



FREE GUIDE · PRIVACY ACTION NETWORK

Using an AI assistant on your own computer, *safely*.

The safe-use protocol: one rule, four levels of risk, a secret sweep, and what to do if something leaks.

- Why an AI on your computer works as you
- What to do at each level, from chat to agents that act alone
- The secret sweep: where passwords hide outside your password manager
- If something leaks: the order that closes the hole

When an AI assistant works on your computer, it works as you.

Whatever you can open without typing a password, it can open too. So the whole protocol comes down to two things:

- 1 **Keep secrets behind a lock:** a password manager, an encrypted disk, or a machine that isn't this one.
- 2 **Keep the AI's reach small:** the folder it works in, the tools it has, and how much it does without asking.

How secrets end up with an AI

Neither route needs the assistant to "want" anything.

- **It gets tricked.** A web page, email, PDF or code repository can carry instructions aimed at the AI (prompt injection). If the assistant can act, those instructions can become actions.
- **It's helpful in the wrong place.** It opens a file with a password in it while fixing something else. Now that password is in the conversation, which means it's been sent to the provider and saved in a transcript on your disk.

Pick your level

Levels go by what the AI can do, because that's what the risk grows with. Each level includes everything in the levels above it.

LEVEL	THE AI CAN...	EXAMPLES
1 · Chat	only see what you type or attach	claude.ai, ChatGPT, Gemini in a browser tab
2 · Hands	read files, run commands, drive a browser, while you approve or watch	Claude Code or Claude Desktop in Manual or Accept-edits mode, Claude in Chrome
3 · Hands, nobody watching	act without asking you each time	auto mode, background agents, overnight or scheduled runs
4 · Crown jewels	should get nowhere near it	client records, legal and money documents, anyone at real risk

LEVEL 1

Chat.

Do

- **Keep every password, key, token and recovery code out of the chat.** If one slips in, change it now. Deleting the chat doesn't un-send it.
- **Check the data setting.** On Claude's consumer plans, conversations are kept for 5 years if you let them be used to improve Claude, and 30 days if you don't (Anthropic's data-usage page, September 2026). You'll find it under `claude.ai` → Settings → Privacy. Other providers have similar settings: look for training or data controls.
- **Treat the AI account like your email account:** a unique password from your password manager, plus a passkey or a hardware key.
- **Cover the password-manager basics** (Protocol 6 in the PAN framework): a unique password everywhere, auto-lock, and a clipboard that clears itself.

Don't

- Paste other people's personal details (clients, family, customers) unless you're happy for the provider to hold them.
- Share screenshots that show a password, a 2FA QR code or a recovery sheet.

LEVEL 2

Hands, while you're watching.

The assistant can now read and write files and run commands, all as you.

Do

- **Start it in a project folder**, never your home folder or a whole drive. The folder you start in is where it works most freely.
- **Keep secrets out of that folder, and out of plain files anywhere.** See the secret sweep below.
- **Say no in its settings, not only in conversation.** In Claude Code, deny rules block in every mode. A starting point for `~/.claude/settings.json` (check the pattern syntax for your version):

```
{
  "permissions": {
    "deny": [
      "Read(~/.ssh/**)",
      "Read(~/**/.env)",
      "Read(~/**/*.*kdbx)",
      "Read(~/**/*.*keyx)",
      "Bash(ssh *)",
      "Bash(scp *)"
    ],
    "blockReadsOutsideWorkingDirectories": true
  }
}
```

Deny rules match commands as written, so treat them as a fence, not a wall.

- **Give the AI its own browser profile.** Some assistants drive your real browser with your logins (Claude in Chrome does). If your password manager's extension is unlocked there, it can fill a password into a page the AI is working on. Make a separate profile for the AI, with no password-manager extension and only the logins the job needs.
- **Set up your password manager for this.** The setting names here are KeePassXC's; other managers have equivalents.
 - *Security:* Lock databases after inactivity (a few minutes); Lock databases when session is locked or lid is closed; Lock databases after minimizing the window; Clear clipboard after 10–30 seconds; Hide passwords in the entry preview panel.
 - *Browser integration:* keep "Never ask before accessing credentials" and "Allow limited access to all entries in connected databases" switched off. When the Browser Access Request appears, don't tick Remember for high-value entries: email, banking, servers, the AI account, and the password manager's own account.
 - *SSH agent,* on each key: Remove key from agent when database is closed/locked, and Require user confirmation when this key is used. Consider Remove key from agent after a set time. A program running as you can't copy a key held in an agent, but it can use it.
- **Don't run the AI app as an administrator.** KeePassXC blocks ordinary programs from reading its memory on Windows, but a program with administrator rights can get round protections like that.
- **Only connect what the job needs.** Every connected service (email, drive, calendar, design tools) is another door the assistant can open.
- **Mind the transcripts.** Claude Code keeps a plain-text record of every session on your disk (`~/ .claude/projects/`) for 30 days by default (the `cleanupPeriodDays` setting). Sessions from the desktop app are exempt from that clean-up by default, so they stay until you delete them. If a secret ever lands in a session, change the secret first, then delete that transcript and any backup of the folder.

- **Take most care with “read this, then do that”.** Tasks that read outside content (a web page, an email, a downloaded repository) and then act on it are where prompt injection bites.
- **Look after the machine under the AI:** full-disk encryption on, updates current, a short screen lock.

Don't

- Paste a secret so the AI can “set it up for you”. Put the secret in place yourself, then tell the AI where it lives.
- Let it read your password database or its key file (the deny rules above cover this). A key file stored next to the database adds very little protection.
- Keep passwords in the built-in password store of a browser the AI drives.

LEVEL 3

Hands, with nobody watching.

Know what “auto” means. In Claude Code, auto mode is the default starting mode on Pro, Max and Team plans (as of September 2026). A second model, the classifier, reviews each action instead of you. By default it blocks a lot: printing a live credential into the chat or a file, sending data where it shouldn't go, and browser actions that could carry cookies or credentials off the site. But it's a model, not a wall. **File reads run without asking;** only the first read outside the working folder prompts you.

Do

- **Lock your password manager** before you walk away from an auto-mode session.
- **Turn on `blockReadsOutsideWorkingDirectories`**. File-reading tools then refuse anything outside the project, in every mode.
- **Run unattended work on a separate machine or VM that holds nothing worth stealing:** no password manager, no browser logins, no keys to your servers, no personal files.
- **Use the sandbox where it exists.** Claude Code's Bash sandbox runs on macOS, Linux and WSL2, but not on native Windows. On Windows, the permission rules and the classifier are your only fences, which is why unattended work belongs on another machine. On Linux, set the sandbox to fail closed (`sandbox.failIfUnavailable: true`) and turn off unsandboxed retries (`allowUnsandboxedCommands: false`). By default, if the sandbox can't start, it warns you and carries on without it.
- **Give it work a test can check,** on a branch, in git. The agent commits; a person reviews and merges. It never has the right to deploy, send, post or pay anything.
- **Put limits on it:** a time limit, a turn limit, and one job at a time.

Don't

- Use “bypass permissions” (`--dangerously-skip-permissions`) outside a container or VM.

- Give an unattended agent any remote it could push to that deploys.
- Let it install packages. A new dependency is a question for a human.

LEVEL 4

Crown jewels.

- Keep crown-jewel material **off any machine where an AI agent runs**. A separate machine is the strongest line; a separate operating-system user or a VM is next best.
- Use a **local model** for sensitive text, and know what that does and doesn't fix. The model stays on your machine, but your files still sit on a computer that's connected to the internet.
- Put a **hardware key** on both the password manager and the AI account.
- Client data never enters an AI session without the client's agreement. Redact before you paste.

ONCE, THEN MONTHLY

The secret sweep.

Secrets inside your password manager are safe. The danger is copies outside it. For each one you find, move it into the password manager (as a note or an attachment) or off the machine, **then delete the plain copy**. Archiving a plain-text secret just keeps it.

WHERE	WHAT TO LOOK FOR
<code>~/.ssh/</code>	Private keys without a passphrase. Test: <code>ssh-keygen -y -f <key></code> should ask you for one
Project folders	<code>.env</code> files, config files holding keys, <code>*.pem</code>
Dot-files in your home folder	<code>.git-credentials</code> (plain-text git passwords), tokens in <code>.npmrc</code> , <code>.docker/config.json</code> , cloud command-line credentials
Downloads, Desktop, Documents	Password exports (CSV), recovery-code files, screenshots of 2FA QR codes
Website backups	<code>wp-config.php</code> (database passwords), database dumps holding customer data
Shell history	On Windows, PowerShell saves every command in <code>%APPDATA%\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt</code> , so a token typed on the command line sits there in plain text
Notes apps	Passwords written down "just for now"
AI transcripts	Anything the assistant ever read, or you pasted

If an AI helps with the sweep, make it report where, never what. It should give you file paths and counts only. A sweep that prints the secrets it finds has just copied them into a transcript.

If something leaks

- 1 **Change or revoke it first:** the password, token or key. That's the only step that actually closes the hole.
- 2 Then delete it wherever it landed: the chat, the transcript, and backups of the transcript folder.
- 3 Check the account's recent activity for anything you didn't do.

TWO QUESTIONS PEOPLE ASK

Is copy-and-paste from the password manager safe?

Mostly, with a well-built manager. KeePassXC marks what it copies so Windows leaves it out of clipboard history and cloud clipboard. It also clears the clipboard only if your password is still what's on it, so it never wipes something else you've copied since (checked in its source code, September 2026). The gap that remains is the clear time itself: until the clipboard clears, any program can read it, and a clipboard-manager app that ignores the marker can keep a copy. Safer still is to let the browser extension fill websites, which skips the clipboard and checks the site's address (and that also beats phishing). Use Auto-Type for desktop apps.

Can an AI get into my password manager while it's unlocked?

Not directly, if the manager is well built. KeePassXC's database file is encrypted, it stops ordinary programs reading its memory on Windows, and its browser extension has to ask you before it hands over an entry. What an AI, or anything else running as you, *can* reach is whatever has left the manager:

- the clipboard, until it clears
- sites you've ticked Remember for
- SSH keys loaded into an agent
- every secret you keep outside the manager

So this protocol is about the doors around the password manager, not the manager itself.

Want a hand?

I help people and small teams use AI safely: the right settings, private AI that runs on your own computer, and clear rules for what never goes in. Remote wherever you are, or on-site in Gippsland.

privacyaction.network/services/private-ai

Sources, checked 11 September 2026. Claude Code documentation: permission modes (auto mode, what the classifier blocks, the first read outside the working folder, `blockReadsOutsideWorkingDirectories`), sandboxing (platforms, `failIfUnavailable`, `allowUnsandboxedCommands`), data usage (retention, local transcripts, `cleanupPeriodDays`) and authentication. KeePassXC source code (develop branch): `src/gui/Clipboard.cpp`, `src/core/Bootstrap.cpp`, and the settings, browser-access and SSH-agent screens. Products change; check the current documentation for your version. General information, not advice for your specific situation.