Threat Model v1: ERC-4337 Smart Wallet Signature Security Threats

Overview

This document outlines the key signature-related security threats identified for our ERC-4337 smart wallet fuzzing framework. These threats represent potential attack vectors that could compromise wallet security if not properly mitigated.

1. Zero-Signature Attack

Threat Description

Attackers submit UserOperations with signatures composed entirely of zero bytes (65 bytes of 0x00), attempting to bypass signature validation.

Attack Mechanism

- **Vector**: Signature validation logic
- **Method**: Submit $\text{signature} = \text{bytes}(65)$ (all zeros)
- **Target**: EntryPoint's signature verification function
- **Bypass Attempt**: Exploit implementations that fail to validate signature content

Potential Impact

- **Severity**: CRITICAL
- **Consequences**: Complete loss of funds, attacker can forge any transaction
- **Scope**: All wallet operations requiring signatures

Testing Approach

- Construct malicious UserOperation with zero-signature
- Submit to `handleOps()` function
- Verify transaction rejection with appropriate error

2. Short-Signature Attack

Threat Description

Attackers submit signatures with non-standard lengths, attempting to trigger unexpected behavior or buffer overflows in validation logic.

Attack Mechanism

- **Vector**: Signature length validation
- **Methods**:
  - Empty signature (`bytes(0)`)
  - 1-byte signature

- 32-byte signature (only r component)
- 64-byte signature (missing v component)
- 66-byte signature (extra bytes)
- **Target**: Signature parsing and recovery functions

Potential Impact

- **Severity**: MEDIUM to CRITICAL
- **Consequences**: Denial of service, memory corruption, or signature bypass
- **Scope**: All signature-based operations

Testing Approach

- Generate signatures of various lengths
- Test each length with valid transaction content
- Monitor for crashes, unexpected acceptance, or gas anomalies

3. Invalid v-Value Signature Attack

Threat Description

Attackers manipulate the signature's v component (recovery identifier) to use values outside the valid range (27 or 28 for Ethereum).

Attack Mechanism

- **Vector**: Signature recovery parameter
- **Methods**:
  - Set $v = 0$ or $v = 1$
  - Use chain-specific values (e.g., $v = 37$ for EIP-155)
  - Extremely high values ($v = 255$)
- **Target**: $ecrecover()$ or equivalent recovery functions

Potential Impact

- **Severity**: MEDIUM
- **Consequences**: Failed signature recovery, incorrect signer identification
- **Scope**: Transactions across different chains or implementations

Testing Approach

- Generate signatures with modified v values
- Test recovery function behavior
- Verify proper rejection of invalid v values

4. Replay Attack (Same Nonce)

Threat Description

Attackers resubmit previously executed UserOperations, attempting to duplicate transactions and drain funds.

Attack Mechanism

- **Vector**: Nonce management system
- **Method**:

1. Capture a valid signed UserOperation
2. Resubmit identical operation after original execution
3. Exploit nonce tracking failures

- **Target**: EntryPoint's nonce validation

Potential Impact

- **Severity**: CRITICAL
- **Consequences**: Double spending, repeated fund transfers
- **Scope**: All sequential operations

Testing Approach

1. Execute a valid UserOperation
2. Record nonce, signature, and operation details
3. Attempt to resubmit identical operation
4. Verify rejection with "Invalid nonce" error

Security Assumptions

1. **Cryptographic Primitives**: ECDSA with secp256k1 curve is secure

2. **Nonce Management**: EntryPoint properly increments and tracks nonces
3. **Signature Format**: Standard 65-byte (r[32], s[32], v[1]) format expected
4. **Recovery Logic**: $\text{ecrecover}()$ correctly implements signature recovery

System Boundaries

- **In Scope**: Signature validation within EntryPoint contract
- **Out of Scope**: Network-layer attacks, front-running, MEV extraction
- **Assumed Secure**: Underlying blockchain consensus, RPC endpoint integrity

Testing Invariants

1. **Signature Validity**: No transaction with invalid signature should succeed
2. **Nonce Monotonicity**: Nonce values must strictly increase per account
3. **Length Enforcement**: Signatures must be exactly 65 bytes
4. **Recovery Bounds**: v value must be 27 or 28 for standard transactions

Mitigation Strategies

1. **Zero-Signature**: Add explicit check for zero-value signatures
2. **Length Validation**: Require exact 65-byte signatures
3. **v-Value Checking**: Validate $\text{v} \in \{27, 28\}$ for standard transactions
4. **Nonce Tracking**: Implement strict nonce incrementing with storage

Evidence of Threat Realization

During initial testing, we have already confirmed:

- ✅ **Zero-Signature Attack**: Successfully accepted by vulnerable implementation
- ✅ **Short-Signature Attack**: Empty signatures accepted in some cases
- ✅ **Replay Attack**: Properly prevented by nonce mechanism (defense working)
- ⚠️ **Invalid v-Value**: Requires deeper contract-level testing

Next Steps for M2

1. **Quantify Risks**: Measure success rates for each attack vector
2. **Develop Payloads**: Create specialized attack payloads for each threat
3. **Automate Detection**: Build invariant checking for production systems
4. **Remediation Guidance**: Provide concrete fixes for identified vulnerabilities