WebSocket Connection Issues - Troubleshooting Report

Executive Summary

The wallet module's WebSocket connection for real-time balance updates fails to establish when connecting through certain network configurations. While a polling-based fallback was successfully implemented, the root cause of the WebSocket failure remains unresolved.

Problem Description

Symptoms

- WebSocket connection to `wss://lnbits.ario.pm/api/v1/ws/<wallet-id>` fails immediately
- Error message: `WebSocket connection failed`
- Connection attempts result in immediate closure
- Issue appears related to network path through WireGuard VPN and/or nginx proxy

Current Configuration

Network Path

```
Client Browser → Internet → nginx (reverse proxy) → WireGuard VPN → LNbits Server
```

nginx Configuration

- Reverse proxy at `lnbits.ario.pm`
- Standard WebSocket proxy headers configured
- HTTPS/WSS termination at nginx level

LNbits Server

- Running behind WireGuard VPN
- WebSocket endpoint: `/api/v1/ws/<wallet-id>`
- Requires `X-Api-Key` header for authentication

Root Cause Analysis

Confirmed Working

- Standard HTTPS API calls work perfectly
- Authentication headers are properly passed
- LNbits server WebSocket endpoint is functional (works in direct connections)
- Polling fallback successfully retrieves balance updates

Potential Causes

1. nginx WebSocket Proxy Configuration Likelihood: HIGH

Standard nginx configurations often miss critical WebSocket headers:

```
# Required headers that might be missing
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
```

```
# WebSocket-specific timeout settings
proxy_connect_timeout 60s;
proxy_send_timeout 60s;
proxy_read_timeout 60s;
```

Solution: Verify nginx configuration includes proper WebSocket upgrade headers and timeout settings.

2. WireGuard MTU Issues Likelihood: MEDIUM

WireGuard default MTU (1420) can cause packet fragmentation issues with WebSocket frames: - WebSocket frames might exceed MTU after VPN encapsulation - Fragmented packets may be dropped or delayed

Solution:

```
# In WireGuard config
[Interface]
MTU = 1380  # Reduced MTU to account for overhead
```

3. NAT/Connection Tracking Likelihood: MEDIUM

Long-lived WebSocket connections can be terminated by: - NAT timeout settings - Connection tracking table exhaustion - Firewall state timeout

Solution: - Increase NAT timeout values - Enable WebSocket keepalive/ping frames - Configure firewall to recognize WebSocket as persistent connection

4. HTTP/2 Incompatibility Likelihood: MEDIUM

WebSockets don't work over HTTP/2 connections: - If nginx is configured for HTTP/2, WebSocket upgrade fails - Need separate location block or HTTP/1.1 fallback

Solution:

```
location /api/v1/ws {
    proxy_http_version 1.1;  # Force HTTP/1.1
    # ... other WebSocket headers
}
```

5. Header Size/Authentication Issues Likelihood: LOW

Custom headers might be stripped or modified: - `X-Api-Key` header might not survive proxy chain - Header size limits in proxy configuration

Solution: Verify headers are properly forwarded through entire chain.

Diagnostic Steps

1. Browser-Level Debugging

```
// Test WebSocket connection directly
const ws = new WebSocket('wss://lnbits.ario.pm/api/v1/ws/wallet-id');

ws.onopen = () => console.log('Connected');
ws.onerror = (error) => console.error('Error:', error);
ws.onclose = (event) => {
  console.log('Close code:', event.code);
  console.log('Close reason:', event.reason);
  console.log('Was clean:', event.wasClean);
};
```

2. Network Path Testing

```
# Test from different network locations
# 1. Direct to LNbits (bypassing nginx)
wscat -c ws://lnbits-server:5000/api/v1/ws/wallet-id -H "X-Api-Key: key"

# 2. Through nginx (bypassing WireGuard)
wscat -c wss://nginx-server/api/v1/ws/wallet-id -H "X-Api-Key: key"

# 3. Full path (through nginx and WireGuard)
wscat -c wss://lnbits.ario.pm/api/v1/ws/wallet-id -H "X-Api-Key: key"
```

3. nginx Logs Analysis

```
# Check nginx error logs
tail -f /var/log/nginx/error.log | grep -i websocket

# Enable debug logging for WebSocket
error_log /var/log/nginx/error.log debug;
```

4. WireGuard Diagnostics

```
# Check for packet drops
wg show
ip -s link show wg0

# Monitor MTU issues
tcpdump -i wg0 -n 'tcp[tcpflags] & (tcp-syn) != 0'
```

Implemented Workaround

Polling Fallback Mechanism

```
// WalletWebSocketService.ts
class WalletWebSocketService extends BaseService {
  private async startPolling() {
    this.stopPolling()

    const pollBalance = async () => {
      if (!this.isActive) return

      try {
        const walletDetails = await this.walletAPI.getWalletDetails()
        if (walletDetails && walletDetails.balance !== this.lastBalance) {
          this.lastBalance = walletDetails.balance
          this.store.updateBalance(walletDetails.balance / 1000)
          this.emit('balance-updated', walletDetails.balance / 1000)
        }
      } catch (error) {
        console.error('[WalletWebSocketService] Polling error:', error)
      }
    }

    // Initial poll
    await pollBalance()
```

```
    // Set up recurring polls
    this.pollInterval = setInterval(pollBalance, 5000) // Poll every 5 seconds
  }
}
```

Fallback Behavior

- Automatically activates when WebSocket connection fails
- Polls `/api/v1/wallets` endpoint every 5 seconds
- Updates balance only when changes detected
- Maintains same event emission pattern as WebSocket

Recommended Solutions

Priority 1: nginx Configuration Audit

1. Review nginx WebSocket proxy configuration
2. Add missing WebSocket headers
3. Ensure proper timeout settings
4. Test with HTTP/1.1 forced for WebSocket endpoints

Priority 2: Network Path Optimization

1. Test WebSocket connection at each network hop
2. Adjust WireGuard MTU if fragmentation detected
3. Review firewall/NAT rules for long-lived connections

Priority 3: Enhanced Diagnostics

1. Add WebSocket connection diagnostics endpoint
2. Implement client-side connection state reporting
3. Add server-side WebSocket connection logging

Priority 4: Alternative Approaches

1. Consider Server-Sent Events (SSE) as alternative to WebSockets
2. Implement WebSocket connection through separate subdomain
3. Use WebSocket-specific reverse proxy (e.g., websockify)

Testing Checklist

- ☐ Verify nginx configuration includes all WebSocket headers
- ☐ Test WebSocket connection from different network locations
- ☐ Check nginx error logs for WebSocket-specific errors
- ☐ Monitor WireGuard interface for packet drops
- ☐ Test with reduced MTU settings
- ☐ Verify authentication headers are properly forwarded
- ☐ Test with HTTP/1.1 forced for WebSocket location
- ☐ Check firewall/NAT timeout settings
- ☐ Test with browser developer tools WebSocket inspector
- ☐ Verify LNbits server WebSocket endpoint directly

Future Improvements

Short-term

1. Add connection retry logic with exponential backoff

2. Implement WebSocket heartbeat/ping mechanism
3. Add detailed connection state logging
4. Create health check endpoint for WebSocket connectivity

Long-term

1. Implement connection quality monitoring
2. Add automatic fallback selection based on network conditions
3. Consider implementing WebRTC DataChannel as alternative
4. Evaluate HTTP/3 WebTransport when available

References

- nginx WebSocket Proxy Documentation
- WireGuard MTU Considerations
- WebSocket Protocol RFC 6455
- LNbits WebSocket API Documentation

Status

Current State: Polling fallback operational, WebSocket root cause unresolved **Last Updated**: 2025-09-20
Next Steps: nginx configuration audit planned