AladdinDAO f(x) Oracle PR

Security Assessment (Summary Report)

July 10, 2024

Prepared for:

AladdinDAO

Prepared by: **Alexander Remie and Troy Sargent**

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel.

We maintain an exhaustive list of publications at https://github.com/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon.

We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on Twitter and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact, or email us at info@trailofbits.com.

Trail of Bits, Inc.
497 Carroll St., Space 71, Seventh Floor
Brooklyn, NY 11215
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

This report is considered by Trail of Bits to be public information; it is licensed to AladdinDAO under the terms of the project statement of work and has been made public at AladdinDAO's request. Material within this report may not be reproduced or distributed in part or in whole without the express written permission of Trail of Bits.

The sole canonical source for Trail of Bits publications is the Trail of Bits Publications page. Reports accessed through any source other than that page may have been modified and should not be considered authentic.

Test Coverage Disclaimer

All activities undertaken by Trail of Bits in association with this project were performed in accordance with a statement of work and agreed upon project plan.

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

Trail of Bits uses automated testing techniques to rapidly test the controls and security properties of software. These techniques augment our manual security review work, but each has its limitations: for example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. Their use is also limited by the time and resource constraints of a project.

Table of Contents

Project Summary

Contact Information

The following project manager was associated with this project:

Anne Marie Barry, Project Manager
annemarie.barry@trailofbits.com

The following engineering director was associated with this project:

Josselin Feist, Engineering Director, Blockchain
josselin.feist@trailofbits.com

The following consultants were associated with this project:

Alexander Remie, Consultant
alexander.remie@trailofbits.com

Troy Sargent, Consultant
troy.sargent@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
May 28, 2024	Pre-project kickoff call
June 4, 2024	Status update meeting #1
June 11, 2024	Delivery of report draft; report readout meeting
July 10, 2024	Delivery of summary report

Project Targets

The engagement involved a review and testing of the following target.

aladdin-v3-contracts/contracts/f(x)

Repository	https://github.com/AladdinDAO/aladdin-v3-contracts/pull/198
Version	`1c9bdad5189ed4db2cbaf321773ccdfc159d80e9`
Type	Solidity
Platform	Ethereum

Executive Summary

Engagement Overview

AladdinDAO engaged Trail of Bits to review the security of PR #198, which implements a redesign of the use of oracles within the f(x) protocol. The PR replaces the use of a TWAP oracle with the use of multiple spot price oracles along with a TWAP oracle to determine more accurate prices for the assets used within the f(x) protocol.

A team of two consultants conducted the review from May 28 to June 7, 2024, for a total of two engineer-weeks of effort. With full access to source code and documentation we performed static and dynamic testing of the changes introduced in the PR, using automated and manual processes.

Observations and Impact

This review was scoped to review the changes present in PR #198. We reviewed all of the changes and new files and uncovered four issues: one medium-severity, one low-severity, and two informational issues. The medium-severity issue (TOB-FX-3) concerns the lack of proper Chainlink price feed answer validation, which could lead to undefined behavior if the price feed provides incorrect prices.

Recommendations

Based on the findings identified during the security review, Trail of Bits recommends that AladdinDAO take the following steps:

- **Remediate the findings disclosed in this report.** These findings should be addressed as part of a direct remediation or as part of any refactor that may occur when addressing other recommendations.

- **Develop an incident response plan.** Such a plan will help the AladdinDAO team to prepare for failure scenarios and will outline the appropriate responses to them. Refer to our guidelines for guidance on creating an incident response plan.

Summary of Findings

The table below summarizes the findings of the review, including type and severity details.

ID	Title	Type	Severity
1	Missing event emission	Auditing and Logging	**Informational**
2	Missing zero-address checks in constructors	Data Validation	**Informational**
3	Lack of validation of Chainlink price feed answers	Data Validation	**Medium**
4	Lack of validation of updates to system configuration parameters	Data Validation	**Low**

Detailed Findings

1. Missing event emission

Severity: **Informational**	Difficulty: **Low**
Type: Auditing and Logging	Finding ID: TOB-FX-1
Target: `contracts/f(x)/oracle/FxBTCDerivativeOracleBase.sol`	

Description

The critical operation `updateOnchainSpotEncodings` does not emit an event. Having an event emitted to reflect changes to this critical storage variable will allow other system/off-chain components to detect suspicious behavior in the system.

```
 88    function updateOnchainSpotEncodings(bytes memory encodings) external
onlyOwner {
 89      // validate encoding
 90      uint256[] memory prices = _getSpotPriceByEncoding(encodings);
 91      if (prices.length == 0) revert();
 92
 93      onchainSpotEncodings_BTCDerivativeUSD = encodings;
 94    }
```

Figure 1.1: The `updateOnchainSpotEncodings` function in `FxBTCDerivativeOracleBase.sol#L88-L94`

Events generated during contract execution aid in monitoring, baselining of behavior, and detecting suspicious activity. Without events, users and blockchain-monitoring systems cannot easily detect behavior that falls outside the baseline conditions; malfunctioning contracts and attacks could go undetected.

Recommendations

Short term, emit an event in the `updateOnchainSpotEncodings` function.

Long term, ensure all state-changing operations are always accompanied by events. In addition, use static analysis tools such as Slither to help prevent such issues in the future.

2. Missing zero-address checks in constructors

Severity: **Informational**	Difficulty: **High**
Type: Data Validation	Finding ID: TOB-FX-2
Target: `contracts/f(x)/oracle/*.sol`	

Description

None of the constructors in the various oracle contracts validate that their address arguments do not equal the zero address. As a result, important immutable state variables might be set to the zero address during deployment, effectively making the given contract unusable and requiring a redeployment.

```
34    constructor(address _Chainlink_BTC_USD_Twap) {
35      Chainlink_BTC_USD_Twap = _Chainlink_BTC_USD_Twap;
36
37      _updateMaxPriceDeviation(1e16); // 1%
38    }
```

Figure 2.1: The constructor in `FxBTCDerivativeOracleBase.sol#L34-L38`

```
18    address public immutable Chainlink_BTC_USD_Twap;
```

Figure 2.2: The `Chainlink_BTC_USD_Twap` variable in `FxBTCDerivativeOracleBase.sol#L18`

Recommendations

Short term, add a check to each constructor to ensure that each address argument does not equal the zero address.

Long term, use the Slither static analyzer to catch common issues such as this one. Consider integrating a Slither scan into the project's CI pipeline, pre-commit hooks, or build scripts.

3. Lack of validation of Chainlink price feed answers

Severity: **Medium**	Difficulty: **High**
Type: Data Validation	Finding ID: TOB-FX-3
Target: `contracts/f(x)/oracle/FxSpotOracleBase.sol`	

Description

The validation of the price returned by Chainlink is incomplete, which means that incorrect prices could be used in the protocol. This could lead to loss of funds or otherwise cause internal accounting errors that might break the correct functioning of the protocol.

```
 46    function _readSpotPriceByChainlink(bytes32 encoding) internal view returns
(uint256) {
 47      address aggregator;
 48      uint256 scale;
 49      uint256 heartbeat;
 50      assembly {
 51        aggregator := shr(96, encoding)
 52        scale := and(shr(32, encoding), 0xffffffffffffffff)
 53        heartbeat := and(encoding, 0xffffffff)
 54      }
 55      (, int256 answer, , uint256 updatedAt, ) =
AggregatorV3Interface(aggregator).latestRoundData();
 56      if (block.timestamp - updatedAt > heartbeat) revert("expired");
 57      return uint256(answer) * scale;
 58    }
```

Figure 3.1: The `_readSpotPriceByChainlink` function in `FxSpotOracleBase.sol#L46-L58`

Because the Chainlink-returned price is of type `int256`, the following two scenarios could happen:

- The price feed answer could be a negative integer. First off, this is highly unlikely for the particular price feeds used by f(x). However, if a negative integer is returned, it will be unsafely cast to an unsigned integer (`uint256`) on line 57 of `_readSpotPriceByChainlink`. This will likely lead to a revert because the unsigned value of a cast signed negative integer will likely be very high, but it might also lead to the use of an incorrect price.

- A Chainlink price feed can also return zero as the answer. In this case, the `isValid` Boolean will be set to `false`, which will ensure the incorrect price is not actually used, as shown in figure 3.2.

```
103    function getPrice()
104      external
105      view
106      override
107      returns (
108        bool isValid,
109        uint256 twap,
110        uint256 minPrice,
111        uint256 maxPrice
112      )
113    {
114      twap = _getLSDUSDTwap();
115      (minPrice, maxPrice) = _getLSDMinMaxPrice(twap);
116      unchecked {
117        isValid = (maxPrice - minPrice) * PRECISION < maxPriceDeviation * minPrice;
118      }
119    }
```

Figure 3.2: The `getPrice` function in `FxLSDOracleV2Base.sol#L103-L119`

Exploit Scenario

The Chainlink price feed returns a negative price, which when cast to an unsigned integer is considered valid. As a result, an incorrect price is used.

Recommendations

Short term, add a check inside the `_readSpotPriceByChainlink` function that ensures `answer` is greater than `0`.

Long term, add validation of returned results from all external sources.

4. Lack of validation of updates to system configuration parameters

Severity: **Low**	Difficulty: **Medium**
Type: Data Validation	Finding ID: TOB-FX-4
Target: `contracts/f\(x\)/oracle/FxBTCDerivativeOracleBase.sol`, `contracts/f\(x\)/v2/LeveragedTokenV2.sol`, `contracts/price-oracle/spot/SpotPriceOracle.sol`	

Description

Several configuration functions (figures 4.1–4.3) do not validate that updates to configuration parameters actually result in a change in value. Although setting a parameter to its current value is benign, it may obscure a logical error in a peripheral program that would be readily identifiable if the update were to revert and raise an alarm.

```
108    function _updateMaxPriceDeviation(uint256 newMaxPriceDeviation) private {
109      uint256 oldMaxPriceDeviation = maxPriceDeviation;
110      maxPriceDeviation = newMaxPriceDeviation;
111
112      emit UpdateMaxPriceDeviation(oldMaxPriceDeviation, newMaxPriceDeviation);
113    }
```

Figure 4.1: The `_updateMaxPriceDeviation` function in `FxBTCDerivativeOracleBase.sol#L108-L113`

```
129    function _updateCoolingOffPeriod(uint256 _newCoolingOffPeriod) private {
130      uint256 oldCoolingOffPeriod = coolingOffPeriod;
131      coolingOffPeriod = _newCoolingOffPeriod;
132
133      emit UpdateCoolingOffPeriod(oldCoolingOffPeriod, _newCoolingOffPeriod);
134    }
```

Figure 4.2: The `_updateCoolingOffPeriod` function in `LeveragedTokenV2.sol#L129-L134`

```
130    function updateReader(uint256 poolType, address newReader) external
onlyOwner {
131      address oldReader = readers[poolType];
132      readers[poolType] = newReader;
133
134      emit UpdateReader(poolType, oldReader, newReader);
135    }
```

Figure 4.3: The `updateReader` function in `SpotPriceOracle.sol#L130-L135`

Recommendations

Short term, add validation to these functions to require that the new value is not equal to the previous value.

Long term, add validation to all configuration functions to ensure they either perform a configuration state update or cause a revert.

A. Vulnerability Categories

The following tables describe the vulnerability categories, severity levels, and difficulty levels used in this document.

Vulnerability Categories	
Category	**Description**
Access Controls	Insufficient authorization or assessment of rights
Auditing and Logging	Insufficient auditing of actions or logging of problems
Authentication	Improper identification of users
Configuration	Misconfigured servers, devices, or software components
Cryptography	A breach of system confidentiality or integrity
Data Exposure	Exposure of sensitive information
Data Validation	Improper reliance on the structure or values of data
Denial of Service	A system failure with an availability impact
Error Reporting	Insecure or insufficient reporting of error conditions
Patching	Use of an outdated software package or library
Session Management	Improper identification of authenticated users
Testing	Insufficient test methodology or test coverage
Timing	Race conditions or other order-of-operations flaws
Undefined Behavior	Undefined behavior triggered within the system

Severity Levels	
Severity	**Description**
Informational	The issue does not pose an immediate risk but is relevant to security best practices.
Undetermined	The extent of the risk was not determined during this engagement.
Low	The risk is small or is not one the client has indicated is important.
Medium	User information is at risk; exploitation could pose reputational, legal, or moderate financial risks.
High	The flaw could affect numerous users and have serious reputational, legal, or financial implications.

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploitation was not determined during this engagement.
Low	The flaw is well known; public tools for its exploitation exist or can be scripted.
Medium	An attacker must write an exploit or will need in-depth knowledge of the system.
High	An attacker must have privileged access to the system, may need to know complex technical details, or must discover other weaknesses to exploit this issue.

B. Fix Review Results

When undertaking a fix review, Trail of Bits reviews the fixes implemented for issues identified in the original report. This work involves a review of specific areas of the source code and system configuration, not comprehensive analysis of the system.

On June 24, 2024, Trail of Bits reviewed the fixes and mitigations implemented by the Aladdin team for the issues identified in this report. We reviewed each fix to determine its effectiveness in resolving the associated issue.

In summary, of the four issues described in this report, Aladdin has resolved two issues and has not resolved the remaining two issues.

ID	Title	Status
1	Missing event emission	Unresolved
2	Missing address zero checks in constructors	Unresolved
3	Chainlink price of zero and negative handled incorrectly	Resolved
4	Lack of validation when updating system configurations	Resolved

Detailed Fix Review Results

TOB-FX-1: Missing event emission

Unresolved.

The client provided the following context for this finding's fix status:

> *There are lots of other tools to monitor storage changes. We don't really need to emit such events. So we decided not to fix this issue for now.*

TOB-FX-2: Missing address zero checks in constructors

Unresolved.

The client provided the following context for this finding's fix status:

> *During the deployment our deploy script will make sure this never happens. So we decided not to fix this issue for now.*

TOB-FX-3: Chainlink price of zero and negative handled incorrectly

Resolved in PR#206. Validation was added to ensure that the price returned from Chainlink is not negative. The case of a zero price being returned is handled elsewhere; this will cause the `isValid` variable to be set to `false`.

TOB-ADFX-4: Lack of validation when updating system configurations

Resolved in PR#207. All of the mentioned functions now include checks that ensure that the new value differs from the current value. In addition, a "max cooling off period" was implemented and is now checked inside the `_updateCoolingOffPeriod` function.

C. Fix Review Status Categories

The following table describes the statuses used to indicate whether an issue has been sufficiently addressed.

Fix Status	
Status	**Description**
Undetermined	The status of the issue was not determined during this engagement.
Unresolved	The issue persists and has not been resolved.
Partially Resolved	The issue persists but has been partially resolved.
Resolved	The issue has been sufficiently resolved.