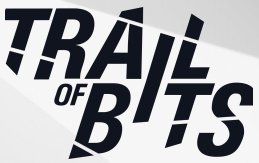

ScopeLift Stealth Address Contracts

Security Assessment (Summary Report)

February 26, 2024

Prepared for:
Ben DiFrancesco and Gary Ghayrat
ScopeLift

Prepared by: **Anish Naik**

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:

Mary O'Brien, Project Manager
mary.obrien@trailofbits.com

The following engineering director was associated with this project:

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

The following consultant was associated with this project:

Anish Naik, Consultant
anish.naik@trailofbits.com

Project Timeline

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

Date	Event
February 8, 2024	Pre-project kickoff call
February 16, 2024	Delivery of report draft and report readout meeting
February 26, 2024	Delivery of summary report with fix review appendix

Executive Summary

Engagement Overview

ScopeLift engaged Trail of Bits to review the security of its `ERC5564Announcer` and `ERC6538Registry` contracts at commit hash 8131727. The `ERC5564Announcer` contract implements a portion of the ERC-5564 standard and is responsible for emitting events when a transaction is sent to a stealth address. The `ERC6538Registry` contract implements the ERC-6538 standard and acts as a registry for stealth meta-addresses. A stealth address allows users to have privacy-preserving accounts. A stealth meta-address is a set of one or two public keys that can be used to compute a stealth address.

One consultant conducted the review from February 12 to February 16, 2024, for a total of one engineer-week of effort. With full access to source code and documentation, we performed static and dynamic testing of the codebase, using automated and manual processes.

Observations and Impact

During the audit, we had a few core areas of focus. First, we assessed whether the two in-scope contracts comply with their associated specifications. Next, we reviewed the signature validation performed in the `ERC6538Registry` contract for general correctness and compliance with EIP-1271 and EIP-712. Additionally, we checked whether the process is resilient to replay attacks, signature malleability attacks (TOB-LIFT-1), front-running attacks, and blockchain forks. We also reviewed whether the two contracts can be deployed to the same address (using the `CREATE2` opcode) across all EVM-compatible chains (assuming the same deployer address and salt). Finally, we looked for any EVM-level deviations across EVM-compatible chains (e.g., Arbitrum or Optimism) to assess whether these deviations could adversely affect the contracts' expected behavior.

Outside of the one reported informational issue (TOB-LIFT-1), the codebase is compliant with the ERC-5564 and ERC-6538 specifications and is resilient to common cryptographic attacks associated with signature validation. Additionally, the codebase follows best practices with regards to data validation, low-level memory manipulation, and testing.

Recommendations

Before deployment, we recommend that the ScopeLift team remediate the findings disclosed in this report. Additionally, we recommend that the team complete and finalize the ERC-5564 and ERC-6538 specifications with a focus on the security considerations related to applications and users that wish to integrate. For example, the security considerations for ERC-6538 should highlight that applications must correctly implement EIP-1271 (e.g., smart contract wallets) and that general users should be aware of the risks related to key theft.

Codebase Maturity Evaluation

Trail of Bits uses a traffic-light protocol to provide each client with a clear understanding of the areas in which its codebase is mature, immature, or underdeveloped. Deficiencies identified here often stem from root causes within the software development life cycle that should be addressed through standardization measures (e.g., the use of common libraries, functions, or frameworks) or training and awareness programs.

Category	Summary	Result
Arithmetic	The in-scope contracts use Solidity version 0.8.23, which has overflow protection by default. Neither contract has any arithmetic calculations or operations that require review.	**Not Applicable**
Auditing	Most state-changing operations emit the necessary events for proper off-chain monitoring and compliance with ERC-5564 and ERC-6538.	**Satisfactory**
Authentication / Access Controls	The `ERC6538Registry` contract uses ECDSA signature validation for authorizing the registration of stealth meta-addresses. This logic is resilient to common cryptographic attacks related to signature validation.	**Satisfactory**
Complexity Management	The codebase has a clear separation of logic, and all functions are easy to test and reason through.	**Strong**
Cryptography and Key Management	The `ERC6538Registry` contract uses the `ecrecover` function for signature recovery without any additional logic to prevent signature malleability (TOB-LIFT-1).	**Moderate**
Decentralization	Both contracts are permissionless by design, are not upgradeable, and have no privileged actors that may act as single points of failure.	**Strong**
Documentation	The two contracts have sufficient inline documentation and have thorough specifications as reference materials. However, the ERC-6538 specification remains incomplete and should be finalized. Additionally, both specifications should be peer-reviewed before the deployment of the associated contracts.	**Satisfactory**

Low-Level Manipulation	The `ERC6538Registry` contract uses low-level assembly to retrieve user signatures from memory. We did not identify any concerns with this logic. However, we recommend documenting the use of assembly more thoroughly in the contract.	**Satisfactory**
Testing and Verification	The codebase does not have any unit tests but has extensive fuzz tests. We recommend adding unit tests to test specific edge cases since it is not guaranteed that any given fuzz test will test every specific input.	**Satisfactory**
Transaction Ordering	The codebase is resilient to traditional front-running attacks related to signature validation. We did not identify any additional transaction ordering vectors that could adversely affect the expected behavior of the system.	**Satisfactory**

Summary of Findings

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

ID	Title	Type	Severity
1	Insufficient protection against signature malleability	Cryptography	**Informational**

1. Insufficient protection against signature malleability

Severity: **Informational**	Difficulty: **Low**
Type: Cryptography	Finding ID: TOB-LIFT-1
Target: `src/ERC6538Registry.sol`	

Description

User signatures for stealth meta-address registration have no protection against signature malleability.

The `ERC6538Registry.registerKeysOnBehalf` function can be used to register a stealth meta-address on behalf of a user through a user-provided signature. The signature recovery process is performed by the `ecrecover` Solidity precompile.

```
  59    /// @notice Sets the `registrant`'s stealth meta-address for the given
scheme ID.
  60    /// @param registrant Address of the registrant.
  61    /// @param schemeId Identifier corresponding to the applied stealth address
scheme, e.g. 1 for
  62    /// secp256k1, as specified in ERC-5564.
  63    /// @param signature A signature from the `registrant` authorizing the
registration.
  64    /// @param stealthMetaAddress The stealth meta-address to register.
  65    /// @dev Supports both EOA signatures and EIP-1271 signatures.
  66    /// @dev Reverts if the signature is invalid.
  67    function registerKeysOnBehalf(
  68     address registrant,
  69     uint256 schemeId,
  70     bytes memory signature,
  71     bytes calldata stealthMetaAddress
  72    ) external {
  73     bytes32 dataHash;
  74     address recoveredAddress;
  75
  76     unchecked {
  77       dataHash = keccak256(
  78         abi.encodePacked(
  79           "\x19\x01",
  80           DOMAIN_SEPARATOR(),
  81           keccak256(
  82             abi.encode(
  83               ERC6538REGISTRY_ENTRY_TYPE_HASH, schemeId, stealthMetaAddress,
nonceOf[registrant]++
  84             )
  85           )
```

```
 86          )
 87        );
 88      }
 89
 90      if (signature.length == 65) {
 91        bytes32 r;
 92        bytes32 s;
 93        uint8 v;
 94        assembly ("memory-safe") {
 95          r := mload(add(signature, 0x20))
 96          s := mload(add(signature, 0x40))
 97          v := byte(0, mload(add(signature, 0x60)))
 98        }
 99        recoveredAddress = ecrecover(dataHash, v, r, s);
100      }
101
102      if (
103        (
104          (recoveredAddress == address(0) || recoveredAddress != registrant)
105            && (
106              IERC1271(registrant).isValidSignature(dataHash, signature)
107                != IERC1271.isValidSignature.selector
108            )
109        )
110      ) revert ERC6538Registry__InvalidSignature();
111
112      stealthMetaAddressOf[registrant][schemeId] = stealthMetaAddress;
113      emit StealthMetaAddressSet(registrant, schemeId, stealthMetaAddress);
114    }
```

Figure 1.1: User signatures submitted to `registerKeysOnBehalf` are malleable. (`stealth-address-erc-contracts/src/ERC6538Registry.sol#L59–L114`)

The `ecrecover` precompile, by default, does not protect against signature malleability. Thus, user signatures are nonunique, so an attacker can generate another valid signature without having access to the user's private key.

However, this signature malleability vector is benign because of the highlighted logic in figure 1.1 (lines 76–88). Since the signed hash's preimage holds the stealth meta-address that is being registered, an attacker cannot register their own stealth meta-address for a given registrant. Thus the end result, whether the attacker or an honest user submits the signature, is the same.

Recommendations

Short term, implement additional logic to prevent signature malleability. The OpenZeppelin `ECDSA.recover` function can be used as a reference or it can be used directly as a dependency.

Long term, add unit testing to ensure that only unique (nonmalleable) signatures are accepted by the `ERC6538Registry.registerKeysOnBehalf` function. If the short-term recommendation is not implemented, add unit testing to ensure that signature malleability does not allow an attacker to register their own stealth meta-address for a given registrant.

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. Code Maturity Categories

The following tables describe the code maturity categories and rating criteria used in this document.

Code Maturity Categories	
Category	**Description**
Arithmetic	The proper use of mathematical operations and semantics
Auditing	The use of event auditing and logging to support monitoring
Authentication / Access Controls	The use of robust access controls to handle identification and authorization and to ensure safe interactions with the system
Complexity Management	The presence of clear structures designed to manage system complexity, including the separation of system logic into clearly defined functions
Cryptography and Key Management	The safe use of cryptographic primitives and functions, along with the presence of robust mechanisms for key generation and distribution
Decentralization	The presence of a decentralized governance structure for mitigating insider threats and managing risks posed by contract upgrades
Documentation	The presence of comprehensive and readable codebase documentation
Low-Level Manipulation	The justified use of inline assembly and low-level calls
Testing and Verification	The presence of robust testing procedures (e.g., unit tests, integration tests, and verification methods) and sufficient test coverage
Transaction Ordering	The system's resistance to transaction-ordering attacks

Rating Criteria	
Rating	**Description**
Strong	No issues were found, and the system exceeds industry standards.
Satisfactory	Minor issues were found, but the system is compliant with best practices.
Moderate	Some issues that may affect system safety were found.

Weak	Many issues that affect system safety were found.
Missing	A required component is missing, significantly affecting system safety.
Not Applicable	The category is not applicable to this review.
Not Considered	The category was not considered in this review.
Further Investigation Required	Further investigation is required to reach a meaningful conclusion.

C. Non-Security–Related Findings

The following recommendations are not associated with specific vulnerabilities. However, they enhance code readability and may prevent the introduction of vulnerabilities in the future.

- **Fix a typo in the ERC-6538 specification.** The ERC-6538 specification says, "This contract defines an `ERC5564Registry` that stores the stealth meta-address for entities." However, the contract name should be `ERC6538Registry`.
- **Emit an event in the `incrementNonce` function.** The `incrementNonce` function is a state-changing function that should emit an event when invoked (figure C.1).

```
116     /// @notice Increments the nonce of the sender to invalidate existing
signatures.
117     function incrementNonce() external {
118      unchecked {
119        nonceOf[msg.sender]++;
120      }
121     }
```

Figure C.1: The `incrementNonce` function does not emit an event. (stealth-address-erc-contracts/src/ERC6538Registry.sol#L116–L121)

D. 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 February 26, 2024, Trail of Bits reviewed the fix implemented by the ScopeLift team for the issue identified in this report to determine its effectiveness in resolving the associated issue.

In summary, ScopeLift resolved the one issue identified during the audit. For additional information, please see the Detailed Fix Review Results below.

ID	Title	Status
1	Insufficient protection against signature malleability	Resolved

Detailed Fix Review Results

Finding 1: Insufficient protection against signature malleability
Resolved in PR #9. The ScopeLift team added unit tests to ensure that a signature malleability attack does not allow an attacker to update the original signer's stealth meta-address.

E. 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.