SECURITY OVERVIEW · DISTRIBUTIVE

DCP Worker for Windows (screensaver)

Installer **dcp-worker-windows-x64.msi** 8.0.1, pre-release build 3100729098 Status as of **November 22, 2026**
Tested on **Windows 11**

> The DCP Worker runs compute jobs from the Distributive network while a Windows PC is idle, as a screensaver. Job code is treated as untrusted. Each job slice runs in its own evaluator process that cannot connect to the internet or the local network, cannot start programs and has no privileges, on JavaScript and WebGPU engines at the patch level of current Chrome stable. The worker service runs under its own virtual service account. Every executable in the installer is signed by Distributive. All 94 JavaScript-engine vulnerabilities disclosed in Chrome security releases from March to November 2026 are fixed, and GPU (WebGPU) jobs run, including at the sign-in screen with nobody signed in.

What is in the installer

COMPONENT	ROLE	VERSION
Worker service	Talks to the Distributive scheduler, fetches work, returns results. Runs only Distributive code, as the Windows service `DCP Worker`.	`Node.js 26.13.1 LTS`
Evaluator	Runs job code (JavaScript and WebAssembly) in a sandboxed engine. Started by the screensaver and stopped when it is dismissed.	`V8 15.8.62.61`
WebGPU engine	Gives GPU jobs access to the graphics card through Direct3D 12, inside the evaluator. Ships with Microsoft's shader compiler.	`Dawn, Chrome 158 branch`
Screensaver	`dcp-screensaver.scr`: shows an image or a local status page while computing; on the sign-in screen and for signed-in users.	`8.0.1`
Updater	Distributive's signed updater installs new versions when Distributive publishes them (optional).	`Optional`

Programs install under `C:\Program Files\Distributive\DCP`. The installer resets the permissions of the data folders under `C:\ProgramData` so that only SYSTEM and Administrators can change them; ordinary users can read but not write. The worker's identity key lives in the service account's own profile.

How job code is contained

The main boundary is the engine: job code has no file, process or network functions at all. The other layers are defence in depth. They matter only if an attacker first found a flaw that let code escape the engine, and they limit what such code could then do.

- **Engine sandbox.** Each job slice runs in its own evaluator process with Node.js system APIs removed. Job code cannot open files, start processes or open sockets.
- **No network.** A Windows Firewall rule blocks every outbound connection from the evaluator, to the internet and the local network. It can reach only the local worker over loopback.
- **No new programs.** Each slice's process is started with a Windows policy that forbids it from creating any child process.
- **Exploit mitigations.** Windows process mitigation policies are enforced on each slice's process: full address-space randomisation, strict handle checks, no injected extension DLLs, no non-system fonts, and no loading of code from network shares or low-integrity files.
- **No privileges, lower integrity.** Each slice's process has every privilege removed from its token and runs at Medium integrity. On the sign-in screen, where Windows runs the screensaver as LOCAL SERVICE at System integrity, that is two levels lower, so it cannot write to or tamper with system-level processes and objects.
- **Memory-aware.** The worker measures each job's memory and starts its slices only while they fit a budget, by default the machine's memory, up to one per CPU core.
- **Short-lived.** All evaluator processes end the moment the screensaver is dismissed; a Windows job object makes sure nothing is left behind.
- **Separate accounts.** The worker service runs as the virtual account `NT SERVICE\DCP Worker`, not as SYSTEM or the signed-in user; job code never runs in it.
- **Signed code only.** All 33 executables and native modules are signed by Distributive, so the installer runs under Smart App Control and similar allow-listing.

Limits. Integrity levels stop writing, not reading: code that escaped the engine could read files the screensaver's account can read. On the sign-in screen that account is LOCAL SERVICE, which has little to read; for a signed-in user it is that user. Low

integrity, which is stricter, is not possible because Windows' screensaver desktop does not admit Low-integrity processes.

Network footprint

- Outbound HTTPS only, from the worker service to the Distributive scheduler, verified against the Windows certificate store (corporate inspection proxies with an installed root CA work).
- No inbound ports. Two local ports listen on loopback only: `9000` (worker to evaluator) and `9080` (local status page). The installer adds Windows Firewall rules for the evaluator: inbound limited to loopback, and every outbound connection blocked.
- The organisation's compute groups decide which jobs a machine accepts; the worker can be limited to the organisation's own groups.

Patch status

AREA	STATUS	DETAIL
JavaScript engine (V8)	Current	15.8.62.61 contains the exact V8 shipped in Chrome 158.0.8127.36, the stable release of November 17, 2026, plus 30 later security and stability merges from V8's 15.8 branch (to November 20).
Disclosed V8 CVEs	0 open	All 94 V8, Turbofan, Maglev and WebAssembly CVEs in Chrome security releases from March 24 to November 18, 2026 are fixed. 89 confirmed by locating the fix in the engine source; the other 5 have no public fix commit and are covered by containing Chrome 158's engine.
WebGPU engine (Dawn)	Current	Built from Chrome 158's Dawn branch (to November 20, 2026). Dawn fixes from Chrome 157 and 158 security releases are included; see "Open V8 Vulnerabilities".
Node.js runtime	Current	Worker service on 26.13.1, the November 10, 2026 security release of the Node 26 LTS line.
Native modules	Reduced	Only Windows x64 builds of native modules are shipped; builds for other platforms are removed, and every remaining one is signed.

No Node.js release yet carries V8 15.8, so the evaluator embeds it through the Node.js development line (28.0.0-pre); the worker service stays on the Node.js LTS release.

Testing

- Install and compute cycle of this build on a Windows 11 PC with Smart App Control enforcing: 25 of 25 checks passed, including signatures, loopback-only ports, registration key absent from installer logs, and real jobs computed and paid at the sign-in screen.
- GPU jobs on the same PC's Intel Iris Plus graphics, at the sign-in screen with nobody signed in: correct results.
- A running slice's process checked: Medium integrity, no privileges, all mitigations and the child-process ban enforced; outbound firewall rule enabled.
- V8's published regression tests for this year's engine CVEs that can run outside V8's own test shell: 18 of 18 pass on this build. 11 of them fail on an engine without the fixes, which confirms the tests detect the flaws.
- A job that reads its input from one approved website and stores its results on another ran correctly, with both sites set as allowed origins in the worker's configuration.
- Memory: slices of 1.2 GB each ran 5 at once on an 8 GB PC, and at most 2 with a 3 GB budget in the configuration file.

Verifying the installer

The installer and every program in it carry Distributive Corp.'s Authenticode signature (EV code signing). Check it before installing:

```
Get-AuthenticodeSignature .\dcp-worker-windows-x64.msi
```

The status must be `Valid` and the signer Distributive Corp.; Smart App Control or an allow-listing policy blocks a changed file.

Keeping it current

The JavaScript and WebGPU engines follow Chrome's stable branches, and updated installers follow Chrome and Node.js security releases. With automatic updates on, Distributive's signed updater installs them; administrators can instead deploy each MSI themselves.

Distributive Corp. Installer SHA-256 `9699e7e58a66ba735de470968cad64dbdd84b7d311b2b70b3a0a8f8a78b634f6`. Per-vulnerability list: "Open V8 Vulnerabilities".