Ante Protocol

Fix Review

June 22, 2022

Prepared for:
REDACTED
Ante Labs LLC

Prepared by: **David Pokora 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 80+ 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.
228 Park Ave S #80688
New York, NY 10003
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

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.

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.

Table of Contents

Executive Summary

Engagement Overview

Ante Labs LLC engaged Trail of Bits to review the security of its Ante Protocol. From May 16 to May 20, 2022, a team of two consultants conducted a security review of the client-provided source code, with two person-weeks of effort. Details of the project's scope, timeline, test targets, and coverage are provided in the original audit report.

Ante Labs LLC contracted Trail of Bits to review the fixes implemented for issues identified in the original report. From June 12 to June 13, 2022, one consultant conducted a review of the client-provided source code.

Summary of Findings

The original audit uncovered significant flaws that could impact system confidentiality, integrity, or availability. A summary of the original findings is provided below.

EXPOSURE ANALYSIS

Severity	*Count*
High	2
Medium	1
Informational	2

CATEGORY BREAKDOWN

Category	*Count*
Auditing and Logging	1
Data Validation	3
Denial of Service	1

Overview of Fix Review Results

Ante Labs LLC has sufficiently addressed most of the issues described in the original audit report.

Project Summary

Contact Information

The following managers were associated with this project:

Dan Guido, Account Manager
dan@trailofbits.com

Sam Greenup, Project Manager
sam.greenup@trailofbits.com

The following engineers were associated with this project:

David Pokora, Consultant
david.pokora@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 12, 2022	Pre-project kickoff call
May 23, 2022	Delivery of report draft and report readout meeting
June 22, 2022	Delivery of final report
June 22, 2022	Delivery of fix review report

Project Methodology

Our work in the fix review included the following:

- A review of the findings in the original audit report
- A manual review of the client-provided source code and configuration material

Project Targets

The engagement involved a review of the fixes implemented in the following target.

Ante Core

Repository	https://github.com/antefinance/ante-v0-core
Version	`fdd0d8d68a5697415cde511aa5dc98c469871bb7`
Type	Solidity smart contracts
Platform	Ethereum-based nodes

Summary of Fix Review Results

The table below summarizes each of the original findings and indicates whether the issue has been sufficiently resolved.

ID	Title	Status
1	AntePoolFactory does not validate create2 return addresses	Resolved
2	Events emitted during critical operations omit certain details	Resolved
3	Insufficient gas can cause AnteTests to produce false positives	Unresolved
4	Looping over an array of unbounded size can cause a denial of service	Partially Resolved
5	Reentrancy into AntePool.checkTest scales challenger eligibility amount	Resolved

Detailed Fix Review Results

1. AntePoolFactory does not validate create2 return addresses	
Status: **Resolved**	
Severity: **Informational**	Difficulty: **High**
Type: Data Validation	Finding ID: TOB-ANTE-1
Target: `contracts/AntePoolFactory.sol`	

Description
The `AntePoolFactory` uses the `create2` instruction to deploy an `AntePool` and then initializes it with an already-deployed `AnteTest` address. However, the `AntePoolFactory` does not validate the address returned by `create2`, which will be the zero address if the deployment operation fails.

```
bytes memory bytecode = type(AntePool).creationCode;
bytes32 salt = keccak256(abi.encodePacked(testAddr));

assembly {
    testPool := create2(0, add(bytecode, 0x20), mload(bytecode), salt)
}

poolMap[testAddr] = testPool;
allPools.push(testPool);

AntePool(testPool).initialize(anteTest);

emit AntePoolCreated(testAddr, testPool);
```

Figure 1.1: `contracts/AntePoolFactory.sol#L35-L47`

This lack of validation does not currently pose a problem, because the simplicity of `AntePool` contracts helps prevent deployment failures (and thus the return of the zero address). However, deployment issues could become more likely in future iterations of the Ante Protocol.

Fix Analysis
This issue has been resolved. The Ante Labs LLC team has added a zero address check for the result of the `create2` operation performed in the `AntePoolFactory`.

2. Events emitted during critical operations omit certain details	
Status: **Resolved**	
Severity: **Informational**	Difficulty: **N/A**
Type: Auditing and Logging	Finding ID: TOB-ANTE-2
Target: `contracts/AntePoolFactory.sol`, `contracts/AntePool.sol`	

Description

Events are generally emitted for all critical state-changing operations within the system. However, the `AntePoolCreated` event emitted by the `AntePoolFactory` does not capture the address of the `msg.sender` that deployed the `AntePool`. This information would help provide a more complete audit trail in the event of an attack, as the `msg.sender` often refers to the externally owned account that sent the transaction but could instead refer to an intermediate smart contract address.

```
emit AntePoolCreated(testAddr, testPool);
```

Figure 2.1: `contracts/AntePoolFactory.sol#L47`

Additionally, consider having the `AntePool.updateDecay` method emit an event with pool share parameters.

Fix Analysis

This issue has been resolved. The Ante Labs LLC team has added the recommended events and event parameters to improve the audit trail in the event of a system failure.

3. Insufficient gas can cause AnteTests to produce false positives

Status: **Unresolved**	
Severity: **High**	Difficulty: **High**
Type: Data Validation	Finding ID: TOB-ANTE-3
Target: `contracts/AntePool.sol`	

Description

Once challengers have staked ether and the challenger delay has passed, they can submit transactions to predict that a test will fail and to earn a bonus if it does. An attacker could manipulate the result of an `AnteTest` by providing a limited amount of gas to the `checkTest` function, forcing the test to fail. This is because the `anteTest.checkTestPasses` function receives 63/64 of the gas provided to `checkTest` (per the 63/64 gas forwarding rule), which may not be enough.

This issue stems from the use of a `try-catch` statement in the `_checkTestNoRevert` function, which causes the function to return `false` when an EVM exception occurs, indicating a test failure. We set the difficulty of this finding to high, as the outer call will also revert with an out-of-gas exception if it requires more than 1/64 of the gas; however, other factors (e.g., the block gas limit) may change in the future, allowing for a successful exploitation.

```
if (!_checkTestNoRevert()) {
    updateDecay();
    verifier = msg.sender;
    failedBlock = block.number;
    pendingFailure = true;

    _calculateChallengerEligibility();
    _bounty = getVerifierBounty();

    uint256 totalStake = stakingInfo.totalAmount.add(withdrawInfo.totalAmount);
    _remainingStake = totalStake.sub(_bounty);
```

Figure 3.1: Part of the `checkTest` function

```
/// @return passes bool if the Ante Test passed
function _checkTestNoRevert() internal returns (bool) {
    try anteTest.checkTestPasses() returns (bool passes) {
        return passes;
    } catch {
        return false;
    }
}
```

Figure 3.2: `contracts/AntePool.sol#L567-L573`

Fix Analysis

This issue is currently unresolved. The Ante Labs LLC team has indicated that it is exploring options to fix this issue.

4. Looping over an array of unbounded size can cause a denial of service

Status: **Partially Resolved**	
Severity: **Medium**	Difficulty: **High**
Type: Denial of Service	Finding ID: TOB-ANTE-4
Target: `contracts/AntePool.sol`	

Description

If an `AnteTest` fails, the `_checkTestNoRevert` function will return `false`, causing the `checkTest` function to call `_calculateChallengerEligibility` to compute `eligibleAmount`; this value is the total stake of the eligible challengers and is used to calculate the proportion of `_remainingStake` owed to each challenger. To calculate `eligibleAmount`, the `_calculateChallengerEligibility` function loops through an unbounded array of challenger addresses. When the number of challengers is large, the function will consume a large quantity of gas in this operation.

```
function _calculateChallengerEligibility() internal {
    uint256 cutoffBlock = failedBlock.sub(CHALLENGER_BLOCK_DELAY);
    for (uint256 i = 0; i < challengers.addresses.length; i++) {
        address challenger = challengers.addresses[i];
        if (eligibilityInfo.lastStakedBlock[challenger] < cutoffBlock) {
            eligibilityInfo.eligibleAmount = eligibilityInfo.eligibleAmount.add(
                _storedBalance(challengerInfo.userInfo[challenger], challengerInfo)
            );
        }
    }
}
```

Figure 4.1: `contracts/AntePool.sol#L553-L563`

However, triggering an out-of-gas error would be costly to an attacker; the attacker would need to create many accounts through which to stake funds, and the amount of each stake would decay over time.

Fix Analysis

This issue has been partially resolved. The Ante Labs LLC team has added an upper bound to the number of users that can partake in a pool. This should prevent out-of-gas errors with the gas costs on the current fork of Ethereum. Future forks of Ethereum may change the gas costs associated with operations, which may affect the viability of this fix. However, the Ante Labs LLC team has indicated that this is a short-term fix and is planning a long-term fix that minimizes gas costs.

5. Reentrancy into AntePool.checkTest scales challenger eligibility amount

Status: **Resolved**	
Severity: **High**	Difficulty: **Medium**
Type: Data Validation	Finding ID: TOB-ANTE-5
Target: `contracts/AntePool.sol`	

Description

A malicious `AnteTest` or underlying contract being tested can trigger multiple failed `checkTest` calls by reentering the `AntePool.checkTest` function. With each call, the `_calculateChallengerEligibility` method increases the `eligibleAmount` instead of resetting it, causing the `eligibleAmount` to scale unexpectedly with each reentrancy.

```
function checkTest() external override testNotFailed {
    require(challengers.exists(msg.sender), "ANTE: Only challengers can checkTest");
    require(
        block.number.sub(eligibilityInfo.lastStakedBlock[msg.sender]) >
CHALLENGER_BLOCK_DELAY,
        "ANTE: must wait 12 blocks after challenging to call checkTest"
    );

    numTimesVerified = numTimesVerified.add(1);
    lastVerifiedBlock = block.number;
    emit TestChecked(msg.sender);
    if (!_checkTestNoRevert()) {
        updateDecay();
        verifier = msg.sender;
        failedBlock = block.number;
        pendingFailure = true;

        _calculateChallengerEligibility();
        _bounty = getVerifierBounty();

        uint256 totalStake = stakingInfo.totalAmount.add(withdrawInfo.totalAmount);
        _remainingStake = totalStake.sub(_bounty);

        emit FailureOccurred(msg.sender);
    }
}
```

Figure 5.1: `contracts/AntePool.sol#L292-L316`

```
function _calculateChallengerEligibility() internal {
    uint256 cutoffBlock = failedBlock.sub(CHALLENGER_BLOCK_DELAY);
    for (uint256 i = 0; i < challengers.addresses.length; i++) {
        address challenger = challengers.addresses[i];
        if (eligibilityInfo.lastStakedBlock[challenger] < cutoffBlock) {
            eligibilityInfo.eligibleAmount = eligibilityInfo.eligibleAmount.add(
                _storedBalance(challengerInfo.userInfo[challenger], challengerInfo)
            );
        }
    }
}
```

Figure 5.2: `contracts/AntePool.sol#L553-L563`

Fix Analysis

This issue has been resolved. The Ante Labs LLC team has added OpenZeppelin's `ReentrancyGuard` modifiers to the appropriate methods to prevent this vulnerability.

A. 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.

B. 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.