How to choose an uninstaller for Mac

A transparent evaluation methodology

Objective criteria for comparing uninstallers on macOS. This methodology is based on hands-on experience in developing and testing macOS utilities (Nektony, since 2011).

Table of Contents

TL;DR

For users

A good uninstaller must:

- Remove only the app-related files, leaving your documents and other data untouched
- Quit all active system processes after removal

- Show you exactly what will be removed before you click the uninstall button
- Handle pkg apps, VPNs, and helpers, not just simple .app bundles

> To check any uninstaller quickly, use the calculator

For reviewers and QA

48 criteria across 3 levels (9 blockers + 20 functional + 19 bonuses), maximum score is 86. 9 test application classes, 5 trap scenarios, a step-by-step testing protocol with a baseline/ground truth approach.

Why this methodology exists

In the Mac App Store and on developer websites, there are dozens of uninstallers. They all promise "complete removal with no leftovers." But how can you tell which one is actually safe?

Picking the wrong tool can cost you:

- User documents deleted from ~/Documents or ~/Desktop
- An active root helper left behind with privileged system access
- A shared vendor folder removed, and other apps broken (Adobe, Microsoft)
- A broken launchd service: the binary is gone, but the plist remains, and the system endlessly tries to launch a ghost process
- A VPN profile keeps running after the "uninstalled" client is gone

99% of articles on uninstallers are just opinions. "I liked it," "nice interface," "found 5 GB of junk." And it's unclear which app was uninstalled, on which macOS, what was treated as "leftovers," or whether launchd services and pkg receipts were checked.

With this methodology, you get actual test results:

- **Unified criteria** for comparing any uninstallers
- **Objective tests** with measurable results
- **Safety first:** what actually matters to users

- **Reproducibility:** anyone can repeat the tests and verify the results

Who this methodology is for

Audience	How to use	What to read
Regular users	Quickly evaluate an uninstaller before purchase	Calculator, criteria tables
Journalists and reviewers	Run a fair comparison for a publication	Full testing procedure
QA engineers	Full product testing	All phases, all application classes
Developers	Understand quality standards in the category	L1 criteria, risk model

Terms and definitions

Before you get started, check the terminology out:

Term	Definition
Application leftover (artifact)	A file or process created by an application. Examples: ~/Library/Application Support, LaunchAgents, caches
User data	Files created by the user WITH the application. ~/Documents, exports, projects. Must not be removed without confirmation
Gray zone (shared)	Elements shared by multiple apps or important settings. Shared vendor folders, Keychain entries. Require an action by the user
Evidence-based detection	Finding files based on a provable link: bundle ID, team ID, pkg receipts. The opposite of "search by folder name"

Term	Definition
Bundle ID	A unique application identifier (com.vendor.appname). The primary way to link files to an application
PKG Receipt	An entry of a .pkg package in /var/db/receipts/. Provides an exact list of installed files
pkgutil --forget	A command that removes ONLY the entry, but does NOT remove the actual files. If an uninstaller says "Removed," that's not true
Shared components	Files belonging to multiple apps from the same vendor. Example: /Library/Application Support/Adobe/
Ground truth	The complete list of an app's artifacts. Obtained by comparing the system BEFORE and AFTER installation
Precision	Of everything suggested for removal, how much actually belongs to the app
Recall	How many real artifacts were found

Priority in metrics conflict: Precision > Recall. It's better to miss a cache file than delete needed data.

Metrics and result calculation

For objective evaluation purposes, we use standard metrics from Information Retrieval:

Core metrics

Metric	Formula	What it measures
Precision	TP / (TP + FP)	Of everything suggested, how many actually relate to the app

Metric	Formula	What it measures
Recall	TP / (TP + FN)	How many artifacts were found

Where:

- **TP (True Positive)**: a ground truth file found by the uninstaller
- **FP (False Positive)**: a file NOT in ground truth, suggested for removal
- **FN (False Negative)**: a ground truth file NOT found by the uninstaller

Three numbers per test

For each uninstallation, we capture:

- **Suggestion for removal**: what the uninstaller showed in its list (this is where false positives are caught)
- **Actual removal**: what was actually removed after confirmation
- **Collateral damage**: what extra was removed, what was broken (other apps, launchd, settings)

This comes down to checking three things against separate L1/L2 criteria and the results table by application class.

Calculation example

Let's assume FooApp created 15 files during installation (ground truth = 15).

The uninstaller suggested removing 18 files:

- 13 of them are real FooApp artifacts (TP = 13)
- 5 are files from another app (FP = 5)
- 2 FooApp artifacts were not found (FN = 2)

Result:

Precision = 13 / (13 + 5) = 72% - a critical failure (L1 requires 100%)

Recall = 13 / (13 + 2) = 87% - below the L2 threshold (requires ≥ 90%)

Core principle: safety over completeness

When evaluating uninstallers, we follow a clear risk priority:

Data Loss > Security/Privacy > System Stability > Cleanliness

- **Data Loss (highest risk)**: deleting user documents, photos, projects
- **Security/Privacy**: a root helper left behind, a VPN profile still active
- **System Stability**: broken launchd services, stuck extensions
- **Cleanliness (low risk)**: remaining caches, logs

A fast uninstaller that deletes the wrong files is dangerous. A slow but accurate one is useful.

One serious failure is worse than a dozen minor flaws.

Scoring system: 3 levels of criteria

All 48 criteria are divided into three levels by importance:

Level	Name	Criteria	Weight	Max points
L1	Must Have (safety)	9	×3	27
L2	Should Have (functionality)	20	×2	40
L3	Nice to Have (bonuses)	19	×1	19
	Total	**48**		**86**

How to interpret the result

Score	Rating
67–86	✓ Recommended
44–66	⚠️ With caveats

Score	Rating
< 44	❌ Not recommended

Threshold logic:

67 = all L1 (27) + ~75% L2 (30) + ~50% L3 (10) - a safe, functional tool with basic conveniences.

44 = all L1 (27) + ~40% L2 (17) - safe, but with noticeable gaps in functionality.

Important: Failing any Level 1 criterion automatically falls under "Not recommended," regardless of the total score.

Evaluation criteria

L1 Level 1: Must Have (blockers)

Safety criteria. Failing any of them means a risk of data loss or system damage.

ID	Criterion	Description
1.1	Precision = 100%	Does not suggest removing files unrelated to the app
1.2	No user data removal without opt-in	Does not remove ~/Documents, ~/Desktop, ~/Downloads without explicit consent
1.3	Correct handling of shared folders	Checks dependencies before removing a shared vendor folder
1.4	No root helpers left behind	No active helpers in /Library/PrivilegedHelperTools after removal
1.5	No broken launchd services	Removes plist AND binary as a pair. Unloads the service

ID	Criterion	Description
1.6	Correct handling of PKG receipts	Does not use pkgutil --forget without removing files
1.7	Profiles/Extensions handled	Configuration Profiles and Extensions removed or instructions shown
1.8	Confirmation before removal	Removal only after explicit confirmation
1.9	Stability	No crashes. Handles permission errors correctly

Red flags

If an uninstaller does any of the following, don't use it:

🚩 **Safety**

- *Suggests removing a file just because it has the same name as the app*
- *Removes files without confirmation*
- *Auto-checks shared folders and user data for removal*
- *Root helper left behind after removal*

🚩 **Transparency**

- *"37 GB of threats!" with no evidence, pushes you to remove*
- *Scans for free, charges for removal, without warning you first*
- *Price not visible until the purchase*

L2 Level 2: Should Have (important)

Functionality and coverage criteria. Without them, a tool can still be used, but with limitations.

Coverage and detection

ID	Criterion	Description
2.1	Recall ≥ 90% of standard paths	Finds Application Support, Preferences, Caches, Containers, Logs
2.2	Finds LaunchAgents/Daemons	User-level and system-level agents and daemons
2.3	Finds PrivilegedHelperTools	Root helpers in /Library/PrivilegedHelperTools
2.4	Finds PKG receipts and components	Uses pkgutil to find files from .pkg
2.5	Finds browser extensions	Safari, Chrome extensions
2.6	Finds Login Items/Background Items	Startup elements
2.20	Finds symlinks, aliases, wrappers	Links in /usr/local/bin, Homebrew paths

Removal safety

ID	Criterion	Description
2.7	Stops processes before removal	Unload/stop for launchd before deleting files
2.11	Works with sandbox apps	Containers and Group Containers for App Store apps
2.12	Finds leftovers without .app	Finds artifacts of manually removed apps
2.17	Distinguishes bundle IDs	Does not confuse apps with the same name

Interface and UX

ID	Criterion	Description
2.8	File list preview	Full list with selection options
2.9	Move to Trash by default	To Trash, not permanently
2.10	Undo/restore	Option to reverse the removal
2.13	Access error handling	Clear messages, keeps working
2.14	Shows what it can't remove	Explains why and what to do
2.15	Evidence-based explanations	Why the file is linked to the app
2.16	Separates app data/user data	Safe defaults (user data NOT pre-selected)
2.18	Scan speed	< 15 seconds per app
2.19	Monetization transparency	Price visible, limitations clear, no traps

L3 Level 3: Nice to Have (bonuses)

Missing them is not critical, but having them makes the overall experience better.

ID	Criterion	Description
3.1	Dark mode	Supports macOS dark theme
3.2	VoiceOver/accessibility	Basic VoiceOver navigation works
3.3	Localization	Supports 5+ interface languages
3.4	Onboarding	First-launch tutorial, explains permissions
3.5	Help/support	Built-in help, FAQ, support contacts
3.6	Warnings explain the risk	Warnings describe the risk and recommend an action
3.7	Operation log	Removal history with review option

ID	Criterion	Description
3.8	Report export	Exports scan/removal results to a file
3.9	App reset	Removes preferences and caches without uninstalling the app itself
3.10	Post-uninstall verification	Checks for leftovers after removal, warns about critical ones
3.11	Keychain entries (with opt-in)	Finds related Keychain entries, suggests removing them with explicit choice
3.12	Menu bar agents, input methods	Finds menu bar agents and input methods installed by the app
3.13	Network Extensions/KEXT	Shows network extensions and KEXT or provides removal instructions
3.14	Search/filter in results	Search or filter field in the file list
3.15	Result sorting	Sorts by size, type, location
3.16	Session saving	Saves results after quitting the tool
3.17	Drag & drop uninstallation	Dragging .app into the window triggers removal
3.18	Help on error	Shows advice on how to fix the problem when an error occurs
3.19	No telemetry without consent or third-party software	Does not send data to servers without the user's knowledge, does not install third-party software

Test application classes

For thorough testing, you need to check the uninstaller against apps of different classes. Each class creates its own set of artifacts and presents its own risks.

#	Class	Characteristics	What to check
1	Drag & Drop .app	Simple app	Application Support, Preferences, Caches
2	App Store sandbox	Containers	~/Library/Containers, Group Containers
3	PKG installer (simple)	Receipts	pkgutil, components in /Library
4	PKG + helpers/daemons	Installer + processes	LaunchDaemons, PrivilegedHelperTools
5	With LaunchAgent/Daemon	Startup	Service unloading, removing plist + binary pairs
6	With Privileged Helper	Root access	/Library/PrivilegedHelperTools
7	VPN/AV/Firewall	Extensions, Profiles	System Settings, Network Extensions, Configuration Profiles
8	"Family" (Adobe, JetBrains)	Shared components	Shared vendor folders are not removed
9	Already removed manually	Leftovers only, no .app	Works by bundle ID without .app

App examples for each class

These examples are provided to make it clear what types of artifacts are typical for each class. This is not about evaluating these apps. The choice of specific apps for testing is up to the QA.

Examples are current as of publication (March 2026). Apps may change their installation mechanism between versions; check before testing.

Class	Examples
1	Telegram, Figma, Spotify, Notion
2	Pages, Keynote, Xcode, Slack (Mac App Store)
3	Wireshark, R for macOS, LibreOffice
4	Adobe Creative Cloud, AutoCAD, Logitech Options
5	Dropbox, Google Drive, Syncthing
6	VMware Fusion, Parallels Desktop, Little Snitch
7	Tunnelblick, WireGuard, Sophos Home, Little Snitch
8	Adobe (Photoshop + Illustrator), JetBrains (IntelliJ + PyCharm), Microsoft (Word + Excel)
9	Any from classes 1–8, removed manually

Minimum set: one app from each class. For a basic L1 check, a short set of 4 tests (see below). For a full L2, all 9 classes.

Typical artifacts by class

Class	Artifacts	Where to look
1	Preferences, Caches, Application Support	~/Library/
2	Containers, Group Containers	~/Library/Containers/[bundle-id]/
3	Files from receipt, components	/Library/, /var/db/receipts/
4	All from class 3 + LaunchDaemons, Helpers	/Library/LaunchDaemons/, /Library/PrivilegedHelperTools/

Class	Artifacts	Where to look
5	LaunchAgents (plist + binary)	~/Library/LaunchAgents/, /Library/LaunchAgents/
6	Privileged Helper Tool	/Library/PrivilegedHelperTools/
7	Configuration Profiles, Network Extensions	System Settings, /Library/SystemExtensions/
8	Shared vendor folder	/Library/Application Support/[Vendor]/
9	Orphaned files (without .app)	All paths from classes 1–8

Performance baseline

For Apple Silicon SSD:

Operation	Excellent	Acceptable	Slow	Unacceptable
Scanning a single app	< 5 sec	5-15 sec	15-30 sec	> 30 sec
Removal (after confirmation)	< 5 sec	5-15 sec	15-30 sec	> 30 sec
Scanning all apps	< 30 sec	30-90 sec	90-180 sec	> 180 sec

For Intel Macs, multiplying these values by 1.5–2x is acceptable.

Speed measurements: 3 runs with a cold start (`sudo purge && sleep 30`), the median to be captured. Dispersion (max - min)/median ≤ 15%.

Memory (RAM): < 200 MB excellent, < 500 MB acceptable, > 1 GB unacceptable.

Risks and common pitfalls

Read this before testing!

Dangerous scenarios

Scenario	Risk
Removing a shared vendor folder	Breaks other apps from the same vendor
Root helper left behind	A process with root privileges runs without its app
Broken launchd service	The system endlessly tries to launch a non-existent process
pkgutil --forget without removing files	The UI says "removed," but the files are still there
VPN profile after removal	Traffic keeps going through the removed VPN
Keychain removal by name	May affect entries belonging to other apps

Common testing mistakes

- Not capturing a baseline before installing the test app
- Not checking ground truth (not knowing what the app actually installed)
- Testing only on simple .app (class 1) doesn't cover pkg, helpers, extensions
- Not checking whether other apps still work after uninstallation
- Comparing App Store and non-App Store versions of the uninstaller without taking into account the Sandbox limitations

Testing procedure

> ⚠️ **Warning:** the uninstallers you're testing may remove files, damage the file system, or break macOS. Before you start, create a full system backup (Time Machine or equivalent). Don't test on a production machine with real data.

Phase 0

Preparation

0.1 Set up the test environment

1. Create a new macOS user or use a fresh session.
2. Gather your configuration:
   - macOS version: _______________
   - Hardware (processor, RAM, disk): _______________
   - Test date: _______________
3. Quit all unnecessary apps.

0.2 Prepare the uninstaller

1. Gather the following details:
   - Name and version: _______________
   - Source (App Store/website): _______________
   - Price/monetization model: _______________
2. Check the signature and notarization:

   ```
   codesign -dv --verbose=2 /path/to/app.app
   spctl --assess --verbose /path/to/app.app
   ```

   Signed? Yes / No. Notarized? Yes / No

   An uninstaller gets Full Disk Access and root privileges. An unsigned tool with that level of access is an increased risk. If unsigned, note it in the report.
3. Install the uninstaller. During installation: does it install third-party software or show ads?
4. Grant Full Disk Access.
5. Launch it, go through the onboarding.

0.3 Placeholders used in commands

The scripts and instructions below use placeholders, replace them with the actual values for your test app:

Placeholder	Description	Examples
[AppName]	Tool name (as in /Applications)	Slack, Figma, Wireshark
[VendorName]	Developer/company name	Adobe, JetBrains, Nektony
[bundle_id]	Bundle ID from Info.plist	com.tinyspeck.slackmacgap, com.figma.Desktop, org.wireshark.Wireshark
[vendor]	Part of the developer name for grep filtering	tinyspeck, jetbrains, adobe
[package-id]	Package ID from pkgutil --pkgs	com.wireshark.chmodbpf.pkg, com.adobe.pkg.ACCCx7_0_0_407, org.nmap.npcap

How to find the bundle ID: open the app in Finder → right-click → "Show Package Contents" → Contents/Info.plist → CFBundleIdentifier key.
Or in Terminal: `mdls -name kMDItemCFBundleIdentifier /Applications/[AppName].app`

0.4 Prepare test apps

For each test app:

1. Take a screenshot of a baseline (system state BEFORE installation).
2. Install the test app, launch it, and grant all permissions.
3. Take a screenshot of ground truth (state AFTER installation).

Baseline capture script (BEFORE installation):

```
mkdir -p ~/uninstall-audit/baseline
OUT=~/uninstall-audit/baseline
```

```
touch "$OUT/BEFORE_INSTALL.marker"
pkgutil --pkgs | sort > "$OUT/pkgutil_pkgs.txt"
ls -1 ~/Library/LaunchAgents 2>/dev/null | sort >
"$OUT/user_launchagents.txt"
sudo ls -1 /Library/LaunchAgents 2>/dev/null | sort >
"$OUT/lib_launchagents.txt"
sudo ls -1 /Library/LaunchDaemons 2>/dev/null | sort >
"$OUT/lib_launchdaemons.txt"
sudo ls -1 /Library/PrivilegedHelperTools 2>/dev/null | sort >
"$OUT/privileged_helpers.txt"
ls -1 ~/Library/Containers 2>/dev/null | sort >
"$OUT/containers.txt"
ls -1 ~/Library/Group\ Containers 2>/dev/null | sort >
"$OUT/group_containers.txt"
```

Ground truth capture script (AFTER installation):

```
mkdir -p ~/uninstall-audit/after
OUT=~/uninstall-audit/after
pkgutil --pkgs | sort > "$OUT/pkgutil_pkgs.txt"
ls -1 ~/Library/LaunchAgents 2>/dev/null | sort >
"$OUT/user_launchagents.txt"
sudo ls -1 /Library/LaunchAgents 2>/dev/null | sort >
"$OUT/lib_launchagents.txt"
sudo ls -1 /Library/LaunchDaemons 2>/dev/null | sort >
"$OUT/lib_launchdaemons.txt"
sudo ls -1 /Library/PrivilegedHelperTools 2>/dev/null | sort >
"$OUT/privileged_helpers.txt"
ls -1 ~/Library/Containers 2>/dev/null | sort >
"$OUT/containers.txt"
ls -1 ~/Library/Group\ Containers 2>/dev/null | sort >
"$OUT/group_containers.txt"
```

Calculate the difference:

```
BASE=~/uninstall-audit/baseline
AFTER=~/uninstall-audit/after
GT=~/uninstall-audit/ground_truth
mkdir -p "$GT"
comm -13 "$BASE/pkgutil_pkgs.txt" "$AFTER/pkgutil_pkgs.txt" >
"$GT/new_receipts.txt"
comm -13 "$BASE/user_launchagents.txt"
"$AFTER/user_launchagents.txt" > "$GT/new_user_la.txt"
comm -13 "$BASE/lib_launchagents.txt"
"$AFTER/lib_launchagents.txt" > "$GT/new_lib_la.txt"
comm -13 "$BASE/lib_launchdaemons.txt"
"$AFTER/lib_launchdaemons.txt" > "$GT/new_lib_ld.txt"
comm -13 "$BASE/privileged_helpers.txt"
"$AFTER/privileged_helpers.txt" > "$GT/new_helpers.txt"
comm -13 "$BASE/containers.txt" "$AFTER/containers.txt" >
"$GT/new_containers.txt"
comm -13 "$BASE/group_containers.txt"
"$AFTER/group_containers.txt" > "$GT/new_gcontainers.txt"
```

By time marker (files not in the lists):

```
MARK="$BASE/BEFORE_INSTALL.marker"
find ~/Library/Application\ Support -newer "$MARK" 2>/dev/null >
"$GT/changed_appsupport.txt"
find ~/Library/Preferences -newer "$MARK" 2>/dev/null >
"$GT/changed_prefs.txt"
find ~/Library/Caches -newer "$MARK" 2>/dev/null >
"$GT/changed_caches.txt"
```

0.5 Set up traps

Before the main tests, set up traps to check for false positives:

- **Trap A "Naive name search":** Create ~/Documents/[AppName]/ with pdf, jpg, and DO_NOT_DELETE.txt files

- **Trap B "Vendor name":** Create ~/Documents/[VendorName]/ and fill it with files
- **Trap C "Exports and backups":** Create in ~/Documents: [AppName]-export-2026.zip, [AppName]-backup.json
- **Trap D "Shared folder":** Install 2 apps from the same vendor
- **Trap E "Same name":** Install two apps with the same name but different bundle IDs

Alternative: minimum set for a quick check

If you are short on time for the full protocol, use a short set of 4 tests:

#	What to test	Class	App example	What to check
1	Simple .app	1	Any from /Applications	Basic recall, preview
2	PKG app	3	App with .pkg installer	PKG receipts, components in /Library
3	App with LaunchAgent	5	App with background processes	launchd unloading, plist+binary pair removal
4	Trap A	-	~/Documents/[AppName]/ with files	False positives, precision

Set-up for Trap A:

```
~/Documents/
└── [AppName]/
    ├── report-2026.pdf
    ├── photo.jpg
    └── DO_NOT_DELETE.txt
```

Expected results:

Test	Pass	Fail
Simple .app	Recall ≥ 90%, all ground truth files found	Recall < 90%
PKG app	Files from the receipt found and removed	Receipt forgotten, but files remain
LaunchAgent	Service unloaded, plist + binary removed as a pair	Orphaned plist or binary left behind
Trap A	~/Documents/[AppName] NOT suggested for removal	Suggested and pre-selected by default

The set covers criteria: 1.1, 1.2, 1.5, 1.6, 1.8, 1.9, 2.1, 2.4, 2.8

The set does NOT cover (key L1 gaps):

Criterion	Why it's not covered	What you need to check it
1.3 Shared folders	Needs 2 apps from the same vendor	Class 8 (Adobe, JetBrains)
1.4 Root helpers	Needs an app with a Privileged Helper	Class 6
1.7 Profiles/Extensions	Needs a VPN/Firewall app	Class 7
2.11 Sandbox	Needs an App Store app	Class 2

For a complete evaluation, use all 9 classes.

Phase 1

Level 1 criteria check (Safety gates)

Goal: Check critical safety requirements. Failing any of them means the tool is "Not recommended."

1.1 Precision = 100%

Why it matters: The uninstaller must not suggest removing files that don't belong to the app. A mistake here means losing user data.

1. Run the uninstallation of a class 1 test app (simple .app).
2. Review the list of suggested files.
3. For each file, check: does it relate to the app being removed (by bundle ID, path)?
4. Compare what's suggested against ground truth.
5. Check traps A, B, C.
6. Repeat for classes 3, 5, 8.

✔ All suggested files relate to the app. Traps passed. | ❌ FAIL: At least one file from another app or system found.

1.2 No user data removal without opt-in

Why it matters: User documents must not be removed without explicit consent.

1. Use Trap A: ~/Documents/[AppName]/, ~/Desktop/[AppName]/, ~/Downloads/[AppName] files.
2. Proceed with uninstalling this app.
3. Check: does the uninstaller suggest removing these folders/files?
4. If it does, are they pre-selected by default?

✔ Does not show ~/Documents, ~/Desktop, ~/Downloads as leftovers
✔ Shows them but does NOT pre-select by default and warns
⚠ Shows them, NOT pre-selected, but no warning (half score)
❌ FAIL: Pre-selects by default or removes without a separate step

1.3 Correct handling of shared folders

Why it matters: Removing a shared folder will break other apps from the same vendor.

1. Use Trap D: 2 apps from the same vendor.
2. Remove only one via the uninstaller.
3. Check: does it suggest removing the shared vendor folder?

✔ Does not suggest removing the shared folder
✔ Suggests only the app-specific subfolder
⚠ Suggests with a dependency warning
❌ FAIL: Suggests removing the entire vendor folder without checking

1.4 No root helpers left behind

Why it matters: A root helper left behind is a process with full privileges running without its app.

1. Pick a class 6 app (with Privileged Helper)
2. Before removal: `sudo ls /Library/PrivilegedHelperTools/` and `sudo launchctl list | grep [vendor]`
3. Remove the app
4. Run the commands again, is the helper gone? Is the service unloaded?

✔ Helper removed, service unloaded
⚠ Shown, but requires a password, this is explained
❌ FAIL: Helper remains active without warning

1.5 No broken launchd services

Why it matters: If the binary is deleted but the plist remains, launchd will endlessly try to launch the process.

1. Pick a class 5 app (with LaunchAgent/Daemon).
2. Before removal, make a screenshot of all plists and binaries.
3. Remove via the uninstaller.
4. Check: for each plist, is the binary removed? And vice versa.
5. `launchctl list | grep [bundle_id]` - should return nothing.

✔ All plist + binary pairs removed, service unloaded | ❌ FAIL: An orphaned plist or binary remains

1.6 Correct handling of PKG receipts

Why it matters: `pkgutil --forget` without removing files is not true. The UI says "removed," but the files are still there.

1. Pick a class 3 or 4 app (PKG installer).
2. Before removal: `pkgutil --pkgs | grep [vendor]` and `pkgutil --files [package-id]`
3. Remove via the uninstaller.
4. Check: are the files from the receipt removed? Is the receipt forgotten?

✔ Files removed AND receipt forgotten
✔ Files removed, receipt not forgotten (acceptable)
❌ FAIL: Receipt forgotten, but files remain

1.7 Profiles/Extensions handled

Why it matters: A VPN profile still active after "uninstalling" the client is a privacy violation.

1. Pick a class 7 app (VPN or Firewall).
2. Before removal: System Settings → VPN & Network, Login Items & Extensions.
3. Remove via the uninstaller.
4. Check: are profiles and extensions removed, or is a warning shown?

✔ Profiles/extensions removed
✔ Warning shown with instructions
❌ FAIL: No mention of profiles/extensions, they keep running

1.8 Confirmation before removal

Why it matters: An accidental click should not lead to removal.

1. Select any app for removal.
2. Click the remove button.
3. Does a confirmation dialog appear?

✔ Shows confirmation with details | ❌ FAIL: Removes immediately without confirmation

1.9 Stability

Why it matters: The uninstaller must not crash, including when it runs into permission errors.

1. Uninstall apps from all tested classes one after another.
2. Check whether there is any crash or freeze.
3. Try running without Full Disk Access.

✔ Runs stable, handles errors correctly | ❌ FAIL: Crash, freeze, or incorrect behavior

Level 1 summary

#	Criterion	Result	Comment
1.1	Precision = 100%	✔ / ❌	
1.2	No user data removal without opt-in	✔ / ❌	
1.3	Shared folders	✔ / ⚠ / ❌	
1.4	Root helpers	✔ / ⚠ / ❌	
1.5	Launchd services	✔ / ❌	
1.6	PKG receipts	✔ / ❌	
1.7	Profiles/Extensions	✔ / ❌	
1.8	Confirmation	✔ / ❌	
1.9	Stability	✔ / ❌	

Is there at least one ❌ ? → "Not recommended"

Phase 2

Level 2 criteria check

Goal: Check functionality and coverage. Each criterion is checked on an app of the corresponding class.

2.1 Recall ≥ 90% of standard paths

Why it matters: The more real artifacts the uninstaller finds, the less junk remains in the system. Recall below 90% means a significant portion of files will remain.

1. Uninstall class 1, 2, 3 apps via the uninstaller.
2. For each app, compare the suggested list against the ground truth.
3. Calculate: Recall = found/ground truth × 100%.
4. Check standard paths: Application Support, Preferences, Caches, Containers, Logs.

✔ Recall ≥ 90% for all test apps | ❌ Recall < 90%

2.2 Finds LaunchAgents/Daemons

Why it matters: LaunchAgents and LaunchDaemons are an app's background processes. If you don't remove them, they will keep running and consuming resources.

1. Use a class 5 app.
2. Before removal, make a screenshot of the plist and binary paths.
3. Check: does it show user-level and system-level?

✔ Shows both levels | ❌ Does not find them

2.3 Finds PrivilegedHelperTools

Why it matters: Root helpers in /Library/PrivilegedHelperTools run with admin privileges. The uninstaller should at least show them.

1. Use a class 6 app.
2. Before removal: `sudo ls /Library/PrivilegedHelperTools/`
3. Run the uninstaller, does it show the helper in the list?

✔ Finds and shows helpers | ❌ Does not find them

2.4 Finds PKG receipts and components

Why it matters: Apps installed via .pkg scatter files across /Library. Without pkgutil, these files are invisible.

1. Use a class 3 or 4 app.
2. Before removal: `pkgutil --pkgs | grep [vendor]`
3. Check: does it find files from the receipt?

✔ Finds files from the receipt | ❌ Does not use receipts

2.5 Finds browser extensions

Why it matters: Some apps install extensions in Safari or Chrome. Leftover extensions can affect browser privacy and performance.

1. Install an app with a Safari or Chrome extension.
2. Run the uninstaller.
3. Check: does it show the extension in the list?

✔ Finds extensions | ❌ Does not find them

2.6 Finds Login Items/Background Items

Why it matters: Login Items and Background Items launch at every login. If you don't remove them, the app "comes back to life" every time you log in.

1. Use an app with login items.
2. Before removal: System Settings → General → Login Items & Extensions.
3. Check: does it show login items in the list?

✔ Finds login items | ❌ Does not find them

2.7 Stops processes before removal

Why it matters: If you delete files of a running process, you can end up with a broken system state. The right sequence: stop the service first, then delete the files.

1. Uninstall a class 5 app.
2. Before removal: `launchctl list | grep [bundle_id]`
3. Watch: does it unload/stop before deleting?
4. After removal: `launchctl list | grep [bundle_id]` - Is it empty?

✔ Stops/unloads before removal | ❌ Deletes files without stopping

2.8 File list preview

Why it matters: You should see the full list of files BEFORE removal and be able to uncheck anything you don't want removed.

1. Select an app for removal.
2. Is a full file list shown?
3. Can you view each file (path, size)?
4. Can you check/uncheck items?

✔ Full list with selection options | ❌ No list or can't change the selection

2.9 Move to Trash by default

Why it matters: Moving to Trash gives you a chance to recover. Permanent removal is dangerous, you can't undo a mistake.

1. Uninstall an app via the uninstaller.
2. Open Trash, are the files there?
3. Is there a choice between "Trash" and "permanently"?

✔ Trash by default, with a choice | ❌ Removes permanently

2.10 Undo/restore

Why it matters: Even when removing to Trash, an undo button in the uninstaller is more convenient, it restores files to their original locations rather than just pulling them out of Trash into a single folder.

1. Uninstall an app.
2. Is there an undo/restore button?
3. Does the restore work correctly?

✔ There is an undo/restore option | ❌ No way to undo

2.11 Works with sandbox apps

Why it matters: App Store apps store their data in ~/Library/Containers and ~/Library/Group Containers. If the uninstaller doesn't know about these paths, it'll miss most of the data.

1. Uninstall a class 2 app (App Store sandbox).
2. Does it find ~/Library/Containers/[bundle-id]?
3. Does it find ~/Library/Group Containers/?

✔ Finds Containers and Group Containers | ❌ Does not find them

2.12 Finds leftovers without .app

Why it matters: The user may have already deleted the .app manually (moved it to Trash), but the leftovers in Library remain. A good uninstaller should be able to find these remaining files.

1. Delete the .app manually (to Trash).
2. Run the uninstaller.
3. Does it find the leftovers? Does it offer to remove them?

✔ Finds leftovers by bundle ID | ❌ Can't see leftovers without the .app

2.13 Access error handling

Why it matters: Without Full Disk Access (FDA), the uninstaller can't remove some files. It should make the situation clear, not crash or silently skip.

1. Revoke Full Disk Access.
2. Try to uninstall an app with files in protected paths.
3. Crash? Clear message? Does it keep working?

✔ Clear message, keeps working | ❌ Crash or silent skip

2.14 Shows what it can't remove

Why it matters: If the uninstaller can't remove a file (password required, SIP protection), it should show this and explain why so you can remove it manually.

1. There should be files requiring a password or manual action.
2. Does it display them separately?
3. Does it explain why it can't remove them and what to do?

✔ Shows and explains | ❌ Silently skips

2.15 Evidence-based explanations

Why it matters: You should understand WHY a file is considered linked to the app. "Because the name is similar" is a bad answer. "Because the bundle ID matches" is a good one.

1. Open the list of files to be removed.
2. Can you see a reason why each file relates to the app?
3. Does it use bundle ID, pkg receipt, or path as evidence?

✔ Explains the linkage (bundle ID, receipt, path)
⚠ Partially
❌ Just a list with no reasoning

2.16 Separates app data/user data

Why it matters: Caches and preferences are safe to remove. But user documents (exports, projects) are not. The uninstaller should separate these categories.

1. Select an app with user data.
2. Is there a separate section for user data?
3. Is user data NOT pre-selected by default?

✔ Separated, safe defaults | ❌ Everything in one list

2.17 Distinguishes bundle IDs

Why it matters: Two apps can have the same name (e.g., "Notes") but different bundle IDs. The uninstaller should tell them apart by ID, not by name.

1. Use Trap E: two apps with the same name, different bundle IDs.
2. Uninstall one of them.
3. Does it leave other files untouched?

✔ Distinguishes correctly | ❌ Confuses the apps

2.18 Scan speed

Why it matters: Nobody wants to wait a minute for each app. Scanning should take seconds, not minutes.

1. Cold start: `sudo purge && sleep 30`
2. Run a scan of a single app, time it.
3. Repeat steps 1–2 twice more (3 runs total).
4. Make a note of the median. Dispersion: (max - min)/median ≤ 15%

✔ Median < 15 seconds
⚠ Median 15–30 seconds (half score)
❌ Median > 30 seconds

2.19 Monetization transparency

Why it matters: The "scans for free, charges for removal" trap is a common trick. You should know about the limitations BEFORE the tool spends time scanning.

1. On first launch: are the free version's limitations clear?
2. Is the price visible before purchase?
3. No trap without warning?

✔ Everything is transparent, price visible | ❌ Paywall trap or hidden price

2.20 Finds symlinks, aliases, wrappers

Why it matters: Some apps create symbolic links in /usr/local/bin or Homebrew paths. If you don't remove them, broken links remain and can interfere with other tools.

1. Use an app with CLI tools.
2. Run the uninstaller.
3. Does it find the links?

✔ Finds and offers to remove | ❌ Does not find them

Level 2 summary

#	Criterion	Result
2.1	Recall ≥ 90%	✔ / ❌
2.2	LaunchAgents/Daemons	✔ / ❌
2.3	PrivilegedHelperTools	✔ / ❌
2.4	PKG receipts	✔ / ❌
2.5	Browser extensions	✔ / ❌

#	Criterion	Result
2.6	Login Items	✔ / ❌
2.7	Stops processes	✔ / ❌
2.8	File list preview	✔ / ❌
2.9	Move to Trash	✔ / ❌
2.10	Undo	✔ / ❌
2.11	Sandbox apps	✔ / ❌
2.12	Leftovers without .app	✔ / ❌
2.13	Access errors	✔ / ❌
2.14	What it can't remove	✔ / ❌
2.15	Evidence-based	✔ / ⚠ / ❌
2.16	App data / user data	✔ / ❌
2.17	Bundle ID	✔ / ❌
2.18	Speed	✔ / ⚠ / ❌
2.19	Monetization	✔ / ❌
2.20	Symlinks/aliases	✔ / ❌

Phase 3

Level 3 criteria check

Goal: Check additional features. Missing them is not critical, but having them makes the overall experience better.

3.1 Dark mode

Turn on macOS dark mode (System Settings → Appearance → Dark). Does the uninstaller switch?

✔ Supports system theme | ❌ No support

3.2 VoiceOver / accessibility

Turn on VoiceOver (Cmd+F5). Try navigating: menu, app list, and the remove button.

✔ Basic navigation works | ❌ VoiceOver doesn't work

3.3 Localization (5+ languages)

System Settings → General → Language & Region. Switch the language. Does it support 5+ languages?

✔ 5+ languages | ❌ English only or less than 5

3.4 Onboarding

On first launch: are there tutorial screens? Do they explain how to use it and what permissions are needed?

✔ There is a clear onboarding | ❌ No tutorial

3.5 Help / support

Help menu or support button. Is there built-in help, FAQ, or a link to support?

✔ The tool comes with help and contacts | ❌ No help

3.6 Warnings explain the risk

When removing risky items (shared folder, system extension), does the warning describe the risk? Recommend an action?

✔ Explains the risk and recommends an action | ❌ Warning without explanation

3.7 Operation log

Is there a removal history? Can you see what was removed yesterday/last week?

✔ The tool comes with a log/history | ❌ No history

3.8 Report export

Can you export scan or removal results to a file (CSV, PDF, text)?

✔ I can export | ❌ No export

3.9 App reset

Is there a "reset to factory settings" feature, removing preferences and caches without uninstalling the app itself?

✔ The tool comes with a reset feature | ❌ No

3.10 Post-uninstall verification

After removal, does the uninstaller check what's left? Does it warn about critical leftovers (helpers, daemons)?

✔ The tool checks and warns | ❌ The tool doesn't check after removal

3.11 Keychain entries (with opt-in)

Does it find Keychain entries linked to the app? Does it offer to remove them with an explicit choice (opt-in)?

✔ Finds, offers with opt-in | ❌ Doesn't find or removes without asking

3.12 Menu bar agents, input methods

If the app installs a menu bar agent or input method, does the uninstaller find these components?

✔ The tool finds them | ❌ The tool doesn't find them

3.13 Network Extensions / KEXT

For apps with network extensions (VPN, firewall) or KEXT, does the uninstaller show them? Does it provide removal instructions?

✔ The tool shows or instructs | ❌ The tool doesn't mention them

3.14 Search / filter in results

Is there a search or filter field in the list of found files?

✔ There is a search/filter | ❌ No

3.15 Result sorting

Can you sort the found files by size, type, and location?

✔ Yes | ❌ No

3.16 Session saving

Close the uninstaller after scanning. Open it again. Are the results still there?

✔ Results are saved | ❌ Lost on close

3.17 Drag & drop uninstallation

Drag the .app from Finder into the uninstaller window. Does it trigger removal?

✔ Drag & drop works | ❌ Not supported

3.18 Help on error

When an error occurs (no permissions, file locked), does it show advice on how to fix the problem?

✔ Shows advice | ❌ Just an error message without help

3.19 No telemetry without consent or third-party software

Install a network monitor (Little Snitch, FireWally, or similar). Run the uninstaller, perform a typical scenario. Does it send data to external servers without the user's knowledge (besides checking for updates)? Did it install third-party software during installation?

✔ Does not send data without the user's knowledge AND does not install third-party software | ❌ Sends data without consent OR installs third-party software

Level 3 summary

#	Criterion	Result
3.1	Dark mode	✔ / ❌

#	Criterion	Result
3.2	VoiceOver/accessibility	✔ / ❌
3.3	Localization (5+ languages)	✔ / ❌
3.4	Onboarding	✔ / ❌
3.5	Help/support	✔ / ❌
3.6	Warnings explain the risk	✔ / ❌
3.7	Operation log	✔ / ❌
3.8	Report export	✔ / ❌
3.9	App reset	✔ / ❌
3.10	Post-uninstall verification	✔ / ❌
3.11	Keychain entries (with opt-in)	✔ / ❌
3.12	Menu bar agents, input methods	✔ / ❌
3.13	Network Extensions/KEXT	✔ / ❌
3.14	Search/filter in results	✔ / ❌
3.15	Result sorting	✔ / ❌
3.16	Session saving	✔ / ❌
3.17	Drag & drop uninstallation	✔ / ❌
3.18	Help on error	✔ / ❌
3.19	No telemetry/third-party software	✔ / ❌

Total score calculation

Formula

Total = (passed L1 × 3) + (passed L2 × 2) + (passed L3 × 1)

Some criteria allow a partial result, half the full score for the criterion (0.5 × level weight).

Calculation example

Level	Passed	Partial	Failed	Score
L1 (×3)	8 of 9	1	0	(8 + 0.5) × 3 = 25.5
L2 (×2)	15 of 20	2	3	(15 + 1) × 2 = 32
L3 (×1)	12 of 19	0	7	12 × 1 = 12
TOTAL				**69.5 / 86**

Result: 69.5 ≥ 67 → ✔ Recommended (provided all L1 criteria are passed or partially passed, with no failures).

⚠ If at least L1 = 0 (failure) → ❌ Not recommended, regardless of the total score.

Final report template

App details

Description	Value
Uninstaller name	
Version	
Source (App Store/website)	
Price/monetization model	

Description	Value
Test date	
macOS version	
Hardware (processor, RAM, disk)	
QA	

Performance

Operation	Time	RAM (MB)	Crash	Baseline
Scanning a single app				< 5 excellent / < 15 acceptable
Removal (after confirmation)				< 5 excellent / < 15 acceptable
Scanning all apps				< 30 excellent / < 90 acceptable

Total score

```
Level 1: ___ / 9 x 3 = ___ / 27

Level 2: ___ / 20 x 2 = ___ / 40

Level 3: ___ / 19 x 1 = ___ / 19
___________________________

TOTAL: ___ / 86
```

Conclusion

```
Level 1 failure: YES / NO

If YES, which criterion: _______________

Blockers:
______________________________________________________________________

Key drawbacks:
______________________________________________________________________

Advantages:
______________________________________________________________________

RECOMMENDATION:
* [ ] Recommended (67–86, no L1 failures)
* [ ] With caveats (44–66)
* [ ] Not recommended (< 44 or L1 failure)
```

Evaluate the app yourself

Use the interactive calculator to quickly evaluate any uninstaller:

Open the evaluation calculator

The calculator:

- Contains all 48 criteria
- Calculates the score automatically
- Applies the L1 disqualification rule
- Saves progress in the browser
- Lets you export results as a text file

Download the full methodology

The full version of the methodology includes detailed instructions for each test, scripts for baseline/ground truth, and report templates.

Download QuickTest.zip

Frequently asked questions

Nektony develops its own uninstaller; isn't the methodology biased?

Yes, we develop App Cleaner & Uninstaller. That's exactly why the methodology is transparent; anyone can verify our results and evaluate our product by the same rules. We welcome independent verification.

Do I have to test all 9 application classes?

Yes, for a full evaluation, you need to test all 9 classes. But if you need a quick check, the minimum set of 4 tests (classes 1, 3, 5 + Trap A) covers the key L1 criteria. The result will be incomplete but sufficient for an initial assessment.

What if the uninstaller doesn't work with a certain class?

That's acceptable for L2/L3, just mark the criterion as failed. But if the missing feature leads to dangerous behavior (e.g., suggesting removal of a shared folder without checking), that's an L1 failure.

How often is the methodology updated?

The test application set rotates every 6 months. We update the methodology when new macOS versions are released (new extension types, SIP/TCC changes) or when coverage gaps are discovered. Current version: 1.0.5.

Can the methodology be used for commercial reviews?

Yes, the CC BY 4.0 license allows commercial use with attribution.

Disclaimer

This methodology is provided "as is" for reference only.

Limitations

- Test results may vary depending on the app version, macOS version, hardware configuration, and test apps used.
- The methodology is not exhaustive and does not cover every possible use case.
- Scores and recommendations are advisory in nature.

Testing risks

- The uninstallers you're testing may remove user files, damage the file system, or break macOS.
- A malfunctioning test tool can lead to irreversible data loss.
- Testing on a production machine with real data is risky. Use a dedicated test environment or a backup (Time Machine).

Liability

- Nektony shall not be liable for any damage resulting from testing under this methodology, including but not limited to: data loss, system damage, app or hardware malfunction.
- The user shall be solely liable for choosing their software, testing it, and for the consequences of using it.
- Nektony shall not be liable for data loss, deletion of needed files, system damage, or any other damage resulting from using uninstallers evaluated under this methodology.
- The methodology describes evaluation criteria, but does not guarantee the safety of any specific tool; even a high-scoring tool may contain bugs in specific versions or configurations.

- **Before you start testing, we strongly recommend creating a full system backup (Time Machine or equivalent).**

Conflict of interest

Nektony develops App Cleaner & Uninstaller, a tool in the category being tested. We publish the methodology transparently and apply it to our own products. Independent verification of results is welcome.

License

Questions and feedback

Found a mistake? Want to suggest an improvement? Have questions about testing?

Contact: support@nektony.com

Last updated: March 2026 (version 1.0.5)

Related materials

- Duplicate Finder Evaluation Calculator
- Comparison of Popular Duplicate Finder Apps for Mac (2026)
- Duplicate File Finder by Nektony