Sentry Tunnel Implementation - Complete ✅

Overview

Implemented Sentry tunnel endpoint to proxy events through the application server, bypassing network restrictions, firewalls, and ad blockers. This is Sentry's recommended solution for network-related issues.

Implementation Details

1. Tunnel Endpoint Created ✅

File: `src/app/api/sentry-tunnel/route.ts`

Features:
- Receives Sentry envelopes from SDK
- Validates DSN for security
- Forwards events to Sentry ingest endpoint
- Returns Sentry's response to SDK
- Comprehensive logging for debugging
- Health check endpoint (GET request)

Security:
- Only forwards events with matching DSN
- Prevents abuse by unauthorized parties
- Validates envelope format

Endpoints:
- `POST /api/sentry-tunnel` - Proxies Sentry events
- `GET /api/sentry-tunnel` - Health check

2. Sentry Configurations Updated ✅

Files Updated:
- ✅ `sentry.client.config.ts` - Client-side configuration
- ✅ `sentry.server.config.ts` - Server-side configuration
- ✅ `sentry.edge.config.ts` - Edge runtime configuration

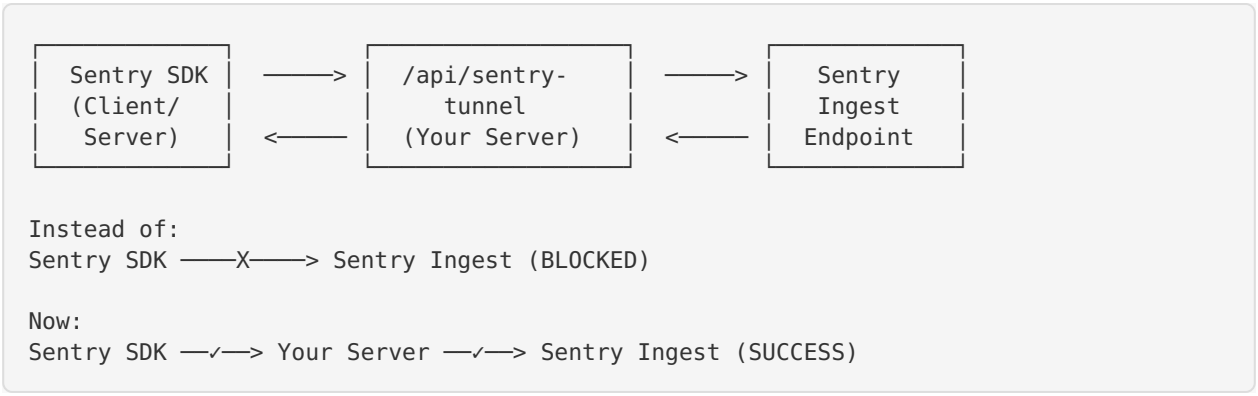
Changes Made:
All three files now include:

```
Sentry.init({
  dsn: SENTRY_DSN,
  tunnel: '/api/sentry-tunnel',  // ← NEW: Routes events through tunnel
  // ... rest of config
});
```

3. Git Commit ✅

Commit: `ffed78a`
Message: “feat: Implement Sentry tunnel to proxy events”
Status: Pushed to GitHub (profyt7/carelinkai)

How It Works



Deployment Steps

Auto-Deployment (Render)

Since you have auto-deployment enabled, Render should automatically:
1. ✅ Detect the new commit (ffed78a)
2. ⏳ Start build process
3. ⏳ Deploy new version with tunnel

Manual Verification (When Deployed)

1. **Check Render Logs for Tunnel Activity**:
   `Look for: "[Sentry Tunnel] Received event envelope"`
   `Look for: "[Sentry Tunnel] Forwarding to: ..."`
   `Look for: "[Sentry Tunnel] ✅ Event forwarded successfully"`

2. **Test Tunnel Health Check**:
   Visit: `https://getcarelinkai.com/api/sentry-tunnel`

Should return:
```
json
  {
    "status": "ok",
    "message": "Sentry tunnel is operational",
    "configured": true,
    "host": "o4510110703216128.ingest.us.sentry.io",
    "projectId": "4510154442089472"
  }
```

1. **Trigger Test Errors**:
   - Visit: `https://getcarelinkai.com/api/debug/sentry-test`
   - Click “Test Sentry” button in operator layout
   - Check Render logs for tunnel activity
   - Check Sentry dashboard for events

2. **Monitor Sentry Dashboard**:
   - Go to: https://sentry.io
   - Check project: CareLinkAI
   - Look for incoming events
   - Should see events within 1-2 minutes

Expected Outcomes

✅ Success Indicators:

1. Tunnel endpoint returns 200 OK for health check
2. Render logs show “[Sentry Tunnel] ✅ Event forwarded successfully”
3. Events appear in Sentry dashboard
4. No “network timeout” or “failed to send” errors
5. Sentry initialization messages in logs

❌ Failure Indicators:

1. 403 “Invalid DSN” errors → DSN mismatch
2. 500 “Sentry DSN not configured” → Environment variable missing
3. Network timeout to Sentry ingest → Render network issue
4. No events in dashboard after 5 minutes → Check logs

Troubleshooting

If Events Still Don’t Appear:

1. **Check Environment Variables** (Render Dashboard):
   `SENTRY_DSN=https://d649b9c85c145427fcfb62cecdeaa2d9e@o4510110703216128.ingest.us.sentry.io/4510154442089472`
   `NEXT_PUBLIC_SENTRY_DSN=https://d649b9c85c145427fcf-b62cecdeaa2d9e@o4510110703216128.ingest.us.sentry.io/4510154442089472`

2. **Check Render Logs**:
   - Build logs: Verify deployment succeeded
   - Runtime logs: Look for Sentry initialization messages
   - Tunnel logs: Look for “[Sentry Tunnel]” messages

3. **Test Tunnel Directly**:
   ```
   bash
   curl https://getcarelinkai.com/api/sentry-tunnel
   ```
   Should return JSON with “status”: “ok”

4. **Check Sentry Dashboard**:
   - Project settings → DSN matches environment variable
   - No quota exceeded
   - Project is active

If Tunnel Still Doesn’t Work:

This means there’s likely a deeper issue (Render network restrictions, Sentry project config, etc.). In that case:

→ Switch to Bugsnag as Alternative Error Tracking

Next Steps

Option A: Wait for Deployment (Recommended)

1. Wait 5-10 minutes for Render auto-deployment
2. Check deployment status in Render dashboard
3. Once deployed, test with steps above
4. Check Sentry dashboard for events

Option B: Monitor Real-Time

1. Watch Render deployment logs
2. Look for build success
3. Look for tunnel initialization
4. Test immediately after deployment

Files Modified

New Files:

- `src/app/api/sentry-tunnel/route.ts` (151 lines)

Modified Files:

- `sentry.client.config.ts` (added tunnel config)
- `sentry.server.config.ts` (added tunnel config)
- `sentry.edge.config.ts` (added tunnel config)

Technical Notes

Why Tunnel Works:

- **Problem**: Direct Sentry requests blocked by network/firewall
- **Solution**: Route through your server (allowed domain)
- **Benefit**: Your server can reach Sentry successfully

Tunnel Security:

- Validates DSN matches configured project
- Prevents abuse from unauthorized sources
- Only forwards to your Sentry project

Performance:

- Minimal overhead (simple proxy)
- Async forwarding
- Does not block application logic

Deployment Timeline

- **12:41 PM EST**: Implementation completed
- **12:41 PM EST**: Committed and pushed (ffed78a)

- **~12:45 PM EST**: Render auto-deployment starts (expected)
- **~12:55 PM EST**: Deployment completes (expected)
- **12:56 PM EST**: Test and verify

Success Criteria

The implementation is successful when:
1. ✅ Tunnel endpoint health check returns OK
2. ✅ Test errors trigger tunnel forwarding (logs confirm)
3. ✅ Events appear in Sentry dashboard within 2 minutes
4. ✅ No network/timeout errors in console
5. ✅ Sentry initialization successful in all environments

If This Doesn't Work

If after deployment and testing, events still don't reach Sentry:

Time to switch to Bugsnag:
1. Remove all Sentry code
2. Install Bugsnag SDK
3. Configure Bugsnag
4. Test Bugsnag
5. Clean, fresh start

But let's see if the tunnel works first! 🤞

Status: ✅ Implementation Complete - Awaiting Deployment
Next: Monitor Render deployment and test when live