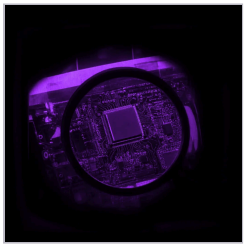
TECHDOM & CODE · CHAPTER 4

~$ open check-a-tool-before-you-install-it

Check a tool before you install it

A techdom tool runs on your side of the screen, next to your sign-ins and your saved card. Ten minutes of checking before you install it shows what it can read, what it sends and whether you can stop it.

ENTRY T4 CATEGORY TECHDOM & CODE CHECKED 3 NOVEMBER 2026
READING 7 MIN SECTIONS 6



ls cover/ T4.webp

EVIDENCE ◆ Verified ◆ Reported ◆ Vivian's field notes ◆ Unverified

IN 30 SECONDS

01 In my experience most risky tools aren't malware. They do what they say, with a hidden price or no way out.

02 Read the permissions, the addresses it talks to, the updater, the defaults and the stop.

03 Never test an unknown tool on the computer where you're signed in.

04 A check lowers the risk. It never makes an unknown tool safe.

05 Remote access is a red line, and you can remove a tool you doubt.

YOU WILL LEARN

- ✓ What a tool can read, send and change once it's installed.
- ✓ A ten-minute check, in order, with three small scripts that run for real.
- ✓ How to word a verdict about a tool without overclaiming.
- ✓ Which lines to refuse, and what to do if a tool is already installed.
- ✓ The same checks from the maker's side.

> *Don't trust me, a label or a friendly maker. Spend ten minutes first, and walk away from any tool that won't let you check. It makes you a lot harder to surprise.*
>
> — VIVIAN

~/01-why-check-at-all

Why check at all

A techdom tool runs on your side of the screen, often inside your signed-in browser, next to a saved card. ◆ See How techdom tools work for

the families.

In my experience the usual danger is a tool that does exactly what it says, with a hidden price or no way out. Malware is the rarer case. ◆ So the check covers design (stop, limit, running total, defaults) as well as code (permissions, addresses, updater).

Press describes a scene flooded with scammers who use stolen photos of attractive women. ◆ A free tool offered by a stranger deserves the same doubt as a beautiful profile. ◆

A check lowers the risk, and that's all it does. Your best protection is the limit you set before you start. See Breaks that hold. If you need a checklist to feel safe, not installing is a fine answer. ◆

~/02-a-ten-minute-check

A ten-minute check

Do the steps in order. If you can't do one, that's your answer.

```
# Ten-minute check
```

ten-minute-check.md · 8 lines copy

1. Find the maker. A maker with a history you can check counts, and a persona or a handle counts too. A tool from someone with no history at all has already failed.
2. Open the permission list and the site list. Write down every line you can't explain.
3. List the addresses it talks to and find out who runs each one.
4. Find the updater. Can it fetch and run new files, and who decides when?
5. Check the defaults: does it start switched off? Does it show a running total? Is there a limit you set?
6. Test the stop in every state: idle, running, locked, just after a payment. A stop that fails in one state is a defect.
7. Compare the file you got with the fingerprint the maker published.
8. Try it in a virtual machine or a fresh browser profile with none of your sign-ins.

> // Vivian's field notes
>
> *I never run an unknown program on the computer where I'm signed in. A throwaway setup costs time, and that's cheap next to a drained card.*
>
> ◆

Reading the permissions

When you add an extension, Chrome shows permission warnings that describe what the permissions allow. ◆ Host permissions are what let an

extension work with the sites that match its listed patterns. ◆ Some permissions are optional: granted while the extension runs, not at install. ◆

After install you can open Extensions, then Manage extensions, then a tool's Details page to see which sites it may access. ◆ That page has a setting for whether the tool may read and change your data on every website you visit. ◆

If an update adds a permission that triggers a warning, Chrome disables the extension until you accept it. ◆ An update that stays within the permissions it already has asks you nothing. ◆ The Chrome Web Store requires extensions to disclose how they handle user data, even when it never leaves the device. ◆ A tool you load from a folder skips that listing, so there's no disclosure to read. ◆

Here are common permissions in my own words. Chrome's own pages are the reference. ◆

What common permissions let a tool do

tabs

WHAT IT LETS THE TOOL DO

See the address of every open tab

WHAT TO ASK

Why does a tool for one site need all of them?

cookies

WHAT IT LETS THE TOOL DO

Read and write cookies

WHAT TO ASK

Does it need your sign-in on any site?

webRequest

WHAT IT LETS THE TOOL DO

See every request your browser makes

WHAT TO ASK

What does it do with them?

scripting

WHAT IT LETS THE TOOL DO

Insert code into pages

WHAT TO ASK

Into which pages, and who wrote that code?

Site access to all websites

WHAT IT LETS THE TOOL DO

Work on every page you open

WHAT TO ASK

Can it be limited to the one site it serves?

An update address

WHAT IT LETS THE TOOL DO

Fetch new versions from a server

WHAT TO ASK

Who runs it, and can it send any file?

Code you can run

Three short scripts show the idea. Each reads made-up data, opens no network connection and changes nothing on your computer.

Three small checks

manifest_check.py

```
# manifest_check.py - read an extension manifest the way a
stranger would, before installing.
# It reads two invented manifests that sit inside this file. It
opens no file and no network.

SAMPLES = {
    "tight": {
        "manifest_version": 3,
        "name": "Example fan tool",
        "permissions": ["storage", "alarms"],
        "host_permissions": ["https://shop.example.test/*"],
        "content_scripts": [{"matches":
["https://shop.example.test/*"], "js": ["main.js"]}],
    },
    "wide": {
        "manifest_version": 3,
        "name": "Example wide tool",
        "permissions": ["storage", "tabs", "cookies",
"webRequest", "scripting"],
        "host_permissions": ["<all_urls>"],
        "content_scripts": [{"matches": ["*://*/*"], "js":
["main.js"]}],
        "update_url": "https://updates.example.test/feed.xml",
    },
}

RISKY = {
    "tabs": "sees the address of every tab",
    "cookies": "reads and writes cookies",
    "webRequest": "sees every request",
    "scripting": "injects code into pages",
    "history": "reads browsing history",
    "nativeMessaging": "talks to a program on the computer",
    "clipboardRead": "reads the clipboard",
}
BROAD = {"<all_urls>", "*://*/*", "http://*/*", "https://*/*"}
```

```

def review(manifest):
    findings = []
    for permission in manifest.get("permissions", []):
        if permission in RISKY:
            findings.append("permission " + permission + ": " +
RISKY[permission])
    patterns = list(manifest.get("host_permissions", []))
    for script in manifest.get("content_scripts", []):
        patterns += script.get("matches", [])
    for pattern in sorted(set(patterns)):
        if pattern in BROAD:
            findings.append("site access " + pattern + ": every
website")
    if "update_url" in manifest:
        findings.append("it updates itself from " +
manifest["update_url"])
    return findings


for name, manifest in SAMPLES.items():
    findings = review(manifest)
    if findings:
        print(name, "->", len(findings), "finding(s) to explain
before installing")
    else:
        print(name, "->", "no broad access found (the code is
still unread)")
    for line in findings:
        print("  -", line)
```

```
output $ python manifest_check.py
tight -> no broad access found (the code is still unread)
wide -> 7 finding(s) to explain before installing
  - permission tabs: sees the address of every tab
  - permission cookies: reads and writes cookies
  - permission webRequest: sees every request
  - permission scripting: injects code into pages
```

```
  - site access *://*/*: every website
  - site access <all_urls>: every website
  - it updates itself from https://updates.example.test/feed.xml
```

host_report.py

```
# host_report.py - group a list of request addresses by host and label each host,
# to answer "who does this page or tool talk to?" before trusting it.
# The addresses are invented and sit in this file. Nothing is fetched.
from collections import Counter

KNOWN = {
    "app.example.test": "the tool's own server",
    "cdn.example.test": "the tool's own server",
    "metrics.tracker.test": "analytics",
}

requests = [
    "https://app.example.test/page",
    "https://app.example.test/api/items",
    "https://cdn.example.test/img/1.png",
    "https://cdn.example.test/img/2.png",
    "https://metrics.tracker.test/collect",
    "https://mystery.host.test/x.js",
]


def host_of(address):
    return address.split("//", 1)[1].split("/", 1)[0]


counts = Counter(host_of(address) for address in requests)
for host, count in counts.most_common():
    print(count, "|", host, "|", KNOWN.get(host, "UNKNOWN: find out who runs it before trusting it"))
```

```
output $ python host_report.py
2 | app.example.test | the tool's own server
2 | cdn.example.test | the tool's own server
1 | metrics.tracker.test | analytics
1 | mystery.host.test | UNKNOWN: find out who runs it before
trusting it
```

pinned_hash.py

```
# pinned_hash.py - compare a file against the fingerprint its
author published.
# The "file" is a few bytes inside this script, so nothing is
downloaded or opened.
import hashlib

published = hashlib.sha256(b"the file the author
released").hexdigest()


def matches(data):
    return hashlib.sha256(data).hexdigest() == published


for label, data in [
    ("the file you got, same bytes", b"the file the author
released"),
    ("the file you got, one byte added", b"the file the author
released!"),
]:
    print(label, "->", "same as published" if matches(data)
else "different: do not run it")
```

```
output $ python pinned_hash.py
the file you got, same bytes -> same as published
the file you got, one byte added -> different: do not run it
```

The fingerprint in the last script is computed inside it. In real life you copy it from the maker's page and compare.

They show a method. They don't give a verdict. A manifest check can't see what the code does, and a matching fingerprint only proves you hold the file the maker published. ◆

~/03-what-a-check-can-t-tell-you

What a check can't tell you

A verdict has two parts: what the check found, and what it didn't cover. ◆

Words for a verdict

$ Malicious today

The tool does harm now. Only say this after reading the whole source, and with the evidence. ◆

$ Not malicious today, able to become so with one update

Nothing harmful runs now, but an update could change that, for example when the updater can fetch any file and your sign-ins sit on the same computer. Serious, but not malware. ◆

$ Harmless in code, unsafe in use

Nothing hidden is harmful, but the defaults expose you: a machine identifier sent from your names, a safety-word or quick-exit off by default. ◆

$ Not checked

You didn't read it, or couldn't. Treat it as unsafe for any computer that holds your sign-ins. ◆

For a made-up example of a first alarm and a fuller reading, see Fake tools.

Scrambled code doesn't prove bad intent, because makers scramble code to stop people copying it. It does mean you can't verify the tool, and you shouldn't take a tool that holds a signed-in session on trust. ◆

- `No malware was found, so the tool is safe.`

+ A tool can do exactly what it says and still hide its price or give you no way out. Check the design as well as the code. ◆

- `The maker says it’s safe, so it’s safe.`

+ A friendly label proves nothing. Trust is the one thing a spending tool shouldn’t need. ◆

- `An assistant read the code and found nothing, so it passed an audit.`

+ An assistant can miss logic hidden in scrambled code. A summary isn’t an audit. ◆

`~/04-red-flags-and-the-red-line`

Red flags and the red line

These lines come from the person offering the tool or the session. The answer is yours.

What you may hear, and the answer

WHAT THEY SAY OR DO

01 It’s safe, I made it myself.

THE COUNTER

That’s a claim until someone checks it. The same list applies to every tool, including one made by Vivian.

WHAT THEY SAY OR DO

02 Install this zip, the store would never allow it.

THE COUNTER

It skips the store’s review. Read its stated limits and expect to remove it.

WHAT THEY SAY OR DO

03 Send me the code on your phone, or your password, so I can set it up.

THE COUNTER

Never. A real dom doesn’t need your password, your login code or your connection code.

WHAT THEY SAY OR DO

04 There’s no stop, it ends when it ends.

THE COUNTER

Walk away. A stop that depends on your willpower is no stop. Closing the program always works, but don’t build your safety on it.

WHAT THEY SAY OR DO

05 Test it with real money first, one test can’t hurt.

THE COUNTER

A tiny amount you chose, a test account or a recording. Never open-ended.

WHAT THEY SAY OR DO

06 Open a session and I'll do it for you, or let me read your chats myself.

THE COUNTER

Say no, and offer nothing from your accounts or your chats. If a session is already open, close it and change your passwords from another device.

Remote access

A remote-session hands your computer to someone else: files, saved passwords and screens. ◆ A US government alert advises people not to give control of their computers or mobile devices to any stranger. ◆ A remote-access vendor tells users not to give device access to people they do not know or trust. ◆ More on the risks in Fake tools.

TRY IT

A dom asks for a session

A dom asks you to open a remote session so she can read your messages. She says it'll take a minute.

A Decline, offer nothing from your accounts, and state your own limit once

The right answer. A session shows everything, and you can't take it back. A capture of your messages hands over private content too.

B Accept, but only for a minute

Not this one. One minute is long enough to open files and saved passwords.

C Accept, because she promised not to look around

Not this one. A promise made in a scene can't be enforced. None of them can.

~/05-if-you-already-installed-it

If you already installed it

A tool that stays installed is an ongoing risk: its maker can push a harmful update later. ◆ Removing it is the simplest way out.

On Windows, uninstall from Settings, then Apps, then Installed apps. ◆ Some apps can't be removed there. ◆ To stop one starting with the computer, switch it off in Settings, Apps, Startup, or in Task Manager's Startup tab. ◆ You remove a browser extension from the Manage extensions page. ◆

```
# If a tool you doubt is installed
```

if-a-tool-you-doubt-is.md · 6 lines copy

1. Stop it. Close the program, or turn the computer off. That always works.
2. Remove it, and remove its browser extension too.
3. Sign out everywhere it could reach, then change those passwords from a different computer.
4. Check your order history and bank statement for sends you didn't choose.
5. Set a limit or an alert at your bank.
6. If money left that you didn't mean to send, go to If you already sent. If stopping feels hard, go to Where to get help.

// Vivian's field notes

I treat a signed-in session like a password. Whoever holds it acts as you, and a two-step login doesn't help once the session is taken. After a tool I doubt, I sign out everywhere as well as changing the password.

◆

~/06-for-makers-the-same-list-from-the

For makers: the same list from the other side

If you build tools, your users should be able to do every check above. ◆

- Strip source files, comments and metadata before handing a tool out. Leftover handles point to your accounts. ◆

- Check the licence of every third-party part and ship it next to the part.
- Send access keys privately and make them revocable.
- Remove every bypass before a build leaves your machine, and test the shipped build. A build step can change behaviour.
- Publish what the tool sends and to whom, write a plain readme, and test the stop in every state by trying to break it.

A tool described by its own maker, including a tool made by Vivian, is a claim until someone else has checked it.

TRY IT

Check yourself

QUESTION 1 OF 4

A tool asks to read and change your data on every website. What's the first thing to do?

- ◯ Accept, since the maker looks friendly
- ◯ Ask whether it can be limited to the one site it serves, and walk away if not
- ◯ Accept now and review it later

Answer. Ask whether it can be limited to the one site it serves, and walk away if not. Access to every website is the widest access a tool can ask for. Narrow it or leave.

QUESTION 2 OF 4

An update adds a new permission that triggers a warning. What does Chrome do?

- ◯ Nothing until you restart
- ◯ It disables the extension until you accept the new permission
- ◯ It removes the extension

Answer. It disables the extension until you accept the new permission. Chrome disables the extension until the user accepts. ◆

QUESTION 3 OF 4

An assistant read a tool's code and found nothing wrong. Which verdict is honest?

- ◯ The tool is safe
- ◯ Nothing obvious was found in the part that was read
- ◯ The tool passed an audit

Answer. Nothing obvious was found in the part that was read. A summary covers the part that was read. It can't see what's hidden or what an update may add.

QUESTION 4 OF 4

A dom asks you to open a remote session so she can fix your tool. What do you do?

- ◯ Open it, but watch the screen
- ◯ Decline, and offer nothing from your accounts
- ◯ Open it and block your own keyboard so you don't interfere

Answer. Decline, and offer nothing from your accounts. Full control of your computer can't be taken back.

REMEMBER IT

- In my experience most risky tools do what they say, with a hidden price or no way out.
- Check design (stop, limit, total, defaults) as well as code (permissions, addresses, updater).
- An update can change a tool without asking, so treat the update channel as a permission.
- Never test an unknown tool on the computer where you're signed in.
- Remote access is a red line, and a promise can't protect it.
- A verdict states what was read and what wasn't.

SOURCES AND EVIDENCE

Check it

Every claim with its evidence level and its sources.

EVIDENCE LEVELS

Verified	Checked against the original source.
Reported	Stated by a serious secondary source read in full.
Community	Widely said in the scene; no document proves it.
Vivian's field notes	First-hand experience, not a study.

◆ Unverified — Plausible, not proven yet.

01 A techdom tool runs on your side of the screen, often inside your signed-in browser, next to a saved card.

◆

02 Malware is the rarer case.

◆

03 Press describes a scene flooded with scammers who use stolen photos of attractive women.

◆

Financial Domination Has a Scamming Problem (https://www.vice.com/en/article/financial-domination-has-a-scamming-problem)

Archive (https://web.archive.org/web/20260827104045/https://www.vice.com/en/article/financial-domination-has-a-scamming-problem/)

Accessed 3 November 2026

04 A free tool offered by a stranger deserves the same doubt as a beautiful profile.

◆

05 Your best protection is the limit you set before you start.

◆

06 If you need a checklist to feel safe, not installing is a fine answer.

◆

07 I never run an unknown program on the computer where I'm signed in. A throwaway setup costs time, and that's cheap next to a drained card.

◆

08 When you add an extension, Chrome shows permission warnings that describe what the permissions allow.

◆

Permission warnings (Chrome for Developers, extensions documentation) (https://developer.chrome.com/docs/extensions/develop/concepts/permission-warnings)

Archive (https://web.archive.org/web/20261102194346/https://developer.chrome.com/docs/extensions/develop/concepts/permission-warnings)

Accessed 3 November 2026

09 Host permissions are what let an extension work with the sites that match its listed patterns.

◆

Declare permissions | Chrome Extensions | Chrome for Developers (https://developer.chrome.com/docs/extensions/develop/concepts/declare-permissions)

Archive (https://web.archive.org/web/20261006143033/https://developer.chrome.com/docs/extensions/develop/concepts/declare-permissions)

Accessed 3 November 2026

10 Some permissions are optional: granted while the extension runs, not at install.

◆

Declare permissions | Chrome Extensions | Chrome for Developers (https://developer.chrome.com/docs/extensions/develop/concepts/declare-permissions)

Archive (https://web.archive.org/web/20261006143033/https://developer.chrome.com/docs/extensions/develop/concepts/declare-permissions)

Accessed 3 November 2026

11 After install you can open Extensions, then Manage extensions, then a tool's Details page to see which sites it may access.

◆

Install and manage extensions - Chrome Web Store Help (https://support.google.com/chrome_webstore/answer/2664769)

Archive (https://web.archive.org/web/20261015165420/https://support.google.com/chrome_webstore/answer/2664769)

Accessed 3 November 2026

12 That page has a setting for whether the tool may read and change your data on every website you visit.

◆

Install and manage extensions - Chrome Web Store Help (https://support.google.com/chrome_webstore/answer/2664769)

Archive (https://web.archive.org/web/20261015165420/https://support.google.com/chrome_webstore/answer/2664769)

Accessed 3 November 2026

13 If an update adds a permission that triggers a warning, Chrome disables the extension until you accept it.

◆

Permission warnings (Chrome for Developers, extensions documentation) (https://developer.chrome.com/docs/extensions/develop/concepts/permission-warnings)

Archive (https://web.archive.org/web/20261102194346/https://developer.chrome.com/docs/extensions/develop/concepts/permission-warnings)

Accessed 3 November 2026

14 An update that stays within the permissions it already has asks you nothing.

◆

15 The Chrome Web Store requires extensions to disclose how they handle user data, even when it never leaves the device.

◆

Updated Privacy Policy & Secure Handling Requirements | Chrome Web Store - Program Policies | Chrome for Developers (https://developer.chrome.com/docs/webstore/program-policies/user-data-faq)

Archive (https://web.archive.org/web/20261018233331/https://developer.chrome.com/docs/webstore/program-policies/user-data-faq)

Accessed 3 November 2026

16 A tool you load from a folder skips that listing, so there's no disclosure to read.

◆

17 Chrome's own pages are the reference.

◆

18 A manifest check can't see what the code does, and a matching fingerprint only proves you hold the file the maker published.

◆

19 A verdict has two parts: what the check found, and what it didn't cover.

◆

20 Only say this after reading the whole source, and with the evidence.

◆

21 Serious, but not malware.

◆

22 Nothing hidden is harmful, but the defaults expose you: a machine identifier sent from your names, a safety-word or quick-exit off by default.

◆

23 Treat it as unsafe for any computer that holds your sign-ins.

◆

24 It does mean you can't verify the tool, and you shouldn't take a tool that holds a signed-in session on trust.

◆

25 Check the design as well as the code.

◆

26 Trust is the one thing a spending tool shouldn't need.

◆

27 A summary isn't an audit.

◆

28 A remote-session hands your computer to someone else: files, saved passwords and screens.

◆

29 A US government alert advises people not to give control of their computers or mobile devices to any stranger.

◆

IC3 Issues Alert on Tech Support Fraud (https://www.cisa.gov/news-events/alerts/2018/03/29/ic3-issues-alert-tech-support-fraud)

Archive (https://web.archive.org/web/20250716102123/https://www.cisa.gov/news-events/alerts/2018/03/29/ic3-issues-alert-tech-support-fraud)

Accessed 3 November 2026

30 A remote-access vendor tells users not to give device access to people they do not know or trust.

◆

TeamViewer and scamming (https://www.teamviewer.com/en/global/support/knowledge-base/teamviewer-remote/security/teamviewer-and-scamming/)

Archive (https://web.archive.org/web/20260519191922/https://www.teamviewer.com/en/global/support/knowledge-base/teamviewer-remote/security/teamviewer-and-scamming/)

Accessed 3 November 2026

31 A tool that stays installed is an ongoing risk: its maker can push a harmful update later.

◆

32 On Windows, uninstall from Settings, then Apps, then Installed apps.

◆

Uninstall or remove apps and programs in Windows | Microsoft Support (https://support.microsoft.com/en-us/windows/uninstall-or-remove-apps-and-programs-in-windows-4b55f974-2cc6-2d2b-d092-5905080eaf98)

Archive (https://web.archive.org/web/20261024175047/https://support.microsoft.com/en-us/windows/uninstall-or-remove-apps-and-programs-in-windows-4b55f974-2cc6-2d2b-d092-5905080eaf98)

Accessed 3 November 2026

33 Some apps can't be removed there.

◆

Uninstall or remove apps and programs in Windows | Microsoft Support (https://support.microsoft.com/en-us/windows/uninstall-or-remove-apps-and-programs-in-windows-4b55f974-2cc6-2d2b-d092-5905080eaf98)

Archive (https://web.archive.org/web/20261024175047/https://support.microsoft.com/en-us/windows/uninstall-or-remove-apps-and-programs-in-windows-4b55f974-2cc6-2d2b-d092-5905080eaf98)

Accessed 3 November 2026

34 To stop one starting with the computer, switch it off in Settings, Apps, Startup, or in Task Manager's Startup tab.

◆

Configure Startup applications in Windows | Microsoft Support (https://support.microsoft.com/en-us/windows/configure-startup-applications-in-windows-115a420a-0bff-4a6f-90e0-1934c844e473)

Archive (https://web.archive.org/web/20260624053016/https://support.microsoft.com/en-us/windows/configure-startup-applications-in-windows-115a420a-0bff-4a6f-90e0-1934c844e473)

Accessed 3 November 2026

35 You remove a browser extension from the Manage extensions page.

◆

36 I treat a signed-in session like a password. Whoever holds it acts as you, and a two-step login doesn't help once the session is taken. After a tool I doubt, I sign out everywhere as well as changing the password.

◆

37 If you build tools, your users should be able to do every check above.

◆

38 Leftover handles point to your accounts.

◆

39 Check the licence of every third-party part and ship it next to the part.

◆

40 Send access keys privately and make them revocable.

◆

41 A build step can change behaviour.

◆

42 Publish what the tool sends and to whom, write a plain readme, and test the stop in every state by trying to break it.

◆

43 A tool described by its own maker, including a tool made by Vivian, is a claim until someone else has checked it.

◆

44 Chrome disables the extension until the user accepts.

◆

Permission warnings (Chrome for Developers, extensions documentation) (https://developer.chrome.com/docs/extensions/develop/concepts/permission-warnings)

Archive (https://web.archive.org/web/20261102194346/https://developer.chrome.com/docs/extensions/develop/concepts/permission-warnings)

Accessed 3 November 2026