Methodology for evaluating macOS app cleaners/uninstallers

Version 1.0.6 | April 2026

Table of Contents

Executive summary

This methodology is an open standard for objectively evaluating tools that uninstall applications and their leftovers on macOS.

Key characteristics:

- 48 evaluation criteria across 3 priority levels

- Maximum score: 86
- Focus: removal safety and cleanup thoroughness
- 9 test application classes
- 4 testing phases with step-by-step instructions
- Trap scenarios (negative controls)

Scoring system:

- Level 1 (Must Have): 9 criteria × 3 = 27 points
- Level 2 (Should Have): 20 criteria × 2 = 40 points
- Level 3 (Nice to Have): 19 criteria × 1 = 19 points

Interpreting the result:

- 67–86 points: Recommended
- 44–66 points: With caveats
- < 44 points or Level 1 failure: Not recommended

Threshold logic:

- 67 = all L1 (27) + ~75% L2 (30) + ~50% L3 (10). A safe, accurate, functional tool with basic conveniences.
- 44 = all L1 (27) + ~40% L2 (17). A safe tool, but with noticeable gaps in functionality.

Testing time: 4–8 hours per tool (full cycle across all classes).

Introduction

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 and which one can remove needed user data or leave active system processes behind?

At the same time, 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, where other apps depend on that folder
- A broken launchd service: the binary is gone, but the plist remains
- A VPN profile keeps running after the "uninstalled" client is gone

99% of articles on uninstallers are just opinions. "I liked it," "nice interface." And it's unclear which app was uninstalled, on which macOS, what was treated as "leftovers."

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

Who this methodology is for

- **Journalists and reviewers:** for fair comparison reviews
- **QA engineers:** for complete product testing
- **Advanced users:** for informed decision
- **Developers:** for understanding quality standards

Authorship

This methodology was developed by the Nektony team, a company that has been building macOS utilities since 2011, including disk space analyzer, duplicate file finder, and uninstaller.

Over that time, we have:

- Analyzed the behavior of dozens of competing products
- Collected feedback from hundreds of thousands of users
- Identified recurring issues that lead to data loss or system damage
- Shaped the criteria that truly matter for safe uninstallation

We publish this methodology transparently in order to:

- Raise quality standards in the category
- Give users an independent evaluation tool
- Ensure transparency in our own testing

Contact: support@nektony.com

Purpose and scope

Goal: a unified standard for evaluating uninstallers on macOS.

Focus: removal safety and cleanup thoroughness.

Audience:

- QA engineers: full testing across all phases and criteria
- Reviewers/journalists: comparative analysis for publications
- Advanced users: evaluating a tool before purchase

User intent: Remove an application completely with its leftovers, without harming the system, so that the application is gone as if it was never here.

How to make sure the intent has been achieved:

- All service files of the application to be removed have been found
- Extensions, services, and settings installed by the application have been found
- Helper processes have been found and removed
- No extra files, system files, or files related to other apps have been removed
- Shared modules used by multiple apps have not been removed without a dependency check

Use cases

1. Removing a standard app completely (drag & drop .app)
2. Removing an App Store application (sandbox)
3. Removing an application installed via a pkg installer
4. Removing an application with helper processes (helpers, daemons)
5. Removing VPN/antivirus/firewall (system extensions and profiles)
6. Removing a "family" application (Adobe, Microsoft, JetBrains)
7. Finding and removing leftovers from a previously manually deleted app
8. Post-removal cleanup after using an application's native uninstaller

Out of scope

- Removing SIP-protected components
- Removing antivirus software completely (leftover post-removal cleanup is possible only)

Terms and definitions

Term	Definition
Application leftover	A file or process created by an application when installing or using it. Examples: files in ~/Library/Application Support, LaunchAgents,

Term	Definition
(artifact)	caches.
User data	Files created by the user WITH the application. Examples: documents in ~/Documents, exports, projects. Must not be removed without the user's explicit consent.
Gray zone (shared)	Elements shared by multiple apps or important settings. Examples: shared vendor folders, Keychain entries, browser extensions. They require an action by the user.
Evidence-based detection	Finding files based on a provable link: bundle ID, team ID, pkg receipts, standard system locations. The opposite of "search by folder name."
Bundle ID	A unique application identifier (com.vendor.appname). The primary way to link files to an application.
PKG Receipt	A system entry indicating what was installed by .pkg. Stored in /var/db/receipts/. Receipt provides an exact list of installed files.
pkgutil --forget	A command that removes ONLY the entry, but does NOT remove the files. If an uninstaller uses --forget and says "Removed successfully," that's not true.
Shared components	Files and folders used by multiple apps from the same vendor. Example: /Library/Application Support/Adobe/ is used by all Adobe products. Removing them without dependency check is a critical failure.
Dependency check	Verifying whether a file or folder is used by other apps before removing it.
Precision	Out of everything suggested for removal, how much actually relates to the app. The higher it is, the lower the risk of deleting unrelated files.

Term	Definition
Recall	Out of all real app artifacts, how many real artifacts were found. The higher it is, the fewer leftovers remain after removal.
Ground truth	The complete list of artifacts generated by an app when installing it. Obtained by comparing the system BEFORE and AFTER installation.

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

Risk model and priorities

Risk priority (from critical to low):

Data Loss > Security/Privacy > System Stability > Cleanliness

For example:

Deleting user needed data or documents > root helper left behind, a VPN profile still active > broken launchd services, stuck extensions > remaining caches, logs

Principle: One serious failure is worse than a dozen minor flaws.

What an uninstaller MUST remove (without prompting):

- App caches, logs, and preferences
- App LaunchAgents/Daemons (with proper unloading)
- App Application Support folder (if not shared)

What an uninstaller SHOULD OFFER with opt-in:

- Shared vendor folders (with a dependency warning)
- Keychain entries

- User data created by the application
- Browser extensions installed by the application

What an uninstaller MUST leave untouched:

- ~/Documents, ~/Desktop, ~/Downloads (user folders)
- Files related to other applications
- System components protected by SIP
- Shared frameworks that other applications depend on

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	Recommendation
67–86	Recommended
44–66	With caveats
< 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). A safe tool, but with noticeable gaps in functionality.

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

Level 1: Must Have (blockers)

Safety criteria. Failing any of them means a risk of data loss or system damage, which results in Not recommended tool.

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 from ~/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 of the uninstalled app in /Library/PrivilegedHelperTools and LaunchDaemons after removing an app.
1.5	No broken launchd services	If it removes plist, it'll remove binary, as well (and vice versa). Unloads the service correctly.
1.6	Correct handling of PKG receipts	Does not use pkgutil --forget without removing files. Receipt forgotten means files were actually deleted.

ID	Criterion	Description
1.7	Profiles/Extensions handled	Configuration Profiles, System/Network Extensions removed or the user is informed on how to remove them.
1.8	Confirmation before removal	Removal only after the user's explicit confirmation. Shows what exactly it will remove.
1.9	Stability	No crashes when using the tool. Handles permission errors correctly.

Level 2: Should Have (important)

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

ID	Criterion	Description
2.1	Recall ≥ 90% of standard paths	Finds files in Application Support, Preferences, Caches, Containers, Group Containers, Logs.
2.2	Finds LaunchAgents/Daemons	Finds and shows user-level and system-level agents/daemons.
2.3	Finds PrivilegedHelperTools	Finds root helpers in /Library/PrivilegedHelperTools.
2.4	Finds PKG receipts and components	Uses pkgutil to find files installed via .pkg.
2.5	Finds browser extensions	Finds Safari, Chrome extensions installed by an app.

ID	Criterion	Description
2.6	Finds Login Items/Background Items	Finds startup elements.
2.7	Stops processes before removal	Unload/stop for launchd before deleting files.
2.8	File list preview	Shows a full list of files before removal. The user sees what will be removed.
2.9	Move to Trash by default	Removes to Trash, not permanently. There is an option to restore files.
2.10	Undo/restore	Option to reverse the removal or recover the removed files.
2.11	Works with sandbox apps	Correctly shows and removes ~/Library/Containers and Group Containers for App Store apps.
2.12	Finds leftovers without .app	Finds artifacts of manually removed apps (by dragging .app to Trash).
2.13	Access error handling	Correctly handles: no rights, file blocked, SIP-protected path. Keeps working.
2.14	Shows what it can't remove	Shows elements, where user password or action is required. Explains why and what to do.
2.15	Evidence-based explanations	Shows why the file is linked to the app (bundle id, receipt, path).
2.16	Separates app data/user data	A specific toggle or section for user data. Safe defaults (user data NOT pre-selected).
2.17	Distinguishes bundle IDs	Does not confuse apps with the same name, but different bundle ID.

ID	Criterion	Description
2.18	Scan speed	Scans a single app within reasonable time.
2.19	Monetization transparency	Price is visible before purchase. Free version limitations are clear beforehand. No paywall trap.
2.20	Finds symlinks, aliases, wrappers	Removes links in /usr/local/bin, Homebrew paths, aliases created by an app.

Level 3: Nice to Have (bonuses)

Extra features. 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 support and keyboard navigation
3.3	Localization	Supports 5+ interface languages
3.4	Onboarding	First-launch tutorial
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	Removed files history. It's possible to see previously deleted items.

ID	Criterion	Description
3.8	Report export	Exports results to CSV, PDF, or text
3.9	App reset	Option to reset an app to default settings 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 app's Keychain entries, suggests removing them with explicit choice
3.12	Menu bar agents, input methods	Finds and removes integration panels from System Settings, menu bar agents, and input methods
3.13	Network Extensions/KEXT	Gives removal instructions if can't remove automatically
3.14	Search/filter in results	Search or filter field in the list of found files
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 external servers without the user's knowledge. Does not install third-party software.

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 risky elements for removal (shared folders, user data)*
- *Root helper left behind after removal*
- *Removes without dependency check*

Transparency

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

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 without installer	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 + helper processes	LaunchDaemons, PrivilegedHelperTools, service unloading
5	With LaunchAgent/Daemon	Startup	Service unloading, removing plist + binary as a pair
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.

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 (by dragging .app to Trash)

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

Minimum set for testing

- One app from each class
- For a basic L1 check: a short set: classes 1, 3, 5 + Trap A
- For a full L1 check: classes 1, 3, 5, 6, 7, 8 (for class 8: two apps from the same vendor)
- For L2: at least one of each class (9 apps)
- Plus trap scenarios (see section below)

Recommendations for selecting test apps

- Choose real, widely used applications

- For class 8 ("family"): install at least 2 products from the same vendor and uninstall only one
- For class 9: manually remove the .app by moving it to Trash, then run the uninstaller to search for leftovers

---

QuickTest: express check (~1 hour)

Goal: a quick first-pass filter. If an uninstaller fails QuickTest, there's no need to run the full test (4–8 hours).

QuickTest does not replace the full test. It neither covers all application classes nor produces a total score. It checks the L1 Safety Gates using synthetic apps with a precisely known ground truth.

How it works

The `quicktest-setup.sh` script creates a synthetic test environment in ~30 seconds:

- 4 synthetic apps with unique bundle IDs:
  - Alpha: `com.methodology-alpha.testcleaners-alpha`
  - Beta: `com.methodology-beta.testcleaners-beta`
  - Gamma: `com.methodology.testcleaners-gamma`
  - Delta: `com.methodology.testcleaners-delta`

  Gamma and Delta have a common prefix `com.methodology.*` (family imitation)
- All artifact types: Application Support, Preferences, Caches, Containers, LaunchAgent, LaunchDaemon, Privileged Helper, PKG receipt, Configuration Profile, Keychain, symlinks
- Shared vendor folder (two apps from the same vendor)
- Traps A–E: user files with similar but non-matching names in ~/Documents, ~/Downloads, ~/Desktop

- Accurate manifest of all created artifacts

The QA shall uninstall 3 apps via the uninstaller, then `quicktest-verify.sh` automatically checks:

- What was found and deleted (True Positives)
- What was missed (False Negatives → Recall)
- What was deleted by mistake: traps, Delta, vendor (False Positives → Precision)
- Launchd, PKG receipts, profile status
- Keychain – info only (not included in the metrics)
- Verdict on the L1 Safety Gates

QuickTest coverage

Class	Synthetic app	What it tests
1	Alpha App	App Support, Prefs, Caches, Containers, Keychain
3+5	BetaUtil (PKG + LaunchAgent/Daemon + Helper)	PKG receipt, LaunchAgent, LaunchDaemon, Helper, Symlink, Configuration Profile
8	FamilyOne + FamilyTwo (family)	Shared vendor folder (one deleted, two remaining untouched)
Trap E	AlphaApp Tool (different bundle ID)	Bundle ID distinction

L1 coverage: 1.1 ✔ 1.2 ✔ 1.3 ✔ 1.4 ✔ 1.5 ✔ 1.6 ✔ 1.7 ✔ 1.8 (manual) 1.9 (manual)

Files:

- quicktest-setup.sh (setting up)
- quicktest-verify.sh (checking)
- quicktest-teardown.sh (cleanup)

Steps

1. `chmod +x quicktest-*.sh`
2. `sudo ./quicktest-setup.sh` (without sudo: user-level artifacts only)
3. If a `.mobileconfig` was created, install the configuration profile manually:
   `open ~/uninstall-audit/BetaUtil-Profile.mobileconfig`
   → System Settings → General → Device Management → Install
4. Run the uninstaller you're testing.
5. Uninstall: AlphaApp, BetaUtil, FamilyOne.
   Do NOT uninstall: FamilyTwo, "AlphaApp Tool" (Trap E)
6. `sudo ./quicktest-verify.sh`
7. Clean up: `sudo ./quicktest-teardown.sh`

Interpreting results

- **L1 PASS** → the app is ready for the full test
- **L1 FAIL** → skip the full test, record the failures

Test environment and reproducibility

Test environment requirements

- Physical Mac (not a virtual machine, as it skews results)
- Latest macOS
- A clean user account or a fresh session
- SIP enabled (standard mode)
- Capture: macOS version, hardware (processor, RAM, disk), test date

- Test an uninstaller with default settings. Do not change settings between tests of different applications

Measurement reproducibility

- Speed: 3 runs, the median to be captured
- Cold start before each run: `sudo purge && sleep 30`
- Dispersion: (max – min)/median × 100%, threshold ≤ 15%
- If dispersion > 15%: identify the cause (background processes, indexing), resolve, and repeat
- Precision/Recall: should be identical across runs. If results differ, get them captured and explained

Baseline → ground truth approach

Stage	What we do
BASELINE	Capture the system state BEFORE installing the test application
INSTALL	Install the test application, launch it, grant all requested permissions
GROUND TRUTH	Capture what appeared AFTER installation. The difference between baseline and after = the artifact reference set
UNINSTALL	Run the uninstaller being tested
VERIFY	Check: what remains, what was removed by mistake

Key monitoring zones (what we capture in baseline and ground truth):

Zone	Path
Autostart (user)	~/Library/LaunchAgents/

Zone	Path
Autostart (system)	/Library/LaunchAgents/
Daemons	/Library/LaunchDaemons/
Privileged Helpers	/Library/PrivilegedHelperTools/
Application Support (user)	~/Library/Application Support/
Application Support (system)	/Library/Application Support/
Preferences	~/Library/Preferences/
Caches	~/Library/Caches/
Containers	~/Library/Containers/
Group Containers	~/Library/Group Containers/
Logs	~/Library/Logs/
PKG Receipts	pkgutil --pkgs (list)
Applications	/Applications/, ~/Applications/

Baseline script (run BEFORE installing the test application)

```
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"
ls -1 ~/Library/Logs 2>/dev/null | sort > "$OUT/logs.txt"
ls -1 /Applications 2>/dev/null | sort > "$OUT/apps_system.txt"
ls -1 ~/Applications 2>/dev/null | sort > "$OUT/apps_user.txt"
```

Ground truth script (run AFTER installation, before uninstall)

```
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 >
"$OUT/group_containers.txt"
ls -1 ~/Library/Logs 2>/dev/null | sort > "$OUT/logs.txt"
ls -1 /Applications 2>/dev/null | sort > "$OUT/apps_system.txt"
ls -1 ~/Applications 2>/dev/null | sort > "$OUT/apps_user.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"
comm -13 "$BASE/logs.txt" "$AFTER/logs.txt" > "$GT/new_logs.txt"
comm -13 "$BASE/apps_system.txt" "$AFTER/apps_system.txt" >
"$GT/new_apps_system.txt"
comm -13 "$BASE/apps_user.txt" "$AFTER/apps_user.txt" >
"$GT/new_apps_user.txt"
```

Additionally, by time marker (covers 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"
```

Ground truth means a combination of:

- Set-diff of lists (what appeared after installation)
- Changed-after-marker (what was created/modified after installation)

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

Step 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.

Step 0.2 Prepare the uninstaller

1. Gather 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.

Note: An uninstaller gets Full Disk Access and root privileges. An unsigned tool with that level of access is an increased risk. If a tool is unsigned, note it in the report.

3. Install the uninstaller. During installation: does it install third-party software? Yes/No. Does it show ads during installation? Yes/No.
4. Grant required permissions (Full Disk Access).
5. Launch it, go through the onboarding (if any).

Step 0.3 Prepare test apps

1. Select at least one application from each class (see the table).
2. For each test application: take a baseline screenshot (script above), install the test app, launch it, grant all permissions, take a ground truth screenshot (script above).
3. Prepare trap scenarios (see section below).

PHASE 1

Level 1 (Safety gates) criteria check

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

Test 1.1 Precision = 100%

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

How to check:

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, contents)?
4. Compare what's suggested against ground truth.
5. Check traps A, B, C (see Trap scenarios section).
6. Repeat for classes 3, 5, 8.

Result:

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

Test 1.2 No user data removal without opt-in

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

How to check:

1. Use Trap A: create ~/Documents/[AppName]/[some-words] and ~/Downloads/[AppName]-[some-words] with pdf, jpg, and DO_NOT_DELETE.txt.pdf files.
2. Proceed with uninstalling this app.
3. Check: does the uninstaller suggest removing this folder?
4. If it does, is it pre-selected by default?

Result:

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

Test 1.3 Correct handling of shared folders

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

How to check:

1. Install 2 apps from the same vendor (Adobe, Microsoft, JetBrains).
2. Remove only one via the uninstaller.
3. Check: does it suggest removing the shared vendor folder?

Result:

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

Test 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.

How to check:

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

Result:

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

Test 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 nonexisting process.

How to check:

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

Result:

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

Test 1.6 Correct handling of PKG receipts

Why it matters: Using `pkgutil --forget` without removing files is not true. The UI says "removed successfully," but the files are still sitting in /Library.

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?

Result:

✔ Files removed and receipt forgotten
✔ Files removed, receipt not forgotten (acceptable)
❌ FAIL: Receipt forgotten, but files remain. The UI says "removed successfully."

Test 1.7 Profiles/Extensions handled

Why it matters: A VPN profile or network extension that keeps running after the client is "uninstalled" is a real privacy and security problem.

How to check:

1. Pick a class 7 app (VPN or Firewall).

2. Before removal, check: System Settings → VPN & Network and System Settings → General → Login Items & Extensions.
3. Remove via the uninstaller.
4. Check: are profiles and extensions removed? Or is a warning shown with instructions?

Result:

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

Test 1.8 Confirmation before removal

Why it matters: An accidental click should not trigger removal.

How to check:

1. Pick any app for removal.
2. Click the remove button.
3. Does a confirmation dialog appear?
4. Does it show the details: number of files, total size, and where they'll be deleted?

Result:

✔ Shows a confirmation dialog with details
❌ FAIL: Removes immediately without confirmation

Test 1.9 Stability

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

How to check:

1. Uninstall apps from each tested class, one after another.
2. Check whether there is any crash or freeze (beachball).

3. Try removing an app without Full Disk Access: revoke FDA, relaunch the uninstaller, and try uninstalling.

Result:

✔ Runs stable; handles errors correctly
❌ FAIL: Crash, freeze, or incorrect behavior when permissions are lacking

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	✔/❌	
Total L1: ___ / 9. Is there at least one ❌? → "Not recommended"			

PHASE 2

Level 2 (Should Have) criteria check

Goal: Check functionality and coverage.

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.

How to check:

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.

Result:

✔ 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 keep running and consuming resources.

How to check:

1. Use a class 5 app (with LaunchAgent/Daemon).
2. Before removal, take a screenshot of the plist and binary paths.
3. Run the uninstaller.
4. Check: does it show both user-level (~/Library/LaunchAgents) and system-level (/Library/LaunchAgents, /Library/LaunchDaemons) agents?

Result:

✔ Shows both levels | ❌ Does not find agents/daemons

2.3 Finds PrivilegedHelperTools

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

How to check:

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

Result:

✔ Finds and shows helpers | ❌ Does not find them

2.4 Finds PKG receipts and components

Why it matters: Apps installed via .pkg scatter files all across /Library. Without pkgutil, those files are invisible for searching by name.

How to check:

1. Use a class 3 or 4 app (PKG installer).
2. Before removal: `pkgutil --pkgs | grep [vendor]` and `pkgutil --files [package-id]`.
3. Run the uninstaller.
4. Check: does it find files from the receipt?

Result:

✔ Finds files from the receipt | ❌ Does not use receipts; misses files from /Library

2.5 Finds browser extensions

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

How to check:

1. Install an app that adds a Safari or Chrome extension.
2. Run the uninstaller.
3. Check: does it show the extension in the list of found items?

Result:

✔ 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.

How to check:

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

Result:

✔ 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.

How to check:

1. Uninstall a class 5 app (with LaunchAgent).
2. Before removal: `launchctl list | grep [bundle_id]`
3. Watch the uninstaller: does it unload/stop the processes before deleting files?
4. After removal: `launchctl list | grep [bundle_id]`, should return nothing.

Result:

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

2.8 File list preview

Why it matters: You should be able to see the full list of files BEFORE anything is removed and uncheck whatever you want to keep.

How to check:

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

Result:

✔ Full list with selection options | ❌ No list, or the selection can't be changed

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.

How to check:

1. Uninstall an app via the uninstaller.
2. Open Trash, are the files there?
3. If the tool removes permanently, is there a choice between "Trash" and "Delete permanently"?

Result:

✔ Trash by default, with a choice | ❌ Deletes permanently without a recovery option

2.10 Undo/restore

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

How to check:

1. Uninstall an app.

2. Can you undo or restore files (not from Trash, but in the interface)?
3. If so, does restoring actually work?

Result:

✔ Undo/restore is available | ❌ No way to undo

2.11 Works with sandbox apps

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

How to check:

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

Result:

✔ Finds Containers and Group Containers | ❌ Does not find them, or shows the wrong items

2.12 Finds leftovers without .app

Why it matters: The user may have already dragged the .app to Trash manually, leaving the Library files behind. A good uninstaller should be able to track those down even without the app present.

How to check:

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

Result:

✔ 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 reach certain files. It should make the situation clear, not crash or silently skip.

How to check:

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

Result:

✔ Clear message; keeps running | ❌ 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.

How to check:

1. Make sure the test app has files that require a password or manual action.
2. Does the uninstaller display those items separately?
3. Does it explain why it can't remove them and what to do instead?

Result:

✔ Shows them 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.

How to check:

1. Open the list of files to be removed.

2. Can you see a reason WHY each file is linked to the app (bundle ID, pkg receipt, or path)?

Result:

✔ Explains the linkage (bundle ID, receipt, path)
⚠ Partially: shows the path, but without an explanation (half score)
❌ 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 keep these two categories apart.

How to check:

1. Select an app that generates user data.
2. Is there a separate section or toggle for user data?
3. Is user data NOT pre-selected for removal by default?

Result:

✔ Separated; safe defaults, or user data not pre-selected | ❌ Everything in one list, no differentiation

2.17 Distinguishes bundle IDs

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

How to check:

1. Use Trap E: install two apps with the same name but different bundle IDs.
2. Uninstall one of them via the uninstaller.
3. Does it leave the other app's files untouched?

Result:

✔ Distinguishes correctly by bundle ID | ❌ Confuses the apps

2.18 Scan speed

Why it matters: Nobody wants to wait a minute per app. A scan should take seconds, not minutes.

How to check:

1. Cold start: `sudo purge && sleep 30`
2. Run a scan of a single app.
3. Time how long it takes from start to results appearing.
4. Repeat steps 1–3 twice more (3 runs total).
5. Note the median. Dispersion: (max – min)/median ≤ 15%.

Result:

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

2.19 Monetization transparency

Why it matters: The "scan for free, pay for removal" trap is a well-known trick. You should know about the limitations BEFORE the tool spends your time scanning.

How to check:

1. On first launch: are the free version's limitations clear?
2. Is the price visible before purchase?
3. There isn't a "scan for free, pay for removal" trap, is there?

Result:

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

2.20 Finds symlinks, aliases, wrappers

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

How to check:

1. Use an app that creates CLI tools or symlinks in /usr/local/bin or Homebrew paths.
2. Run the uninstaller.
3. Does it find these links?

Result:

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

Level 2 summary

#	Criterion	Result	Comment
2.1	Recall ≥ 90%	✔/❌	
2.2	LaunchAgents/Daemons	✔/❌	
2.3	PrivilegedHelperTools	✔/❌	
2.4	PKG receipts and components	✔/❌	
2.5	Browser extensions	✔/❌	
2.6	Login Items/Background Items	✔/❌	
2.7	Stops processes	✔/❌	
2.8	File list preview	✔/❌	
2.9	Move to Trash	✔/❌	
2.10	Undo/restore	✔/❌	
2.11	Sandbox apps	✔/❌	
2.12	Leftovers without .app	✔/❌	

#	Criterion	Result	Comment
2.13	Access error handling	✔/❌	
2.14	Shows what it can't remove	✔/❌	
2.15	Evidence-based explanations	✔/⚠/❌	
2.16	App data/user data	✔/❌	
2.17	Distinguishes bundle IDs	✔/❌	
2.18	Scan speed	✔/⚠/❌	
2.19	Monetization transparency	✔/❌	
2.20	Symlinks, aliases, wrappers	✔/❌	
Total L2: ___ / 20			

PHASE 3

Level 3 (Nice to Have) criteria check

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

3.1 Dark mode

How to check:

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

Result:

✔ Supports the system theme | ❌ No support

3.2 VoiceOver/accessibility

How to check:

1. Turn on VoiceOver (Cmd+F5).
2. Try navigating: the menu, the app list, and the remove button.
3. Are all elements read aloud correctly?

Result:

✔ Basic navigation works | ❌ VoiceOver doesn't work or reads elements incorrectly

3.3 Localization (5+ languages)

How to check:

1. System Settings → General → Language & Region.
2. Switch the system language.
3. Does the uninstaller support 5 or more languages?

Result:

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

3.4 Onboarding

How to check:

1. On first launch: are there tutorial screens?
2. Do they walk you through how to use the tool and which permissions it needs?

Result:

✔ There is a clear onboarding | ❌ No tutorial

3.5 Help/support

How to check:

1. Open Help menu or find the support button.

2. Is there built-in help, FAQ, or a link to support?

Result:

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

3.6 Warnings explain the risk

How to check:

1. When removing risky items (shared folder, system extension), does the warning describe what could go wrong?
2. Does it recommend a specific action?

Result:

✔ Explains the risk and recommends an action | ❌ Warning with no explanation, or no warning at all

3.7 Operation log

How to check:

1. Uninstall a few applications.
2. Check: is there a removal history?
3. Can you see what was removed yesterday/last week?

Result:

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

3.8 Report export

How to check:

1. Run a scan or uninstall an app.
2. Can you export the results to a file (CSV, PDF, or text)?

Result:

✔ Export is available | ❌ No export

3.9 App reset

How to check:

1. Is there a "reset to default settings" feature?
2. Does it remove preferences and caches without uninstalling the app itself?

Result:

✔ Reset feature is available | ❌ No

3.10 Post-uninstall verification

How to check:

1. Uninstall an app via the uninstaller.
2. Does the uninstaller check what's left behind?
3. Does it warn about critical leftovers (helpers, daemons)?

Result:

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

3.11 Keychain entries (with opt-in)

How to check:

1. Uninstall an app that created Keychain entries.
2. Does the uninstaller find Keychain entries linked to the app?
3. Does it offer to remove them with an explicit choice (opt-in)?

Result:

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

3.12 Menu bar agents, input methods

How to check:

1. If the app installs a menu bar agent, a System Settings integration panel, does the uninstaller find those components?

Result:

✔ The tool finds them | ❌ The tool misses them

3.13 Network Extensions/KEXT

How to check:

1. For apps with network extensions (VPN, firewall) or KEXT, does the uninstaller show these components?
2. If it can't remove them automatically, does it provide removal instructions?

Result:

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

3.14 Search/filter in results

How to check:

1. Run a scan.
2. Is there a search or filter field in the list of found files?

Result:

✔ There is a search/filter | ❌ No

3.15 Result sorting

How to check:

1. Run a scan.
2. Can you sort the found files by size, type, and location?

Result:

✔ Sorting is available | ❌ No

3.16 Session saving

How to check:

1. Run a scan.
2. Close the uninstaller.
3. Reopen it. Are the scan results still there?

Result:

✔ Results are saved | ❌ Lost on close

3.17 Drag & drop uninstallation

How to check:

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

Result:

✔ Drag & drop works | ❌ Not supported

3.18 Help on error

How to check:

1. Trigger an error (no permissions, file locked).
2. Does the uninstaller show actionable advice on how to fix it?

Result:

✔ Shows advice | ❌ Just an error message with no help

3.19 No telemetry without consent or third-party software

How to check:

1. Install a network monitor (Little Snitch, FireWally, or equivalent).

2. Launch the uninstaller and run a typical scenario (scan, removal).
3. Check what connections the app makes: does it send data to external servers (other than update checks)?
4. Think back to the installation step (Phase 0, Step 0.2): did it install any third-party software (bundleware, adware)?

Result:

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

Level 3 summary

#	Criterion	Result	Note
3.1	Dark mode	✔/❌	
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)	✔/❌	

#	Criterion	Result	Note
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 L3: ___ / 19			

Trap scenarios (negative controls)

Traps verify that an uninstaller does NOT do dangerous things. Set them up BEFORE the main tests (the results to be used in tests 1.1 and 1.2).

Trap A: "Naive name search"

Preparation:

1. Pick a test app, e.g. FooApp.
2. Create these folders:
   - ~/Documents/FooApp/
   - ~/Desktop/FooApp Export/

3. Put some files in there: test.pdf, photo.jpg, DO_NOT_DELETE.txt.

Test:

1. Uninstall FooApp via the uninstaller.
2. Check the list of suggested files.

Evaluation:

- **Found:** did it flag ~/Documents/FooApp as a "leftover"?
- **Default selection:** was the folder pre-selected for removal?
- **Delete behavior:** did it remove without a final warning?

Fail if: it shows ~/Documents as leftovers AND pre-selects it by default.

Trap B: "Vendor name"

Preparation:

1. Create ~/Documents/[VendorName]/ (the name of the vendor whose app you're about to remove).
2. Fill it with files.

Test: Uninstall that vendor's app.

Fail if: the uninstaller treats the vendor-named folder in Documents as junk to be removed.

Trap C: "Exports and backups"

Preparation:

1. Create the following files in ~/Documents:
    - FooApp-export-2026-02-09.zip
    - FooApp-backup.json
    - FooApp-report.pdf

Test: Uninstall FooApp.

Fail if: artifacts are pre-selected for removal by default.

Trap D: "Shared folder"

Preparation:

1. Install App A and App B from the same vendor (Adobe Photoshop + Illustrator, or equivalent).
2. Both share a common folder: /Library/Application Support/[Vendor]/.

Test: Uninstall only App A via the uninstaller.

Pass: The vendor folder is NOT removed (or only the app-specific subfolder is).

Fail: It offers to remove the entire vendor folder without dependency check.

Trap E: "Same name, different apps"

Preparation:

1. Install two apps with the same or similar name but different bundle IDs.

Test: Uninstall one of them. Check: does it affect the files of the second app?

Fail if: it suggests files from the second app as "leftovers" of the first one.

Metrics and how to calculate them

Core metrics

Metric	Formula	What it measures
Precision	TP / (TP + FP)	Of everything suggested for removal, how many actually relate to the app
Recall	TP / (TP + FN)	Of all the real app artifacts, how many were actually 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

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

Target values: Precision = 100% (required for L1). Recall ≥ 90% (required for L2).

3 numbers per test

1. **Suggested for removal:** what the uninstaller showed in its list (this is where false positives are caught)
2. **Actually removed:** what was actually removed after confirmation (compare the system state before and after)
3. **Collateral damage:** what was removed by mistake, and what got broken (check other apps, launchd services, and system settings)

Penalties

Type	Description	Effect
Critical Fail	Safety gate (L1) failure	Disqualification
Major	False positive on a shared component	Major drawback

Type	Description	Effect
Minor	Missed cache or log	Minor drawback

Performance baseline

For Apple Silicon SSD:

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

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

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

Risks and common pitfalls

Read this before you start testing!

Dangerous scenarios

Scenario	Risk
Removing a shared vendor folder	Breaks other apps from the same vendor

Scenario	Risk
Root helper left behind	A process with root privileges keeps running without its app
Broken launchd service	The system endlessly tries to launch a non-existent process
pkgutil --forget without removing files	The UI claims "removed," but the files are still there
Configuration Profile after removal	Settings (VPN, Wi-Fi, proxy) keep applying
Removing Keychain entries by name	May affect entries related to other apps

False results

Case	Problem
Searching "by folder name"	Flags ~/Documents/AppName as a leftover
Ignoring shared components	Treats a vendor folder as junk related to one app
Not using pkg receipts	Misses files installed to /Library via .pkg
Not distinguishing bundle IDs	Confuses apps with the same name

Common testing mistakes

- Not capturing a baseline before installing the test app
- Not checking the ground truth (not knowing what the app actually put on disk)
- Testing only on simple .app (class 1) doesn't cover pkg, helpers, and extensions
- Not checking whether other apps still work after the uninstallation

- Comparing the App Store and non-App Store versions of the uninstaller without taking into account the Sandbox limitations

---

Templates

Criteria results table template

ID	Criterion	Level	Weight	Result	Score	Test
1.1	Precision = 100%	1	3		___	Phase 1
1.2	No user data removal	1	3		___	Phase 1
…	…	…	…	…	…	…
2.1	Recall ≥ 90%	2	2		___	Phase 2
…	…	…	…	…	…	…
3.1	Dark mode	3	1		___	Phase 3
…	…	…	…	…	…	…

Application class results table template

Class	Application	Recall	Precision	FP	FN	Note
1. Drag & Drop	_____	___%	___%	___	___	_____
2. Sandbox	_____	___%	___%	___	___	_____
3. PKG	_____	___%	___%	___	___	_____

Class	Application	Recall	Precision	FP	FN	Note
…	…	…	…	…	…	…

Final report template

TEST INFORMATION

```
Uninstaller: _____________
Version: _____________
Source (App Store/website): _____________
Price/monetization model: _____________
Test date: _____________
macOS version: _____________
Hardware: _____________
QA: _____________
```

RESULTS

```
Level 1: ___ / 9 x 3 = ___
Level 2: ___ / 20 x 2 = ___
Level 3: ___ / 19 x 1 = ___

_________________________________
TOTAL:                 ___ / 86
```

Signed and notarized: Yes / No (Phase 0)

CONCLUSION

```
Level 1 failure: YES / NO
```

```
If YES, which criterion: _____________
```

RECOMMENDATION

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

```
NOTES
Blockers: _____________
Key drawbacks: _____________
Advantages: _____________
```

Versioning and updates

Current version: 1.0.6 | April 2026

Changelog

- **v1.0.6 (April 2026):** interim result ❗ added for criterion 2.15; minimum set for a full L1 check clarified; baseline scripts extended with /Applications zones
- **v1.0.5 (March 2026):** version synchronization between the methodology and the article; "Configuration Profile" replaces "VPN profile" in dangerous scenarios;
- **v1.0.4 (March 2026):** see CHANGELOG.md
- **v1.0.3 (March 2026):** copy-paste-ready scripts, placeholder table with examples, expanded L2 sections, QuickTest, fix for pre-code styles

- **v1.0.2 (March 2026):** expanded Liability block; synchronization between methodology and article
- **v1.0.1 (March 2026):** strengthened disclaimer: "Testing risks" block, warning callout before the testing procedure
- **v1.0 (February 2026):** first public release

Update principles

1. **Test application set rotation.** The set of test apps is refreshed every 6 months. There are two sets: a public demo set (for self-evaluation) and a closed rotatable set (for independent reviews).
2. **Class-based testing, not name-based.** The methodology defines 9 application classes. Specific test apps are picked from each class but are never left permanently.
3. **Synthetic tests.** Apps with randomly generated bundle IDs are also used — ones that can't be known in advance.
4. **Updates for new macOS versions.** When a new version of macOS is released, the following are reviewed: new extension types and restrictions, changes to SIP/TCC, new system paths and mechanisms. The methodology is updated as needed.
5. **Backward compatibility.** Test results are tied to the methodology version. Results are only comparable within the same version.

Disclaimer

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

Limitations

- Results may vary depending on the uninstaller 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 delete 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

This methodology is distributed under the **Creative Commons Attribution 4.0 (CC BY 4.0)** license.

You are free to:

- Copy and distribute
- Adapt and create derivative works
- Use it for commercial purposes

Provided that you give attribution:

"Methodology for evaluating macOS app uninstallers" © Nektony, 2026

Contact: support@nektony.com