Wiren Board Connectivity Issue from Russian ISPs

# Summary

Users in Russia are experiencing connection issues accessing Wiren Board's cloud interface at `\*.wirenboard.cloud`. Network logs show that while initial assets are loading, key JavaScript files (including React-based front-end scripts) fail with `net::ERR\_HTTP2\_PROTOCOL\_ERROR`. This prevents the interface from functioning properly.

# Observed Behavior

* Occurs without VPN on multiple Russian ISPs.
* Different browsers and systems affected equally.
* Failures observed in scripts critical to React application initialization.
* Network tools show `net::ERR\_HTTP2\_PROTOCOL\_ERROR` and `ERR\_BLOCKED\_BY\_ORB`.
* Static resources (e.g. CSS) load correctly, but JavaScript files fail.

# Technical Analysis

* The application is built with React (evident from file names like `react.70de11236f5807c6b07c.js` and usage of `main.[hash].js`).
* React applications rely heavily on dynamic loading and client-side routing.
* React frontends often serve minimal HTML shells and depend on JavaScript bundles to render and manage the UI. Blocking or corruption of these assets results in a completely non-functional app.
* React uses dynamic imports and sometimes HTTP/2 multiplexing, both of which can be disrupted by DPI (deep packet inspection) or protocol interference.
* Failure in just one of the initial JavaScript bundles (like `main.js`, `vendor.js`, or `react.js`) prevents the entire app from bootstrapping.
* In contrast to traditional server-rendered apps, React apps often lack usable fallbacks, which makes these failures more visible and disruptive.

# Likely Cause

* Selective deep packet inspection (DPI) or protocol throttling by Russian regulators (Roskomnadzor) affecting access to foreign services like DigitalOcean, Cloudflare, etc.
* HTTP/2-specific filtering or corruption—likely during SSL/TLS traffic inspection.
* Blocking or corruption of fingerprinted JS files (frequently updated by CDNs) can disrupt front-end frameworks like React.

# Why This Breaks React Apps More Easily

* React single-page applications (SPA) are sensitive to the failure of initial JS bundles.
* Partial loading (e.g. stylesheets OK, scripts blocked) leads to a blank or broken UI.
* Unlike traditional websites, fallback content is often minimal, compounding the visibility of the issue.
* React apps often depend on background requests and component lazy-loading. If these requests are blocked or delayed, critical parts of the UI may never appear.

# Advice for Wiren Board Team

* Implement fallback HTML or minimal functional UI for broken JS situations.
* Provide alternate domains hosted through geographically or politically neutral CDNs.
* Avoid over-reliance on HTTP/2 and consider fallback routes for HTTP/1.1.
* Split large JS bundles to reduce triggering DPI filters that inspect for signatures.
* Add `noscript` and meta-refresh fallbacks to redirect to a lightweight diagnostics page.
* Use server-side rendering (SSR) or partial hydration where possible to allow minimal interactivity even if JS fails.

# Workaround Options for Users (Non-VPN Methods)

* Try HTTP over HTTP/1.1: Use proxy servers that downgrade HTTP/2 to HTTP/1.1.
* Use mobile networks or different providers not yet affected.
* Download and self-host control panel locally (if provided by manufacturer).
* Request Wiren Board to provide static snapshots or embedded local access versions.
* Corporate firewalls or DNS-based routing tools can allow selective domain-based routing for critical services.

# Conclusion

The issue is not on the user's local device or browser. The connectivity failure results from a combination of how React-based apps rely on CDN-loaded dynamic JS and ongoing regional filtering by Russian authorities affecting HTTP/2 and international CDNs. Alternative delivery or internal hosting options may help circumvent the problem without resorting to VPNs.