contact@namiq.io · namiq.io

NamiQ

SECURITY ADVISORY

Agent sandbox, no login

CVE-2026-112346 · OpenHands Software Agent SDK

Assessment & Action Plan

OpenHands' agent sandbox listens on every interface with authentication off by default. What that gives an attacker, why it is not the same as owning the host, and what to check today.

PREPARED BY	Hung Le, NamiQ	DATE	2 November 2026
FACTS AS OF	26 October 2026	VERSION	1.1
SEVERITY	**Critical: CVSS 3.1 9.8 (4.0: 9.3)**	STATUS	Fixed in 1.52.0

NamiQ © 2026 NamiQ. All rights reserved. This document is the property of NamiQ. It may be shared unaltered, with attribution.
Companion article: namiq.io/insights/openhands-sandbox-cve-2026-112346

Update, 2 November 2026: this note now ends with a section on checking your own servers and projects with lattice-scan, for IT admins, AI agent skills and CI/CD, and the analysis report is reissued with it. Everything else is as of 26 October 2026; check CISA's catalog for any change since.

On 26 October 2026 a critical flaw was published for the OpenHands Software Agent SDK: CVE-2026-112346. It is not a flaw in a model. It is an ordinary default: a service that listens on every network interface and has no key set. What makes it worth a close read is what an agent sandbox contains.

This note is for you if you run coding agents through the SDK's Docker or Apptainer workspaces. Everything is as of 26 October 2026 and links to its source at the end.

The short version

- Affected: OpenHands Software Agent SDK before 1.52.0. Fixed in 1.52.0.
- The DockerWorkspace and Apptainer workspace launchers bind the sandbox agent server to 0.0.0.0 with an empty session_api_keys list, which disables authentication by default.
- Anyone who can reach the published port can send HTTP and WebSocket requests that run shell commands as root in the sandbox, reach the mounted workspace and read plaintext secrets from the secrets store.
- CVSS 3.1 9.8 (network, no privileges, no interaction). CVSS 4.0 9.3. CWE-1327, binding to an unrestricted IP address.
- The record does not report exploitation, and we have not read any exploit code.

What the bug is

The SDK starts an agent server inside a container so that an agent can run commands and edit files in a workspace. That server is meant to be controlled only by the process that launched it. Before 1.52.0 it listened on all interfaces, and an empty key list meant it accepted any caller. The published port is therefore a remote shell for whoever can reach it.

The weakness class is not exotic. It is the same shape as a database or a message broker left open with no password, and it is found the same way: by scanning for the port.

Root in the sandbox is not root on the host

Some summaries say the flaw gives code execution on the host that runs the SDK. The record says root inside the sandbox, with access to the mounted workspace and the secrets store. Those are different, and the difference decides your response.

What the attacker can do is bounded by what you put in the sandbox and what its network can reach. For many teams that is a source repository, cloud and model-provider tokens, and a route to internal services, which is already serious. Reaching the host is a further step, and it depends on how the container was started. A privileged container, or one with the Docker socket mounted into it, is a

separate and much shorter path to the host. That is our reading of how containers work, not a statement in the advisory.

Our SOC analyst's assessment

- Treat it as same-day if the port is reachable from anywhere you do not control. The fix is a version bump, so the cost of acting now is small, and the cost of waiting rises if exploit code appears.
- Exposure depends on where the port is published, not on the version alone. A developer laptop on a home network, a shared development VM and a CI runner are three different answers.
- The first move when you cannot patch is to take the port off the network: bind it to 127.0.0.1 or a private address and close any load balancer, ingress or firewall rule that publishes it.
- Assume what the sandbox could see is exposed if the port was reachable. Rotate those secrets as a precaution. This is a judgement, not something the advisory asks for.
- Expect exploit code to follow within days to weeks. That is an estimate from how similar flaws have played out, not a fact about this one.

> *The agent's execution environment is part of the attack surface, even when the model itself is secure.*

Step 1: find where it runs and what it exposes

```
# Agent containers and how their ports are published
docker ps --format '{{.Names}}\t{{.Image}}\t{{.Ports}}'
# Look for 0.0.0.0: mappings on agent containers

# Installed SDK version, from the interpreter that runs it
pip list 2>/dev/null | grep -i openhands

# Is anything listening beyond localhost?
ss -tlnp
```

Check the running environment, not the lockfile. Include CI runners, shared development hosts and any cloud instance that launches agent workspaces. Apptainer is common on research clusters, where a compromised node can reach shared storage, so count that blast radius too.

Step 2: upgrade and verify

- Upgrade to 1.52.0 or later through your normal pipeline, testing workspace launch and tool execution first.
- Set session API keys explicitly rather than relying on a default, and keep them out of the workspace.

- Publish the port on 127.0.0.1 or a private network only.
- Verify the running version after deployment, and confirm from another host that the port no longer answers.

Step 3: look for prior use

- Workspace or container launches you did not initiate, especially from addresses outside your own network.
- Shell processes inside agent containers that no task explains, and new files, cron entries or container images created outside normal working windows.
- Outbound connections from agent containers to hosts they do not normally use.
- If anything suspicious turns up, treat it as an incident and start from the secrets the sandbox could read.

Hardening that outlives the patch

- Run agent sandboxes unprivileged, without the Docker socket, on an isolated network with only the egress they need.
- Keep secrets out of the mounted workspace where you can, and scope the ones that must be there.
- Add authentication and network policy in front of the sandbox service even after the fix.
- Log workspace launches and shell execution inside agent containers so that an unexpected one is visible.

Check your own servers and projects

The facts above are as of 26 October 2026. To compare your own estate with this CVE, the open-source lattice-scan command line checks the parts it can see. Naming the CVEs needs no key, only the CVE ids leave your machine, and a match is a candidate for a person to confirm, not a finding.

On a server: run it on the server itself, not on the laptop or CI runner that deploys to it. It reads installed packages (dpkg, rpm or apk), snaps, the names of running container images and the kernel version, compares them on the machine, and keeps the inventory there. A distribution can fix a CVE under an unchanged version number, so on Debian-family hosts read the changelog line the report shows.

```
lattice-scan --cve CVE-2026-112346
```

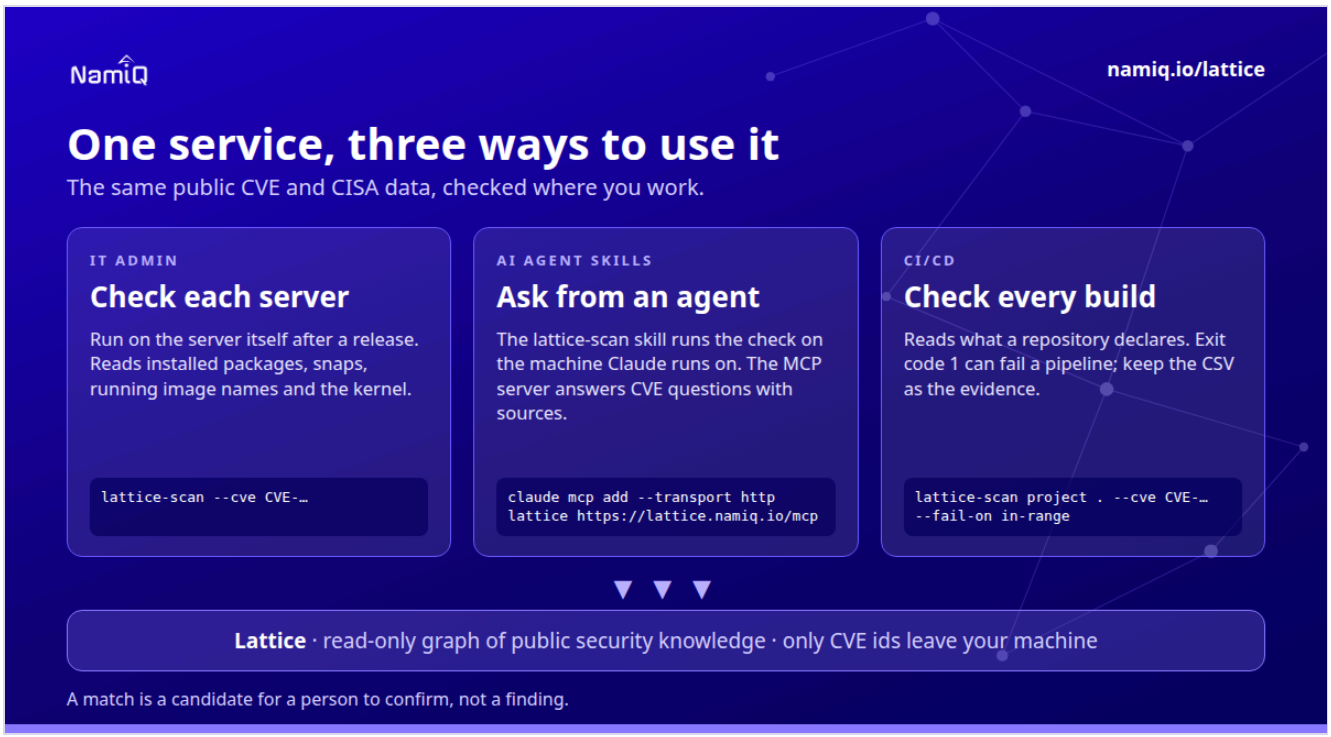
On a repository: the project check reads what it declares, in Python, JavaScript, Java, Rust, C and C++, Make, and Docker and compose files. It installs and runs nothing. A declared range or a tag without a patch number cannot show whether a fix is in, so such a row is undetermined and needs a person.

```
lattice-scan project . --cve CVE-2026-112346
```

The host check reads the names of the running images, not what is inside the sandbox image; use the project check on the image's Dockerfile and requirements files.

Who can use it, and how

For	What to do	Command
IT admin	Run the host check on each server after a release, and again when a new entry lands.	lattice-scan --cve CVE-2026-112346
AI agent skills	The lattice-scan skill runs the check on the machine Claude runs on. The Lattice MCP server answers questions about the CVE with its sources.	claude mcp add --transport http lattice https://lattice.namiq.io/mcp
CI/CD	Check the repository on every pull request. Exit code 1 fails the pipeline; keep the CSV as the evidence.	lattice-scan project . --cve CVE-2026-112346 --fail-on in-range



Where the same check is used. The three ways read the same public data; only CVE ids leave your machine.

Guide: namiq.io/lattice/guides/choose-your-channel. MCP server (io.namiq/lattice in the MCP registry): namiq.io/lattice. Package: pypi.org/project/lattice-scan, or namiq.io/apt.

What we do not know yet

- Whether anyone has accessed an exposed instance. The record does not say.
- Whether public exploit code appears.

- How many instances are reachable from the internet.

If you take one thing from this: an agent's sandbox is a service with a port, a user and a set of secrets. Treat it like the other services you run, with a login, a network boundary and a log.

Sources

- CVE-2026-112346 record (VulnCheck CNA): https://www.vulncheck.com/advisories/openhands-software-agent-sdk-unauthenticated-sandbox-agent-server-rce
- OpenHands software-agent-sdk release v1.52.0: https://github.com/OpenHands/software-agent-sdk/releases/tag/v1.52.0
- OpenHands software-agent-sdk pull request 5209: https://github.com/OpenHands/software-agent-sdk/pull/5209
- OpenHands software-agent-sdk fix commit: https://github.com/OpenHands/software-agent-sdk/commit/7b66584ff8899144eb4ce679c2811861342af9de
- CISA Known Exploited Vulnerabilities catalog: https://www.cisa.gov/known-exploited-vulnerabilities-catalog