

# Native Executable Creation for Windows and macOS

## 1. Overview

This document describes the approach used to package the **Network Monitoring Dashboard (HPE Project)** into **native executable applications** for **Windows and macOS**.

The objective was to deliver a **single-click, dependency-free application** that:

- Runs directly on the host operating system
- Captures real network traffic using system interfaces
- Does not require Python or Node.js installation on the user machine
- Starts both frontend and backend together

Linux deployment is handled separately using Docker and is **out of scope for this document**.

## 2. Why Native Executables Were Chosen

### **2.1 Limitations of Containers on Windows/macOS**

- Docker on Windows/macOS runs inside a **virtual machine**
- VM network adapters cannot access the host's NIC in promiscuous mode
- Packet capture tools (tshark) inside containers only see container traffic
- This makes accurate live packet capture impossible

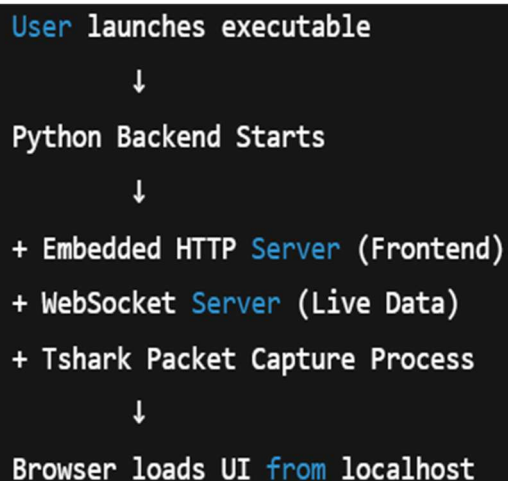
### **2.2 Advantages of Native Execution**

- Direct access to host network interfaces
- Full compatibility with Npcap (Windows) and OS networking stack
- No virtualization overhead
- Accurate system-wide traffic visibility

### **Conclusion:**

Native executables are mandatory for reliable packet capture on Windows and macOS.

## 3. High-Level Architecture



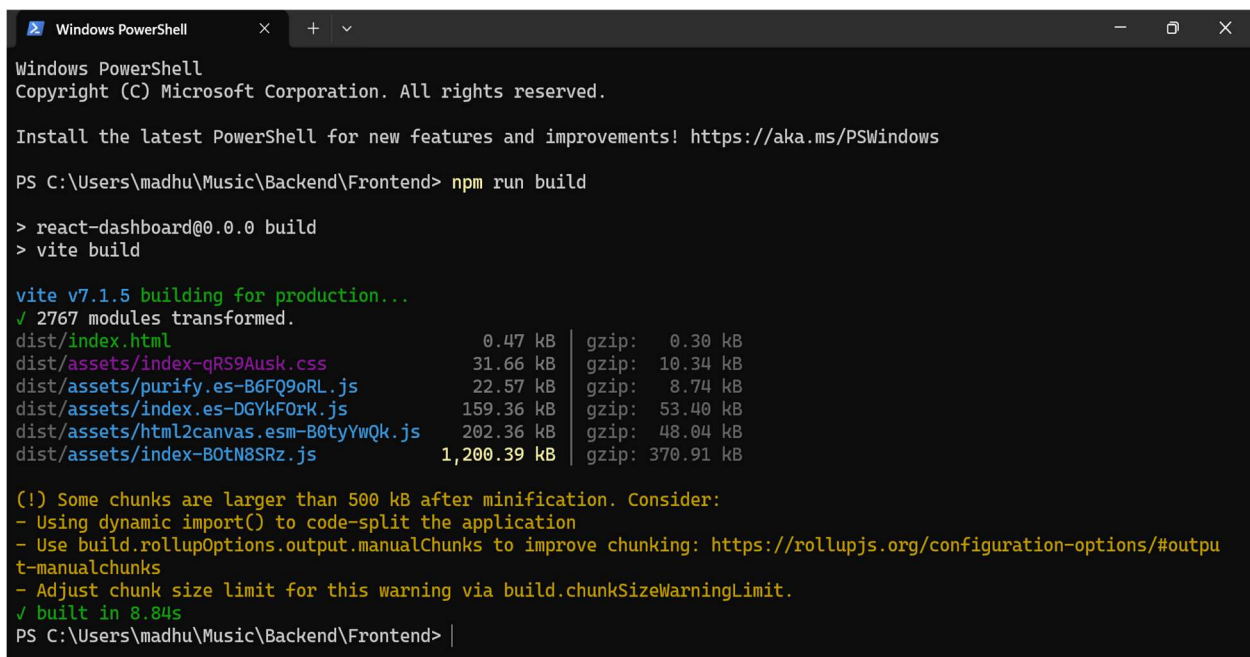
Frontend and backend are bundled and launched together as a single application.

## **4. Frontend Packaging Strategy**

### **4.1 Why Dynamic Frontend Serving Was Avoided**

Dynamic frontend servers (Vite / Node.js) are designed for development and introduce:

- Runtime dependency on Node.js
- Additional background processes
- Complexity in executable packaging
- Increased memory usage



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\madhu\Music\Backend\Frontend> npm run build

> react-dashboard@0.0.0 build
> vite build

vite v7.1.5 building for production...
✓ 2767 modules transformed.
dist/index.html                                0.47 kB | gzip: 0.30 kB
dist/assets/index-qRS9Ausk.css                 31.66 kB | gzip: 10.34 kB
dist/assets/purify.es-B6FQ9oRL.js             22.57 kB | gzip: 8.74 kB
dist/assets/index.es-DGYkF0rK.js              159.36 kB | gzip: 53.40 kB
dist/assets/html2canvas.esm-B0tyYwQk.js       202.36 kB | gzip: 48.04 kB
dist/assets/index-B0tN8SRz.js                  1,200.39 kB | gzip: 370.91 kB

(!) Some chunks are larger than 500 kB after minification. Consider:
- Using dynamic import() to code-split the application
- Use build.rollupOptions.output.manualChunks to improve chunking: https://rollupjs.org/configuration-options/#output-manualchunks
- Adjust chunk size limit for this warning via build.chunkSizeWarningLimit.
✓ built in 8.84s
PS C:\Users\madhu\Music\Backend\Frontend> |
```

### **4.2 Static Build Approach**

The frontend is built using:

*npm run build*

This produces a dist/ directory containing:

- index.html
- Minified JavaScript bundles
- Optimized CSS
- Static assets (images, icons)

After build:

- The frontend no longer requires Node.js
- React runs entirely in the browser

- Files can be served by any HTTP server

### 4.3 Embedding Frontend into Executable

- The dist/ folder is bundled inside the executable
- A Python HTTP server serves these files locally at:
- `http://localhost:8000`
- The frontend behaves like a hosted web app, but runs fully offline

## 5. Backend Packaging Strategy

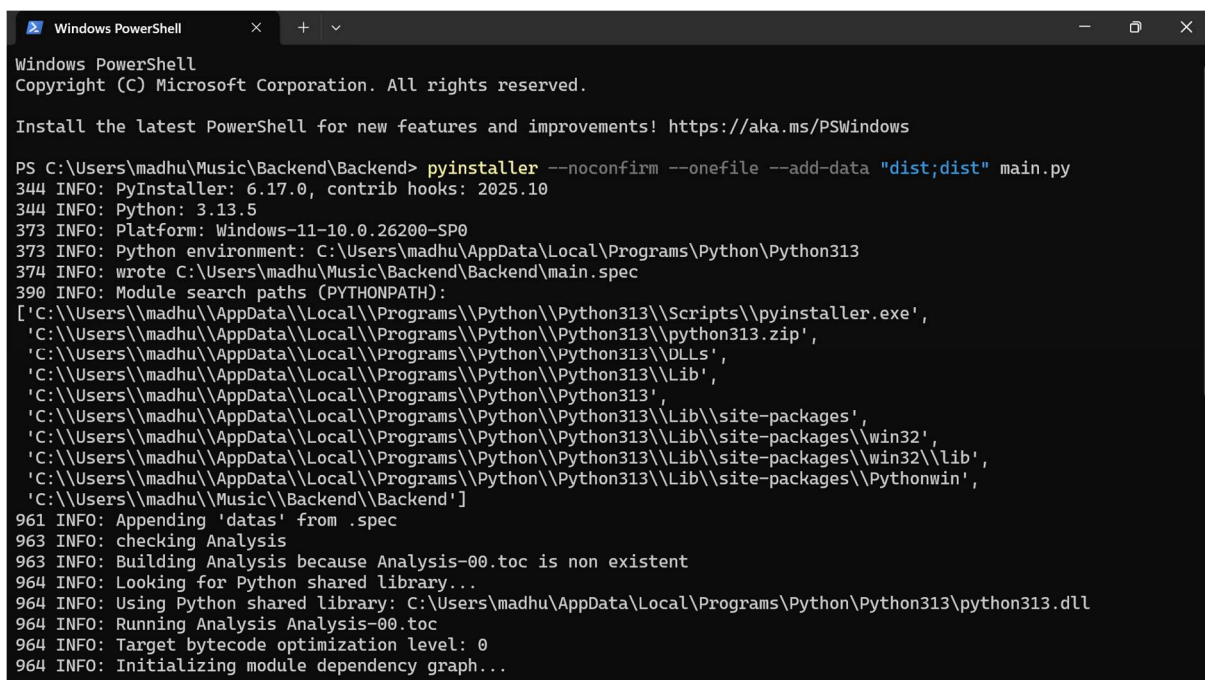
### 5.1 Tool Used: PyInstaller

PyInstaller is used to package the Python backend into a native executable.

#### Key capabilities:

- Bundles Python interpreter
- Includes all required Python libraries
- Packages application code and static assets
- Produces a single native binary:
  - .exe on Windows
  - .app on macOS

The user does **not** need Python installed.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\madhu\Music\Backend\Backend> pyinstaller --noconfirm --onefile --add-data "dist;dist" main.py
344 INFO: PyInstaller: 6.17.0, contrib hooks: 2025.10
344 INFO: Python: 3.13.5
373 INFO: Platform: Windows-11-10.0.26200-SP0
373 INFO: Python environment: C:\Users\madhu\AppData\Local\Programs\Python\Python313
374 INFO: wrote C:\Users\madhu\Music\Backend\Backend\main.spec
390 INFO: Module search paths (PYTHONPATH):
['C:\\Users\\madhu\\AppData\\Local\\Programs\\Python\\Python313\\Scripts\\pyinstaller.exe',
 'C:\\Users\\madhu\\AppData\\Local\\Programs\\Python\\Python313\\python313.zip',
 'C:\\Users\\madhu\\AppData\\Local\\Programs\\Python\\Python313\\DLLs',
 'C:\\Users\\madhu\\AppData\\Local\\Programs\\Python\\Python313\\Lib',
 'C:\\Users\\madhu\\AppData\\Local\\Programs\\Python\\Python313',
 'C:\\Users\\madhu\\AppData\\Local\\Programs\\Python\\Python313\\Lib\\site-packages',
 'C:\\Users\\madhu\\AppData\\Local\\Programs\\Python\\Python313\\Lib\\site-packages\\win32',
 'C:\\Users\\madhu\\AppData\\Local\\Programs\\Python\\Python313\\Lib\\site-packages\\win32\\lib',
 'C:\\Users\\madhu\\AppData\\Local\\Programs\\Python\\Python313\\Lib\\site-packages\\Pythonwin',
 'C:\\Users\\madhu\\Music\\Backend\\Backend']
961 INFO: Appending 'datas' from .spec
963 INFO: checking Analysis
963 INFO: Building Analysis because Analysis-00.toc is non existent
964 INFO: Looking for Python shared library...
964 INFO: Using Python shared library: C:\Users\madhu\AppData\Local\Programs\Python\Python313\python313.dll
964 INFO: Running Analysis Analysis-00.toc
964 INFO: Target bytecode optimization level: 0
964 INFO: Initializing module dependency graph...
```

```
Windows PowerShell
C:\Users\madhu\AppData\Local\Programs\Python\Python313\Lib\site-packages\pydantic\experimental\__init__.py:7: PydanticExperimentalWarning: This module is experimental, its contents are subject to change and deprecation.
  warnings.warn(
57270 INFO: Extra DLL search directories (AddDllDirectory): []
57270 INFO: Extra DLL search directories (PATH): []
57620 INFO: Warnings written to C:\Users\madhu\Music\Backend\Backend\build\main\warn-main.txt
57722 INFO: Graph cross-reference written to C:\Users\madhu\Music\Backend\Backend\build\main\xref-main.html
57832 INFO: checking PYZ
57833 INFO: Building PYZ because PYZ-00.toc is non existent
57833 INFO: Building PYZ (ZlibArchive) C:\Users\madhu\Music\Backend\Backend\build\main\PYZ-00.pyz
58726 INFO: Building PYZ (ZlibArchive) C:\Users\madhu\Music\Backend\Backend\build\main\PYZ-00.pyz completed successfully.
58802 INFO: checking PKG
58802 INFO: Building PKG because PKG-00.toc is non existent
58802 INFO: Building PKG (CArchive) main.pkg
66893 INFO: Building PKG (CArchive) main.pkg completed successfully.
66919 INFO: Bootloader C:\Users\madhu\AppData\Local\Programs\Python\Python313\Lib\site-packages\PyInstaller\bootloader\Windows-64bit-intel\run.exe
66920 INFO: checking EXE
66920 INFO: Building EXE because EXE-00.toc is non existent
66920 INFO: Building EXE from EXE-00.toc
66921 INFO: Copying bootloader EXE to C:\Users\madhu\Music\Backend\Backend\dist\main.exe
67002 INFO: Copying icon to EXE
67048 INFO: Copying 0 resources to EXE
67048 INFO: Embedding manifest in EXE
67088 INFO: Appending PKG archive to EXE
67265 INFO: Fixing EXE headers
67693 INFO: Building EXE from EXE-00.toc completed successfully.
67719 INFO: Build complete! The results are available in: C:\Users\madhu\Music\Backend\Backend\dist
PS C:\Users\madhu\Music\Backend\Backend> |
```

## **6. Static Asset Bundling**

### **6.1 Frontend Assets**

- React build output (dist/) is included using PyInstaller's --add-data
- Assets are extracted at runtime into a temporary directory
- The backend resolves paths dynamically to serve files correctly

### **6.2 Runtime Static Server**

- A lightweight Python HTTP server serves:
  - / → index.html
  - /assets/\* → frontend assets
  - /leaflet/\* → map icons and static resources
- SPA fallback routing is handled for client-side navigation

## **7. Runtime Dependency Validation**

### **7.1 External Dependencies**

The application depends on:

- **Wireshark (TShark)**
- **Npcap** (Windows packet capture driver)
- **A modern browser**

These cannot be bundled due to:

- Kernel-level drivers

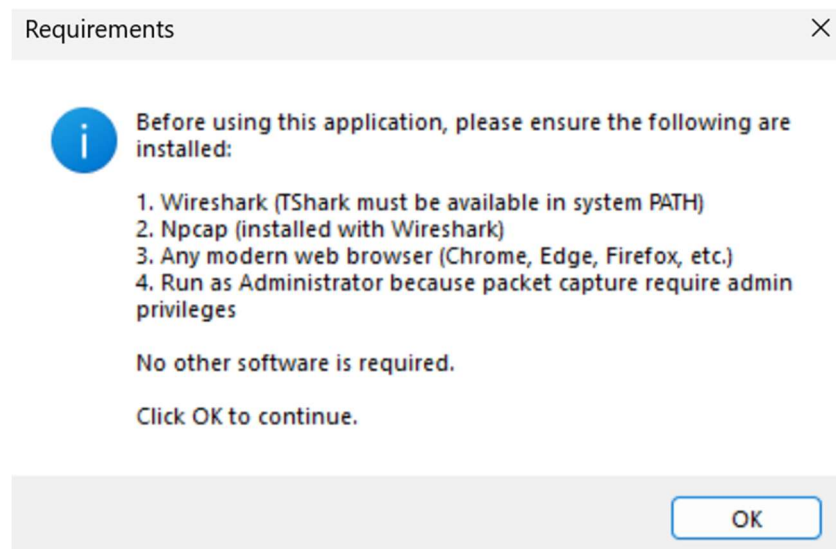
- OS security constraints
- Licensing considerations

## **8. Startup Validation Flow**

### **8.1 Initial Requirements Notification**

On startup, an informational popup explains:

- Required dependencies
- Packet capture constraints
- Browser usage



### **8.2 Tshark Presence Check**

The application checks:

```
shutil.which("tshark")
```

This verifies that:

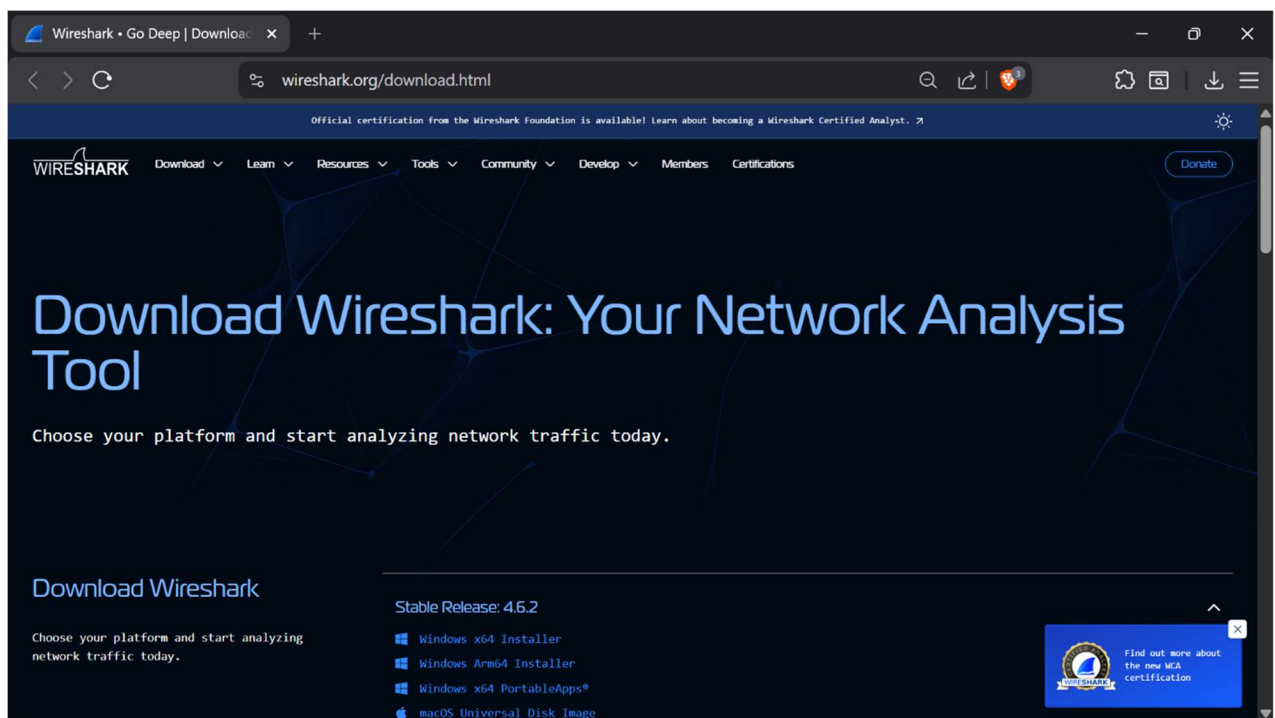
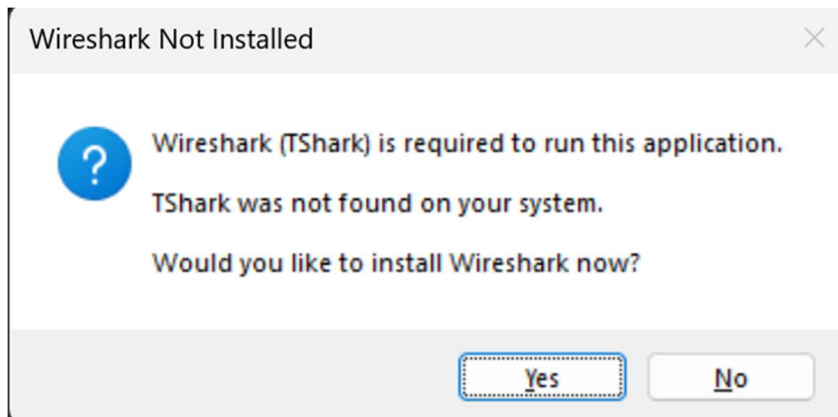
- Wireshark is installed
- Tshark is accessible via system PATH

### **8.3 Missing Dependency Handling**

If tshark is not found:

- A custom popup is shown with two options:
  - **Install Wireshark**
  - **Exit**
- Choosing **Install** redirects the user to the official Wireshark installer
- Application terminates cleanly

This prevents partial startup or silent failures.



## 9. Terminal-Based Execution and Log Visibility

The application executable is intentionally created as a **terminal-based application** rather than a silent background process.

### Purpose:

- To allow users and developers to view runtime logs directly
- To provide immediate feedback on:
  - Backend startup status
  - WebSocket server initialization

- Packet capture lifecycle (start/stop)
- Error and exception messages

```

Network Monitoring Dashboard - Backend
Starting WebSocket server on ws://localhost:8765
Starting HTTP static server at http://localhost:8000 serving C:\Users\madhu\AppData\Local\Temp\_MEI168642\dist
Client 2586592691936 connected. Total clients: 1
Found 8 network interfaces
Found 8 network interfaces
All packets cleared
Starting tshark on interface: 4
Tshark started successfully on interface 4
Metrics calculation took: 0.41ms
Metrics calculation took: 2.90ms
Metrics calculation took: 0.23ms
Metrics calculation took: 0.24ms
Metrics calculation took: 0.83ms
Metrics calculation took: 0.36ms
Metrics calculation took: 0.28ms
Metrics calculation took: 0.19ms
Metrics calculation took: 0.56ms
Metrics calculation took: 0.27ms
Metrics calculation took: 0.42ms
Metrics calculation took: 0.27ms
Metrics calculation took: 0.42ms
Metrics calculation took: 0.53ms

```

#### Benefits:

- Improves transparency during execution
- Simplifies debugging and troubleshooting
- Prevents silent failures during packet capture
- Enables easier validation in testing and enterprise environments

This design choice ensures that operational behavior of the application remains **observable and verifiable**, which is critical for a network monitoring system.

## 10. Windows Executable Creation Process

### 10.1 Steps

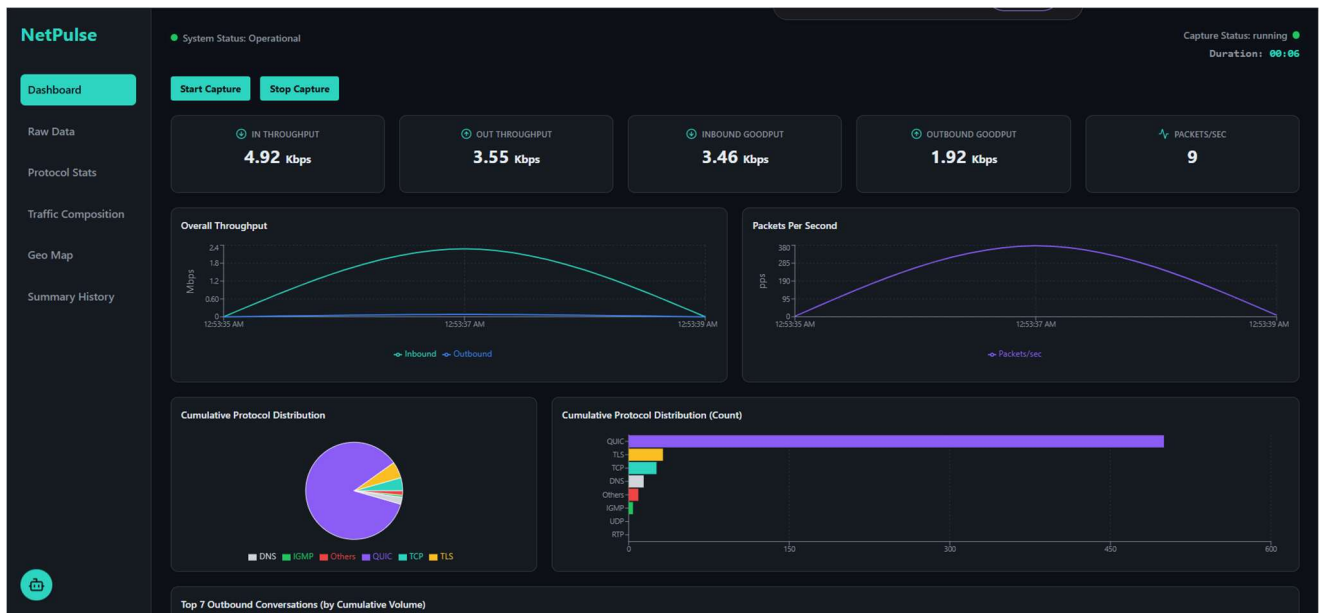
1. Finalize Python backend
2. Build React frontend (npm run build)
3. Move dist/ into backend directory
4. Use PyInstaller with static asset inclusion
5. Generate single .exe file
6. Test on a clean Windows system

### 10.2 Output

- One .exe file
- No Python, Node.js or Vite dependency



- External dependency checks handled at runtime



**NetPulse** System Status: Operational Capture Status: running

**Raw Data of Captured Packets (673 shown of 673)**

Filter by: All Fields Enter value...

No.	Time	Source	Destination	Protocol	Length	Info
1	00:53:34.944	91.108.56.162	192.168.29.228	SSL	159	Continuation Data
2	00:53:34.945	192.168.29.228	91.108.56.162	SSL	143	Continuation Data
3	00:53:35.022	91.108.56.162	192.168.29.228	TCP	54	443 → 61126 [ACK] Seq=106 Ack=90 Win=4838 Len=0
4	00:53:35.208	192.168.29.179	224.0.0.22	IGMPv3	62	Membership Report / Join group 239.255.255.250 for any sources / Join group 239.255.255.251 for any sources
5	00:53:35.208	192.168.29.179	224.0.0.22	IGMPv3	62	Membership Report / Join group 239.255.255.250 for any sources / Join group 239.255.255.251 for any sources
6	00:53:35.208	192.168.29.179	224.0.0.22	IGMPv3	62	Membership Report / Join group 239.255.255.250 for any sources / Join group 239.255.255.251 for any sources
7	00:53:35.758	2405:2015:04ee...	2405:2015:04ee...	DNS	95	Standard query 0x39a9 A www.gstatic.com
8	00:53:35.765	2405:2015:04ee...	2405:2015:04ee...	DNS	111	Standard query response 0x39a9 A www.gstatic.com A 142.250.193.99
9	00:53:35.768	2405:2015:04ee...	2405:2015:04ee...	DNS	95	Standard query 0x7585 A www.gstatic.com
10	00:53:35.768	2405:2015:04ee...	2405:2015:04ee...	DNS	95	Standard query 0x0cb6 AAAA www.gstatic.com
11	00:53:35.770	2405:2015:04ee...	2405:2015:04ee...	DNS	111	Standard query response 0x7585 A www.gstatic.com A 142.250.193.99
12	00:53:35.772	2405:2015:04ee...	2405:2015:04ee...	DNS	123	Standard query response 0x0cb6 AAAA www.gstatic.com AAAA 2404:6800:4002:808::2003
13	00:53:35.773	2405:2015:04ee...	2404:6800:4002...	TCP	86	51280 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
14	00:53:35.784	2404:6800:4002...	2405:2015:04ee...	TCP	86	443 → 51280 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 SACK_PERM WS=256
15	00:53:35.784	2405:2015:04ee...	2404:6800:4002...	TCP	74	51280 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
16	00:53:35.785	2405:2015:04ee...	2404:6800:4002...	SSL	1514	N/A
17	00:53:35.785	2405:2015:04ee...	2404:6800:4002...	TLSv1.2	391	Client Hello (SNL=www.gstatic.com)
18	00:53:35.792	2404:6800:4002...	2405:2015:04ee...	TCP	74	443 → 51280 [ACK] Seq=1 Ack=1441 Win=268032 Len=0

## 11. macOS Executable Creation Process

### 11.1 Steps

- Same Python and frontend codebase
- PyInstaller generates a macOS .app bundle
- Static assets embedded identically
- Runtime dependency checks remain unchanged

## 12. Conclusion



The executable-based deployment strategy enables a **production-ready, enterprise-friendly network monitoring application** that runs reliably on Windows and macOS while maintaining accurate access to host network interfaces.