

Readiness report — SatNOGS Satellite Ground Station Client

Checked 2026-09-03 03:49 · 2 searches, 12 pages fetched

Readiness: SatNOGS Satellite Ground Station Client

Verdict: Ready · 2026-09-02 · est. parts cost: none — software-only upload, no bill of materials **Cybersecurity — software & network: Review before networking**

This is the cold check for this lab: no previous readiness report exists, so this report establishes the baseline that future checks measure drift against. The verdict is about sourcing, and this upload has nothing to source. The snapshot records no bill of materials, no schematics or board files, and nothing to fabricate — it is the satnogs-client software package: packaging files, metadata and Python manifests. With no part numbers, no quantities and no purchase path, there is nothing that can go obsolete, out of stock or overpriced between now and the next check. The external resources the lab genuinely depends on to be usable were confirmed reachable this run: the GitLab source repository (30 entries listed on branch `master`, seven project files fetched from it) and the documentation site both responded. One caveat is outside the instructor's control: the community forum carried a maintenance banner at check time announcing intermittent availability of SatNOGS Network, the service this client pulls its observation jobs from.

Action required

None — this lab has no bill of materials, so there are no parts to source.

Cybersecurity — software & network

Review before networking — Eight findings on a network-connected client, including known flaws in the HTTP stack behind every SatNOGS API call and a base operating system past its standard end-of-life, with three of the five library issues blocked from a straightforward upgrade by the project's own version pins.

Component	Version	How known	Support status
satnogs-client (the lab's application)	not determined	not determined (the package version is generated at build time by versioneer from git metadata; no release is pinned in any fetched file)	—

Component	Version	How known	Support status
Debian (base OS of the reference container)	12	read from project (base image debian:bookworm-slim pinned as ARG DEBIAN_IMAGE_TAG in fetched/08-Dockerfile (docker-gnuradio); the same file adds the SatNOGS apt repository for Debian_12)	extended/LTS support only until 2028-06-30
Python interpreter	3.11	inferred (setup.cfg declares a lower bound, python_requires = >=3.11, and a Programming Language :: Python :: 3.11 classifier - a floor, not an exact pin)	supported until 2027-10-31
GNU Radio (SDR framework, container base image)	3.10.5.1-satnogs	read from project (ARG GNURADIO_IMAGE_TAG=3.10.5.1-satnogs in fetched/06-Dockerfile)	—
urllib3	1.26.20	read from project (pinned in fetched/01-requirements.txt and fetched/03-constraints.txt)	—
requests	2.28.2	read from project (pinned in fetched/01-requirements.txt and fetched/03-constraints.txt)	—
sentry-sdk	1.9.10	read from project (pinned in fetched/01-requirements.txt and fetched/03-constraints.txt)	—
pillow	12.2.0	read from project (pinned in fetched/01-requirements.txt and fetched/03-constraints.txt)	—
python-dotenv	0.21.1	read from project (pinned in fetched/01-requirements.txt and fetched/03-constraints.txt)	—
APScheduler	3.9.1.post1	read from project (pinned in fetched/01-requirements.txt)	—
numpy	1.26.4	read from project (pinned in fetched/01-requirements.txt)	—
matplotlib	3.8.4	read from project (pinned in fetched/01-requirements.txt)	—
h5py	3.13.0	read from project (pinned in fetched/01-requirements.txt)	—
skyfield	1.54	read from project (pinned in fetched/01-requirements.txt)	—
sgp4	2.25	read from project (pinned in fetched/01-requirements.txt)	—
satnogs-predict	0.6	read from project (pinned in fetched/01-requirements.txt)	—
versioneer	0.28	read from project (pinned in fetched/04-pyproject.toml build-system requires and in fetched/01-requirements.txt)	—

Component	Version	How known	Support status
setuptools (build system)	not determined	unpinned (fetched/04-pyproject.toml requires setuptools>=40.8.0 - a lower bound only, so the build takes whatever is current)	—
wheel (build system)	not determined	unpinned (fetched/04-pyproject.toml requires wheel with no version constraint at all)	—
Debian system packages (packages.debian)	not determined	unpinned (fetched/07-packages.debian lists 9 apt packages with no versions; the installed versions are whatever the Debian 12 archive serves at build time)	—

Component	Finding	Severity	Fix available?
pillow	The image library used to render observation waterfalls carries 13 known advisories at the pinned version 12.2.0, several of them memory-corruption bugs reachable by processing a malformed image, the most severe scoring 8.2. — GHSA-xj96-63gp-2gmr , GHSA-jjj6-mw9f-p565 , GHSA-6r8x-57c9-28j4 , GHSA-9hw9-ch79-4vh6 (+2 more)	high	Yes - update to 12.3.0, which fixes all of them; nothing in setup.cfg constrains Pillow (it arrives through matplotlib), so this needs only a regenerated requirements.txt and constraints.txt.
sentry-sdk	The crash-reporting client that sends error telemetry off the ground station to a remote service can leak sensitive session information when the send-default-PII option is switched on, and exposes environment variables to processes it spawns. — GHSA-29pr-6jr8-q5jm , GHSA-g92j-qhmq-64v2	high	No - project pins sentry-sdk ~1.9.0 in setup.cfg, a compatible-release range that excludes the fixed 1.14.0 and later releases.
urllib3	The HTTP library that carries every call the ground station makes to the SatNOGS network has five known flaws at the pinned version 1.26.20, including decompression-bomb protections being bypassed on redirects and sensitive headers being forwarded to a different origin during proxied redirects. — GHSA-38jv-5279-wg99 , GHSA-qccp-gfcp-xxvc , GHSA-2xpw-w6gg-jr37 , GHSA-gm62-xv2j-4w53 (+1 more)	high	Mitigation only - every fixed release is in the urllib3 2.x line while the project pins urllib3==1.26.20 in both requirements.txt and constraints.txt, so clearing these means regenerating both lockfiles onto 2.x, not a drop-in bump.
python-dotenv	The configuration loader at the pinned 0.21.1 follows symbolic links when writing a key back to the settings file, which lets a local attacker who can place a symlink cause an arbitrary file to be overwritten. — GHSA-mf9w-mj56-hr94	moderate	No - project pins python-dotenv ~0.21.0 in setup.cfg, which excludes the fixed 1.2.2.

Component	Finding	Severity	Fix available?
requests	The HTTP client the software uses to pull observation jobs and upload results has four known issues at the pinned 2.28.2, including a proxy-credential header leaking on redirect and certificate verification not being reapplied on a reused session. — GHSA-j8r2-6x86-q33q , GHSA-9wx4-h78v-vm56 , GHSA-gc5v-m9x4-r6x2 , GHSA-9hjg-9r4m-mvj7	moderate	No - project pins requests <code>~=2.28.0</code> in <code>setup.cfg</code> , a compatible-release range that excludes every fixed version (2.31.0 and later).
Debian (base OS of the reference container)	The reference container is built on Debian 12 (Bookworm), which passed the end of its standard support on 2026-07-11 and now receives only Debian LTS security updates, scheduled to end 2028-06-30.	posture	Mitigation only - make sure the host actually pulls the Debian LTS security updates, and plan the move to Debian 13 well before June 2028 rather than at the deadline.
satnogs-client container image	The image creates the client's state directory / <code>var/lib/satnogs-client</code> world-writable (mode 1777), so on a shared ground-station host any other local account can write into the directory that holds observation artifacts and client configuration.	posture	Mitigation only - if that path is bind-mounted onto a shared host, tighten it to the <code>satnogs-client</code> user (0700 or 0755) rather than leaving it world-writable.
setuptools / wheel (build system)	The build configuration pins only <code>versioneer==0.28</code> : <code>setuptools</code> carries a <code>>=40.8.0</code> lower bound and <code>wheel</code> has no constraint at all, so every rebuild of the client pulls whatever those build tools happen to be that day and the build step has no checkable security state.	posture	Mitigation only - pin the build-system requirements to exact versions in <code>pyproject.toml</code> , or build only from the checked-in <code>constraints.txt</code> .

What this means for your lab

This software is a client, not a server: it dials out to the SatNOGS network, and nothing in the project files opens a listening port, sets a default password, or exposes an admin page, so someone else on your network has no obvious front door into it. The risk sits in what the client pulls in. The versions the project locks are old enough that five of its libraries carry publicly known flaws, and the ones that matter most are in the very code that fetches and uploads data over the network and in the image library used to draw observation waterfalls. In practice that means a malformed or hostile response from the network could crash an observation or, in the worst reported cases, corrupt memory in the client. Three of the five cannot simply be upgraded, because the project's own version rules pin them to the old line; the image library is the exception and has a clean fix available. The reference container's operating system, Debian 12, also passed its normal end-of-life this July and now gets only long-term security patches. What to do: put the ground station on its own network segment rather than the main campus network, keep it pulling Debian LTS updates, and if you rebuild

the client, regenerate its locked requirements so at least the image library moves to the fixed release. Two things worth knowing: the separate developer and test requirements file, which a normal deployment does not install, carries its own known issues, so keep development work off the station machine; and if you install on bare metal or a Raspberry Pi instead of using the container, the project never states which operating system version you should be on, so we could not judge that host's support state at all.

Not checked

- POSIX Linux host OS for a bare-metal or Raspberry Pi install: the snapshot's target platform is recorded only as 'POSIX Linux, version not stated'; no fetched file names a host OS version for a non-container install, so its support state is unknown. Only the reference Docker image's base OS (Debian 12) could be determined.
- GNU Radio 3.10.5.1-satnogs: no vulnerability database covers a GNU Radio build distributed as a container image and openSUSE Build Service packages; the version is pinned and known, but there is nothing to check it against
- SatNOGS apt packages from the openSUSE Build Service repository (home:librespace:satnogs2, Debian_12): installed from an external apt repository at image build time; no versions are pinned in any fetched file, so there is no exact version to check
- 9 Debian system packages in packages.debian (git, python3-gps, python3-hamlib, libhdf5-hl-100, libhdf5-103-1, libaec0, libsz2, libgfortran5, libopenblas0-pthread): listed with no version constraints; the versions actually installed are whatever the Debian 12 archive serves at build time
- USRP FPGA images downloaded by uhd_images_downloader at image build time: fetched over the network during the container build with no version or checksum recorded in any fetched file
- setup.cfg abstract dependency ranges (install_requires and the dev extras): declared as compatible-release ranges (~=) and, for pytz, with no constraint at all, so no exact version could be checked from that file. The concrete versions actually installed come from requirements.txt and constraints.txt, which were fetched and fully checked. Note the automated parse of setup.cfg also counted plain configuration keys (author, classifiers, line_length) as dependencies, which is why its unpinned count reads 60.

Parts

None. The snapshot's "Bill of Materials" section contains no entries: "There are no physical resistors, capacitors, semiconductors, connectors, heatsinks, or hardware sub-assemblies to inventory."

No part is listed as "not checked", because no part exists to check. This is an empty bill of materials, not an unchecked one — if a future re-ingest of this lab produces parts, they are new, not previously missed.

Substitutes

None needed.

Verify any substitute against its datasheet before ordering.

Other dependencies

The snapshot lists six external resources. All the load-bearing ones were fetched this run:

- **Project source repository** — gitlab.com/librespacefoundation/satnogs/satnogs-client: live. Listed via the GitLab API, 30 entries on branch `master`; seven project files were fetched from it successfully. This is the lab's single most important external dependency — it serves the code.
- **Project documentation** — docs.satnogs.org/projects/satnogs-client: live, fetched this run. Currently serves the User guide, Developer guide, "Working with Docker" and the module reference.
- **Community Discourse forum** — community.libre.space/c/satnogs: live, fetched this run; Hardware and Software support categories both present. Note the page carried an operational banner at fetch time: "Scheduled maintenance to reconfigure Docker service. Intermittent service availability is expected. Affected services: SatNOGS Network, SatNOGS DB, SatNOGS Dashboard." Because the client pulls its observation jobs from SatNOGS Network, confirm that service is up before treating a failed session as a station fault.
- **Organization website** — librespacefoundation.org: live, fetched this run.
- **Matrix channel** — <https://app.element.io/#/room/#satnogs:matrix.org>: not checked. A single-page web client whose room state a text fetch cannot observe.
- **IRC channel** — <https://web.libera.chat/?channels=satnogs>: not checked, same reason — a browser-based IRC client rather than a fetchable resource.

Both unchecked entries are community chat conveniences, not materials the lab depends on to run.

Fabrication: none. No Gerber, drill, layer stackup or board dimensional data exists in this upload.

Watch list

- **SatNOGS Network service availability** — the client is useless without it, and it was under announced maintenance at check time. The most likely cause of a "broken lab" session in the near term, and not something an instructor can fix locally.
- **The GitLab source repository** — the lab's only code source. A rename, archive or move of this project would break the lab outright; it is the first thing to re-confirm at the next check.

- **The documentation site** — `docs.satnogs.org` is a separate host from the repository and can drift independently; the lab's setup instructions live there.
- **A future re-ingest producing a hardware BOM** — this upload is the client software only, but a real ground station needs an SDR, antenna and rotator. If the lab materials are ever re-uploaded with that hardware included, the next check will be sourcing parts from a cold start with no price baseline.