public view override returns (uint256) {
    AggregatorInterface source = assetsSources[asset];
    if (asset == BASE_CURRENCY) {
        return BASE_CURRENCY_UNIT;
    } else {
        int256 price = source.latestAnswer();
        return uint256(price);
    }
}
```

It may leave the chance for arbitragers to get the profit from the ecosystems if the actual price of the `BASE\_CURRENCY` differs from the `BASE\_CURRENCY\_UNIT`.

**File(s) Affected**

contracts/HopeFallbackOracle.sol #64-73

```
64     function getAssetPrice(address asset) public view override returns (uint256) {
65         AggregatorInterface source = assetsSources[asset];
66
67         if (asset == BASE_CURRENCY) {
68             return BASE_CURRENCY_UNIT;
69         } else {
70             int256 price = source.latestAnswer();
71             return uint256(price);
72         }
73     }
```

**Recommendation**



Consider checking the design of the constant price for the **BASE\_CURRENCY** and the design of the capped **HOPE** price, monitoring their prices, and preventing malicious arbitrage with the help of governance strategies.

**Alleviation** Acknowledged

[HOPE]: **BASE\_CURRENCY** represents the base currency used in our protocol, and **BASE\_CURRENCY\_UNIT** represents the smallest unit of the base currency. This is used to determine the unit of measure in our protocol. HOPE's price discovery is only a temporary data source for our ecosystem, the intrinsic value of HOPE is calculated from the reserve as its price. HOPE is a stablecoin, when the price of the reserve token doubles, HOPE is anchored to \$1, at which point we will change the data source of HOPE.

## Low risk (2)

### 1. Lack of checking the **failoverActive**

 Low risk Security Analyzer

In the `HopeOracle` contract, the **OPERATOR\_ROLE** is able to **activate** or **deactivate** a fallback price oracle:

```
function activateFailover(address asset) external override onlyRole(OPERATOR_ROLE) {
    _activateFailover(asset);
}
```

```
}

function deactivateFailover(address asset) external override onlyRole(OPERATOR_ROLE) {
    _deactivateFailover(asset);
}
```

Common sense is that the fallback price oracle should not be used if it is deactivated. However, in the final inner **else** branch of the **getAssetPrice** function, it lacks checking if the fallback oracle is deactivated or not:

```
function getAssetPrice(address asset) public view override returns (uint256) {
    ...
} else if (priceData.failoverActive) {
    return _fallbackOracle.getAssetPrice(asset);
} else {
    int256 price = priceData.source.latestAnswer();
    if (price > 0) {
        return uint256(price);
    } else {
        return _fallbackOracle.getAssetPrice(asset);
    }
}
}
```

The same case happened in the first **else if** branch of the **getAssetPrice** function:

```
function getAssetPrice(address asset) public view override returns (uint256) {
    PriceData storage priceData = assetsPriceDatas[asset];

    if (asset == BASE_CURRENCY) {
        return BASE_CURRENCY_UNIT;
    } else if (address(priceData.source) == address(0)) {
        return _fallbackOracle.getAssetPrice(asset);
    }
}
```

### File(s) Affected

contracts/HopeOracle.sol #128-145

```
128     function getAssetPrice(address asset) public view override returns (uint256) {
129         PriceData storage priceData = assetsPriceDatas[asset];
130
131         if (asset == BASE_CURRENCY) {
132             return BASE_CURRENCY_UNIT;
133         } else if (address(priceData.source) == address(0)) {
134             return _fallbackOracle.getAssetPrice(asset);
135         } else if (priceData.failoverActive) {
136             return _fallbackOracle.getAssetPrice(asset);
137         } else {
138             int256 price = priceData.source.latestAnswer();
139             if (price > 0) {
140                 return uint256(price);
141             } else {
142                 return _fallbackOracle.getAssetPrice(asset);
143             }
144         }
145     }
```


### Recommendation


1. Consider avoiding using the fallback price oracle if it is deactivated.
2. Is using a deactivated fallback price oracle an intended design?

### Alleviation Acknowledged

**[HOPE]:** We listen for prices returned by third-party data sources that are decoupled from the market price. When the price returned by the third-party data source is out of the market price and not zero, we (the risk control organization) will start a Failover for the Token and use the price of the Alternative Prophet. When the price returned by the third-party data source is in line with the market price, we will stop the Failover of the Token and use the price of the Primary Prophet (i.e. the price of the third-party data source).

## 2. Lack of checking the timestamp for a price

 Low risk

 Security Analyzer

The `_calculateReserveValue` function in the `HOPEPriceFeed` contract fetches price from a third-party price feed but without checking the timestamp/updatedAt for a price:

```
function _calculateReserveValue(uint256 reserveAmount, TokenConfig memory config) internal view returns (uint256) {
    uint256 reservePrice = uint256(config.priceFeed.latestAnswer());
    uint256 reserveDecimals = uint256(config.priceFeed.decimals());
    ...
}

interface AggregatorInterface {
    function latestAnswer() external view returns (int256);
    function latestTimestamp() external view returns (uint256);
    ...
}
```

It may return a stale price, like a price a day ago, which is bad to calculate the actual price of the reserve.

### File(s) Affected

contracts/HOPEPriceFeed/HOPEPriceFeed.sol #79-86

```
79     function _calculateReserveValue(uint256 reserveAmount, TokenConfig memory config) internal view return
80         uint256 reservePrice = uint256(config.priceFeed.latestAnswer());
81         uint256 reserveDecimals = uint256(config.priceFeed.decimals());
82         unchecked {
83             uint256 reserveValue = (reserveAmount * reservePrice * PRICE_SCALE) / (10 ** reserveDecimals);
84             return reserveValue;
85         }
86     }
```

### Recommendation


Consider checking the timestamp/updatedAt for a price to prevent the stale price from a third-party price feed.


### Alleviation Acknowledged

-

## Informational (4)

### 1. Zero address check

 Informational

 Security Analyzer

Zero addresses assigned to address type storage variables by mistake will incur unexpected results.

### File(s) Affected

contracts/HopeOracle.sol #47-48

```
47     _setFallbackOracle(fallbackOracle);
48     _setAssetsSources(assets, sources);
```

contracts/HOPEPriceFeed/HopeAutomation.sol #28-30

```
28     _setHOPEPriceFeed(_priceFeed);
29     _setAggregator(_aggregator);
30     _setHeartbeat(_heartbeat);
```


#### Recommendation


Consider adding zero value checks for assigning values to address type storage variables.

**Alleviation** Acknowledged

The development team acknowledged this finding.

## 2. Lack of set cap value for **heartbeat**

 Informational

 Security Analyzer

The **heartbeat** stands for the interval of two price updates. If an extremely high value is set to **heartbeat** by mistake the **\_checkUpKeep** function will always return false, which makes the **performUpkeep** function works unexpectedly.

#### File(s) Affected

contracts/HOPEPriceFeed/HopeAutomation.sol #91-91

```
91     upkeepNeeded = upkeepNeeded && block.timestamp - lastTimestamp >= heartbeat;
```

contracts/HOPEPriceFeed/HopeAutomation.sol #34-36

```
34     function setHeartbeat(uint256 _heartbeat) external onlyRole(OPERATOR_ROLE) {
35         _setHeartbeat(_heartbeat);
36     }
```


#### Recommendation


Consider setting a cap value for the **heartbeat** in the **\_setHeartbeat** function.

**Alleviation** Acknowledged

The development team acknowledged this finding.

## 3. Gas optimization

 Informational

 Security Analyzer

The storage variable **roundId** in the **transmit** function is read 3 times and written once, which could be optimized by declaring a memory variable for **roundId** and reducing the number of times of read.

#### File(s) Affected

contracts/HOPEPriceFeed/HopeAggregator.sol #25-30

```
25     function transmit(uint256 _answer) external override onlyRole(OPERATOR_ROLE) {
26         roundId++;
27         int192 currentPrice = int192(int256(_answer));
28         transmissions[roundId] = Transmission(currentPrice, uint64(block.timestamp));
29         emit AnswerUpdated(currentPrice, roundId, uint64(block.timestamp));
30     }
```



contracts/HOPEPriceFeed/HopeAggregator.sol #57-60

```
57     function latestRoundData() external view override returns (uint80, int256, uint256, uint256, uint80) {
58         Transmission memory transmission = transmissions[roundId];
59         return (roundId, transmission.answer, transmission.timestamp, transmission.timestamp, roundId);
60     }
```


#### Recommendation


Consider declaring memory variables to reduce the number of read times for storage variables.

**Alleviation** Acknowledged

The development team acknowledged this finding.

## 4. Unused storage variable

 Informational

 Security Analyzer

The storage variable `transmitter` of the `HopeAggregator` contract is unused and can be removed.

#### File(s) Affected

contracts/HOPEPriceFeed/HopeAggregator.sol #11-11

```
11     address public transmitter;
```

#### Recommendation

Consider removing the unused storage variable.

**Alleviation** Fixed

The development team resolved this issue by removing the unused variable in commit 8cedac58f01f75eeca051c7bf3671a5dcbb9a4

## 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 MetaTrust'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 MetaTrust 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. MetaTrust's position is that each company and individual are responsible for their own due diligence and continuous security. MetaTrust'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 MetaTrust 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 Security Assessment AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, MetaTrust HEREBY DISCLAIMS ALL WARRANTIES,

WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, MetaTrust 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, MetaTrust 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, MetaTrust 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 MetaTrust NOR ANY OF MetaTrust'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. MetaTrust 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 MetaTrust'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 Security Assessment MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST MetaTrust WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF MetaTrust 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 MetaTrust 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.