Mondrian Wallet Audit Report

Prepared by: X3 Security

Table of Contents

Protocol Summary

Mondrian Wallet leverages account abstraction (EIP-4337) to provide a flexible and secure wallet solution. By abstracting the account, users are able to sign transactions with various methods, not limited to a private key. This approach enhances the security and usability of the wallet, ensuring that user operations are executed by authorized entities only.

Audit Period: May 09, 2024 - May 16, 2024

Disclaimer

The X3 Security team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

Impact	High	Medium	Low
High	H	H/M	M
Medium	H/M	M	M/L
Low	M	M/L	L

Audit Details

Scope

In Scope:

```
./contracts/
└── MondrianWallet.sol
```

Findings Table

Severity	ID	Title
High	H-01	[H-01] Vulnerability in Signature Validation of `_validateSignature` Function

[H-01] Vulnerability in Signature Validation of `_validateSignature` Function.

Severity: High Risk
Date Modified: May 15th, 2024

Submitted Report

[H-01] Vulnerability in Signature Validation of _validateSignature Function
Severity: High Risk

Relevant GitHub Links

- Link to _validateSignature function lines 113-121
- Link to related code on line 76

Summary

The **_validateSignature** function within the MondrianWallet smart contract is critically flawed due to its failureto validate signatures properly. This function is intended to ensure that user operations are executed only byentities with authorized access. However, it is currently hardcoded to always return **SIG_VALIDATION_SUCCESS**, which bypasses signature verification. This allows any user, regardless of theirauthenticity, to submit operations, posing a severe security risk.

Vulnerability Details

The function utilizes the **ECDSA.recover** method from the OpenZeppelin library to attempt to recover the address from a given signature and a hashed message (userOpHash). Although the function successfully recovers an address from the signature, it does not perform any subsequent checks to verify whether therecovered address is authorized to perform the requested operation. Here is the critical part of the code:

```
function _validateSignature(PackedUserOperation calldata userOp, bytes32
userOpHash)
    internal
    pure
    returns (uint256 validationData)
{
    bytes32 hash = MessageHashUtils.toEthSignedMessageHash(userOpHash);
    ECDSA.recover(hash, userOp.signature); // Recovery is done, but no validation
follows
    return SIG_VALIDATION_SUCCESS; // Always returns success, bypassing security
}
```

Impact

This vulnerability can be exploited to forge signatures and execute unauthorized transactions or state changes. The implications are particularly dire, including potential unauthorized fund transfers, administrative changes, or other malicious activities that could severely compromise the integrity and security of the contract and its stakeholders.

Tools Used

- Manual Review

Recommendations

- **Implement Signature Verification**: Revise the **_validateSignature** function to include a verification step that checks if the recovered address is among the set of addresses authorized to initiate operations:

```
address recoveredAddress = ECDSA.recover(hash, userOp.signature);
require(recoveredAddress == authorizedSigner, "Unauthorized signer detected");
```