SHADOW AI
DISCOVERY · SECSTUDIO

DEPLOYMENT & OPERATIONS GUIDE

Sensor Deployment Guide

Packaging, signing, MDM rollout, configuration, auto-update and fleet operations for the Shadow AI Discovery sensor.

AUDIENCE	VERSION	DATE
IT · Endpoint Engineering · SecOps	Sensor 1.1.0	13 August 2026

Overview

The Shadow AI Discovery sensor ships as a **single self-contained binary per platform** — one file, no Python, no virtualenv, no dependencies to install on the endpoint. It runs as a background service, discovers unsanctioned AI usage, and reports to your central collector.

This guide covers how the binary is built and signed, how it is published with versioning, how to roll it out at scale through your MDM with minimal manual work, what it needs to run, how updates are pushed, and how to verify a healthy fleet.

One binary, three platforms

PLATFORM	ARTIFACT	SERVICE MANAGER	INSTALL PATH
macOS	`shadow-ai-sensor` (universal2) + `.pkg`	LaunchDaemon	`/usr/local/shadow-ai-sensor/`
Windows	`shadow-ai-sensor.exe` + `.msi`	Windows Service	`C:\Program Files\ShadowAISensor\`
Linux	`shadow-ai-sensor` + `.deb` / `.rpm`	systemd	`/opt/shadow-ai-sensor/`

1. Build, sign & publish

Build

Each platform builds from the same source with PyInstaller, producing one frozen binary:

```
# macOS  (universal2, signed + notarized in CI)
bash packaging/build_macos.sh 1.1.0

# Linux  (built on oldest-supported glibc; optional .deb)
bash packaging/build_linux.sh 1.1.0

# Windows (Authenticode-signed .exe + optional MSI)
powershell -File packaging\build_windows.ps1 1.1.0
```

The official release path is the CI workflow `.github/workflows/build-sensor.yml`, which builds all three platforms on native runners, signs each with certificates injected from CI secrets (never stored in git), generates the manifest, and publishes to S3 via short-lived OIDC credentials.

Signing

Real distribution signing uses your organisation's certificates:

- **macOS:** Developer ID Application signature + notarization (`MAC_SIGN_IDENTITY`, `AC_NOTARY_PROFILE`). Gatekeeper then runs the binary without prompts.
- **Windows:** Authenticode signature over the `.exe` and `.msi` (`WIN_CERT_THUMBPRINT`), with an RFC-3161 timestamp so signatures outlive the cert. SmartScreen reputation builds as the signed binary is deployed.
- **Linux:** detached GPG signature over the binary and the manifest; your apt/yum repo verifies it.

Without those certificates the build scripts still run, producing an ad-hoc-signed (macOS) or unsigned artifact clearly marked for development only.

Versioned publishing

Every release is immutable under its own version prefix in the versioned S3 bucket, with a manifest recording each artifact's SHA-256:

```
s3://<sensor-bucket>/sensor/
  1.1.0/
    macos/shadow-ai-sensor
    macos/ShadowAISensor-1.1.0.pkg
    windows/shadow-ai-sensor.exe
    windows/ShadowAISensor-1.1.0.msi
    linux/shadow-ai-sensor
    linux/shadow-ai-sensor_1.1.0_amd64.deb
    manifest.json          # version + per-artifact sha256 + size
  latest/manifest.json     # pointer — advanced only to promote a build
```

Publish with `packaging/upload_s3.sh 1.1.0 <bucket>` (SSE-KMS, private ACL). A new build never auto-promotes: the `latest` pointer is advanced explicitly, so a bad build cannot reach the fleet by accident.

2. Configuration

The sensor reads one config file, `agent_config.json`, resolved from a fixed system location when running as a packaged service:

- macOS / Linux: `/etc/shadow-ai-sensor/agent_config.json`
- Windows: `%ProgramData%\ShadowAISensor\agent_config.json`

```
{
  "collector_url": "https://app.your-domain.com",
  "sensor_key": "<enrollment-secret-from-your-deployment>",
  "ca_cert": null,           // path to pin a private CA (recommended for pilots)
  "insecure_tls": false,     // pilot-only escape hatch; prints a warning
  "auto_update": false,      // enable sensor self-update (see §5)
  "update_url": "https://downloads.your-domain.com/sensor"
}
```

Deliver this file through your MDM (a configuration profile or a packaged payload), **not** baked into the public binary — it carries the enrollment secret. Every value can also be overridden by environment variable (`COLLECTOR_URL`, `SENSOR_KEY`, `COLLECTOR_CA`, `COLLECTOR_INSECURE`) for quick tests.

3. Enterprise deployment by platform

The goal is zero-touch: your MDM installs the signed package silently and the service starts itself. No per-machine manual steps.

macOS (Jamf / Intune / Kandji)

Push two items:

1. The signed `.pkg` as a package/app deployment. Its postinstall registers and starts the LaunchDaemon.
2. A **PPPC configuration profile** granting Full Disk Access to `/usr/local/shadow-ai-sensor/shadow-ai-sensor`, plus a payload dropping `agent_config.json` into `/etc/shadow-ai-sensor/`.

```
# What the pkg does on the endpoint (silently):
#   installs binary → /usr/local/shadow-ai-sensor/
#   installs LaunchDaemon → /Library/LaunchDaemons/com.shadowai.sensor.plist
#   creates /var/lib/shadow-ai-sensor and /var/log/shadow-ai-sensor
#   launchctl bootstrap system/com.shadowai.sensor
```

Verify: `sudo launchctl print system/com.shadowai.sensor` and `tail -f /var/log/shadow-ai-sensor/sensor.log`.

Windows (Intune / SCCM)

Deploy the signed `.msi` (or the `.exe` plus `install-service.ps1`) as a Win32 app with a silent install command. It installs to `C:\Program Files\ShadowAISensor\`, registers the `ShadowAISensor` service (Automatic start), sets crash-recovery failure actions, and starts it. Deliver `agent_config.json` to `%ProgramData%\ShadowAISensor\` via the same app or a config profile.

```
msiexec /i ShadowAISensor-1.1.0.msi /qn
```

Verify: `Get-Service ShadowAISensor` and the Windows service log.

Linux (Ansible / apt / yum)

Install the `.deb` / `.rpm`; the postinstall enables and starts the systemd unit. Or, with configuration management:

```
- hosts: endpoints
  become: true
  tasks:
    - copy: { src: shadow-ai-sensor, dest: /opt/shadow-ai-sensor/shadow-ai-sensor, mode: '0755
    - copy: { src: shadow-ai-sensor.service, dest: /etc/systemd/system/shadow-ai-sensor.servic
    - copy: { src: agent_config.json, dest: /etc/shadow-ai-sensor/agent_config.json, mode: '06
    - systemd: { name: shadow-ai-sensor, enabled: true, state: started, daemon_reload: true }
```

Verify: `systemctl status shadow-ai-sensor` and `journalctl -u shadow-ai-sensor -f`.

4. Permissions the sensor needs

The sensor is read-only: it never modifies user files, installs other software, or changes security settings.

PLATFORM	RUNS AS	GRANT	PURPOSE
macOS	root (LaunchDaemon)	Full Disk Access (PPPC profile)	Read other users' shell profiles, browser-extension and skill folders
Windows	LocalSystem	Standard service rights	Enumerate installed apps, PATH tools, per-user configs
Linux	root (systemd, hardened)	Read under `/home`, `/opt`, `/usr`	Scan manifests, configs, installed tooling

On managed macOS fleets the PPPC profile grants Full Disk Access to the specific binary, so there is no interactive prompt. The Linux unit is hardened (`ProtectSystem=strict`, `ProtectHome=read-only`, `NoNewPrivileges=yes`, `PrivateTmp=yes`) — it can read what it needs and write only its own data and log directories.

5. Auto-update & pushing updates

There are two supported update models. Pick per fleet; you can mix.

Model A — MDM-pushed (recommended for enterprise)

Cut a new version, publish it, then have your MDM deploy the new signed package exactly as the first install. The service swaps binaries and restarts. This keeps the update decision with IT, needs no extra outbound access from endpoints, and gives you MDM's native staged-rollout and rollback.

```
# 1. bump sensor/version.py, build + sign all platforms (CI)
# 2. publish the immutable version
bash packaging/upload_s3.sh 1.2.0 <bucket>
# 3. stage to a pilot ring in your MDM, then broaden
# 4. rollback = redeploy the previous version's package
```

Model B — sensor self-update (for fleets without MDM)

Set `auto_update: true` and an `update_url`. The sensor periodically checks `latest/manifest.json`, and only updates when a **newer** version is advertised. Before installing it: downloads over TLS, verifies the artifact's SHA-256 against the manifest, relies on the OS code signature (Developer ID / Authenticode) as the trust anchor, stages the binary, and applies on the next service restart. A checksum mismatch aborts the update.

Promoting and rolling back

Promotion is a single explicit step — advance the `latest` pointer to the new manifest. Rollback is the same step aimed at the previous version. Because each version is immutable, rollback is always available.

```
aws s3 cp s3://<bucket>/sensor/1.2.0/manifest.json \
          s3://<bucket>/sensor/latest/manifest.json --sse aws:kms
```

6. How enrollment & data capture work

1. **Enroll (once):** on first run the sensor calls `/api/enroll` with the enrollment secret and its hostname/OS/IP/username. The collector issues a unique `device_id` + bearer `token`, stored on the endpoint at `0600`.
2. **Collect:** collectors run on schedule (network 15s, static 10 min) and emit findings.
3. **Spool + deliver:** every batch is written to the local durable spool first, then delivered to `/api/ingest` with the device token; a batch is deleted only after the collector acknowledges it.
4. **Heartbeat (60s):** reports liveness plus the agent and signature versions, so the dashboard shows which sensors are current and flags any that go silent (`stale`).

Findings, devices and their status are then visible in the dashboard, where analysts triage them.

7. Verify a healthy fleet

- **Dashboard:** device count, last-seen, agent/signature version per host; findings by category and severity.
- **Heartbeat:** a device missing heartbeats is flagged `stale` automatically.
- **Endpoint logs:** `/var/log/shadow-ai-sensor/sensor.log` (macOS/Linux) or the Windows service log show per-cycle scan counts and delivery status.
- **Version check:** `shadow-ai-sensor --version` on any endpoint; `--selftest` confirms all collectors import and run.

Log rotation (recommended)

```
# /etc/logrotate.d/shadow-ai-sensor   (Linux/macOS)
/var/log/shadow-ai-sensor/sensor.log {
  weekly
  rotate 8
  compress
  missingok
  copytruncate
}
```

8. Uninstall

- **macOS:** `launchctl bootout system/com.shadowai.sensor`; remove `/Library/LaunchDaemons/com.shadowai.sensor.plist`, `/usr/local/shadow-ai-sensor/`, `/var/lib/shadow-ai-sensor/`.
- **Windows:** `sc.exe stop ShadowAISensor && sc.exe delete ShadowAISensor`; remove `C:\Program Files\ShadowAISensor\` and `%ProgramData%\ShadowAISensor\`.
- **Linux:** `systemctl disable --now shadow-ai-sensor`; remove the unit, `/opt/shadow-ai-sensor/`, `/var/lib/shadow-ai-sensor/` (or `apt remove` / `yum remove`).

Support checklist

Before contacting support, capture: `shadow-ai-sensor --version`, the tail of the sensor log, and the output of the platform's service-status command. These three identify the vast majority of issues (config path, connectivity, or permissions).