Unibot Router

Security Assessment (Summary Report)

January 8, 2024

Prepared for:
Unibot

Prepared by: **Anish Naik and Robert Schneider**

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.
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.

Table of Contents

Project Summary

Contact Information

The following project manager was associated with this project:

Brooke Langhorne, Project Manager
brooke.langhorne@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:

Anish Naik, Consultant
anish.naik@trailofbits.com

Robert Schneider, Consultant
robert.schneider@trailofbits.com

Project Timeline

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

Date	Event
December 27, 2023	Pre-project kickoff call
January 3, 2024	Delivery of summary report draft
January 3, 2024	Report readout meeting
January 9, 2024	Delivery of summary report

Executive Summary

Engagement Overview

Unibot engaged Trail of Bits to review the security of its Unibot Router at commit hash `0f03bfd`. The Unibot Router (the `UnibotRouter_03` contract) is a fork of Uniswap's `SwapRouter` contract with a few modifications. The core modifications include custom multicall functions, functionality to retrieve fees on Unibot transactions, and the ability to interface with allowlisted decentralized exchanges (DEXes) outside of Uniswap V2 or V3.

A team of two consultants conducted the review from December 26 to December 29, 2023, for a total of eight engineer-days of effort. With full access to source code and documentation, we performed a manual review of the diff between the original `SwapRouter` contract and the `UnibotRouter_03` contract.

Observations and Impact

During our review, we focused on the following objectives. First, we reviewed the access controls on system operations to ensure that an attacker cannot bypass these privileges to update the allowlist of DEXes, disable swapping, or update any critical addresses. Next, we reviewed the custom multicall functions for general correctness and to identify whether they introduced any unexpected behavior. Next, we reviewed the fee capability to validate that an attacker cannot steal funds from the system through reentrancy or the use of arbitrary, attacker-controlled tokens. Finally, we validated that an attacker cannot interface with a DEX that is not allowlisted by the system.

We identified two core concerns during the audit. First, there were no unit tests for the custom functionality added by the Unibot Team (TOB-UNIBOT-1). This is problematic since any edge cases within the functionality may remain unidentified. Additionally, any bugs introduced by code changes made on top of the reviewed commit may go unnoticed. Second, no functions that were added by the team emit events (TOB-UNIBOT-2). This is of concern since any suspicious activity cannot be identified and triaged effectively by offchain tooling.

Recommendations

Going forward, we recommend that the Unibot team invest further efforts into developing a test suite. This test suite will aid in ensuring that there are no edge cases that can cause unexpected behavior and no additional code changes can lead to the introduction of latent bugs, and it will also improve the larger security posture and guarantees of the system.

Summary of Findings

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

ID	Title	Type	Severity
1	Missing unit tests	Testing	High
2	Lack of events for critical operations	Auditing and Logging	Low

Detailed Findings

1. Missing unit tests	
Severity: **High**	Difficulty: **Low**
Type: Testing	Finding ID: TOB-UNIBOT-1
Target: `contracts/UnibotRouter_03_edit.sol`	

Description

There were no unit tests provided for the target contract. This is problematic for a few reasons:

1. Smart contracts often have an intricate network of dependencies. Modifying one section of the code can have unforeseen consequences on other sections. Testing helps detect these ripple effects that may not be immediately apparent.
2. Even small changes in the code can introduce vulnerabilities. In the case of smart contracts, these vulnerabilities can be exploited, resulting in substantial financial losses. Testing ensures that the new code does not create any security weaknesses.
3. The system lacks programmatic guarantees around the data validation, access controls, and arithmetic operations performed by the system.
4. Even if the original project was thoroughly tested, the new changes may not be covered by the existing tests. Implementing new tests will help ensure that the modified lines of code are adequately covered.
5. Writing tests can also serve as a form of documentation, helping developers and security researchers understand the intended functionality and any changes made to the contract.
6. For projects that have a community of users or stakeholders, it is essential to maintain trust. One way to reinforce this trust is by demonstrating, through testing, that the contract remains secure and reliable even after changes have been made.

Recommendations

Short term, write unit tests for any existing and future custom functionality.

Long term, consider integrating the test suite of the `SwapRouter` contract and updating any tests that are not relevant or must be updated. This will provide additional

programmatic guarantees that any changes made by the Unibot team do not adversely affect the expected behavior of the router contract.

2. Lack of events for critical operations

Severity: **Low**	Difficulty: **Low**
Type: Auditing and Logging	Finding ID: TOB-UNIBOT-2
Target: `contracts/UnibotRouter_03_edit.sol`	

Description

Several critical operations do not trigger events. As a result, it will be difficult to review the correct behavior of the contracts once they have been deployed.

For example, the `setUnibotSettingAddress` function, which is a privileged operation used to set critical state variables, does not emit an event that specifies which system state variable was updated (figure 2.1).

```
function setUnibotSettingAddress(uint256 actionIndex, address _address, uint256
_factoryValue) external {
    require(msg.sender == ADMIN_WALLET_ADDR, "NO_AUTH");
    // 1000 - FACTORY_SET_VALUE

    // 7777 - SET_ADMIN_WALLET
    // 7555 - SET_FEE_WALLET
    // 9999 = SWAP_ENABLED
    // 9111 = SWAP_DISABLED

    if (actionIndex == 1000) {  // => FACTORY_SET_VALUE
        require(_factoryValue == 0 || _factoryValue == 2 || _factoryValue == 3,
"Invalid: 0 (disabled) || 2 (UniV2) || 3 (UniV3)");
        validFactoryVersion[_address] = _factoryValue;

    } else if (actionIndex == 7777) { // => SET_ADMIN_WALLET
        ADMIN_WALLET_ADDR = _address;
    } else if  (actionIndex == 7555) { // => SET_FEE_WALLET
        FEE_WALLET_ADDR = _address;

    } else if  (actionIndex == 9999) { // => SWAP_ENABLED
        SWAP_ENABLED = true;
    } else if  (actionIndex == 9111) { // => SWAP_DISABLED
        SWAP_ENABLED = false;
```

```
    }
}
```

Figure 2.1: The `setUnibotSettingAddress` function does not emit events (`contracts/UnibotRouter_03_edit.sol#L3732-L3755`)

Without events, users and blockchain-monitoring systems cannot easily detect suspicious behavior.

Exploit Scenario

Eve, an attacker, is able to take ownership of the `ADMIN_WALLET_ADDR` account and calls the `setUnibotSettingAddress` function to update the `FEE_WALLET_ADDR` to her own address, which allows her to steal user funds. Alice, a Unibot team member, is unaware of the change and does not raise a security incident.

Recommendations

Short term, add events for all critical operations that result in state changes. Events aid in contract monitoring and the detection of suspicious behavior.

Long term, consider using a blockchain-monitoring system to track any suspicious behavior in the contracts. The system relies on several contracts to behave as expected. A monitoring mechanism for critical events would quickly detect any compromised system components.

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 January 5, 2024, Trail of Bits reviewed the fixes and mitigations implemented by the Unibot team for the issues identified in this report. We reviewed each fix to determine its effectiveness in resolving the associated issue.

In summary, the Unibot team has resolved the two issues described in this report. For additional information, please see the Detailed Fix Review Results below.

ID	Title	Status
1	Missing unit tests	**Resolved**
2	Lack of events for critical operations	**Resolved**

Detailed Fix Review Results

TOB-UNIBOT-1: Missing unit tests
Resolved in 5b42e9c. The Unibot team has added unit tests that validate the expected functionality of any custom code that was added on top of the original `SwapRouter` contract.

TOB-UNIBOT-2: Lack of events for critical operations
Resolved in 5b42e9c. The Unibot team has added an event for the `setUnibotSettingAddress` 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.