macOS Persistence Cheatsheet

v1.0SO4/2026

Filter by Mechanism: Show all | Hide Sandbox | Hide Root | Hide Persistence | Hide Hijacking | Hide Evasion | Hide Recon | Hide Execution | Hide Stealth

Filter by Trigger: Show all | Boot | Login | Schedule | Event | Hijacking | Persistence | Evasion | Stealth

Filter by Scope: Show all | Global | Sandbox | Root | Persistence

Filter by Signal: Show all | Hijacking | Persistence | Evasion | Stealth | Recon | Execution

Legend: Boot · Login · Schedule · Event · Hijacking · Persistence · Evasion · Recon · Execution · Stealth

Mechanism

This cheat sheet lists persistent Mach-O payload types, macOS, as the primary focus is on both the latest.

Core Launch & Login Persistence

Launch Agents — Boot · Root · Persistence
Boot · Root · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
launchd · launchctl · .plist · launchctl bootstrap

Launch Daemons — Boot · Root · Persistence
Boot · Root · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
Requires root · Runs before any user login · Survives reboots · Runs as root

BTM / SMAppService — Boot · Persistence
Boot · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Sandbox · Persistence
Modern · Sandbox-aware · Managed by launchd · Visible in System Settings

Login Items (legacy / classic) — Login · Persistence
Login · Persistence · Hijacking · Evasion
Sandbox · Persistence
Deprecated · Still widely used · Visible to user · Runs in GUI session

Embedded Login Helpers — Login · Persistence
Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Sandbox · Persistence
Hidden · Sandbox-aware · Managed by parent app · Invisible to user

Shell Startup Files — Boot · Persistence
Boot · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
.zshrc · .zprofile · .zshenv · .bash_profile · .bashrc · .profile · /etc/zshrc · /etc/zprofile · /etc/profile · /etc/bashrc
Runs on every shell · Persists across reboots · Affects all terminal sessions

Shell and Quasi (.rc) — Boot · Persistence
Boot · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
.vimrc · .gdbinit · .lldbinit · .inputrc · .curlrc · .wgetrc · .gitconfig · .gitconfig.local · .tmux.conf · .tmux.conf.local · .screenrc · .dir-locals.el · .mkshrc · .ksh · .tcshrc · .cshrc · .hushlogin
Runs on every shell · Persists across reboots · Affects all terminal sessions

Kernel Extensions (KEXT) — Root · Persistence
Root · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
Requires root · Runs before any user login · Survives reboots · Runs as root

Kernel Extensions (KEXT) — Root · Persistence
Root · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
Requires root · Runs before any user login · Survives reboots · Runs as root

Cron — Schedule · Persistence
Schedule · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
crontab · /etc/crontab · /usr/lib/cron/tabs/ · /var/cron/tabs/
Runs on a schedule · Survives reboots · Runs as root

At / atrun — Schedule · Persistence
Schedule · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
Requires atrun daemon enabled · Can run once or recurrently

Periodic Jobs — Schedule · Persistence
Schedule · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
/etc/periodic/daily · /etc/periodic/weekly · /etc/periodic/monthly · /usr/local/etc/periodic · /opt/local/etc/periodic · /etc/periodic.conf · /usr/local/etc/periodic.conf · /opt/local/etc/periodic.conf
Runs on a schedule · Survives reboots · Runs as root

Emond — Event · Persistence
Event · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
Deprecated · Rarely used · Still supported · Runs as root

Extensions, Browser, Interpreter, and Handler Surfaces

System Extensions — Boot · Persistence
Boot · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
Modern · Managed by sysextd · Survives reboots · Runs as root

KEXTs — Boot · Persistence
Boot · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
Deprecated · Requires root · Still supported · Runs before any user login

Chromium Extensions — Boot · Persistence
Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
Manifest V3 · Sandbox · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence
Manifest V2 · Deprecated · Still supported · Persistence · Hijacking · Evasion · Stealth · Recon · Execution
Global · Sandbox · Root · Persistence

Core Launch & Login Persistence

Mechanism	Required privilege	Scope of Scope	Signal	Collection / Trigger	Trigger	What to review
Launch Agents Plist · launchd Core Launch	Root	Global	Persist	**Artifact Path** /Library/LaunchAgents ~/Library/LaunchAgents /System/Library/LaunchAgents **Mechanism** Program / ProgramArguments MachServices/Sockets / Sockets / QueueDirectories **Evasion** launchctl print gui/<uid> find ~/Library/LaunchAgents -newer /var/db/.AppleSetupDone -name "*.plist" **Impact** launchctl print gui/<uid>/label launchctl bootout gui/<uid> <path> **Legend** launchctl print gui/<uid>/label launchctl bootout gui/<uid> <path> plutil -p <plist>	Boot Login Persistence Evasion	Watch ~/Library/LaunchAgents. Parse Label, Program, ProgramArguments, MachServices, QueueDirectories, Sockets, RunAtLoad, KeepAlive, StartInterval, StartCalendarInterval, WatchPaths, PathState, and ThrottleInterval. Use launchctl blame to identify the actual trigger. Flag unknown scripts in the downgraded paths, or anything with RunAtLoad + KeepAlive, which restarts for hunting on demand. Compare against a known-good baseline. Alert on unknown LaunchAgents and any persistence on disk but absent in user launchctl print. Flag binaries in /tmp, /Users/Shared, or hidden paths. Cross-reference against ~/Library, system-scoped ~/Library, system-scoped /Library.
Launch Daemons Plist System · Windows Core Launch	Root	Root	Persist	**Artifact Path** /Library/LaunchDaemons ~/Library/LaunchDaemons **Mechanism** Program / ProgramArguments MachServices endpoints and loaded binary path **Evasion** launchctl print system find /Library/LaunchDaemons -newerBt "1 week ago" -name "*.plist" **Impact** launchctl print system/<label> launchctl bootout system/<label> plutil -p <plist> **Legend** launchctl print system/<label> launchctl bootout system/<label> plutil -p <plist>	Boot pre-login	Watch /Library/LaunchDaemons. Very high-value persistence. Review Program, ProgramArguments, MachServices, QueueDirectories, and Sockets. Use launchctl blame, see the actual trigger. Flag unknown scripts in the downgraded paths, or anything with RunAtLoad + KeepAlive, which restarts for hunting on demand. Compare against a known-good baseline. Alert on unknown LaunchDaemons and any persistence on disk but absent in user launchctl print. Flag binaries in /tmp, /Users/Shared, or hidden paths. Cross-reference against ~/Library, system-scoped ~/Library, system-scoped /Library.
launchd overrides / disabled state Plist System · Windows Stealth Control	Root	Root	Stealth	**Artifact Path** /Library/LaunchDaemons /Library/LaunchAgents **Mechanism** Persistent disabled / override state **Evasion** Silently stealth disabled state find /var /etc -name "com.apple.launchd.peuser.plist" **Impact** launchctl print disabled system launchctl print disabled gui/<uid> **Legend** launchctl print disabled system launchctl print disabled gui/<uid> plutil -p <plist>	Persistence Evasion Stealth Recon	Watch /Library/LaunchDaemons. Very high-value persistence. Review Program, ProgramArguments, MachServices, QueueDirectories, and Sockets. Use launchctl blame, see the actual trigger. Flag unknown scripts in the downgraded paths, or anything with RunAtLoad + KeepAlive, which restarts for hunting on demand. Compare against a known-good baseline. Alert on unknown LaunchDaemons and any persistence on disk but absent in user launchctl print. Flag binaries in /tmp, /Users/Shared, or hidden paths. Cross-reference against ~/Library, system-scoped ~/Library, system-scoped /Library.
BTM / SMAppService Plist · Binary Modern · Sandbox-aware · Task Management	Boot	Sandbox	Persist	**Artifact Path** ~/Library/Application Support/com.apple.backgroundtaskmanagement/ /private/var/db/com.apple.backgroundtaskmanagement/Library/ /private/var/db/com.apple.backgroundtaskmanagement/Library/ /Applications/<app>/Contents/Library/ **Mechanism** Registered helper binary or app bundle in BTM Backing app bundle, Team ID, and registration record **Evasion** sfltool dumpbtm **Impact** sfltool -v -l /Library/LaunchDaemons/com.foo.bar.plist /System/Library/CoreServices/Applications/Utilities/Terminal.app/Contents/MacOS/Terminal -c "/usr/sbin/sfltool dumpbtm" **Legend** sfltool -v -l /Library/LaunchDaemons/com.foo.bar.plist sfltool dumpbtm	Boot Login Persistence Stealth	Watch ~/Library/Application Support/com.apple.backgroundtaskmanagement/. Review Backgroundtaskmanagement.plist. Use sfltool dumpbtm. Blocking this and /var/db/com.apple.backgroundtaskmanagement/ is usually an anti-forensic step and has each registration to the real app or helper on disk.
Login Items (legacy / classic) Plist · Binary System · Windows Pre-BTM login items	Login	Sandbox	Persist	**Artifact Path** ~/Library/Preferences/com.apple.loginitems.plist **Mechanism** Legacy login item records / alias file size stores Resolved app or helper path (sometimes at user login) **Evasion** sqlite3 ~/Library/Application Support/com.apple.backgroundtaskmanagement/Library/<version>/backgroundtaskmanagement.sqlite3 "select * from items;" **Impact** sqlite3 -header -line ~/Library/Application Support/com.apple.backgroundtaskmanagement/Library/<version>/backgroundtaskmanagement.sqlite3 "select * from items order by id;" **Legend** sqlite3 -header -line ~/Library/Application Support/com.apple.backgroundtaskmanagement/Library/<version>/backgroundtaskmanagement.sqlite3 "select * from items order by id;" plutil -p ~/Library/Preferences/com.apple.loginitems.plist	User Login Persistence	Watch ~/Library/Application Support/com.apple.backgroundtaskmanagement/. Review Backgroundtaskmanagement.plist. Use sfltool dumpbtm. Blocking this and /var/db/com.apple.backgroundtaskmanagement/ is usually an anti-forensic step and has each registration to the real app or helper on disk.
Embedded Login Helpers Plist · Binary Persistence · Hijacking · Evasion · Stealth · Recon · Execution Sandbox-aware · Managed by parent app	Login	Sandbox	Persist	**Artifact Path** ~/Library/Application Support/Slack **Mechanism** Embedded helper app / bundle Parent app bundle, signature, Team ID, and enabled-state **Evasion** sqlite3 ~/Library/Application Support/Slack/storage/root-state.sqlite3 "select * from local_state;" **Impact** codesign -dv --verbosel <helper> /usr/libexec/PlistBuddy -c "Print :CFBundleIdentifier" <app>/Contents/Info.plist **Legend** codesign -dv --verbosel <helper> plutil -p <app>/Contents/Info.plist	Login Persistence Stealth Hijacking	Watch ~/Library/Application Support/<vendor>/. Review root-state.sqlite3 and its WAL. Flag unknown or stale helpers, and parity with the actual UI and app state. Cross-reference against MDM allowlists, and all launchd/LaunchAgents, and background-task inventories.
Shell and Quasi (.rc) Plaintext · Binary Persistence · Hijacking · Evasion · Stealth · Recon · Execution	Root	Sandbox	Persist	**Artifact Path** /etc/zshrc /etc/bashrc ~/.zshrc ~/.zshrc.local ~/.zsh_profile ~/.zsh_login ~/.zshrc ~/.bash_profile ~/.profile **Mechanism** Commands executed by shell at startup files ZDOTDIR overrides file discovery and order **Evasion** echo $ZDOTDIR ls -la ~	grep -E '\.(z|bash|k|t|c|profile|gdb|lldb|inputrc|curlrc|wgetrc|gitconfig|tmux|screen|dir-locals|hushlogin)' **Impact** /usr/libexec/PlistBuddy -c "Print :ProgramArguments:0" /Library/LaunchDaemons/com.foo.bar.plist **Legend** /usr/libexec/PlistBuddy -c "Print :ProgramArguments:0" /Library/LaunchDaemons/com.foo.bar.plist plutil -p /Library/LaunchDaemons/com.foo.bar.plist	Login shell interactive shell shell trigger	Watch /etc/zshrc, /etc/bashrc, ~/.zshrc, ~/.zshrc.local, ~/.zsh_profile, ~/.zsh_login, ~/.zshrc, ~/.bash_profile, ~/.profile. Review ZDOTDIR and its siblings. Flag unknown or stale, obfuscated, or recently modified entries. Cross-reference against MDM, dotfiles, and all launchd/LaunchAgents, and background-task inventories.
SSH authorized_keys Plaintext · Binary System · Windows Remote access persistence	Root	Sandbox	Persist	**Artifact Path** ~/.ssh/authorized_keys ~/.ssh/config /etc/ssh/sshd_config **Mechanism** Authorized key entries accepted by sshd AuthorizedKeysCommand, command, user, principals, principals_command, expiry_command **Evasion** find ~ /var /tmp -name authorized_keys -newerBt "1 week ago" **Impact** sshd -T -f /etc/ssh/sshd_config **Legend** sshd -T -f /etc/ssh/sshd_config ssh-keygen -lf -/.ssh/authorized_keys	Login Persistence Stealth	Watch ~/.ssh/authorized_keys. Review AuthorizedKeysCommand and AuthorizedKeysCommandUser. Flag unknown or stale, obfuscated, or recently modified entries. Cross-reference against MDM, dotfiles, and all launchd/LaunchAgents, and background-task inventories.
Cron Plaintext Classic scheduler	Root	Sandbox	Persist	**Artifact Path** /etc/crontab /var/spool/cron/crontabs/ **Mechanism** Command field and crontab script path The user or system crontab entry actually executed **Evasion** crontab -l ls -la /etc/cron.{hourly|daily|weekly|monthly} **Impact** ls -l /var/spool/cron/crontabs/ **Legend** grep '^[^#]' /etc/crontab /etc/cron.{hourly|daily|weekly|monthly}	Schedule persists	Watch /etc/crontab, /var/spool/cron/crontabs/*. Review command and crontab script path. Flag unknown or stale, obfuscated, or recently modified entries. Cross-reference against MDM, dotfiles, and all launchd/LaunchAgents, and background-task inventories.
At / atrun Plaintext Rarely used on modern systems, can run once	Root	Sandbox	Persist	**Artifact Path** /var/spool/at/ /var/at/jobs/ **Mechanism** atrun daemon enabled **Evasion** ls -ltr /var/at/jobs/ **Impact** launchctl enable system/com.apple.atrun **Legend** launchctl enable system/com.apple.atrun atrun -d <job_id>	Schedule Persistence Stealth	Watch /var/spool/at/ and /var/at/jobs/. Review atrun -d <job_id>. Flag unknown or stale, obfuscated, or recently modified entries. Cross-reference against MDM, dotfiles, and all launchd/LaunchAgents, and background-task inventories.
Periodic Jobs Plaintext System · Windows Rarely used on modern systems	Root	Sandbox	Persist	**Artifact Path** /etc/periodic/daily /etc/periodic/weekly /etc/periodic/monthly /usr/local/etc/periodic /opt/local/etc/periodic (macOS: /opt/local/) **Mechanism** Scripts executed from periodic dirs Any custom path listed/enabled through periodic.conf **Evasion** ls -la /etc/periodic/{daily|weekly|monthly} /opt/local/etc/periodic **Impact** grep -E '^(/etc|/usr|/opt|/var|/home|/Library|/Applications|/tmp|/private|/dev|/bin|/sbin)' /etc/periodic.conf **Legend** grep -E '^(/etc|/usr|/opt|/var|/home|/Library|/Applications|/tmp|/private|/dev|/bin|/sbin)' /etc/periodic.conf find /etc/periodic /usr/local/etc/periodic -type f -perm -u+x	Daily weekly monthly scheduled	Watch /etc/periodic/daily, /etc/periodic/weekly, /etc/periodic/monthly, /usr/local/etc/periodic, /opt/local/etc/periodic. Review /etc/periodic.conf, /usr/local/etc/periodic.conf, /opt/local/etc/periodic.conf. Flag unknown or stale, obfuscated, or recently modified entries. Cross-reference against MDM, dotfiles, and all launchd/LaunchAgents, and background-task inventories.
Emond Plaintext · Binary Deprecated · Rarely used	Root	Sandbox	Persist	**Artifact Path** /private/var/db/emond/ /private/etc/emond.d/ /System/Library/LaunchDaemons/com.apple.emond.plist **Mechanism** Active command or script referenced by the rule Conditions or action chain that leads to code execution **Evasion** ls -la /private/var/db/emond /private/etc/emond.d **Impact** plutil -p /private/var/db/emond/*.plist **Legend** plutil -p /private/var/db/emond/*.plist grep -R '^' /private/var/db/emond /private/etc/emond.d	Event-driven Persistence Stealth	Watch /private/var/db/emond/, /private/etc/emond.d/. Review /System/Library/LaunchDaemons/com.apple.emond.plist. Flag unknown or stale, obfuscated, or recently modified entries. Cross-reference against MDM, dotfiles, and all launchd/LaunchAgents, and background-task inventories.

Extensions, Browser, Interpreter, and Handler Surfaces

Mechanism	Required privilege	Scope of Scope	Signal	Collection / Trigger	Trigger	What to review
System Extensions System · Windows Current macOS	Root	Global	Persist	**Artifact Path** /Library/SystemExtensions /var/db/sysx/sysx.db /System/Extensions **Mechanism** Activated system extensions and matching app Bundle ID, Team ID, and activation state where be systemextensionsctl **Evasion** systemextensionsctl list **Impact** codesign -dv --verbose=4 /path/to.app systemextensionsctl list	Boot activation persistence Stealth	Watch /Library/SystemExtensions. Review /var/db/sysx/sysx.db. Flag unknown or stale, obfuscated, or recently modified entries. Cross-reference against MDM, dotfiles, and all launchd/LaunchAgents, and background-task inventories.
KEXTs System · Windows Legacy, deprecated, but still supported on Apple Silicon	Root	Global	Persist	**Artifact Path** /Library/Extensions **Mechanism** Loaded kext bundle ID and path Actual loaded extension plus the on-disk bundle **Evasion** kmutil showloaded kextstat **Impact** codesign -dv --verbose=4 /kext/ kmutil inspect -b <bundle_id>	Boot persistence Stealth	Watch /Library/Extensions. Review loaded kext bundles, kmutil showloaded, kextstat, and kmutil inspect. Flag unknown or stale, obfuscated, or recently modified entries. Cross-reference against MDM, dotfiles, and all launchd/LaunchAgents, and background-task inventories.
Chromium Extensions Manifest · Binary Chromium-based	Boot	Sandbox	Persist	**Artifact Path** ~/Library/Application Support/Google/Chrome/<profile>/Extensions ~/Library/Application Support/Chromium/<profile>/Extensions ~/Library/Google/GoogleSoftware/Update/Policies/ **Mechanism** Extension, its manifest, and effective policy source Force-installed or managed extension config that is not user-visible **Evasion** find ~/Library/Application Support/{Google/Chrome|Chromium}/*/Extensions/* -name manifest.* **Impact** find ~/Library/Application Support/{Google/Chrome|Chromium}/*/Extensions/* -name manifest.* **Legend**	Boot Persistence Evasion Stealth	Watch ~/Library/Application Support/{Google/Chrome|Chromium}/*/Extensions/*/*/manifest.*. Review extension IDs, policy forced installs, managed profiles, update URLs, and permissions. Flag unknown or stale, obfuscated, or recently modified entries. Cross-reference against MDM, dotfiles, and all launchd/LaunchAgents, and background-task inventories.

```
/usr/bin/sqlite3 -cmd '.output /dev/null' -cmd '.load /tmp/evil.so' -cmd '.exit'
/usr/bin/sqlite3 -cmd '.output /dev/null' -cmd '.load /tmp/evil.so' -cmd '.exit'
```

Safari App Extensions

Class	Approval	Auth	Persist
Extension	Dev	Auto	Install

macOS · Uses TCC, SQLite

Lists TCC-granted extensions.

```
/usr/bin/sqlite3 -cmd '.output /Library/Application Support/com.apple.TCC/TCC.db' -cmd 'select client, auth_value, last_modified from access;' -cmd '.exit'
```

Browser extension persistence

macOS · Safari · Persistence · Extension · Dev · Auto · Install

Safari Web Extensions

Class	Approval	Auth	Persist
Extension	Dev	Auto	Install

macOS · Uses TCC

Extend Safari with extension files.

```
/usr/bin/sqlite3 -cmd '.output /Library/Application Support/com.apple.TCC/TCC.db' -cmd 'select service, client, auth_value, last_modified from access;' -cmd '.exit'
```

Browser extension persistence

macOS · Safari · Persistence · Extension · Dev · Auto · Install

Application / daemon plug-ins

Class	Approval	Auth	Persist
Extension	Mixed	Auto	Install

macOS · Varies by host

Quarantine, Gatekeeper, Notarization, Hardened Runtime, Sandbox, TCC, Code signing, Library validation

Extend a host application with plug-in bundles that load into its process.

Host-dependent hook or hijacking

macOS · Extension · Mixed · Auto · Install · Varies by host

Python Startup Hooks

Class	Approval	Auth	Persist
Hijack	Dev	Auto	Install

macOS · Linux · Windows

Sys Events

```
/usr/bin/sqlite3 -cmd '.output /dev/null' -cmd '.load /tmp/evil.so' -cmd '.exit'
/usr/bin/sqlite3 -cmd '.output /dev/null' -cmd '.load /tmp/evil.so' -cmd '.exit'
```

Python startup hook

macOS · Linux · Persistence · Hijack · Dev · Auto · Install

LaunchServices URL Handlers

Class	Approval	Auth	Persist
Hijack	Dev	Auto	Install

macOS · Uses TCC

Sys Events

```
/usr/bin/sqlite3 -cmd '.output /Library/Preferences/com.apple.LaunchServices/com.apple.launchservices.secure.plist' -cmd '.output /Users/$(whoami)/Library/Preferences/com.apple.LaunchServices/com.apple.launchservices.secure.plist' -cmd '.dump LSHandlers' -cmd '.exit'
```

The agent hijacks URL handlers

macOS · Persistence · Hijacker · Dev · Auto · Install · Uses TCC

Persistence-Adjacent & Legacy Mechanisms

These stay in the sheet because they, hooks or OS-level persistence, persistence, or persistence-adjacent mechanisms in the same sense as launchd, RC, login hooks, or cron jobs.

Mechanism	Class	Approval	Auth	Persist	Collection / Trigger	Trigger	What to review
Installer Packages macOS · Uses TCC · Sys Events · Persistence · Hijacker · Extension	Persistence	Mixed	Auto	Install	Artifacts / Trigger Artifacts: /var/db/receipts/*.plist Resources: /usr/sbin/pkgutil Trigger: pkgutil --pkgs --volume / --regexp com.(?!apple).*	Package installation	Log OS Security Hooks / Log Endpoint Security Hooks Usually a development or dropper surface rather than persistence. Receipt backlog is often less useful than it looks, but it is still a strong persistence. Scope it needed or relevant because packages are a common way to drop agents, daemons, or other user- or system-scoped persistence.
DYLD_* Injection / LSEnvironment macOS · Persistence · Hijacker · Dev · Auto · Install · Uses TCC · Sys Events · Binary analysis · Environment hijacker	Hijacker	Dev	Auto	Install	Artifacts / Trigger Artifacts: ~/Library/LaunchAgents/*.plist /Library/LaunchAgents/*.plist /Library/LaunchDaemons/*.plist Resources: /usr/libexec/PlistBuddy /usr/bin/plutil /usr/sbin/launchctl Trigger: grep -E 'DYLD_(INSERT_LIBRARIES|FRAMEWORK_PATH)' ~/Library/LaunchAgents/*.plist /Library/LaunchAgents/*.plist /Library/LaunchDaemons/*.plist launchctl setenv DYLD_INSERT_LIBRARIES /path/to/evil.dylib launchctl setenv DYLD_FRAMEWORK_PATH /path/to/evil launchctl bootout gui/$(id -u) /path/to/target.plist launchctl bootstrap gui/$(id -u) /path/to/target.plist	Targeted persistence	Log OS Security Hooks / Log Endpoint Security Hooks Separate from Mach-O linked-command hijacking. Do keep it separate from launchd plists, shell startup, and other persistence when the goal is to bend one specific binary or launch path. Strong when paired with launchd persistence or when the target is a trusted or high-value binary.
Application / library modification macOS · Persistence · Hijacker · Dev · Auto · Install · Uses TCC · Sys Events · Binary analysis · Filesystem hijacker	Hijacker	Dev	Auto	Install	Artifacts / Trigger Artifacts: /Applications/*.app ~/Applications/*.app Resources: /usr/bin/find /usr/bin/sqlite3 /usr/sbin/spctl /usr/bin/codesign Trigger: find /Applications ~/Applications -name '*.app' -mtime -7 -ls sqlite3 ~/Library/Preferences/com.apple.LaunchServices.plist spctl --assess --verbose /Applications/<app_name>.app codesign -vvv --deep --strict /Applications/<app_name>.app find /Applications ~/Applications -name '*.app' -exec codesign -vvv --deep --strict {} \;	Trusted binary abuse	Log OS Security Hooks / Log Endpoint Security Hooks Broaden from signature-only persistence to include persistence when the target is a trusted or high-value binary. Keep it separate from dylib hijacking, launchd persistence, and shell startup persistence. Strong when the target is a user-writable or system-writable location.
LC_LOAD_DYLIB Addition macOS · Persistence · Hijacker · Dev · Auto · Install · Binary analysis	Hijacker	Dev	Auto	Install	Artifacts / Trigger Artifacts: ~/Library/Preferences/com.apple.loginitems.plist /Library/Preferences/com.apple.loginitems.plist Resources: /usr/bin/sqlite3 /usr/bin/osascript /usr/sbin/sfltool Trigger: sqlite3 ~/Library/Preferences/com.apple.loginitems.plist osascript -e 'tell application "System Events" to get properties of login items' sfltool dumpbtm sfltool resetbtm /usr/libexec/PlistBuddy -c "Print :SessionItems:CustomItems" ~/Library/Preferences/com.apple.loginitems.plist /usr/libexec/PlistBuddy -c "Print :SessionItems:CustomItems" /Library/Preferences/com.apple.loginitems.plist	Trusted binary abuse	Log OS Security Hooks / Log Endpoint Security Hooks Persistence-adjacent binary abuse. Keep it separate from login items, launchd persistence, and shell startup persistence. Strong when the target is a trusted or high-value binary. Treat as broad scope because the persistence may be in a user-writable or system-writable location.
Dylib Hijacking / Proxying macOS · Persistence · Hijacker · Dev · Auto · Install · Binary analysis · Dylib hijacker	Hijacker	Dev	Auto	Install	Artifacts / Trigger Artifacts: ~/Library/LaunchAgents/*.plist /Library/LaunchAgents/*.plist /Library/LaunchDaemons/*.plist Resources: /usr/bin/find /usr/bin/sqlite3 /usr/sbin/spctl /usr/bin/codesign Trigger: find /Applications ~/Applications -name '*.app' -mtime -7 -ls sqlite3 ~/Library/Preferences/com.apple.LaunchServices.plist spctl --assess --verbose /Applications/<app_name>.app codesign -vvv --deep --strict /Applications/<app_name>.app find /Applications ~/Applications -name '*.app' -exec codesign -vvv --deep --strict {} \;	Binary-level persistence	Log OS Security Hooks / Log Endpoint Security Hooks Broaden from persistence-adjacent binary abuse. Keep it separate from dylib hijacking, launchd persistence, and shell startup persistence. Strong when the target is a trusted or high-value binary. Treat as broad scope because the persistence may be in a user-writable or system-writable location.
Login Hooks macOS · Persistence · Hijacker · Dev · Auto · Install · Uses TCC · Sys Events · Legacy, use only as a last resort	Persistence	Mixed	Auto	Install	Artifacts / Trigger Artifacts: ~/Library/Preferences/com.apple.loginwindow.plist /Library/Preferences/com.apple.loginwindow.plist Resources: /usr/bin/defaults /usr/bin/sqlite3 /usr/sbin/spctl Trigger: defaults read ~/Library/Preferences/com.apple.loginwindow.plist LoginHook defaults read /Library/Preferences/com.apple.loginwindow.plist LoginHook sqlite3 ~/Library/Preferences/com.apple.loginwindow.plist spctl --assess --verbose /path/to/hook.app spctl --assess --verbose /path/to/hook.sh	Legacy / deprecated	Log OS Security Hooks / Log Endpoint Security Hooks Deprecated since 10.11. Hooks can be shell scripts or apps. Keep it separate from launchd persistence, cron persistence, and shell startup persistence. Use it when the goal is an older, simple, persistent login hook. Strong when the artifact still matters in older images, migrations, and mixed fleets.
Startup Items macOS · Persistence · Hijacker · Dev · Auto · Install · Uses TCC · Sys Events · Legacy	Persistence	Dev	Auto	Install	Artifacts / Trigger Artifacts: /Library/StartupItems /System/Library/StartupItems Resources: /usr/sbin/kextstat /usr/sbin/spctl Trigger: ls -la /Library/StartupItems /System/Library/StartupItems kextstat -b com.<vendor_name>.<kext_name> spctl --assess --type exec /Library/StartupItems/<item_name> /System/Library/StartupItems/<item_name>	Boot	Log OS Security Hooks / Log Endpoint Security Hooks Legacy mechanism mostly replaced by OS X 10.5 and earlier compatibility. Keep it separate from modern persistence. Strong if legacy coverage is the artifact still matters in older images, migrations, and mixed fleets.
RC Scripts macOS · Persistence · Hijacker · Dev · Auto · Install · Uses TCC · Sys Events	Persistence	Dev	Auto	Install	Artifacts / Trigger Artifacts: /etc/rc.local /etc/rc.common /usr/local/etc/rc.d/* Resources: /bin/ls /usr/sbin/systemsetup Trigger: ls -la /etc/rc.local /etc/rc.common /usr/local/etc/rc.d/* /bin/ls -la /Library/StartupItems /System/Library/StartupItems /usr/sbin/systemsetup -getrunsystem /usr/sbin/systemsetup -setrunsystem on /usr/sbin/systemsetup -setrunsystem off /usr/sbin/systemsetup -getwaitforstartup /usr/sbin/systemsetup -setwaitforstartup 30 /usr/sbin/systemsetup -getrestartafterfreeze /usr/sbin/systemsetup -setrestartafterfreeze on /usr/sbin/systemsetup -getrestartfreeze /usr/sbin/systemsetup -setrestartfreeze on	Boot-time path	Log OS Security Hooks / Log Endpoint Security Hooks Mostly historical on current macOS. Keep it separate from modern persistence. Strong when legacy boot-time persistence or a modern hook is itself persistence and therefore high signal. Keep it separate from shell startup persistence, cron persistence, or modern hooks. Treat any non-empty result in a production or forensic image as worth a deeper persistence review.

Configuration Profiles & MDM

Configuration profiles, mdmclient, and MDM-pushed persistence are a deployment surface, not a persistence surface. Keep them separate from login items and cron persistence. Use it when the goal is managed persistence, policy-driven persistence, or persistence that looks like enterprise administration. Strong when the artifact is a managed profile, MDM-pushed persistence, or a persistence change that is easier to explain as managed than as ordinary persistence.

Mechanism	Class	Approval	Auth	Persist	Collection / Trigger	Trigger	What to review
Configuration Profiles (/usr/bin/profiles) macOS · Persistence · Hijacker · Dev · Auto · Install · Uses TCC · Binary analysis · Profile hijacker	Profile hijacker	Mixed	Auto	Install	Artifacts / Trigger Artifacts: /var/db/ConfigurationProfiles/Store/ /var/db/ConfigurationProfiles/Settings/ /Library/Managed Preferences/*.plist /Library/Managed Preferences/*/*.plist Resources: /usr/bin/profiles /usr/libexec/PlistBuddy /usr/bin/plutil /usr/bin/sqlite3 Trigger: /usr/bin/profiles -P /usr/bin/profiles -L /usr/bin/profiles -C -v /usr/bin/profiles -C -v -f -I /usr/bin/profiles -P -f -I /usr/bin/profiles -D -f -I /usr/bin/profiles -E -f -I /usr/bin/profiles -V -f -I /usr/bin/profiles -I -F /path/to/profile.mobileconfig /usr/bin/profiles -R -p com.example.profile /usr/libexec/PlistBuddy -c "Print :" /Library/Managed Preferences/com.apple.SoftwareUpdate.plist /usr/libexec/PlistBuddy -c "Print :" /Library/Managed Preferences/*/com.apple.SoftwareUpdate.plist /usr/bin/plutil -p /Library/Managed Preferences/com.apple.SoftwareUpdate.plist /usr/bin/sqlite3 /var/db/ConfigurationProfiles/Store/CurrentProfiles.sqlite "select * from profile_info;" /usr/bin/sqlite3 /var/db/ConfigurationProfiles/Store/CurrentProfiles.sqlite "select * from payload_info;"	Profile install / enrollment / policy change	Log OS Security Hooks / Log Endpoint Security Hooks Keep it separate from login items, cron persistence, and shell startup persistence. Use it when the goal is managed persistence, policy-driven persistence, or persistence that looks like enterprise administration. Strong when the artifact is a managed profile, MDM-pushed persistence, or a persistence change that is easier to explain as managed than as ordinary persistence. Use raw profiles output when the goal is to see what the host thinks is installed. Use parsed plists when the goal is to see what actually is enforced. Use the profile database when the goal is to see what the host has recorded as installed.
High-Impact MDM payloads macOS · Persistence · Hijacker · Dev · Auto · Install · Uses TCC · Sys Events · Enterprise · Binary analysis · Profile hijacker	Profile hijacker	Mixed	Auto	Install	Artifacts / Trigger Artifacts: /var/db/ConfigurationProfiles/Store/ /var/db/ConfigurationProfiles/Settings/ Resources: /usr/bin/profiles /usr/bin/sqlite3 /usr/bin/plutil Trigger: /usr/bin/profiles -P /usr/bin/profiles -L /usr/bin/profiles -C -v /usr/bin/profiles -C -v -f -I /usr/bin/profiles -P -f -I /usr/bin/profiles -D -f -I /usr/bin/profiles -E -f -I /usr/bin/profiles -V -f -I /usr/bin/profiles -I -F /path/to/profile.mobileconfig /usr/bin/profiles -R -p com.example.profile /usr/bin/sqlite3 /var/db/ConfigurationProfiles/Store/CurrentProfiles.sqlite "select * from profile_info;" /usr/bin/sqlite3 /var/db/ConfigurationProfiles/Store/CurrentProfiles.sqlite "select * from payload_info;"	Profile install / enrollment / policy change	Log OS Security Hooks / Log Endpoint Security Hooks Keep it separate from login items, cron persistence, and shell startup persistence. Use it when the goal is managed persistence, policy-driven persistence, or persistence that looks like enterprise administration. Strong when the artifact is a managed profile, MDM-pushed persistence, or a persistence change that is easier to explain as managed than as ordinary persistence. Use raw profiles output when the goal is to see what the host thinks is installed. Use parsed plists when the goal is to see what actually is enforced. Use the profile database when the goal is to see what the host has recorded as installed.

Quick Ref Checklist

- Sources of **launchd** jobs and parts the code with `launchd.plist` and `plist.launchd`
- Scope BTM / login items and stay inside Ventura or later, signature and team identity against the supplied expectations, `sfltool dumpbtm`
- Use `sqlite3 -cmd '.output /dev/null' -cmd '.load /tmp/evil.so' -cmd '.exit'` before concluding nothing changed.
- Prioritize extension/execution through `/usr/sbin/spctl` and `/usr/bin/codesign`
- Do long surfaces, triage by signal. Trace SQLite, XPC, policy, content, path quality, and install prevalence rather than by path alone.
- For every artifact, capture owner, Team ID, path, SHA, mtime or birthtime, parent process, and user context.
- Accumulate installed configurations, `/var/db/ConfigurationProfiles/Store/*.plist`, `/var/db/ConfigurationProfiles/Settings/*.plist`, `/Library/Managed Preferences/*/*.plist`, `/usr/bin/profiles -P -f -I`, `/usr/bin/profiles -L`, `/usr/bin/profiles -C -v`, `/usr/bin/profiles -E -f -I`, `/usr/bin/profiles -V -f -I`, `/usr/bin/profiles -D -f -I`, `/usr/bin/profiles -P -f -I`, `/usr/bin/profiles -I -F /path/to/profile.mobileconfig`, `/usr/bin/profiles -R -p com.example.profile`, `/usr/bin/sqlite3 /var/db/ConfigurationProfiles/Store/CurrentProfiles.sqlite "select * from profile_info;"`, `/usr/bin/sqlite3 /var/db/ConfigurationProfiles/Store/CurrentProfiles.sqlite "select * from payload_info;"`, `/usr/libexec/PlistBuddy -c "Print :" /Library/Managed Preferences/com.apple.SoftwareUpdate.plist`, `/usr/libexec/PlistBuddy -c "Print :" /Library/Managed Preferences/*/com.apple.SoftwareUpdate.plist`, `/usr/bin/plutil -p /Library/Managed Preferences/com.apple.SoftwareUpdate.plist`, `/usr/bin/sqlite3 /var/db/ConfigurationProfiles/Store/CurrentProfiles.sqlite "select * from profile_info;"`, `/usr/bin/sqlite3 /var/db/ConfigurationProfiles/Store/CurrentProfiles.sqlite "select * from payload_info;"`, `/usr/sbin/systemsetup -getrunsystem`, `/usr/sbin/systemsetup -setrunsystem on`, `/usr/sbin/systemsetup -setrunsystem off`, `/usr/sbin/systemsetup -getwaitforstartup`, `/usr/sbin/systemsetup -setwaitforstartup 30`, `/usr/sbin/systemsetup -getrestartafterfreeze`, `/usr/sbin/systemsetup -setrestartafterfreeze on`, `/usr/sbin/systemsetup -getrestartfreeze`, `/usr/sbin/systemsetup -setrestartfreeze on`
- Recursion, `/Library/StartupItems`, `/System/Library/StartupItems`, `/usr/sbin/kextstat -b com.<vendor_name>.<kext_name>`, `/usr/sbin/spctl --assess --type exec /Library/StartupItems/<item_name>`, `/System/Library/StartupItems/<item_name>`, `/usr/sbin/spctl --assess --verbose /path/to/hook.app`, `/usr/sbin/spctl --assess --verbose /path/to/hook.sh`, `/usr/sbin/codesign -vvv --deep --strict /Applications/<app_name>.app`, `/usr/sbin/codesign -vvv --deep --strict {} \;`, `/usr/sbin/sqlite3 ~/Library/Preferences/com.apple.LaunchServices.plist`, `/usr/sbin/sqlite3 ~/Library/Preferences/com.apple.LaunchServices/com.apple.launchservices.secure.plist`, `/usr/sbin/sqlite3 /Library/Preferences/com.apple.LaunchServices/com.apple.launchservices.secure.plist`, `/usr/sbin/sqlite3 ~/Library/Preferences/com.apple.loginitems.plist`, `/usr/sbin/sqlite3 /Library/Preferences/com.apple.loginitems.plist`, `/usr/sbin/osascript -e 'tell application "System Events" to get properties of login items'`, `/usr/sbin/sfltool dumpbtm`, `/usr/sbin/sfltool resetbtm`, `/usr/sbin/defaults read ~/Library/Preferences/com.apple.loginwindow.plist LoginHook`, `/usr/sbin/defaults read /Library/Preferences/com.apple.loginwindow.plist LoginHook`, `/usr/sbin/sqlite3 ~/Library/Preferences/com.apple.loginwindow.plist`, `/usr/sbin/spctl --assess --verbose /path/to/hook.app`, `/usr/sbin/spctl --assess --verbose /path/to/hook.sh`, `/usr/sbin/find /Applications ~/Applications -name '*.app' -mtime -7 -ls`, `/usr/sbin/find /Applications ~/Applications -name '*.app' -exec codesign -vvv --deep --strict {} \;`, `/usr/sbin/spctl --assess --verbose /Applications/<app_name>.app`, `/usr/sbin/codesign -vvv --deep --strict /Applications/<app_name>.app`, `/usr/sbin/find /Applications ~/Applications -name '*.app' -exec codesign -vvv --deep --strict {} \;`, `/usr/sbin/find ~/Library/LaunchAgents/*.plist /Library/LaunchAgents/*.plist /Library/LaunchDaemons/*.plist -ls`, `/usr/sbin/find ~/Library/LaunchAgents/*.plist /Library/LaunchAgents/*.plist /Library/LaunchDaemons/*.plist -exec /usr/sbin/codesign -vvv --deep --strict {} \;`, `/usr/sbin/find ~/Library/LaunchAgents/*.plist /Library/LaunchAgents/*.plist /Library/LaunchDaemons/*.plist -exec /usr/sbin/spctl --assess --verbose {} \;`, `/usr/sbin/find /Applications ~/Applications -name '*.app' -exec /usr/sbin/spctl --assess --verbose {} \;`, `/usr/sbin/find ~/Library/LaunchAgents/*.plist /Library/LaunchAgents/*.plist /Library/LaunchDaemons/*.plist -exec /usr/sbin/sqlite3 {} '.output /dev/null' '.load /tmp/evil.so' '.exit' \;`, `/usr/sbin/find /Applications ~/Applications -name '*.app' -exec /usr/sbin/sqlite3 {} '.output /dev/null' '.load /tmp/evil.so' '.exit' \;`, `/usr/sbin/find ~/Library/LaunchAgents/*.plist /Library/LaunchAgents/*.plist /Library/LaunchDaemons/*.plist -exec /usr/sbin/sqlite3 {} '.output /dev/null' '.load /tmp/evil.so' '.exit' \;`
- Prioritize persistence-adjacent, hijacker, and hook artifacts against the suspicious process tree, signature, and recent execution.

Coordinating Evidence

- **Signature / Team ID**: confirm the persisted component, its owner, app, and any helper all claim to the same signer where that is expected.
- **Quarantine / provenance**: inspect `com.apple.quarantine` and review how the artifact first arrived on the host.
- **Timestamps**: compare birthtime, mtime, and first seen relatively against the suspicious process tree.
- **Unified logs**: pivot to system and kernel events around the entry or enablement time, especially for extensions, kernel extensions, login items, and profiles.
- **Process ancestry**: identify what created or modified the artifact, then compare that process ancestry against known-good baselines.
- **Policy context**: compare managed profiles, enrollment state, and enforced restrictions against what the host should normally accept.
- **Recursion / lag awareness**: recurse, lag, sure-fire timestamp, and on-disk state are signals worth tracking before calling an unexplained change in extension behavior.

Web Signals / Priority Indicators

- Resources: `/usr/bin/sqlite3` · `/usr/bin/osascript` · `/usr/bin/defaults` · `/usr/bin/plutil` · `/usr/libexec/PlistBuddy` · `/usr/bin/find` · `/usr/bin/codesign` · `/usr/sbin/spctl` · `/usr/sbin/sfltool` · `/usr/sbin/kextstat` · `/usr/sbin/systemsetup` · `/usr/sbin/launchctl` · `/usr/sbin/profiles`
- Trigger: `/Library/LaunchAgents/*.plist` · `/Library/LaunchDaemons/*.plist` · `/var/db/ConfigurationProfiles/Store/*.plist` · `/var/db/ConfigurationProfiles/Settings/*.plist` · `/Library/Managed Preferences/*/*.plist` · `/var/db/ConfigurationProfiles/Store/CurrentProfiles.sqlite` · `/Library/StartupItems` · `/System/Library/StartupItems`
- BTM entry or embedded helper whose path is stale, user-writable, or whose signer / Team ID does not match the expected app.
- New GUI app, or an active account with no corresponding earlier activity, before a suspicious event.
- `/usr/bin/profiles -P` or an unusual path or artifact flags that identify source and enrollment state in forensic context.
- Unexpected third-party entry in `/var/db/ConfigurationProfiles/Store/*.plist` or a clear first appearance on a host where one is not normally expected.
- New Chromium extension ID, policy force-install, or managed-profile entry. Gather signer, extension ID, and policy source on the suspicious host.
- `/usr/bin/profiles -P -f -I` containing one-shot `-E` flags, or a new `/Library/Managed Preferences/*/com.apple.SoftwareUpdate.plist` change.
- Trusted binary that suddenly loads a dylib from a user-writable path, hidden directory, or a location inconsistent with the app's normal install layout.

01 Unified Logs for launch activity, especially suspicious launchd labels, repeated failures, and unexpected restarts.

02 backgroundtaskmanagementagent and related BTM records when Login Items or helpers look inconsistent with the installed app set.

03 sysextd, systemextensionsctl output, and approval-related records when a system extension or driver stack is in scope.

04 install.log and package provenance when preinstall / postinstall or helper deployment appears to have introduced persistence.

05 sshd logs plus auth context for unexpected authorized_keys changes, custom AuthorizedKeysFile usage, or unusual remote access timing.

06 Signer, Team ID, notarization, quarantine xattrs, mtime / birthtime, and first-seen telemetry to validate whether the artifact matches its claimed parent app or install flow.