Mass - PRs review

Security Assessment

August 31, 2023

Prepared for:
Rudy Kadoch
Mass
rk@mass.money

Olivier Guimbal
Mass
og@mass.money

Prepared by: **Josselin Feist and Richie Humphrey**

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

About Trail of Bits **1**
Notices and Remarks **2**
Table of Contents **3**
Executive Summary **4**
Project Summary **6**
Project Goals **7**
Project Targets **8**
Project Coverage **9**
Summary of Findings **11**
Detailed Findings **12**
1. Trusted forwarder can take over the WalletFactory contract 12
2. Lack of contract existence check on StaticHyVM 14
3. Address aliasing on optimistic rollups is not considered 16
4. Undocumented expectations for state-changing operations in HyVM 17
5. Invalid EVM versions possible in multi-chain deployment 18
6. executeCall will always revert when sending native tokens 19
A. Vulnerability Categories **21**
B. TOB-NESTED-PRS-01 Test Case **23**
C. Code Quality Recommendations **24**
D. Fix Review Results **25**
Detailed Fix Review Results 26
E. Fix Review Status Categories **27**

Executive Summary

Engagement Overview

Nested Finance engaged Trail of Bits to review the security of several PRs made to the Tetris and HyVM codebases.

A team of two consultants conducted the review from June 26 to June 30, 2023, for a total of one engineer-week of effort. Our testing efforts focused on finding issues introduced by the PRs' changes that would impact the funds or the functionalities of the codebases. With full access to the source code and documentation, we performed static and dynamic testing of the codebases, using automated and manual processes.

Observations and Impact

Most of the PRs are the results of recommendations made during a previous security review. The changes either remove code complexity or implement additional safeguards, which overall increase the codebases' security.

However, some of the issues found during this review highlight the need for more thorough testing, including tests to better cover the access control capabilities of every stakeholder, as well as the support for additional chains.

Recommendations

Based on the findings identified during the security review, Trail of Bits recommends that Nested Finance 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.

- **Increase the testing coverage**, including coverage in the following areas: access controls, support for multiple chains, and transfers of assets.

The following tables provide the number of findings by severity and category.

EXPOSURE ANALYSIS

Severity	*Count*
High	2
Medium	1
Low	0
Informational	3
Undetermined	0

CATEGORY BREAKDOWN

Category	*Count*
Access Controls	1
Configuration	1
Data Validation	2
Undefined Behavior	2

Project Summary

Contact Information

The following managers were associated with this project:

Dan Guido, Account Manager
dan@trailofbits.com

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

The following engineers were associated with this project:

Josselin Feist, Consultant
josselin@trailofbits.com

Richie Humphrey, Consultant
richie.humphrey@trailofbits.com

Project Timeline

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

Date	Event
June 22, 2023	Pre-project kickoff call
July 5, 2023	Delivery of report draft
July 5, 2023	Report readout meeting
August 31, 2023	Delivery of final report

Project Goals

The engagement was scoped to provide a security assessment of several PRs made to Nested Finance's HyVM and Tetris codebases. Specifically, we sought to answer the following non-exhaustive list of questions:

- Can an attacker steal or freeze funds?
- Are there appropriate access control measures in place for users and owners?
- Does the system's behavior match the specification?
- Is it possible to perform system operations without paying the required amount?
- Are there any risks associated with using the UUPS proxy?
- Are all of the common Solidity flaws avoided?

Project Targets

The engagement involved a review and testing of the PRs listed below. All of the code reviewed was in either Solidity or Huff.

Tetris

- PR #110: “Use UUPS proxy for walletFactory: use a solidity proxy for the WalletFactory”
- PR #111: “Simplify the ImplementationResolver by removing the dynamic dispatching”
- PR #114: “Remove custom callHelper and use Address from OZ”
- PR #112: “Allowing user to revert back to the default implementation”
- PR #113: “ERC1271 for NestedWallet”
- PR #119: “Allow same wallet address on different chains”
- PR #121: “Use Nested HyVM fork without state changing operations”
- PR #118: “Signature EIP712”

HyVM

- PR #43: “StaticHyVM”
- PR #1: “Remove state changing operations”

Documentation and Testing

- PR #115: “Various test fixes”
- PR #122: “Doc 1271”
- PR #123: “Rename test file”
- PR #124: “Missing tests walletFactory”

Project Coverage

This section provides an overview of the analysis coverage of the review, as determined by our high-level engagement goals.

We performed static analysis with Slither on both the Tetris and HyVM repositories. For each PR in scope, we used a tailored approach, as follows:

Tetris

- PR #110: We reviewed the integration of the `UUPSUpgradeable` contract and the new owner functionalities, with a focus on reviewing the access controls and the impact of the use of ERC-2771. We also did a line-by-line comparison of the `Proxy.huff` contract with the inline assembly in the `Proxy.sol` contract to ensure that there is no loss of functionality.

- PR #111: We performed a manual review of all Solidity and Huff changes.

- PR #114: We conducted a line-by-line review of the replaced code, comparing it with the new OpenZeppelin implementation.

- PR #112: We reviewed the new default implementation option in the `ImplementationResolver` contract with a heightened focus on identifying access control and data validation issues, including missing implementation addresses. We looked for ways to prevent users from reverting to the default implementation, and we investigated how a malicious wallet factory owner could abuse their position.

- PR #113: We reviewed the implementation of ERC-1271 and the changes introduced with the updated OpenZeppelin `TimelockController` dependency.

- PR #119: We reviewed the changes and investigated whether the removal of the chain ID from the wallet creation logic effectively allows the same wallet address to be used across EVM-compatible chains.

- PR #121: This PR mostly updates the HyVM dependency; therefore, our review was not focused on this PR. However, we checked that the removal of the `executeCall` function does not lead to new issues.

- PR #118: We reviewed the signature verification code and looked for ways to bypass the validation, such as by misusing the `ecrecover` function or conducting signature replay attacks.

HyVM

- PR #43: We reviewed the low-level call manipulation introduced in this PR and looked for ways to execute arbitrary calls to `delegatecall`.

- PR #1: We reviewed the removal of the state-changing operations and looked for further ways to manipulate the state.

Coverage Limitations

Because of the time-boxed nature of testing work, it is common to encounter coverage limitations. The following list outlines the coverage limitations of the engagement and indicates system elements that may warrant further review:

- We focused our review on the PRs; we did not perform a holistic review of the codebases. A previous Trail of Bits audit covered the codebases more broadly, including the HyVM, Tetris, Nested token, and Nested DCA contracts. Nested Finance stated that these PRs cover all of the changes made to the codebases since this review.

- We focused our review on changes to the contracts; we did not review the updates on the tests in depth.

Summary of Findings

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

ID	Title	Type	Severity
1	Trusted forwarder can take over the WalletFactory contract	Access Controls	**High**
2	Lack of contract existence check on StaticHyVM	Data Validation	**High**
3	Address aliasing on optimistic rollups is not considered	Configuration	**Informational**
4	Undocumented expectations for state-changing operations in HyVM	Undefined Behavior	**Informational**
5	Invalid EVM versions possible in multi-chain deployment	Data Validation	**Informational**
6	executeCall will always revert when sending native tokens	Undefined Behavior	**Medium**

Detailed Findings

1. Trusted forwarder can take over the WalletFactory contract	
Severity: **High**	Difficulty: **High**
Type: Access Controls	Finding ID: TOB-NESTED-PRS-01
Target: `WalletFactory.sol`	

Description

The `WalletFactory` contract implements ERC-2771, which enables the use of a trusted forwarder to create wallets on behalf of other accounts. With the introduction of PR #110, the contract is also upgradeable. Therefore, the trusted forwarder contract can call the upgrade function on the `WalletFactory` contract by submitting the owner address as the value of `msg.sender`.

PR #110 adds an override of the `_msgSender` function to the `WalletFactory` contract, which now uses the ERC-2771 context contract; this function allows the trusted forwarder to set any value for the message sender:

```
/// @dev Provides information about the current execution context
///      WalletFactory is inheriting from ERC2771Context and Context through
///      Ownable2Step. It is needed to override to specify which base to use
function _msgSender()
    internal
    view
    virtual
    override(ERC2771Context, Context)
    returns (address)
{
    return ERC2771Context._msgSender();
}
```

Figure 1.1: `WalletFactory.sol#L306-L317`

The `onlyOwner` modifier relies on `_msgSender`:

```
/**
 * @dev Throws if called by any account other than the owner.
 */
modifier onlyOwner() {
    _checkOwner();
    _;
}

[…]

/**
 * @dev Throws if the sender is not the owner.
 */
function _checkOwner() internal view virtual {
    require(owner() == _msgSender(), "Ownable: caller is not the owner");
}
```

Figure 1.2: `OZ/access/Ownable.sol#L42-L64`

Among others, `onlyOwner` is used as an access control of the upgrade function:

```
function _authorizeUpgrade(address newImplementation) internal virtual override
onlyOwner {
    if (newImplementation.code.length == 0) {
        revert AddressNotContract(newImplementation);
    }
}
```

Figure 1.3: `OZ/access/Ownable.sol#L50-L52`

As a result, the trusted forwarder can control all of the owner-specific operations, including the following:

- Upgrading the contract
- Changing the contract's owner

Appendix B contains a test case to trigger this issue.

Exploit Scenario

Bob is a trusted forwarder. Therefore, he expects that his account has only additional rights regarding the creation of new wallets, so he does not properly protect his access. Eve manages to compromise Bob's account and gains access to ownership of the wallet factory. Eve uses her new access to upgrade the wallet factory to a malicious contract that generates malicious wallets. Alice unknowingly generates a new malicious wallet and puts $10 million worth of assets into it. Eve steals the assets.

Recommendations

Short term, prevent the trusted forwarder from having access to owner privileges except where this behavior is expected and documented.

Long term, document and test the access controls of the wallet factory; ensure that the documentation highlights the access control expectations for the trusted forwarder.

2. Lack of contract existence check on StaticHyVM

Severity: **High**	Difficulty: **High**
Type: Data Validation	Finding ID: TOB-NESTED-PRS-02
Target: `StaticHyVM.sol`	

Description

The call to `delegatecall` made in the `StaticHyVM` contract does not check that the `hyvm` contract has code; this call will succeed even if the `hyvm` contract was destroyed.

The `doDelegateCall` function uses a low-level call to `delegatecall` to call `hyvm`:

```
function doDelegateCall(bytes calldata payload) public returns (bytes memory) {
    if (msg.sender != address(this)) revert OnlySelf();

    (bool success, bytes memory data) = hyvm.delegatecall(payload);
    if (!success) _bubbleError(data, "StaticHyVM: delegatecall failed");
    return data;
}
```

Figure 2.1: `StaticHyVM/StaticHyVM.sol#L30-L36`

The Solidity documentation includes the following warning:

> The low-level functions call, `delegatecall` and `staticcall` return `true` as their first return value if the account called is non-existent, as part of the design of the EVM. Account existence must be checked prior to calling if needed.

Figure 2.2: A snippet of the Solidity documentation detailing unexpected behavior related to `delegatecall`

As a result, any call made to a nonexistent contract will return success, even if no code was executed.

We acknowledge that the likelihood that the `hyvm` contract will be destroyed is low; however, the previous audit that we conducted on the Nested Finance codebases determined that this risk is not null. (Refer to finding TOB-NESTED-1 of the previous report.)

Exploit Scenario

Bob creates a bot that takes actions following the execution of `StaticHyVM`. A bug is found in `hyvm`, and the contract is destroyed. However, all of the calls made to `StaticHyVM` continue to return success, leading Bob's bot to take incorrect actions.

Recommendations
Short term, implement a contract existence check before the call to `delegatecall` in `StaticHyVM`.

Long term, carefully review the Solidity documentation, especially the "Warnings" section. Document every assumption made for code optimizations, and ensure that the underlying invariants hold. For example, if an invariant states that all of the contracts that are used exist, extra care must be taken to ensure that all calls' destinations always have code.

3. Address aliasing on optimistic rollups is not considered

Severity: **Informational**	Difficulty: **Low**
Type: Configuration	Finding ID: TOB-NESTED-PRS-03
Target: `WalletFactory.sol`	

Description
The documentation states that the user's wallet address will be the same across different chains, but this does not take into account the address aliasing that some chains apply to sender addresses.

PR #119 changes the logic of the wallet creation process such that a wallet originating from a specific user will be the same across EVM-compatible chains:

```
Allows to share the same address on different EVM equivalent chains.
It should prevent cross chain manipulation.
If a USER1 creates a wallet or USER2 creates on behalf of USER1, the address of the wallet will be the same
and will depend on the USER1 address.
Hence, on different chains, if EVM equivalent, the address of a wallet for a specific user will be the same.
```

Figure 3.1: Documentation included with PR #119

However, some chains—particularly optimistic rollups—apply aliasing on the L1 caller's address if it is a contract. (Refer to Arbitrum's documentation.) In these situations, the address specified for the L2 caller will not be the same as the L1 address. As a result, the assumption that users can keep the same address across chains will not hold for such chains.

Recommendations
Short term, update the documentation to inform users that their wallet addresses may be aliased on some chains.

Long term, create tests targeting every EVM-compatible chain that the system is meant to support.

References
- Address aliasing in Arbitrum
- Address aliasing in Optimism

4. Undocumented expectations for state-changing operations in HyVM

Severity: **Informational**	Difficulty: **Low**
Type: Undefined Behavior	Finding ID: TOB-NESTED-PRS-04
Target: `HyVM.huff`	

Description
PR #1 removed some of the state-changing operations from the `HyVM` contract, but not all of them. It is unclear whether the Nested Finance team intended to keep those remaining operations in the contract.

In particular, the following state-changing operations are still allowed:

- Operations that emit events (such as `log0` and `log1`)
- Operations that are called with nonzero values

Recommendations
Short term, update the documentation to make it clear whether these operations should be allowed. Disable any that should not be allowed.

Long term, maintain an up-to-date design specification to define the behavior of the HyVM, including the operations that are available and the state changes that are allowed, if any. This specification should be a living document that changes as the protocol changes.

5. Invalid EVM versions possible in multi-chain deployment

Severity: **Informational**	Difficulty: **High**
Type: Data Validation	Finding ID: TOB-NESTED-PRS-05
Target: All Solidity files	

Description

The HyVM and Tetris contracts contain Solidity pragmas that constrain the compiler's version to 0.8.16 or above. However, if these contracts were to be compiled with Solidity 0.8.20 and deployed to a chain that has not implemented EIP-3855, the contracts would fail to deploy because Solidity 0.8.20 would default to the "Shanghai" EVM version.

The Shanghai hard fork introduced a new opcode, `PUSH0`. Solidity 0.8.20 adds support for Shanghai, and contracts compiled with Solidity 0.8.20 will include `PUSH0` instructions by default. Given that most non-Ethereum chains have not added support for `PUSH0` yet, contracts compiled with Solidity 0.8.20 will fail to deploy on those chains unless the EVM version is explicitly set.

Recommendations

Short term, add documentation to ensure that the code is built with an EVM version that is supported by the target chain.

Long term, add tests that cover all of the EVM-compatible chains that should be supported by the contracts.

6. executeCall will always revert when sending native tokens

Severity: **Medium**	Difficulty: **Low**
Type: Undefined Behavior	Finding ID: TOB-NESTED-PRS-06
Target: `NestedWallet.sol`	

Description

`executeCall` and `executeMultiCall` are both payable functions that accept a `Call` struct as an argument. These functions use OpenZeppelin's `Address.functionCallWithValue` function to make an arbitrary call to the target address, with `payload` passed as the calldata and the value to be sent equal to `call.value`.

```
function executeCall(Call calldata call)
    public
    payable
    override
    onlyWalletOwner
    returns (bytes memory data)
{
    _wrapNativeToken();
    return Address.functionCallWithValue(call.target, call.payload, call.value);
}
```

Figure 6.1: `NestedWallet.sol#L141-L150`

In both functions, before `Address.functionCallWithValue` is invoked, the `_wrapNativeToken` function is called, which deposits `msg.value` worth of native tokens into the WETH contract, turning any ETH that was sent with the transaction into WETH.

Therefore, by the time the call to `Address.functionCallWithValue` is made, the wallet no longer has the native tokens that were sent along with the call. So when `Address.functionCallWithValue` performs its low-level call, the transaction reverts with the error `Address: insufficient balance for call`.

Exploit Scenario

Bob calls `executeCall` to send 10 ether. The call fails, and Bob is unable to use the Nested wallet as he expected to.

Recommendations

Short term, remove the `_wrapNativeToken();` line from both functions. Additionally, we noted that `_wrapNativeToken` is called in `executeHyVMCall` and `executeSignature712`, but it does not introduce a risk because it is called before a `delegatecall` and the value is not passed along in the same way. However, if this is the desired behavior, then it should be documented.

Long term, include unit and fuzz tests for sending native tokens when testing payable functions.

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. TOB-NESTED-PRS-01 Test Case

The following contains a test case to reproduce finding TOB-NESTED-PRS-01. The test calls the wallet factory as the trusted forwarder and successfully upgrades the contract.

```
/// @notice Test the walletFactoryProxy can be upgraded by a compromised forwarder.
/// @dev Using test setup from WalletFactory.t.sol.
function testCanUpgradeIfTrustedForwarder() public {

    // Create a mock malicious contract.
    UUPSUpgradeableMock maliciousProxy = new UUPSUpgradeableMock();

    // Change the caller to the compromised trusted forwarder address.
    changePrank(address(minimalForwarder));

    // Create calldata to call the upgradeTo function and append OWNER address to
    // end of the calldata per ERC-2771.
    bytes memory data = abi.encodePacked(
        abi.encodeCall(
            walletFactoryProxy.upgradeTo,
            (address(maliciousProxy))
        ),
        OWNER
    );

    // Expect the Upgraded event to be emitted by the wallet factory proxy.
    vm.expectEmit({
        checkTopic1: true,
        checkTopic2: false,
        checkTopic3: false,
        checkData: false,
        emitter: address(walletFactoryProxy)
    });
    emit Upgraded(address(maliciousProxy));

    // Make the call to the walletFactoryProxy (calling as the forwarder).
    (bool success,) = address(walletFactoryProxy).call(data);

    // The call should revert, but because of TOB-NESTED-PRS-01 it does not revert.
    require(success);
}
```

Figure B.1: This custom test works with `WalletFactory.t.sol`.

C. Code Quality Recommendations

- **Add events to the `setCurrentImplementation` and `setDefaultImplementation` functions.** Events will make tracking updates to the implementations easier.
- **Rename the `owner` parameter in the `WalletFactory.initialize` function.** The `owner` parameter name shadows the `owner` state variable.
- **In the `create` and `createAndCall` functions' documentation, specify the actor that can use the function to create wallets.** For example, one can create a wallet on behalf of someone else using `create` but not `createAndCall`. While this restriction is a good practice, its associated reasoning should be documented.
- **Update the comment in `WalletFactory.sol#L317-L318`.** The `ERC2771Context` contract is not inherited through the `Ownable2Step` contract.
- **Remove the reference to "name" in `Proxy.huff#L38-L45`.** "Name" is no longer used.

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 August 21, 2023, Trail of Bits reviewed the fixes and mitigations implemented by the Nested Finance team for the issues identified in this report. We reviewed each fix to determine its effectiveness in resolving the associated issue. Additionally, on August 29, 2023, Trail of Bits reviewed additional code changes that were the result of follow up recommendations for TOB-NESTED-PRS-01.

In summary, Nested Finance has resolved all six of the issues disclosed in this report. For additional information, please see the Detailed Fix Review Results below.

ID	Title	Status
1	Trusted forwarder can take over the WalletFactory	**Resolved**
2	Lack of contract existence check on StaticHyVM	**Resolved**
3	Address aliasing on optimistic rollups is not considered	**Resolved**
4	Undocumented expectations for state-changing operations in HyVM	**Resolved**
5	Invalid EVM versions possible in multi-chain deployment	**Resolved**
6	executeCall will always revert when sending native tokens	**Resolved**
7	ERC-2771 cannot be deactivated	**Resolved**

Detailed Fix Review Results

TOB-NESTED-PRS-1: Trusted forwarder can take over the WalletFactory contract
Resolved in PR #137. Nested Finance replaced the use of OpenZeppelin's `ERC2771Context` contract with a custom version and is now using ERC-2771 only for `createAndCall`; these changes prevent the trusted forwarder from having access to owner privileges.

TOB-NESTED-PRS-2: Lack of contract existence check on StaticHyVM
Resolved in PR #47. Nested Finance added a contract existence check to the affected call to `delegatecall`.

TOB-NESTED-PRS-3: Address aliasing on optimistic rollups is not considered
Resolved in PR #132. Nested Finance updated the documentation to inform users of address aliasing.

TOB-NESTED-PRS-4: Undocumented expectations for state-changing operations in HyVM
Resolved in PR #2. Nested Finance updated the documentation about the state-changing operations.

TOB-NESTED-PRS-5: Invalid EVM versions possible in multi-chain deployment
Resolved in PR #127 and PR #48. Nested Finance pinned the EVM version and updated the documentation of HyVM.

TOB-NESTED-PRS-6: executeCall will always revert when sending native tokens
Resolved in PR #128. Nested Finance removed the call to `_wrapNativeToken`.

TOB-NESTED-PRS-7: ERC-2771 cannot be deactivated
This issue was found by Nested Finance and was resolved in PR #125. Nested Finance added `deactivateERC2771` to the Nested wallet.

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.