NETVANE

Specifications

Technical specifications & system requirements

Document version 1.0

A doorvane project

NetVane v1.3.1 · August 31, 2026

The private, local network scanner and security monitor.

Contents

1. Introduction and overview

What this document covers, and what NetVane is.

This document sets out the technical specifications and system requirements for NetVane v1.3.1. It is intended for readers who need precise facts before installing or evaluating the software: what it runs on, what it needs, what it installs (and does not install), which technologies it is built from, how it scans, what it stores, and where its boundaries lie. It is reference material rather than a tutorial, so it favours tables and exact figures over step-by-step instruction.

NetVane is a local, private network scanner and security monitor. It runs entirely on your own machine and serves a dashboard in your browser at `http://127.0.0.1:8787`. From there it discovers the devices on your network, identifies what each one is, records which services are reachable on them, scores your overall exposure, and tracks how all of that changes over time. It is not a cloud or SaaS product, and it is not a live or authenticated vulnerability scanner. It does match detected service versions against a bundled, offline CVE database (NVD + CISA-KEV), but it reports what is *reachable* on the network rather than actively testing for exploitable flaws. Toward other devices it is non-destructive: it never attempts to exploit or alter them, never writes to them and never changes their configuration. It is not purely passive, though — to establish whether an open port is a genuine exposure it performs read-only protocol confirmations against the services it finds (section 10.1), and it sends credentials only in an opt-in, separately authorized mode that is off by default.

NetVane is a doorvane project. It is distributed as a Windows installer and a portable executable, both downloadable from netvane.doorvane.com; no account is required to obtain or run it.

> **Responsible use**
> Only scan networks you own or are explicitly authorized to test.

2. System requirements

What you need to run the prebuilt Windows binary comfortably.

The prebuilt binary is deliberately light. The minimum column below is what NetVane needs to run at all; the recommended column is what makes larger networks and repeated scans feel responsive. Neither column requires administrator rights, and no separate runtime, driver, or framework has to be installed.

Requirement	Minimum	Recommended
Operating system	Windows 10, 64-bit	Windows 11, 64-bit
CPU	Any modern dual-core processor	Quad-core or better
Memory (free)	~512 MB available RAM	~4 GB available RAM
Free disk space	~200 MB (app + database)	~200 MB on an SSD
Browser	None to set up — NetVane opens its own app window via an installed Chromium-family browser (Brave or Chrome preferred), falling back to a tab in your default browser	Brave or Chrome installed (for the chromeless app-window experience)
Privileges	Standard user (no admin)	Standard user (no admin)
Network	A wired or wireless connection to the network you intend to scan	Wired connection for large scans

The ~200 MB figure covers the application itself plus the local database. The database starts small and grows slowly as scans accumulate; see section 11 for detail on what it stores.

> **A note on RAM**
> Actual memory use is modest. The recommended figure leaves comfortable headroom for the browser dashboard, the map view, and larger device inventories rather than reflecting a hard demand of the scanner itself.

3. Supported platforms

Where NetVane runs — as a binary, and from source.

There are two ways to run NetVane: the prebuilt binaries (recommended) or, for developers, from source. Both currently target **Windows** — the prebuilt executables are Windows-only, and running from source uses a few Windows-native APIs, so macOS and Linux are not supported yet.

3.1 Prebuilt binaries

Distribution	`NetVane-Setup-1.3.1.exe` (per-user installer, recommended) and `NetVane-Portable-1.3.1.exe` (single-file portable)
Operating system	Windows 10 and Windows 11, 64-bit
Installation	Per-user installer — no administrator rights, no UAC prompt; or none at all (portable)
Runtimes required	None (Python and all libraries are bundled)

3.2 From source (advanced)

Operating systems	Windows (uses Windows-native APIs; macOS and Linux not supported yet)
Python	3.10 or newer (3.12 recommended)
Setup	`pip install -r requirements.txt`, then `python run.py`

Running from source is a developer convenience on Windows, not a cross-platform route: the source itself relies on Windows-native APIs, so macOS and Linux are not supported from source either.

4. Distribution and footprint

How NetVane is delivered and how much room it takes up.

Each release ships two self-contained builds of under 20 MB: a per-user **installer** (recommended) and a single-file **portable** executable. Neither ever asks for administrator rights. On launch, NetVane starts its local server and opens the dashboard in its own chromeless app window (section 2).

Delivery	`NetVane-Setup-1.3.1.exe` (installer, ~19 MB) and `NetVane-Portable-1.3.1.exe` (portable, ~18 MB)
Install location	Per-user, under `%LocalAppData%\Programs\NetVane` — with Start-menu and desktop shortcuts; installing a newer version removes the older one automatically
Administrator rights	Not required — no UAC prompt at install or run time
Data folder	`%LocalAppData%\NetVane` — settings (`config.yaml`), the database, and logs
Local database	One SQLite file, `netvane.db`, created on first run
Database growth	Starts small; grows slowly as scans, devices, and events accumulate

Removal is a normal Windows uninstall (Settings → Apps); for the portable build, delete the file. Either way, deleting the `%LocalAppData%\NetVane` data folder removes all stored data (see sections 11 and 12).

5. Dependencies

What the binary needs — and, just as importantly, what it does not.

The prebuilt binary has no external dependencies. Everything it needs to run is bundled inside the single executable. There is exactly one optional dependency, and several things that are commonly assumed to be necessary but are not.

Required for the binary	Nothing beyond a supported version of Windows
Optional	**nmap** — adds OS fingerprinting and richer version detection when you explicitly select the nmap engine in Settings; never used automatically

5.1 Not required

- **Npcap / WinPcap** — not required; the built-in scanner uses ordinary TCP connections and the OS ARP table.
- **.NET** — not required.
- **Python** — not required for the binary (it is bundled). Only needed when running from source.
- **Administrator rights** — not required at any point.

> **Optional nmap**
> NetVane works fully without nmap: the built-in native engine handles scanning on its own by default. nmap runs only when you explicitly select the nmap engine — installing it changes nothing until you do. See section 7.

6. Technology stack

The components NetVane is built from.

NetVane is a Python application with a browser-based front end, packaged into a single Windows executable. Every component is bundled or self-hosted: there are no external fonts, scripts, CDNs, or telemetry endpoints, and the application runs fully offline.

Language / runtime	Python 3.12
Web framework	FastAPI
Application server	uvicorn
Database	SQLite, in WAL (write-ahead logging) mode
Front end	Vanilla-JavaScript single-page dashboard
Network map	vis-network (vendored — bundled, not fetched)
Fonts	Self-hosted Inter and JetBrains Mono
Live updates	Server-Sent Events (SSE)
Background mode	Optional system-tray integration with native Windows notifications and per-user start-at-login — both off by default
Packaging	PyInstaller onefile + an Inno Setup per-user installer
Connectivity	Offline-first — no external fonts, scripts, CDNs, or telemetry. Outbound traffic happens only at your request: optional update checks (no network data sent), a webhook you configure, a support report you send, and the speed test / DNS benchmark when you run them

7. Scan engines

Two engines — the built-in one by default, nmap only if you choose it.

NetVane can scan using its own built-in engine or, when available, nmap. The built-in engine is the default and NetVane never switches away from it on its own; when nmap is the active engine a header pill in the dashboard says so.

7.1 Built-in native engine

The native engine is pure Python and requires nothing to be installed. It runs unprivileged and performs:

- TCP-connect port scanning (standard connections, no raw sockets);
- reading the operating system's ARP table to map MAC addresses to vendors;
- reverse-DNS lookups to resolve device names;
- service banner grabs;
- TCP-connect latency (round-trip time) measurement.

7.2 Optional nmap engine

The nmap engine runs only when you explicitly select it in Settings — NetVane never switches to it on its own, even when nmap is installed. When selected, nmap adds operating-system fingerprinting and richer service-version detection on top of what the native engine provides.

7.3 Engine selection

Settings offers two choices; the third value is accepted in the configuration file for completeness.

Built-in — fast & reliable `auto`	Default. Uses the built-in native engine (fast, unprivileged, zero-dependency).
Deep scan (nmap) `nmap`	Uses nmap — the only setting under which nmap runs.
`native` **(config file only)**	Explicitly pins the built-in engine; currently identical to `auto`, and not offered in Settings.

7.4 Experimental scanning (opt-in)

An **Experimental scanning** toggle in Settings — **off by default** — enables two features that are being validated before they become standard behaviour:

Two-phase deep scans	With the nmap engine, a fast live-host sweep runs first and the slow, detailed scan is pointed only at the hosts that answered — far faster on sparsely-populated ranges.
Multi-subnet auto-targeting	On a fresh configuration, all local IPv4 /24 networks the machine is attached to are detected and targeted (capped at five), instead of a single assumed network.

With the toggle off, scanning behaves exactly as sections 7.1–7.3 and 8 describe.

7.5 Experimental features (opt-in)

Two further capabilities live under **Settings → Experimental features**, each **off by default** and independently toggled while they are validated:

IPv6 device discovery	In addition to the IPv4 scan, NetVane lists IPv6-only devices from the operating system's IPv6 neighbour (NDP) cache — address, MAC/vendor, and reverse-DNS name. IPv4 remains the primary scan; these additional hosts are discovery-only and are not port-scanned. Group, multicast, loopback and unspecified addresses are excluded.
SNMP switch topology	For managed switches, NetVane reads each switch's bridge forwarding table over SNMP v2c (read-only, LAN-only — nothing leaves the machine) and places each device on the switch it actually connects through in the Map. It uses a self-contained SNMP client (no additional dependency). Configure the switch IPs and read community, then use *Sync now*. A manually assigned "Connected via" link always takes precedence over an SNMP-derived one.

With both toggles off, discovery and topology behave exactly as the rest of this document describes.

8. Scan profiles and ports

Five profiles trading depth against speed. All observe only.

A scan profile controls how many ports are checked and how much detection is performed. Every profile is observation-only — none of them attack or change the target devices. Standard is the recommended default.

Profile	Ports covered	Relative speed	Typical use
Discovery	None — live hosts only	Fastest	Quickly find which devices are online.
Quick	~Top 100 ports, light service detection	Fast	A rapid look at the most common services.
Standard	Top 1000 ports + service versions + OS guess	Moderate	Recommended default for routine scans.
Thorough	All 65,535 ports + full version and OS detection	Slow (minutes per host)	Deep coverage when nothing should be missed.
Aggressive	All ports + full OS/version + safe discovery scripts + traceroute	Slowest	A one-off, in-depth audit.

Time scales with depth

Thorough and Aggressive scans check every one of the 65,535 ports per host and can take several minutes for each device. Use them deliberately rather than for routine monitoring.

9. Notable-exposure categories

The categories of reachable service NetVane highlights.

When a scan finds a reachable service that commonly carries risk, NetVane groups it into one of the categories below. Each carries a brief explanation of why it matters; the dashboard adds per-category remediation guidance. The ports listed are representative examples, not the full set.

Category	Representative ports	Why it matters
Cleartext	Telnet 23, FTP 21	Credentials and data travel unencrypted and can be read on the wire.
File-share	SMB 445, NetBIOS 139	Exposed file sharing is a frequent target for lateral movement and ransomware.
Windows RPC	135	Remote procedure call surface that is often better kept off the network.
Remote desktop	RDP 3389, VNC 5900	Remote-control access; a high-value target if reachable or weakly secured.
Database	MySQL 3306, Postgres 5432, MSSQL 1433, Redis 6379, Mongo 27017, Elastic 9200, memcached 11211	Databases exposed to the network can leak or lose data if unauthenticated.
Container	Docker 2375	An open Docker API can allow full control of the host running it.
SSH	22	Administrative access; worth confirming it is intended and locked down.
Printer	9100	Raw print services can leak documents or be abused if left open.
UPnP	1900, 5000	Automatic port-mapping and discovery services that can widen exposure.
Admin panel	8080, 8443	Web management interfaces that should not be broadly reachable.

10. Network behaviour and protocols

What NetVane puts on the wire, and what it never does.

NetVane establishes ordinary connections and reads information that devices already advertise or will volunteer to any client. It sends no exploit traffic and never attempts to change a device's configuration. This table is the complete inventory of what NetVane puts on the wire.

TCP connect	Standard TCP connections to test whether ports are open (no raw packets).
ARP	Reads the local ARP table to map MAC addresses to vendors.
Reverse DNS (PTR)	Resolves device hostnames from their IP addresses.
Service banners	Reads the greeting a service offers on connection, where one is offered.
Risk-validation probes (on by default)	Read-only protocol handshakes against notable open ports, to confirm whether each is a real exposure — see 10.1. Credentials are sent only in the opt-in authorized mode.
SSDP / UPnP discovery (on by default)	Device-identification sweep: a multicast discovery request, then a description fetch restricted to private-IP HTTP URLs. Turn off *device identification* in Settings and a scan sends nothing but TCP connect probes.
UPnP-IGD to the router (on by default)	Reads the gateway's own status for the Internet / WAN health panel and the behaviour watch. LAN-only; the router is the only device asked.
DHCP discovery	Enumerates DHCP servers on the LAN to detect a rogue one.
mDNS (optional)	Used only during an opt-in Identify lookup for richer device names.
NetBIOS (optional)	Used only during an opt-in Identify lookup for richer device names.
Wake-on-LAN (on request)	A magic packet, sent only when you press Wake for a device.
SNMP (off by default)	Read-only queries to managed switches you configure, for switch-port topology.
NetFlow / IPFIX (off by default)	A UDP listener on the port you choose; NetVane receives, and never sends.
DNS benchmark (on request)	Test queries to your configured resolvers and to well-known public ones, when you run the benchmark. Leaves your network by design.
Speed test (on request)	A throughput test against a public endpoint, when you run it. Leaves your network by design.

10.1 Active risk validation

After each scan NetVane probes the notable open ports it found to determine whether each is a genuine exposure or a properly secured service, so the Security view can clear the harmless ones rather than leaving every open port on your action list. Three levels, set in Settings:

`safe` **(default)**	Read-only protocol handshakes and banner reads. No credential is ever sent. For services that answer unauthenticated clients by design (e.g. an exposed Redis or Elasticsearch), a single read-only status command is issued — that response *is* the evidence of exposure.
`off`	No probing. Findings remain unverified and nothing is cleared automatically.
`authenticated`	Additionally attempts a small set of well-known default and blank credentials. Inert unless the operator ticks a target-scoped authorization (auto-revoked if the target range changes) *and* the Aggressive profile is selected. Attempts are hard-capped per host and service to avoid account lockouts.

All levels are bounded by per-host and overall time budgets; a validation pass that runs out of budget keeps its partial results and leaves the remainder unverified.

> **Non-destructive by design**
> NetVane never exploits a device, never writes to one, and never modifies the configuration of anything it scans. It logs in to nothing unless you explicitly select the authenticated validation level and authorize it for your own network.

11. Data and storage

Everything lives in one local file.

All of NetVane's data is kept in a single local SQLite database file, `netvane.db`, on the machine running the app. There is no remote store and no separate data service.

Storage location	Installed build: `%LocalAppData%\NetVane\netvane.db`, the same folder holding `config.yaml` and the logs. Portable build: a `NetVane-Data` folder beside the executable, so nothing is written to your user profile. From source: the working directory.
Scans	Each scan run, its profile, and its results.
Devices	The device inventory, including type, vendor, and identity details.
Observations	Open ports, services, banners, and related findings per device.
Events	Change history — devices arriving/departing, ports opening/closing, and more.
How long history is kept	Scans, observations, and events: everything from the last **180 days**, plus the **120 most recent scans** and the detail recorded with them whatever their age. A scan is removed only when it is both older than 180 days and outside those 120 — so a weekly scan cadence keeps over two years of history, and a monthly one a decade. Alerts and notices not attached to a scan age out at 180 days. Both figures are the defaults and are **adjustable in Settings** (7–3650 days; 30–100,000 scans); lowering them **never deletes history you already have**, because each scan retains the window it was recorded under.
Never removed automatically	Devices, labels, notes, accepted risks, zones, people, and settings. Only the scan-by-scan detail behind them ages out.
Shorter rolling windows	Internet and gateway quality samples: last 7 days. Data-volume samples behind behaviour watch: last 14 days. Speed-test results: last 50. Optional flow collector: an external destination not seen for 60 days leaves that device's learned-normal list.

This tidy-up runs on the local machine at the end of a scan. Nothing is transmitted when it runs and no configuration is required. To retain history beyond these windows, take a backup (**Settings → Data & backup → Download backup**) or copy the database file while the app is closed.

11.1 Exports

NetVane can produce two kinds of export from this data:

- a self-contained, printable HTML security report (use your browser's Print → Save as PDF);
- a CSV inventory export of the device list.

To delete all NetVane data, stop the app and delete the `%LocalAppData%\NetVane` folder along with any exports you saved.

12. Privacy and security posture

Local by default; almost nothing leaves the machine.

NetVane is built to keep your network data on your machine. It has no accounts, no telemetry, and no cloud component. Nothing about your network ever leaves the machine; its only network use is optional, on-demand, integrity-checked app, CVE-database, and device-fingerprint update checks, and it fetches no fonts or scripts.

Accounts	None
Telemetry	None
External calls	Optional, integrity-checked app-update, CVE-database, and device-fingerprint update checks only — they send nothing about your network. No CDNs; fonts and scripts are self-hosted.
Network binding	The dashboard binds to `127.0.0.1:8787` (local machine only)
Only outbound path for network data	An optional, user-configured `https://` webhook for change alerts — the sole traffic that carries information about your network, and only if you set it
Scan authorization	A one-time in-app acknowledgment that you are authorized to scan the network is required before the first scan
Code signing	Authenticode, GRCSAC (Azure Artifact Signing, RFC-3161 timestamped)

> **Signed binaries**
> The NetVane executables are signed as GRCSAC, so Windows names the publisher. A recently issued certificate can still draw a SmartScreen prompt until it builds reputation, and antivirus software may warn on first run. To proceed, choose **More info → Run anyway**. No administrator rights are required.

13. Performance characteristics

What to expect at start-up and during scans.

NetVane is packaged as a single onefile executable, which is unpacked into memory at launch. The first cold start therefore takes longer than subsequent runs.

Cold start	Roughly 10–40 seconds on the first launch; faster afterwards
Per-host scan time	Scales with the chosen profile — from near-instant (Discovery) to minutes per host (Thorough/Aggressive)
Live updates	Server-Sent Events push scan progress and results to the dashboard in real time

Because results stream to the dashboard as they are found, you can watch a scan populate rather than waiting for it to finish. Overall scan duration depends on the profile, the number of live hosts, and network conditions.

14. Limitations

Where NetVane's scope ends.

NetVane is deliberately focused. Knowing what it does not do is as useful as knowing what it does.

- **Windows only.** The ready-to-run executables target Windows 10/11 64-bit, and the source itself uses Windows-native APIs — macOS and Linux are not supported (section 3).
- **IPv4 networks by default.** Discovery targets IPv4 addresses and ranges. With the opt-in IPv6 device discovery of section 7.5 enabled, IPv6 neighbours are listed for awareness but are not port-scanned; with it off, IPv6-only devices are not discovered yet.
- **Offline CVE matching, not a live scanner.** NetVane reports which services are *reachable* and matches their versions against a bundled NVD + CISA-KEV database; it does not actively test for or exploit specific vulnerabilities.
- **Switch-level wiring is not auto-discovered.** A network scan can't see which switch port a device is wired to, so by default devices are drawn one hop under the gateway. You can record it manually per device with the Map's *"Connected via"* control, which always takes precedence. If exactly one Wi-Fi access point is visible on a network, NetVane makes a best-effort guess that nearby camera/smart-home devices connect to it over Wi-Fi — shown as a dashed line with an explanatory note, never presented as a confirmed fact.
- **Double-NAT nesting needs a Deep scan.** When a Deep scan's traceroute data shows multiple devices agreeing that one detected network's own gateway sits upstream of another (e.g. an ISP router feeding your own router), the Map nests the second network under the first — again a guess, marked as such. The default scan engine collects no traceroute data, so this only ever applies to Deep scans.
- **Identify depends on device cooperation.** The optional mDNS and NetBIOS identify lookups only return results if the target device responds and the local firewall does not block the traffic.

> **Reachable, not exploitable**
> A finding means a service can be reached from the network — it is a signal to review, not a confirmation that the service is vulnerable.

15. Document control

Version and status of this document.

Product	NetVane v1.3.1
Document	Specifications & System Requirements
Document version	1.0
Date	August 31, 2026
Publisher	A doorvane project

15.1 Revision history

Updated with each revision of this document.

Version	Date	Summary
1.0	July 30, 2026	Initial release.

> **Subject to change**
> Specifications, figures, and behaviour described here may change as the product develops. Where this document and the running software disagree, the software is authoritative.