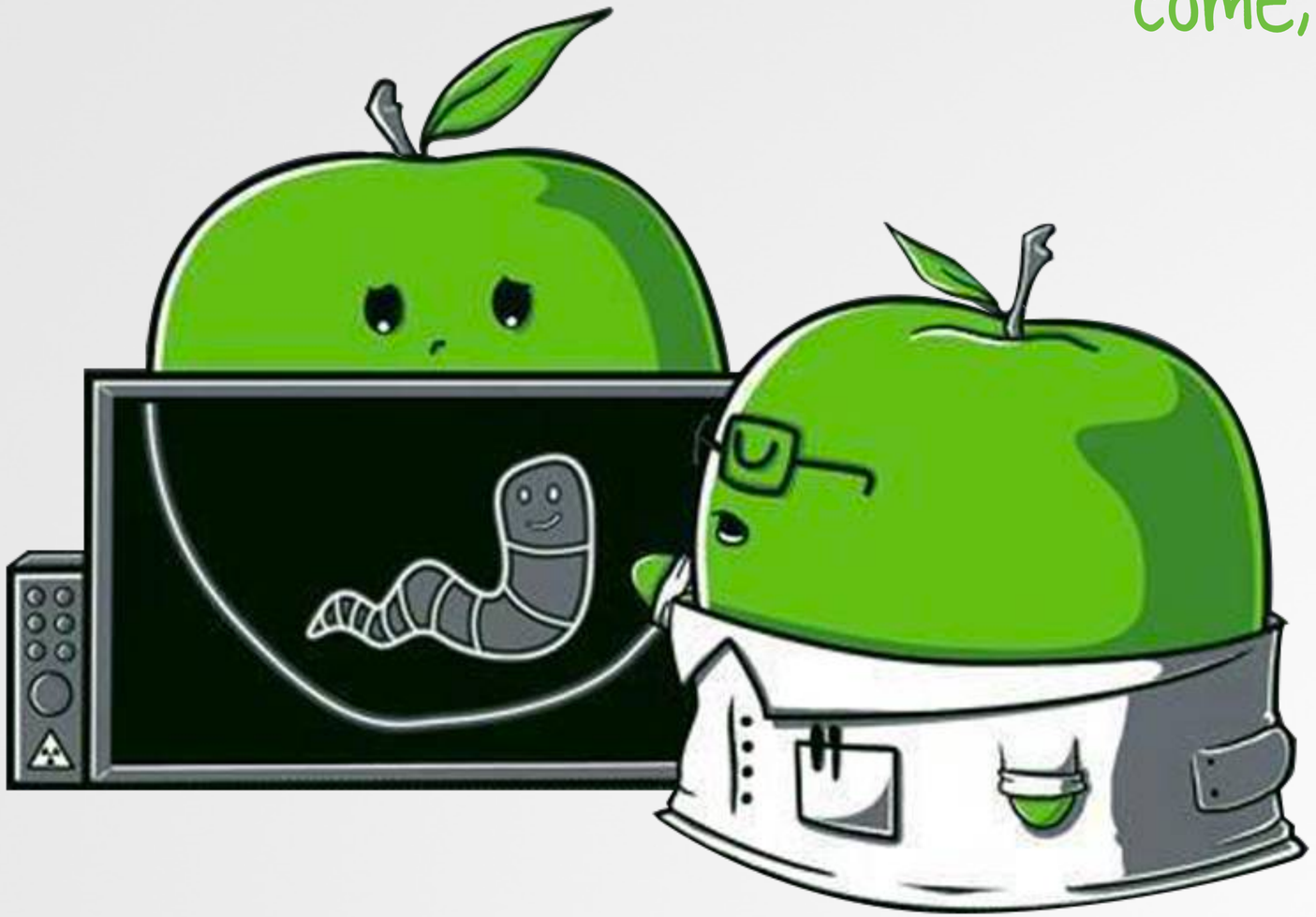
Gatekeeper Exposed

come, see, conquer!



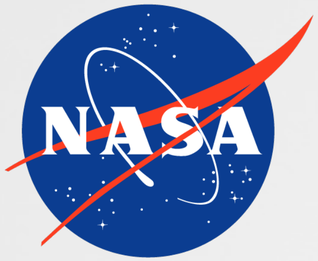
WHOIS

Synack

security for the 21st century

"leverages the best combination of humans and technology to discover security vulnerabilities in our customers' web apps, mobile apps, IoT devices and infrastructure endpoints"

career



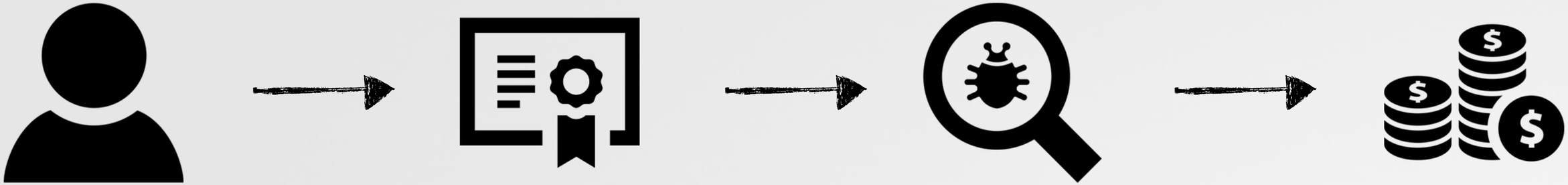
hobby

Objective-See

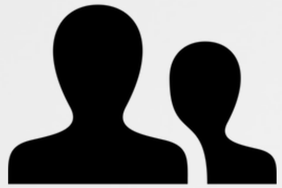
@patrickwardle

Synack & the Synack Red Team (SRT)

join, find bugs, profit!



why Synack?



smaller 'crowd'

+

larger customers

=

more, higher,
faster, payouts

Synack

Outline

all aspects of gatekeeper

Gatekeeper

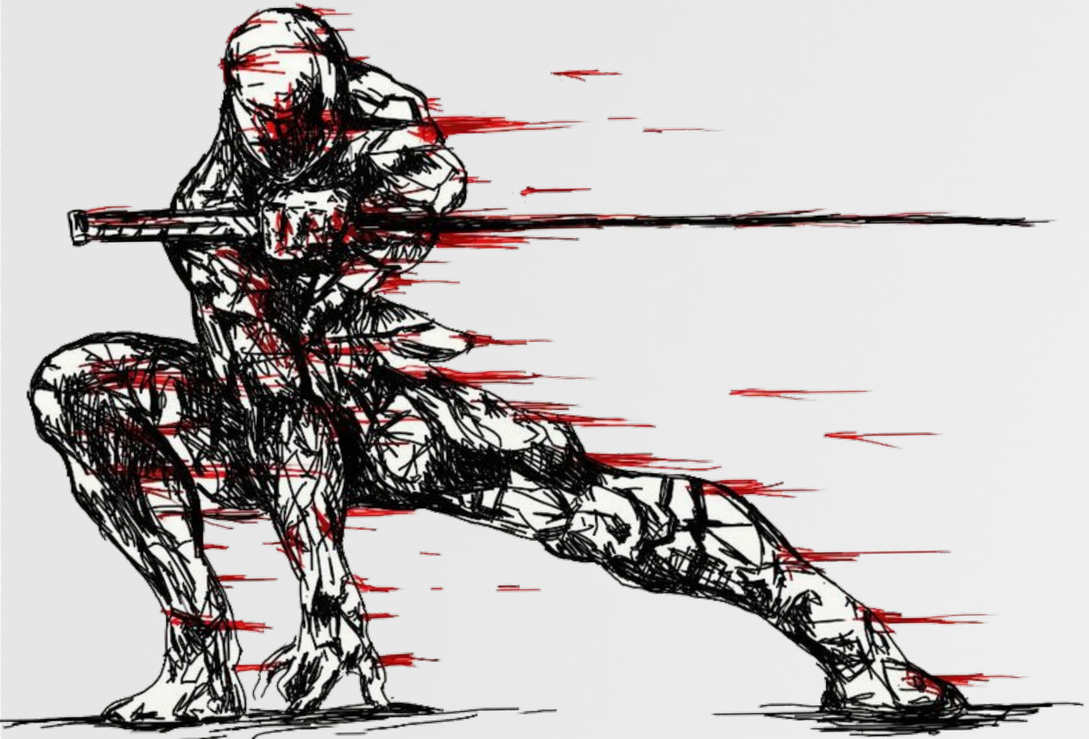
understanding

bypassing

fixing

UNDERSTANDING GATEKEEPER

…under the hood

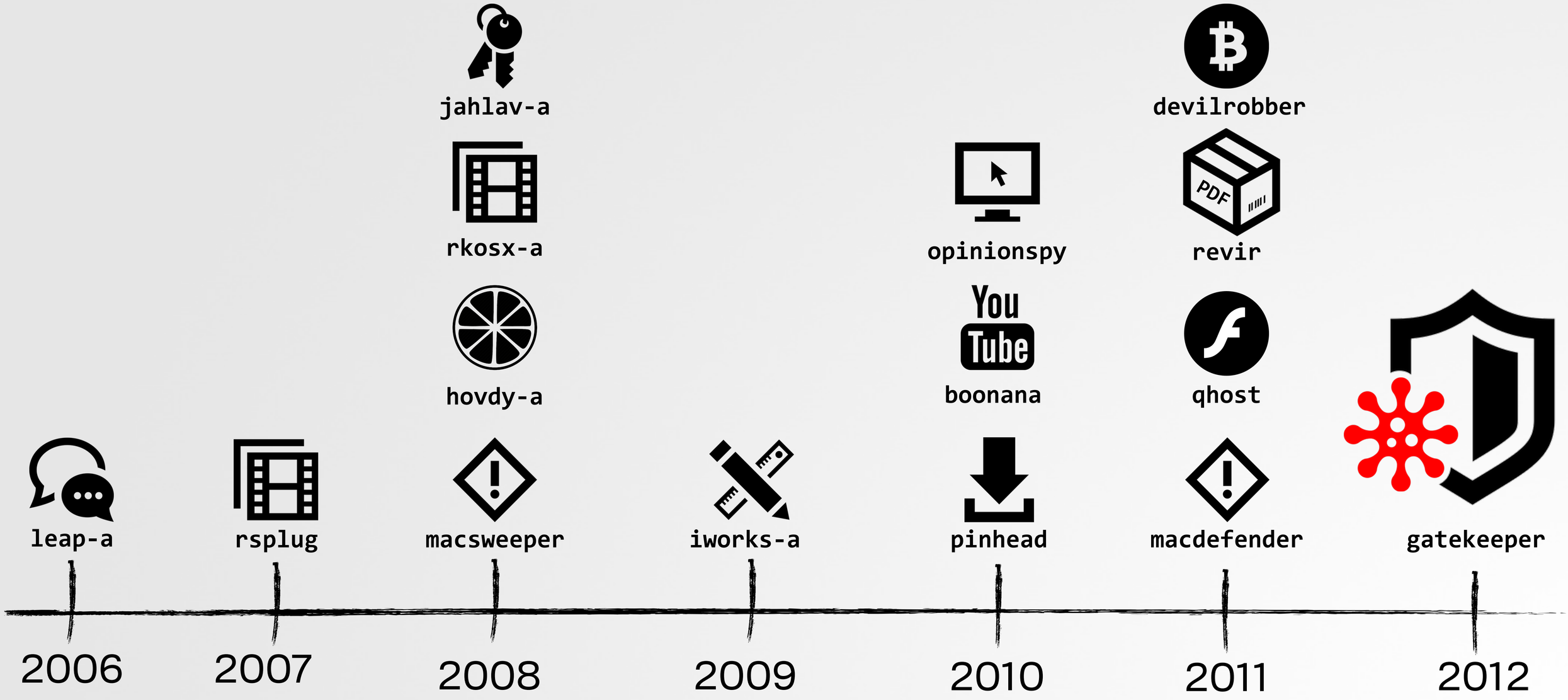


Synack

Life Before Gatekeeper

...os x trojans everywhere? everywhere!

countless OS X users infected



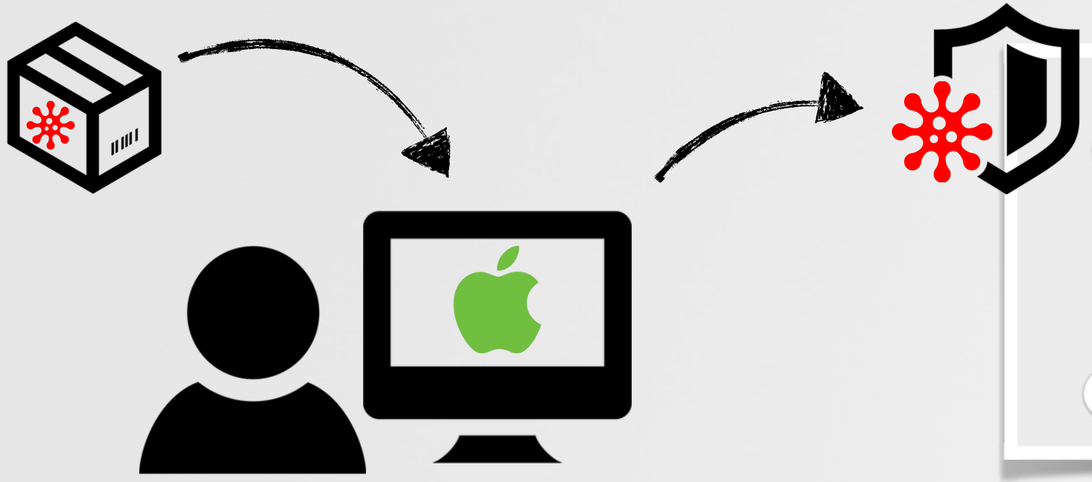
Synack

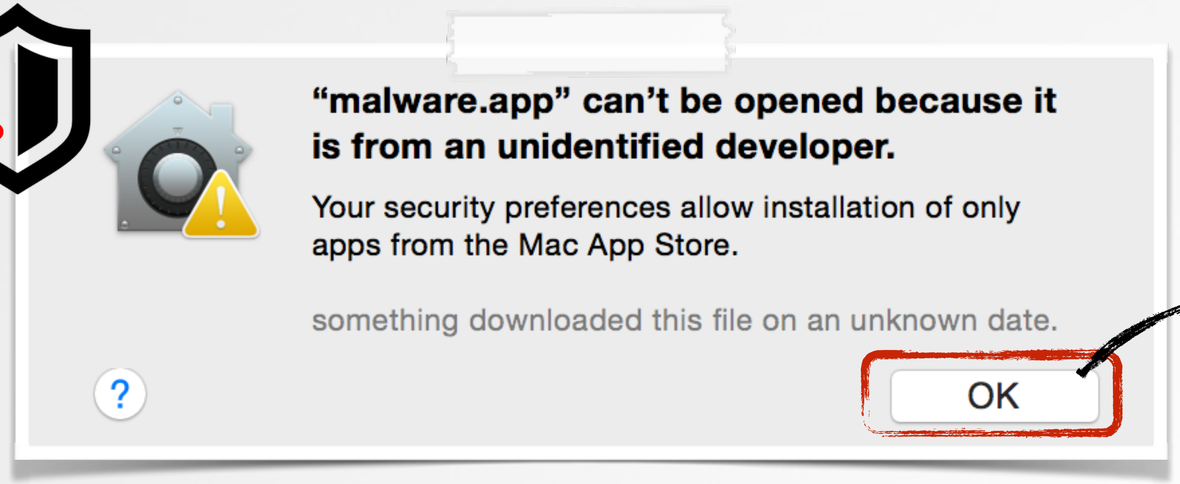
Gatekeeper Aims to Protect

as there is no patch for human stupidity ;)

Gatekeeper is a built-in anti-malware feature of OS X (10.7+)

"*If a [downloaded] app was developed by an unknown developer—one with no Developer ID—or tampered with, Gatekeeper can block the app from being installed*" -apple.com





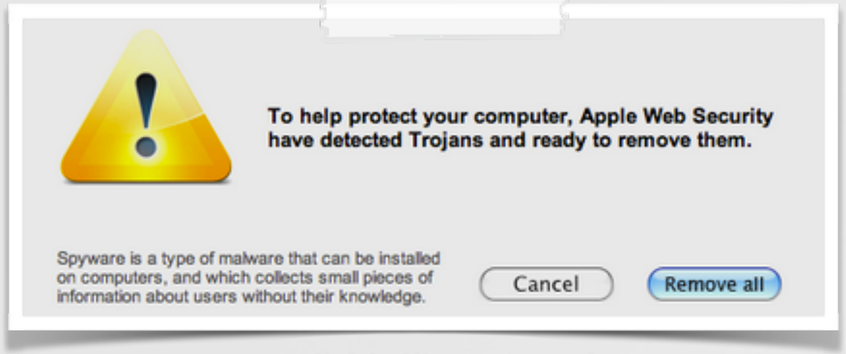
→ TL;DR **block unauthorized code from the internet**

Synack

GATEKEEPER PROTECT USERS

...from low-tech adversaries

"*Gatekeeper Slams the Door on Mac Malware Epidemics*" -tidbits.com

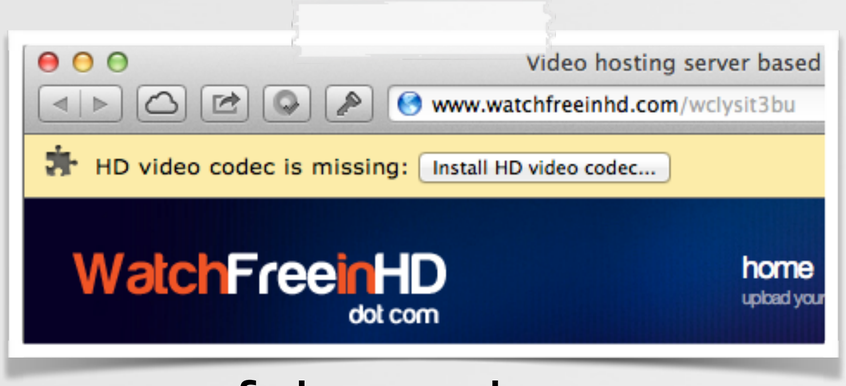


rogue "AV" products

???

poor naive users!

fake installers/updates



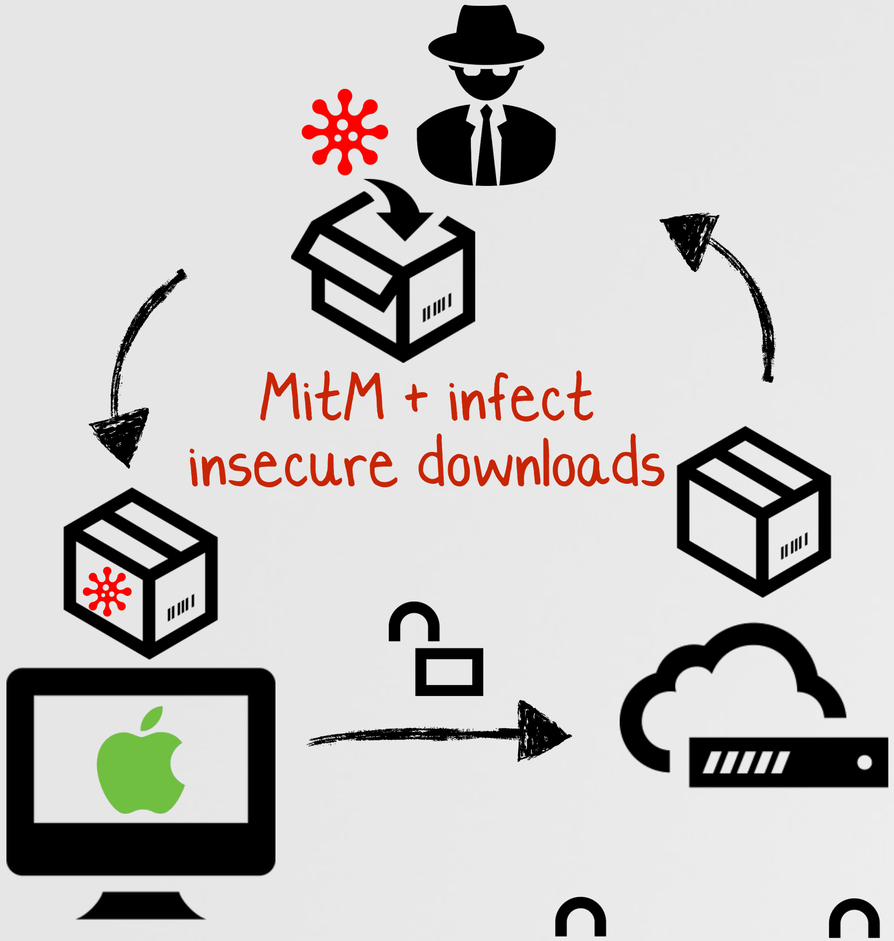
fake codecs

infected torrents

Synack

Gatekeeper Protects Users

...from high-tech adversaries



avast_free_mac_security.dmg
http://download.ff.avast.com/mac/a

bitdefender_antivirus_for_mac.dmg
http://download.bitdefender.com/m

F-Secure-Anti-Virus-for-Mac_JDCQ-
http://download.sp.f-secure.com/SE

LittleSnitch-3.5.1.dmg
http://www.obdev.at/ftp/pub/Produ

savosx_he_r.zip
http://downloads.sophos.com/inst_

eset_cybersecurity_en_.dmg
http://download.eset.com/download

Internet_Security_X8.dmg
http://www.integodownload.com/ma

TrendMicro_MAC_5.0.1149_US-en_T
http://trial.trendmicro.com/US/TM/

NortonSecurity.EnglishTrial.zip
http://buy-download.norton.com/dc

ksm15_0_0_226a_mlg_en_022.dmg
http://downloads-am.kasperskyame

Q1 2015: all security software,
I downloaded -> served over HTTP :(

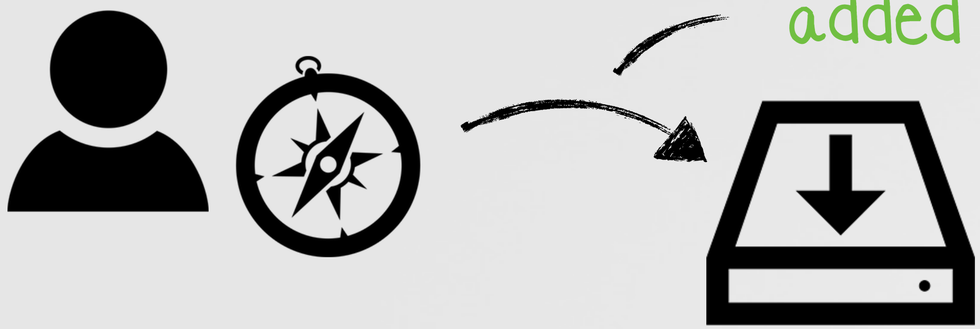
avast!
intego
Avira
KASPERSKY lab
Bitdefender
LittleSnitch
ClamXav
Norton by Symantec
ESET
Sophos
F-Secure
TREND MICRO

my dock

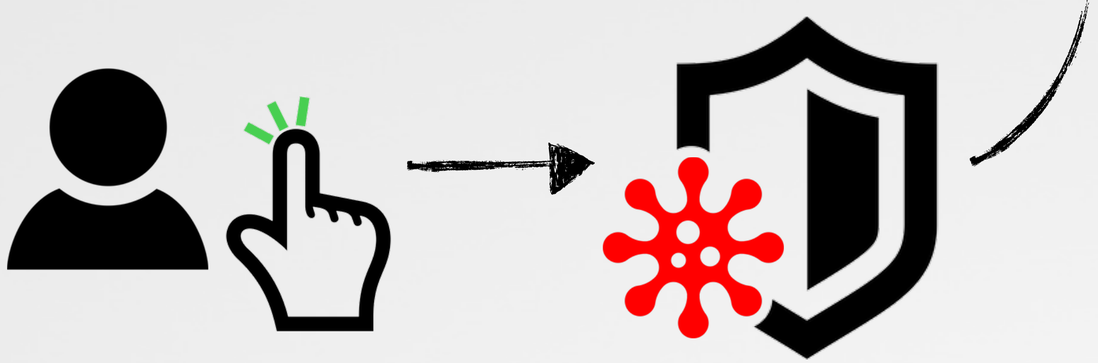
How Gatekeeper Works

an overview

quarantine attribute added



iff quarantine attribute is set!



1

```
//attributes
$ xattr -l ~/Downloads/malware.app
com.apple.quarantine:0001;534e3038;
Safari; B8E3DA59-32F6-4580-8AB3...
```

quarantine attributes

2

Allow apps downloaded from:

- (•) Mac App Store
- ( ) Mac App Store and identified developers
- ( ) Anywhere

gatekeeper settings

3

“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.

gatekeeper in action

Synack

Extended File Attributes

simply put; file metadata

"Mac OS X & iOS Internals"
Jonathan Levin

extended attr. (com.apple.*)	brief details
FinderInfo	information for **Finder.app** (such as folder colors)
metadata	Spotlight data, such as download location & version info
quarantine	indicates that file is from an 'untrusted' source (internet)

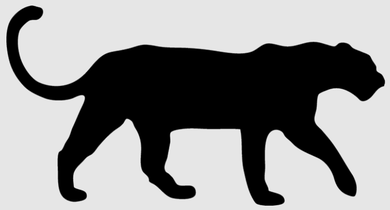
dump w/ **xattr** command

```
$ xattr -l ~/Downloads/eicar.com.txt
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A2 01 02 5F 10 2B 68 74  |bplist00..._.+ht|
00000010  74 70 3A 2F 2F 77 77 77 2E 65 69 63 61 72 2E 6F  |tp://www.eicar.o|
00000020  72 67 2F 64 6F 77 6E 6C 6F 61 64 2F 65 69 63 61  |rg/download/eica|
00000030  72 2E 63 6F 6D 2E 74 78 74 5F 10 27 68 74 74 70  |r.com.txt_.....|

com.apple.quarantine: 0001;55ef7b62;Google Chrome.app;3F2688DE-C34D-4953-8AF1-4F8741FC1326
```

dumping quarantine attributes

'File Quarantine'

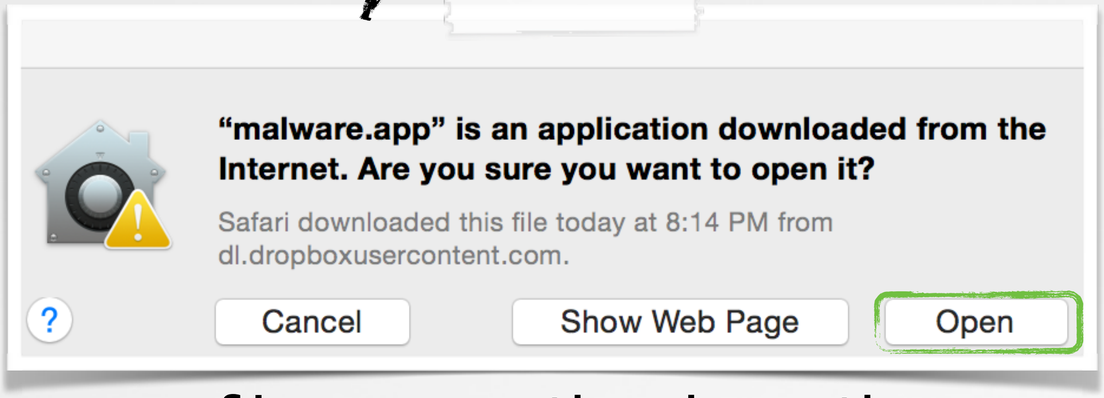
realized by the `com.apple.quarantine` file attribute



added in Leopard

"file from internet"

note; not gatekeeper



file quarantine in action

```
//dictionary for quarantine attributes
NSDictionary* quarantineAttributes = nil;

//get attributes
[fileURL getResourceValue:&quarantineAttributes
         forKey:NSURLQuarantinePropertiesKey error:NULL];
```

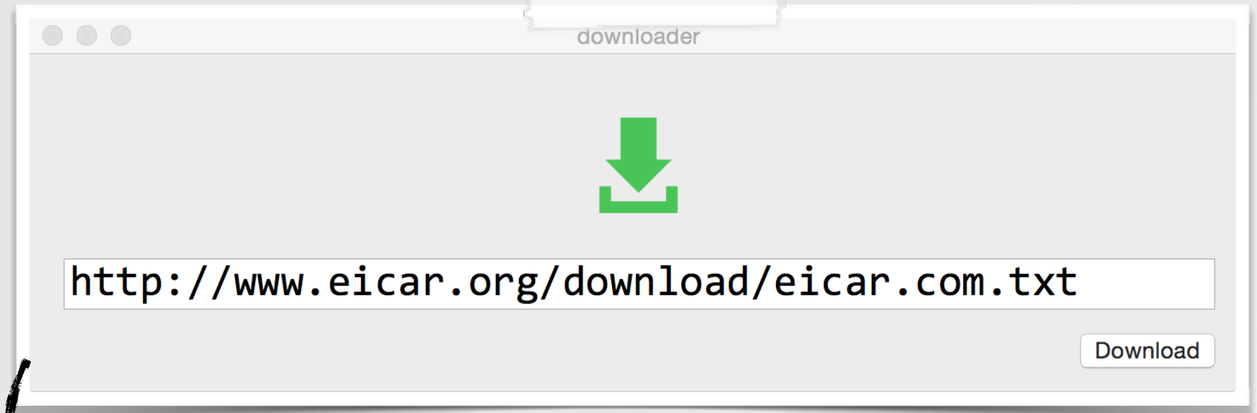
code to get attributes

dumping a file's **`com.apple.quarantine` attribute**

```
$ dumpAttrs ~/Downloads/eicar.com.txt
    LSQuarantineAgentBundleIdentifier = "com.google.Chrome";
    LSQuarantineAgentName = "Google Chrome.app";
    LSQuarantineDataURL = "http://www.eicar.org/download/eicar.com.txt";
    LSQuarantineEventIdentifier = "3F2688DE-C34D-4953-8AF1-4F8741FC1326";
    LSQuarantineOriginURL = "http://www.eicar.org/85-0-Download.html";
    LSQuarantineTimeStamp = "2015-09-09 00:20:50 +0000";
    LSQuarantineType = LSQuarantineTypeWebDownload;
```

Setting The Quarantine Attribute

who done it!?



custom downloader

none; huh?

```
$ xattr -l ~/Downloads/eicar.com.txt
$ dumpAttrs ~/Downloads/eicar.com.txt
$
```

any extended attributes?

```objc
//button handler: download file
-(IBAction)download:(id)sender
{
    //url
    NSURL *remoteFile = [NSURL URLWithString:self.textField.stringValue];

    //local file
    NSString* localFile = [NSString stringWithFormat:@"/tmp/%@", [remoteFile lastPathComponent]];

    //download & save to file
    [[NSData dataWithContentsOfURL:remoteFile] writeToFile:localFile atomically:NO];

    return;
}
```

custom downloader's source code

Setting The Quarantine Attribute

apps can manually add it

consts in **LSQuarantine.h**

```
-(void)setQAttr:(NSString*)localFile
{
    //quarantine attributes dictionary
    NSMutableDictionary* quarantineAttributes = [NSMutableDictionary dictionary];

    //add agent bundle id
    quarantineAttributes[kLSQuarantineAgentBundleIdentifierKey] = [[NSBundle mainBundle] bundleIdentifier];

    //add agent name
    quarantineAttributes[kLSQuarantineAgentNameKey] = [[[NSBundle mainBundle] infoDictionary] objectForKey:kCFBundleNameKey];

    ...

    //manually add quarantine attributes to file
    [[NSURL fileURLWithPath:localFile] setResourceValues:@{NSURLQuarantinePropertiesKey: quarantineAttributes} error:NULL];

    return;
}
```

code to set a file's quarantine attribute

```
$ xattr -l ~/Downloads/eicar.com.txt
com.apple.quarantine:
0000;55efddeb;downloader;ED9BFEA8-10B1-48BA-87AF-623EA7599481

$ dumpAttrs ~/Downloads/eicar.com.txt
    LSQuarantineAgentBundleIdentifier = "com.synack.downloader";
    LSQuarantineAgentName = downloader;
    LSQuarantineDataURL = "http://www.eicar.org/download/eicar.com.txt";
    LSQuarantineEventIdentifier = "ED9BFEA8-10B1-48BA-87AF-623EA7599481";
    LSQuarantineTimeStamp = "2015-09-09 07:21:15 +0000";
    LSQuarantineType = LSQuarantineTypeWebDownload;
```

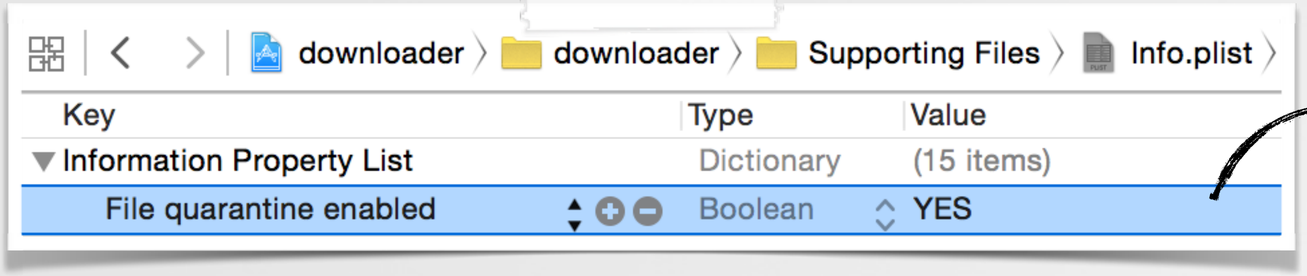
manually set, quarantine attribute

Setting The Quarantine Attribute

or, apps can generically tell the OS to add it

`Info.plist` keys: `LSFileQuarantineEnabled`

"When the value of this key is true, all files created by the application process will be quarantined by OS X" -apple.com




```
$ grep -A 1 LSFileQuarantineEnabled Info.plist
  <key>LSFileQuarantineEnabled</key>
  <true/>
```

app's **`Info.plist`** file updated (**`LSFileQuarantineEnabled`**)

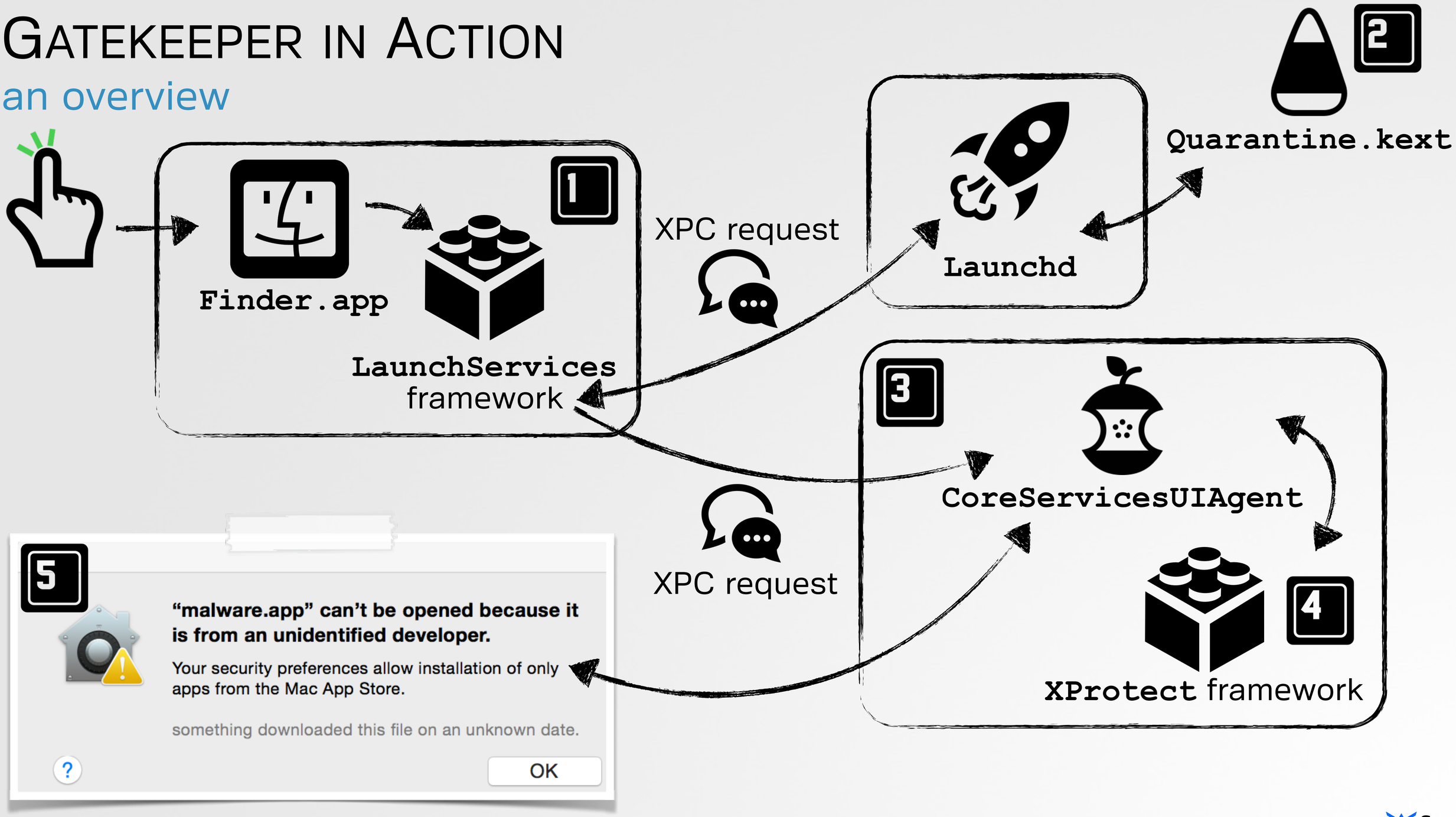
```
$ xattr -l ~/Downloads/eicar.com.txt
com.apple.quarantine: 0000;55f139c4;downloader.app;

$ dumpAttrs ~/Downloads/eicar.com.txt
    LSQuarantineAgentName = "downloader.app";
    LSQuarantineTimeStamp = "2015-09-10 08:05:24 +0000";
```

automatically (OS) set, quarantine attribute

Synack

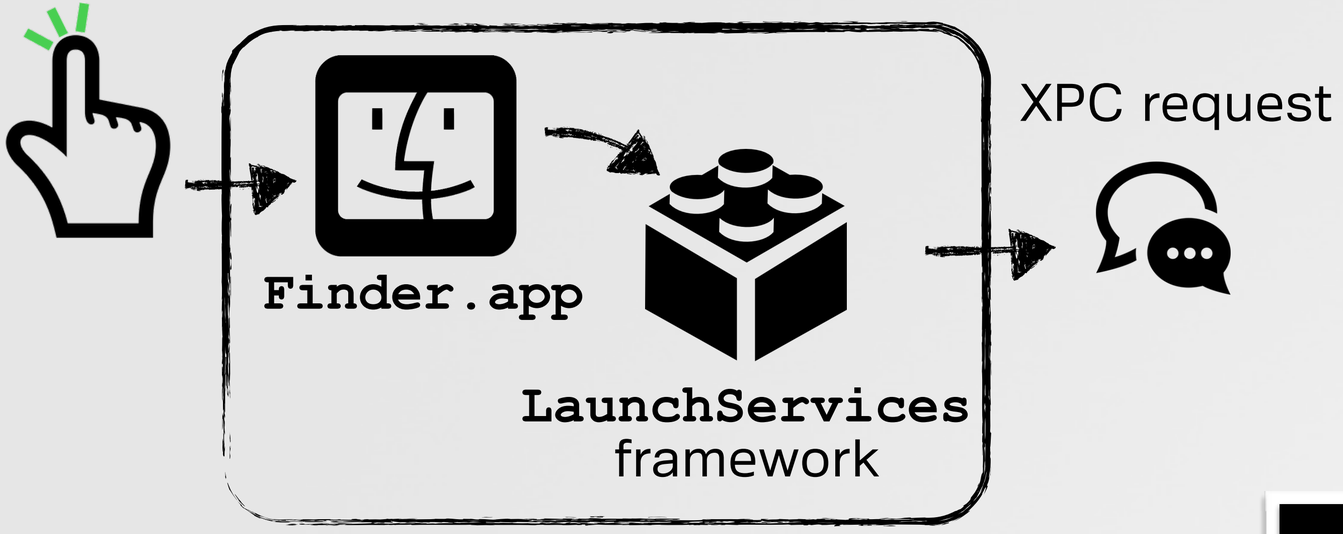
Gatekeeper in Action

an overview



1 Launching the Binary/App

handled by the **launchservices** framework



launch_priv.h

```
pid_t _spawn_via_launchd(
   const char *label,
   const char *const *argv,
   const struct spawn_via_launchd_attr *spawn_attrs,
   int struct_version
);
```

_spawn_via_launchd()

```
libxpc.dylib`_spawn_via_launchd
LaunchServices`LaunchApplicationWithSpawnViaLaunchD
LaunchServices`_LSLaunchApplication
LaunchServices`_LSLaunch
LaunchServices`_LSOpenApp
LaunchServices`_LSOpenStuffCallLocal
LaunchServices`_LSOpenStuff
LaunchServices`_LSOpenURLsWithRole_Common
LaunchServices`LSOpenURLsWithRole
```

call stack

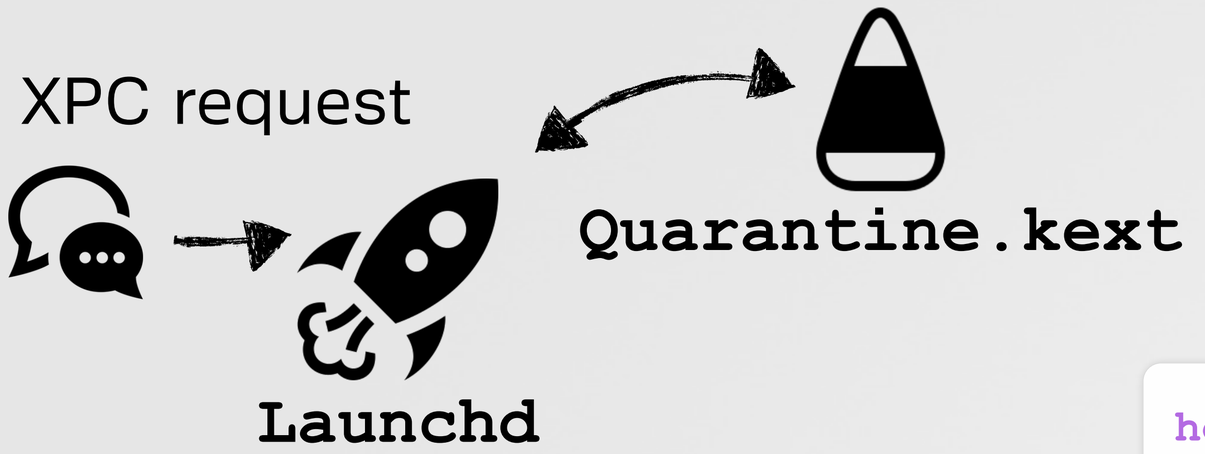
```
(lldb) x/s $rdi
"[0x0-0xb92b92].com.nsa.malware"

(lldb) print *(char**)$rsi
"~/Downloads/Malware.app/Contents/MacOS/Malware"

(lldb)print *(struct spawn_via_launchd_attr*)$rdx
{
    spawn_flags = SPAWN_VIA_LAUNCHD_STOPPED
    ...
}
```

'spawn' attributes, etc.

Synack

2 Policy Enforcement with Quarantine.kext

kernel-mode mac component




```
(lldb) print *(struct mac_policy_conf*)0xFFFFFF7F8B447110
  mpc_name = 0xffffff7f8b446c3a "Quarantine"
  mpc_fullname = 0xffffff7f8b446cb0 "Quarantine policy"
  ...
```

quarantine policy

```
Quarantine`hook_vnode_check_exec
kernel`mac_vnode_check_exec
kernel`exec_activate_image
kernel`exec_activate_image
kernel`posix_spawn
kernel`unix_syscall64
kernel`hndl_unix_scall64
```

call stack

```
hook_vnode_check_exec

    //bail if sandbox'ing not enforced
    cmp     cs:_sandbox_enforce, 0
    jz      leaveFunction

    //bail if file previously approved
    call    _quarantine_get_flags
    and     eax, 40h
    jnz     leaveFunction

    //bail if file is on read-only file system
    call    _vfs_flags          ; mnt flags
    test    al, MNT_RDONLY
    jnz     leaveFunction
```

hook_vnode_check_exec

Synack

3 User Interaction via CoreServicesUIAgent

first, the xpc request

LaunchServices framework

XPC request

CoreServicesUIAgent

pseudo code

```
void ____LSAgentGetConnection_block_invoke(void * _block)
{
  rax = xpc_connection_create_mach_service("com.apple.coreservices.quarantine-resolver",
            dispatch_get_global_queue(0x0, 0x0), 0x0);

  xpc_connection_set_event_handler(rax, void ^(void * _block, void * arg1)
  {
        return;
  });

  xpc_connection_resume(rax);
  return;
}
```

getting XPC connection to **CoreServicesUIAgent**

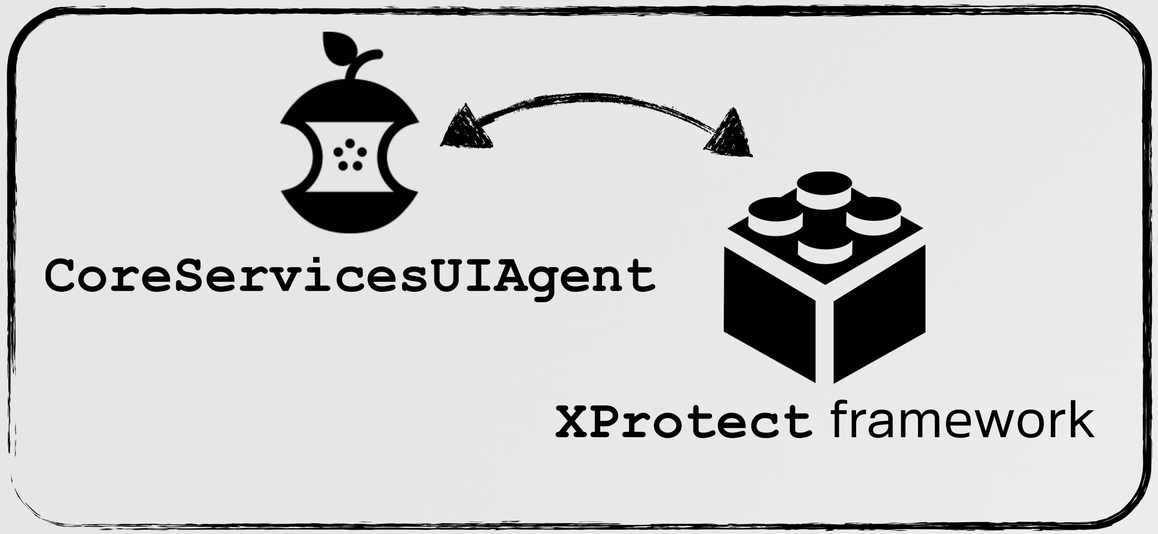
```
(lldb) po $rax
{
    LSQAllowUnsigned = 0;
    LSQAppPSN = 3621748;
    LSQAppPath = "/Users/patrick/Downloads/Malware.app";
    LSQAuthorization = <bed76627 c7cc0ae4 a6860100 00000000 ...
    LSQRiskCategory = LSRiskCategoryUnsafeExecutable;
}
```

XPC message contents

Synack

4 User Interaction via CoreServicesUIAgent

then, analysis via xprotect




```
-[CSUIController handleIncomingXPCMessage:clientConnection:]
  -[GKQuarantineResolver resolve]
   -[GKQuarantineResolver malwareChecksBegin]
     -[GKQuarantineResolver malwareCheckNextItem]
        mov rdi, cs:classRef_XProtectAnalysis
        mov rsi, cs:selRef_alloc
        call     r15  ; _objc_msgSend
        mov      rdi, rax
        mov      rsi, cs:selRef_initWithURL_
        mov      rdx, r14 ;path to app
        call     r15  ; _objc_msgSend

      -[XProtectAnalysis
      beginAnalysisWithDelegate:didEndSelector:contextInfo:]
        +[WorkerThreadClass threadEntry:]
          mov      rdi, [rbp+staticCodeRef]
          lea      rdx, [rbp+signingInfo]
          xor      esi, esi ;flags
          call     _SecCodeCopySigningInformation
```

```
(lldb) po $rdi
{
  FileURL = "file:///Users/patrick/Downloads/Malware.app";
  ShouldShowMalwareSubmission = 0;
  XProtectCaspianContext =     {
        "context:qtnflags" = 33;
        operation = "operation:execute";
  };
  XProtectDetectionType = 3;
  XProtectMalwareType = 2;
}
```

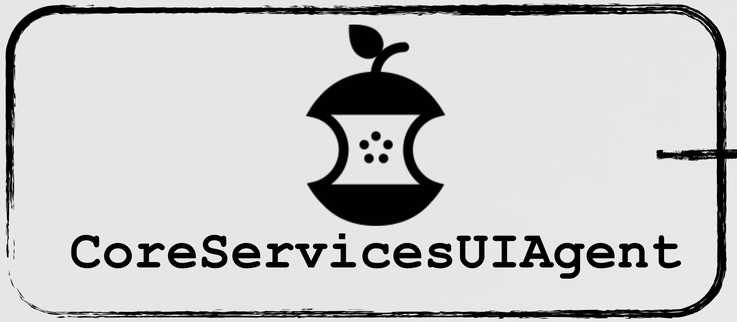
program control flow

XProtectMalwareType	meaning
0x2	unsigned
0x3	modified bundle
0x5	signed app
0x7	modified app

Synack

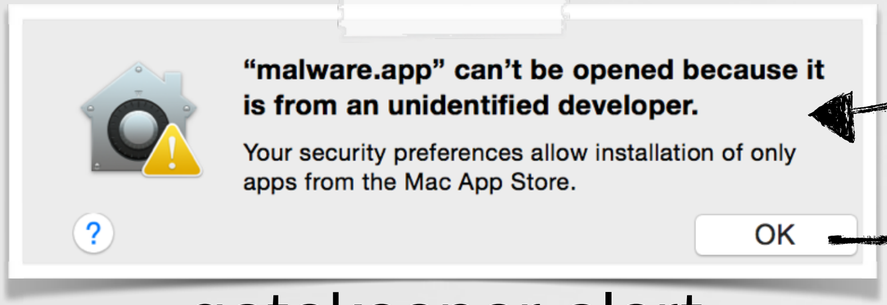
5 User Interaction via CoreServicesUIAgent

finally, display the alert




```
-[GKQuarantineResolver showGKAlertForPath:]
  -[GKQuarantineResolver alertForPath:malwareInfo:]

    mov     rax, _OBJC_IVAR_$_GKQuarantineResolver__allowUnsigned
    mov     rcx, [rbp+GKQuarantineResolver]
    cmp     byte ptr [rcx+rax], 0

    lea     rdi, cfstr_Q_headline_cas ; "Q_HEADLINE_CASPIAN_BAD_DISTRIBUTOR"

    mov     rdi, cs:classRef_NSAlert
    mov     rsi, cs:selRef_alloc
    call    r12  ; _objc_msgSend
```

alert customization



gatekeeper alert

```
mov     rsi, cs:selRef_deny
mov     rdi, r14
call    cs:_objc_msgSend_ptr

-[GKQuarantineResolver deny]
  -[GKQuarantineResolver denyWithoutSettingState]

    mov     rax, _OBJC_IVAR_$_GKQuarantineResolver__appASN
    mov     rsi, [rdi+rax]
    mov     edi, 0FFFFFFFEh
    mov     edx, 2
    call    __LSKillApplication
```

application termination

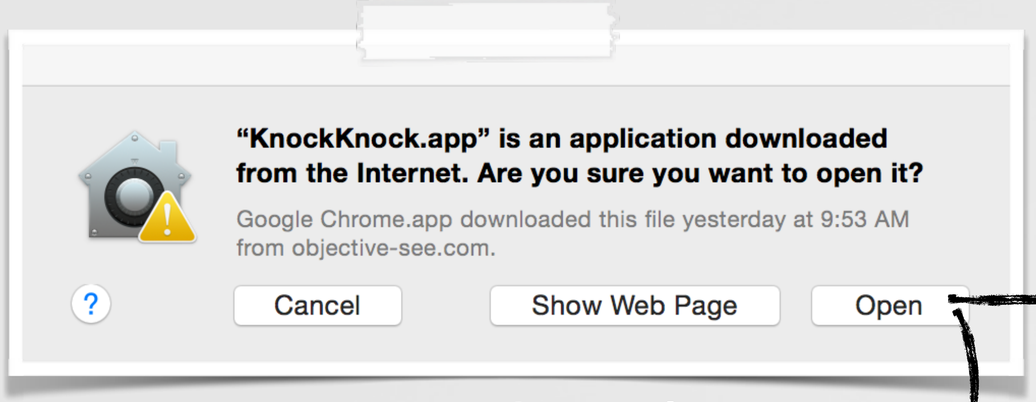
```
$ less QuarantineHeadlines.strings
<key>Q_HEADLINE_CASPIAN_BAD_DISTRIBUTOR</key>
<string>
  "%@" can't be opened because it is from an unidentified developer.
</string>
<key>Q_HEADLINE_CASPIAN_BLOCKED</key>
<string>
  "%@" can't be opened because it was not downloaded from the Mac App Store.
</string>
```

alert strings (`QuarantineHeadlines.strings`)

Synack

What if the App Conforms & is Allowed by the User?

quarantine attributes updated, then application resumed



quarantine alert

```
-[GKQuarantineResolver
approveUpdatingQuarantineTarget:recursively:volume:]

call      __qtn_file_get_flags
or        eax, 40h
mov       rdi, [rbp+var_B8]
mov       esi, eax
call      __qtn_file_set_flags
```

updating quarantine attributes

```
mov       rsi, [r13+r14+0]
mov       rax, __kLSApplicationInStoppedStateKey_ptr
mov       rdx, [rax]
mov       edi, 0FFFFFFFEh
xor       r8d, r8d
mov       rcx, rbx
call      __LSSetApplicationInformationItem

;on error
lea       rsi, "Unable to continue stopped application"
mov       edi, 4
xor       eax, eax
mov       edx, ecx
call      logError
```

resuming application

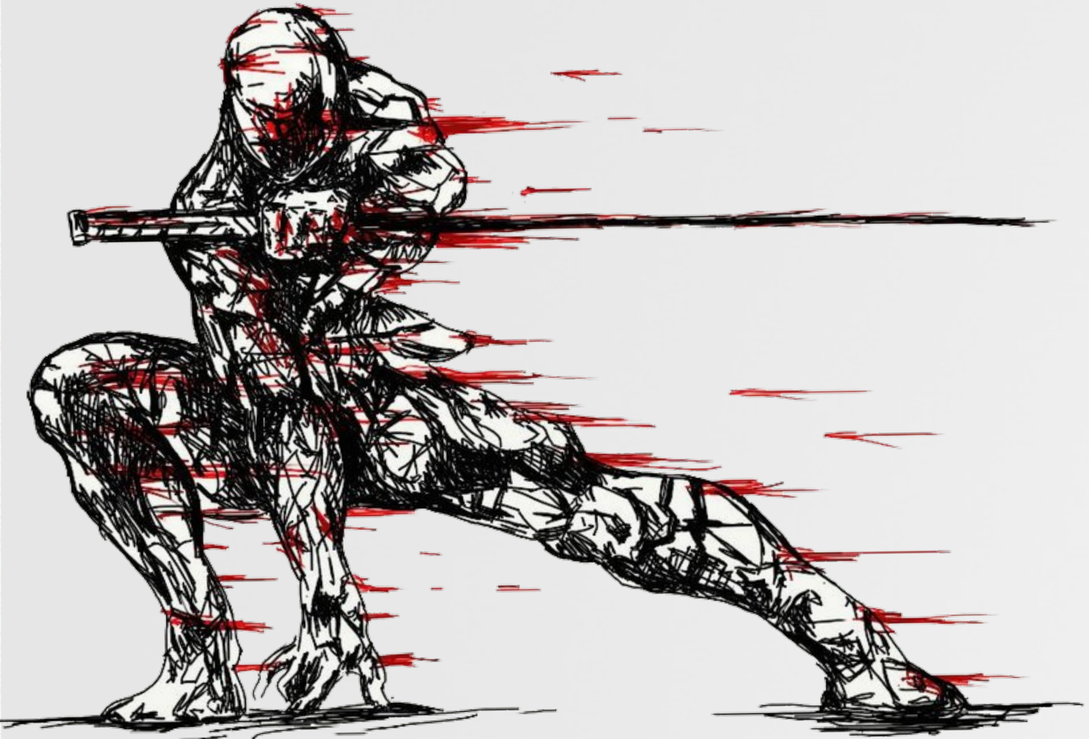
```
$ xattr -l ~/Downloads/KnockKnock.app/Contents/MacOS/KnockKnock
com.apple.quarantine: 0001 55f3313d;Google\x20Chrome.app;FBF45932...

$ xattr -l ~/Downloads/KnockKnock.app/Contents/MacOS/KnockKnock
com.apple.quarantine: 0041 55f3313d;Google\x20Chrome.app;FBF45932...
```

before & after

Synack

BYPASSING GATEKEEPER

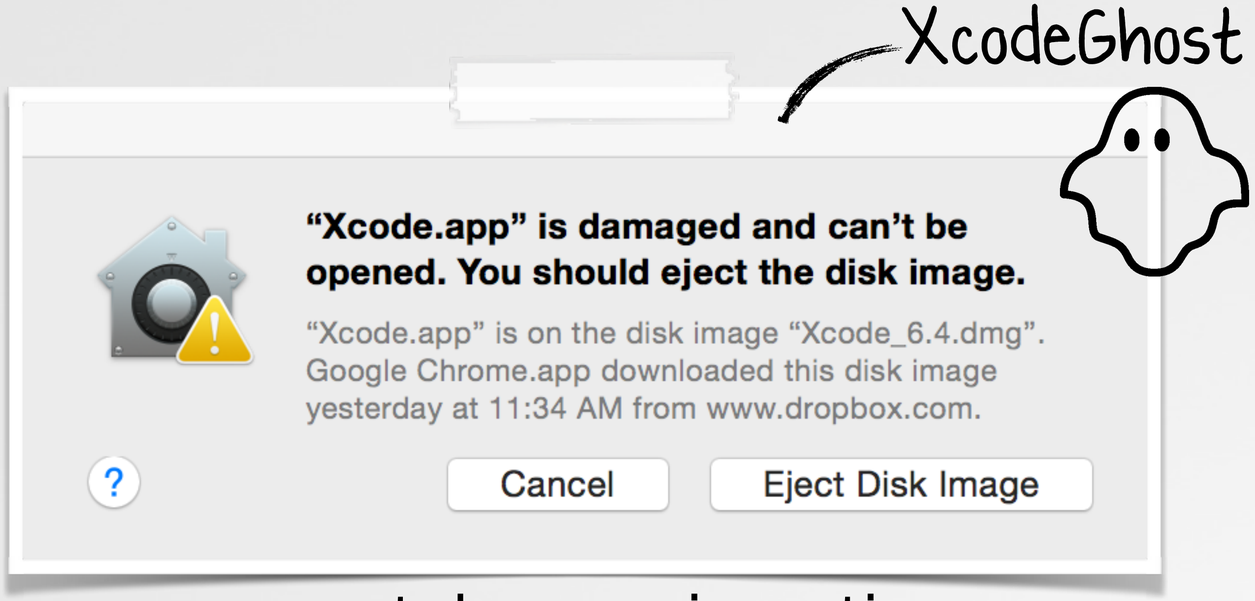
unsigned code allowed!?



Synack

Recall; Gatekeeper Aims to Protect

...unauthorized code should be blocked!

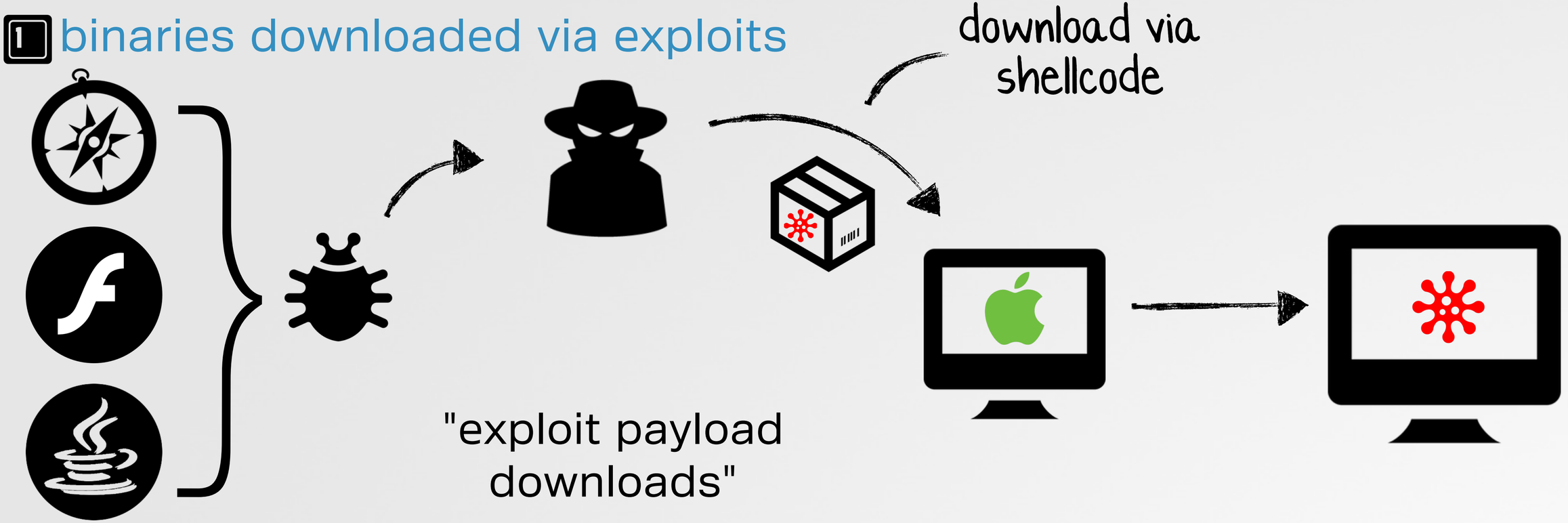


gatekeeper in action

block unauthorized code from the internet

Gatekeeper Shortcomings

1. binaries downloaded via exploits

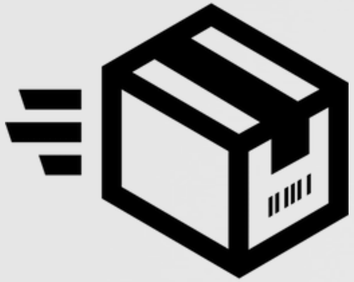


"*malware that comes onto the system through vulnerabilities...bypass quarantine entirely. The infamous Flashback malware, for example, used Java vulnerabilities to copy executable files into the system. Since this was done behind the scenes, out of view of quarantine, those executables were able to run without any user interactions*" -www.thesafemac.com

Gatekeeper Shortcomings

2 downloading app, must 'support' quarantine attribute

attribute added?



→ virus BULLETIN **vb201410-iWorm.pdf**

uTorrent

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

iWorm infected applications

```
$ xattr -p com.apple.quarantine Adobe\ Photoshop\ CC\ 2014.dmg
xattr: Adobe Photoshop CC 2014.dmg: No such xattr: com.apple.quarantine
```

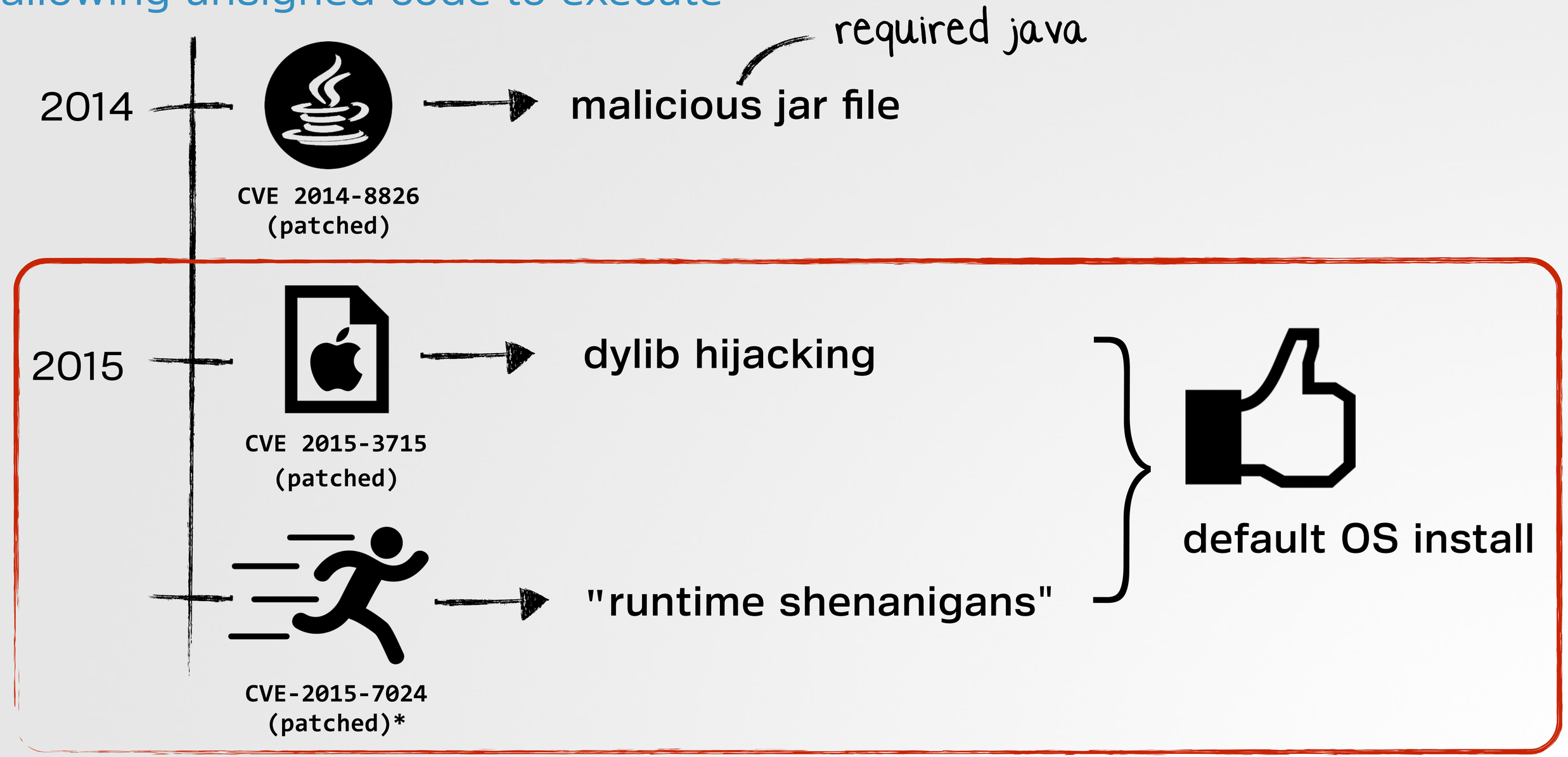
no quarantine attribute :(

"*the quarantine system relies on the app being used for downloading doing things properly. Not all do, and this can result in the quarantine flag not being set on downloaded files*" -www.thesafemac.com

Synack

Gatekeeper Bypasses

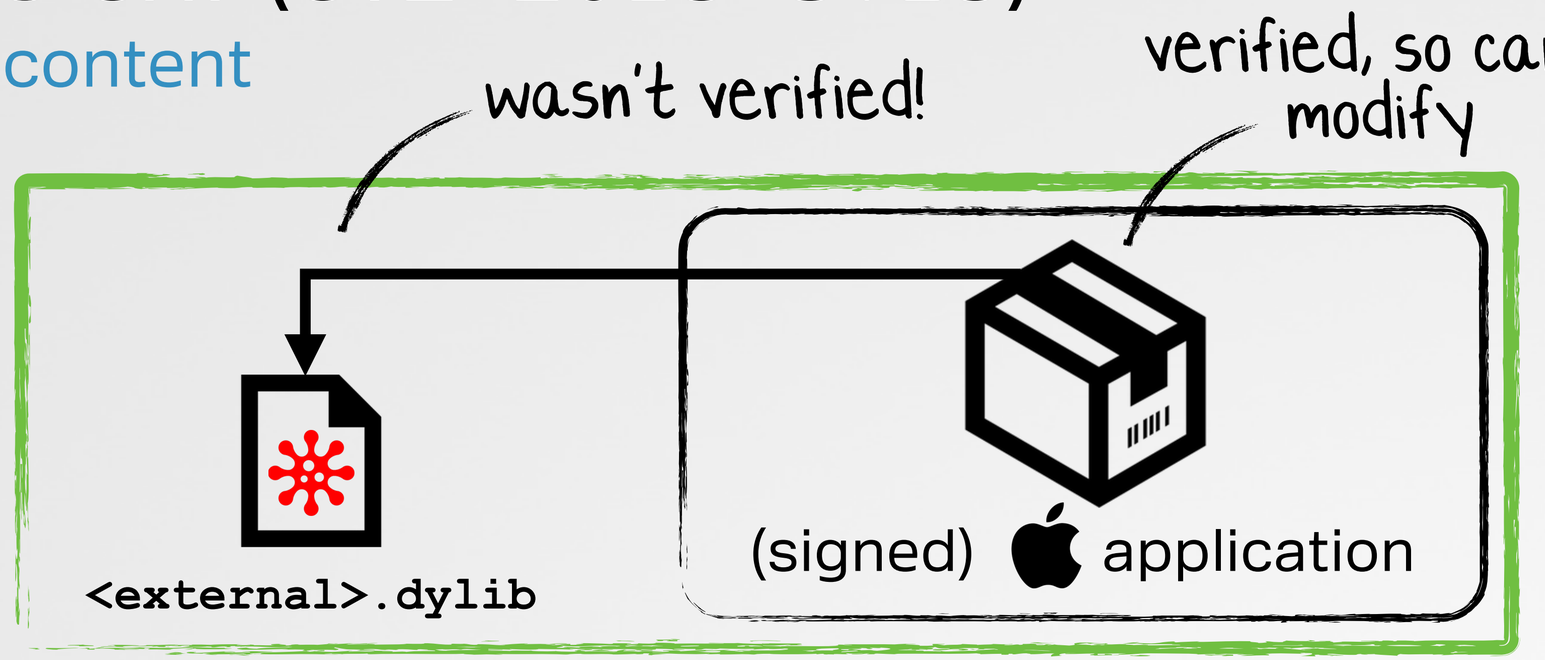
allowing unsigned code to execute



Gatekeeper Bypass 0x1 (CVE 2015-3715)

(dylib) hijacking external content

gatekeeper **only** verified the app bundle!



.dmg/.zip layout

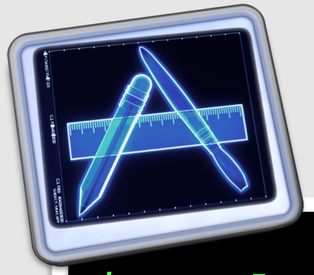
1. find an signed app that contains an **external, relative dependency** to a hijackable dylib
2. create a .dmg/.zip with the necessary folder structure (i.e. placing the malicious dylib in the **externally** referenced location)
3. host online or inject

→ virus BULLETIN white paper **www.virusbtn.com/dylib**

Synack

Gatekeeper Bypass 0x1 (CVE 2015-3715)

1. a signed app that contains an external dependency to hijackable dylib



`spctl` tells you if gatekeeper will accept the app

```
$ spctl -vat execute /Applications/Xcode.app/Contents/Applications/Instruments.app
Instruments.app: accepted
source=Apple System
```

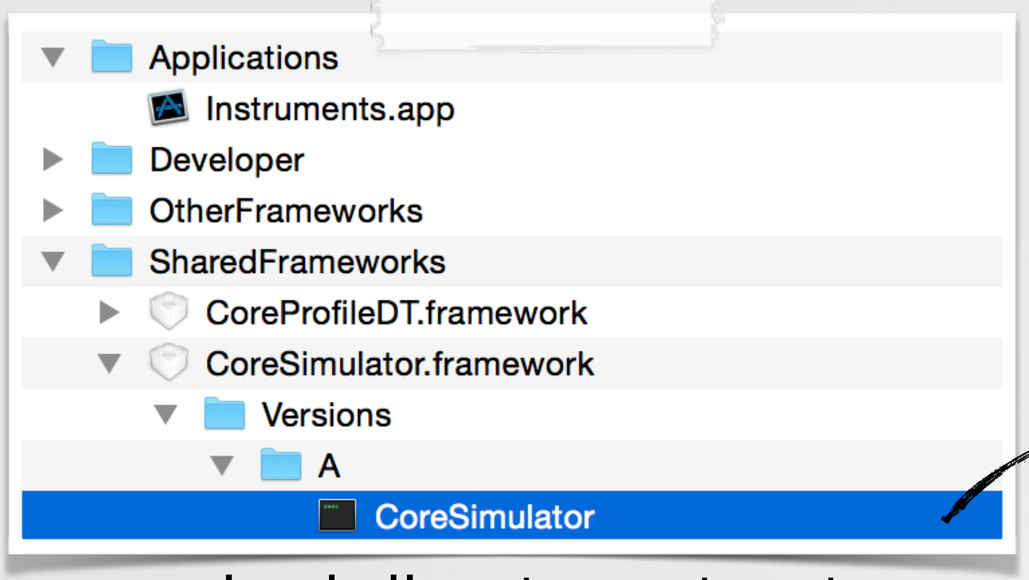
```
$ otool -l Instruments.app/Contents/MacOS/Instruments

Load command 16
          cmd LC_LOAD_WEAK_DYLIB
         name @rpath/CoreSimulator.framework/Versions/A/CoreSimulator

Load command 30
          cmd LC_RPATH
         path @executable_path/../../../../SharedFrameworks
```

`Instruments.app` - fit's the bill

Synack

Gatekeeper Bypass 0x1 (CVE 2015-3715)

2 create a .dmg with the necessary layout



required directory structure

'clean up' the .dmg

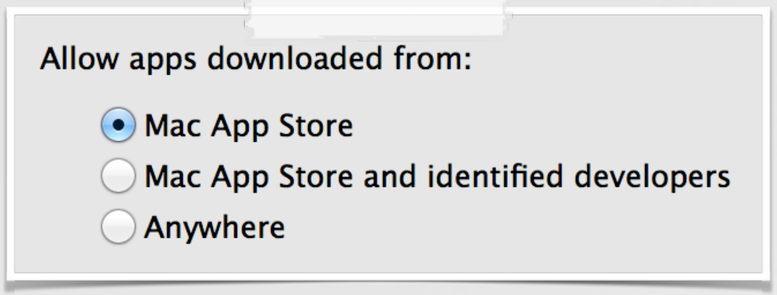
- hide files/folder
- set top-level alias to app
- change icon & background
- make read-only

(deployable) malicious .dmg

Synack

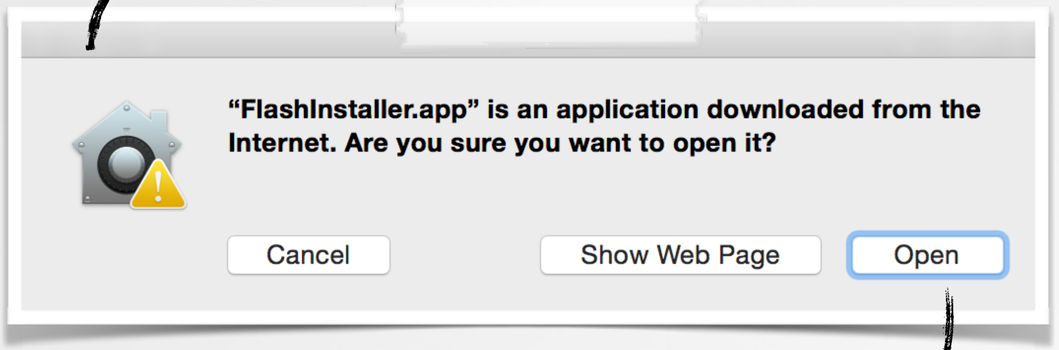
Gatekeeper Bypass 0x1 (CVE 2015-3715)

3 host online or inject into downloads



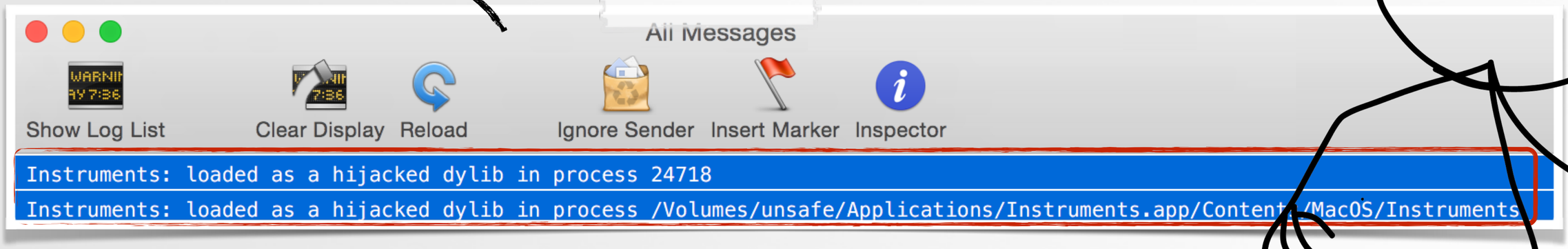
gatekeeper setting's (maximum)

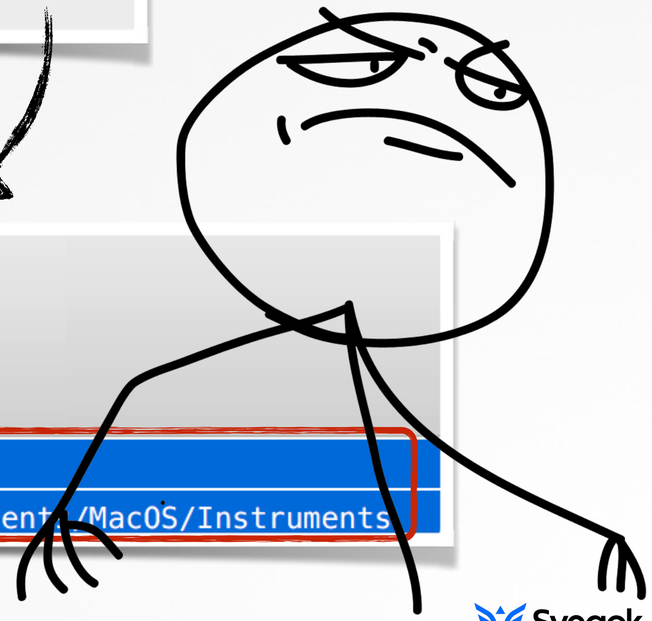
quarantine popup (anything downloaded)



quarantine alert

unsigned (non-Mac App Store) code execution!!





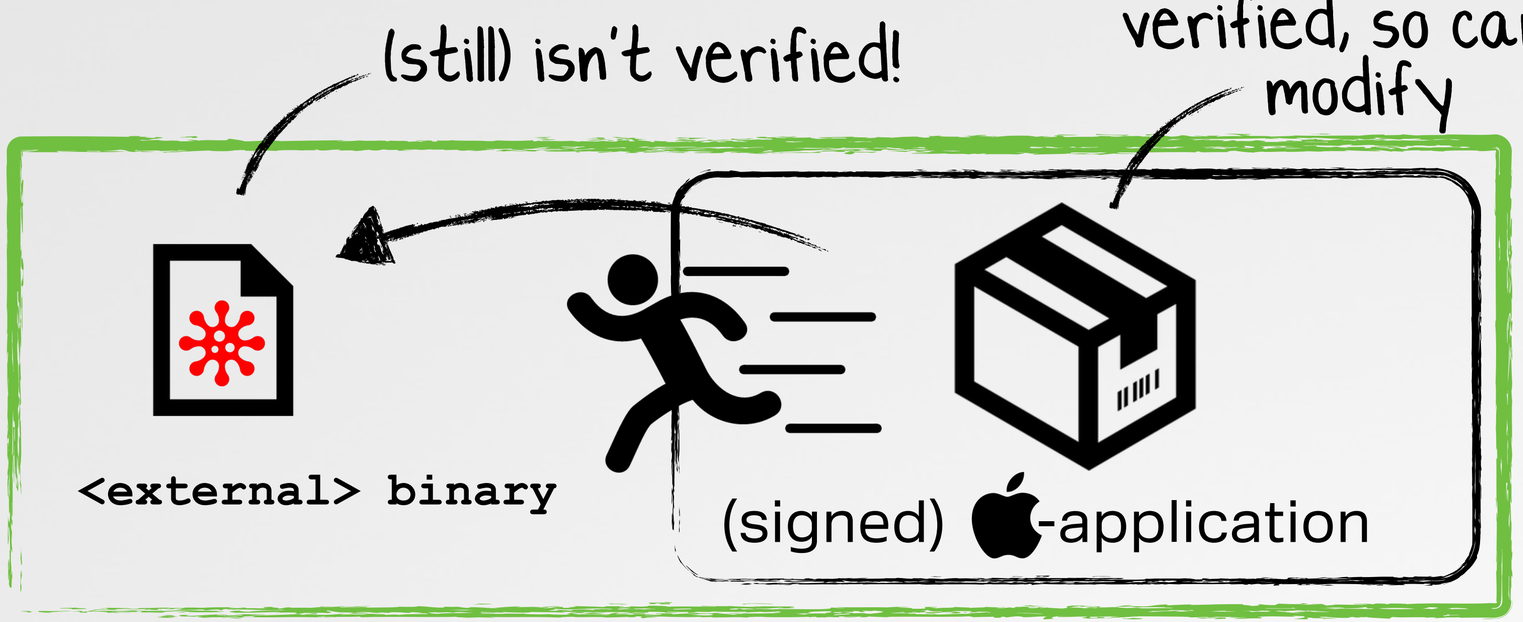
gatekeeper bypass :)

Synack

Gatekeeper Bypass 0x2 (CVE 2015-7024)

runtime shenanigans

gatekeeper only **statically** verifies the app bundle!



.dmg/.zip layout

1. find any signed app that **at runtime**, executes a '**relatively external**' binary
2. create a .dmg/.zip with the necessary folder structure (i.e. placing the malicious binary in the **externally** referenced location)
3. host online/inject into insecure downloads

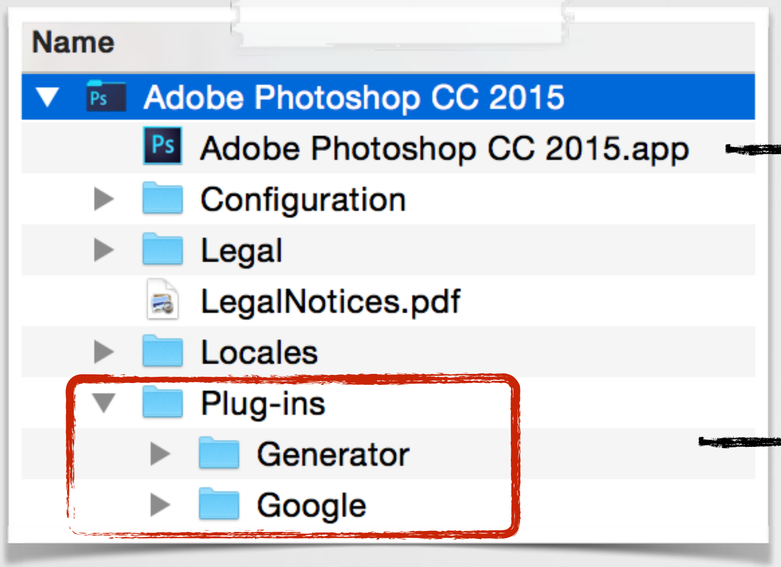
Gatekeeper Bypass 0x2 (CVE 2015-7024)

example 1: Adobe (Photoshop, etc)

3rd party plugins, etc.
-> go outside the bundle!

Q: Can I add/modify files in my signed (app) bundle?

A: "*This is no longer allowed. If you must modify your bundle, do it before signing. If you modify a signed bundle, you must re-sign it afterwards. Write data into files outside the bundle*" -apple.com



Adobe Photoshop

app bundle validates!

not validated

```
NSString* pluginDir = APPS_DIR + @"../Plug-ins";

for(NSString* plugins in pluginDir)
{
    //load plugin dylib
    // ->not validated, can unsigned
}
```

plugin loading pseudo code

Synack

Gatekeeper Bypass 0x2 (CVE 2015-7024)

example 2: Apple (ictool)

```
//execute ibtoold
void IBExecDirectly()
{
     //build path to ibtool
     ibToolPath = IBCopyServerExecutablePath()

     //exec ibtoold
     execv(ibToolPath, ....)
}

//build path to ibtoold
char* IBCopyServerExecutablePath()
{
     //get full path to self (ictool)
     icToolPath = IBCopyExecutablePath()

    //remove file component
    icToolDir = IBCreateDirectoryFromPath(exePath)

    //add 'ibtool'
    ibToolPath = IBCreatePathByAppendingPathComponent
                  (icToolDir, "ibtoold")

    return ibToolPath
}
```

ictool's pseudo code

```
$ spctl -vat execute Xcode.app/Contents/Developer/usr/bin/ictool
Xcode.app/Contents/Developer/usr/bin/ictool: accepted
source=Apple System
```

gatekeeper, happy with **ictool**

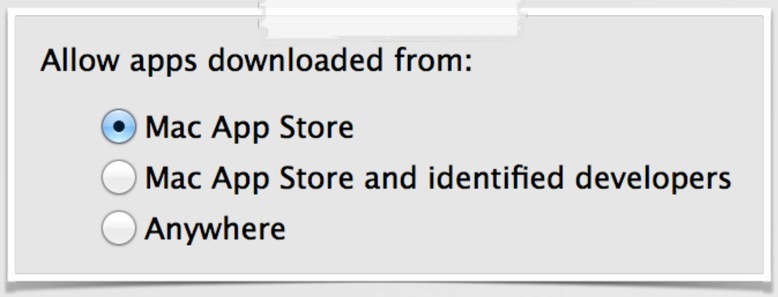
```
$ xattr -l *
ibtoold: com.apple.quarantine: 0001;55ee3be6;Google\x20Chrome.app
ictool: com.apple.quarantine: 0001;55ee3be6;Google\x20Chrome.app

$ codesign -dvv ibtoold
ibtoold: code object is not signed at all
```

...but **ibtoold** is unsigned

Synack

GATEKEEPER BYPASS 0X2 (CVE 2015-7024)

example 2: Apple (ictool)



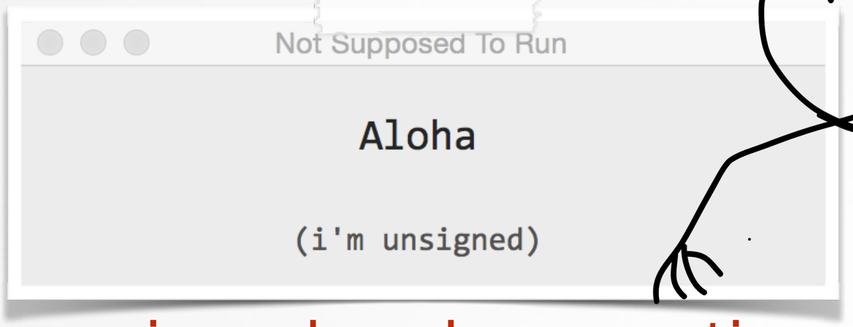
gatekeeper setting's (max.)

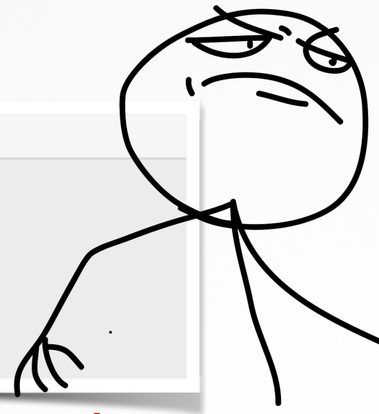
.dmg setup

1. alias to **'update.app' (ictool)**
   ...name & icon attacker controlled
2. apple-signed **'update.app' (ictool)**
   .app extension prevents **Terminal.app** popup

hide

3. unsigned **ibtoold**
   command-line executable
4. unsigned application



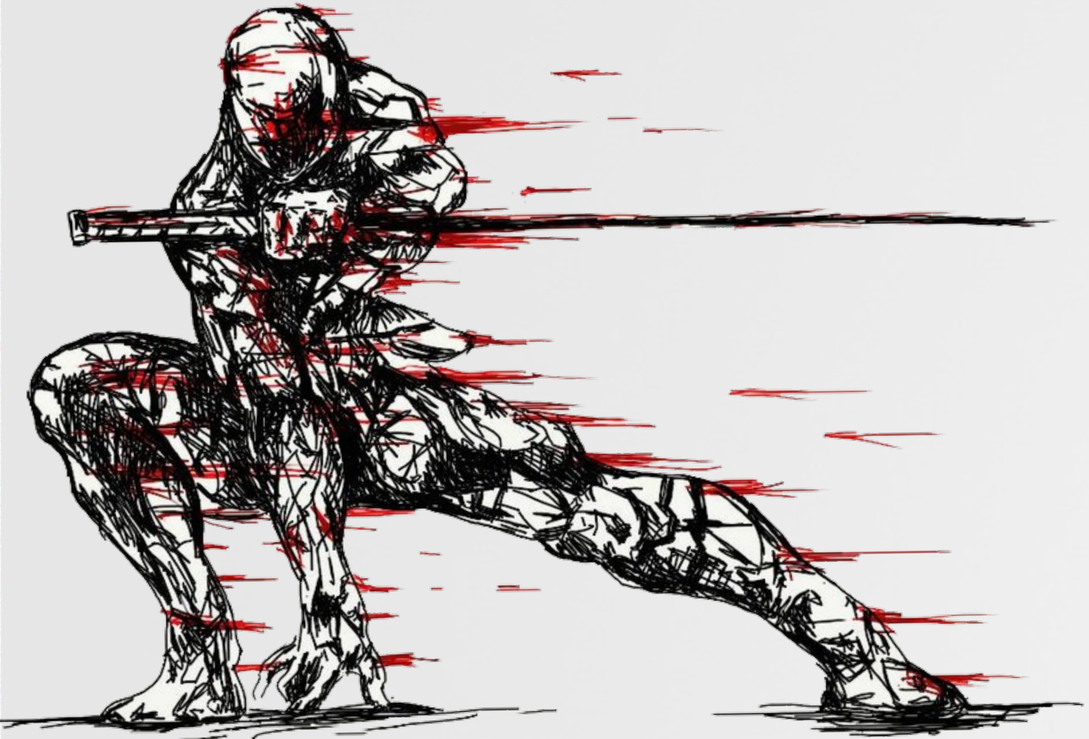


unsigned code execution

Synack

FIXING GATEKEEPER

'patches' & runtime validation



Synack

PATCHES **CVE 2015-3715/2015-7024**

both bypasses now "patched"

- **Security**

  Available for: OS X Mountain Lion v10.8.5, OS X Mavericks v10.9.5, OS X Yosemite v10.10 to v10.10.3

  Impact: A malicious application may be able to bypass code signing checks

  Description: An issue existed where code signing did not verify libraries loaded outside the application bundle. This issue was addressed with improved bundle verification.

  CVE-ID

  CVE-2015–3715 : Patrick Wardle of Synack

CVE 2015-3715
patched in OS X 10.10.4

- **Security**

  Available for: OS X Mavericks v10.9.5, OS X Yosemite v10.10.5, and OS X El Capitan 10.11

  Impact: An Apple-signed binary could be used to load arbitrary files

  Description: Certain Apple-signed executables loaded applications from relative locations. This was addressed through additional checks in Gatekeeper.

  CVE-ID

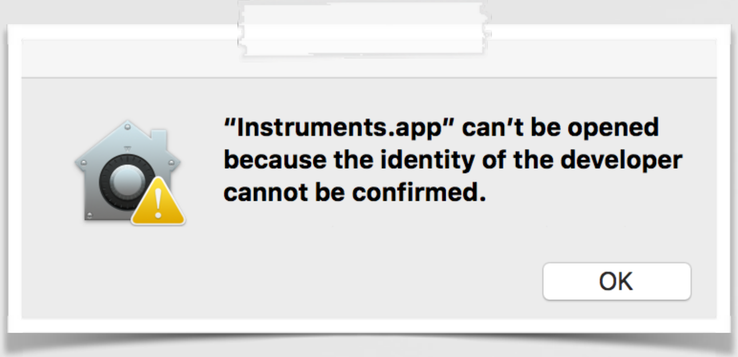
  CVE-2015–7024 : Patrick Wardle of Synack

CVE 2015-7024
patched in OS X 10.11.1

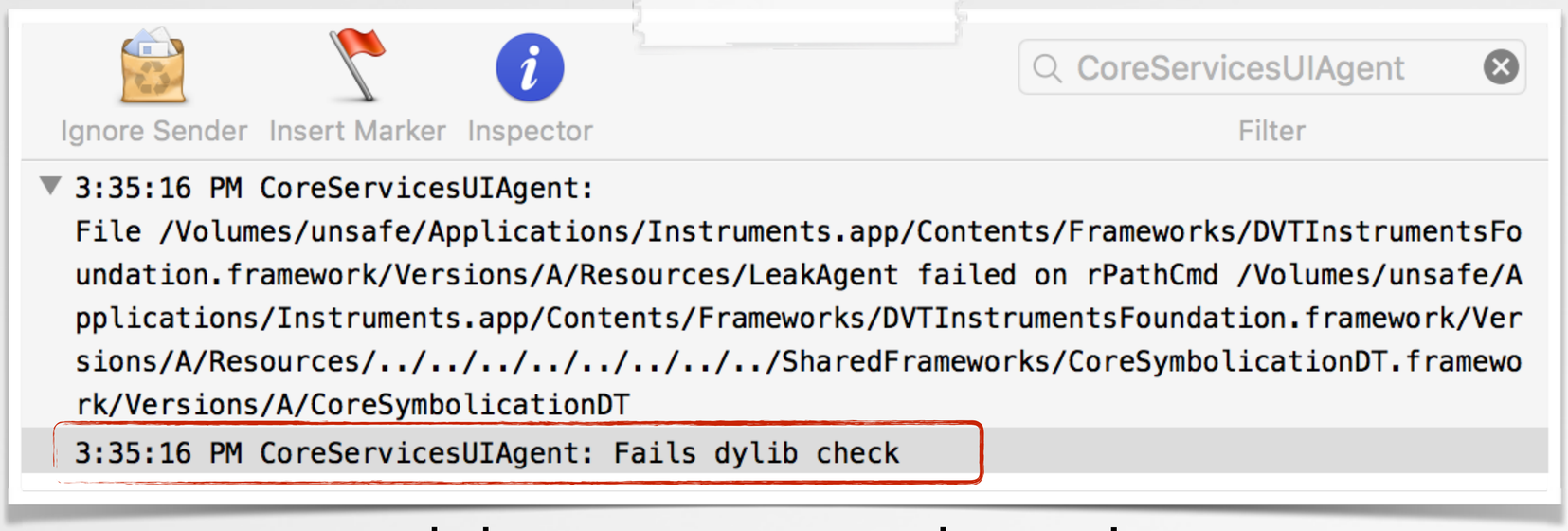
Synack

PATCHING CVE 2015-3715

external dylibs; verified

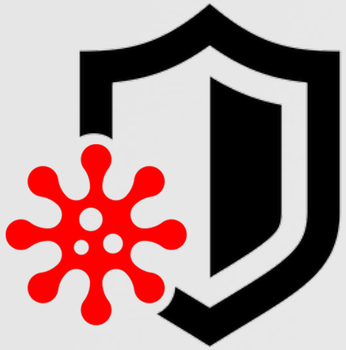


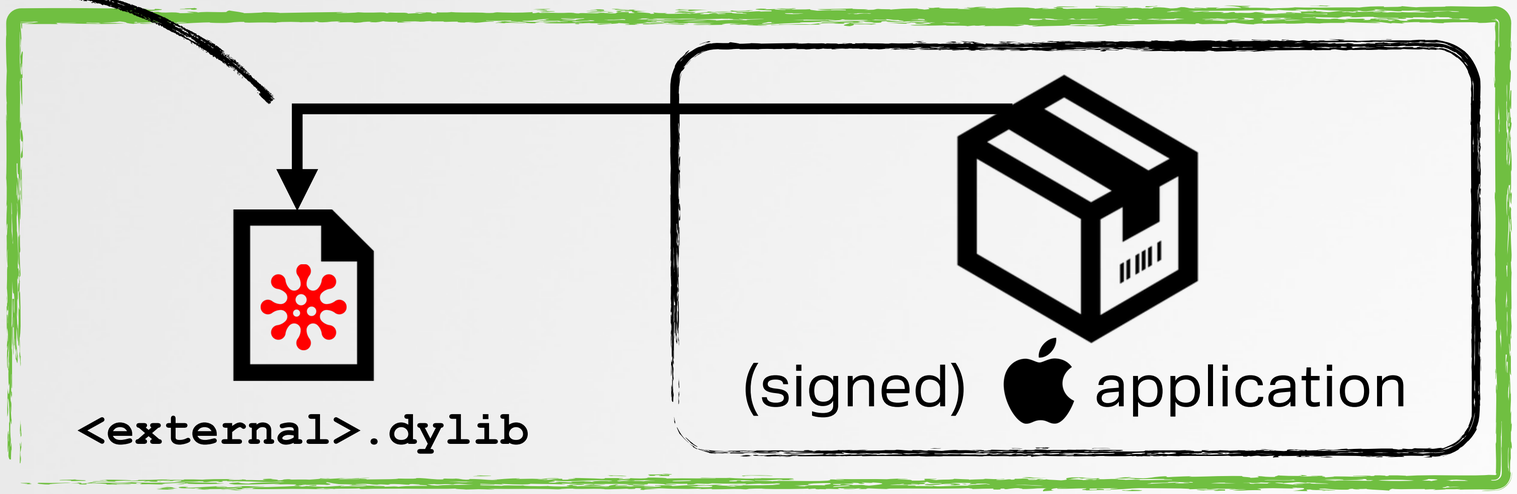
gatekeeper in action



debug messages in syslog

external dylibs,
now verified





malicious .dmg/.zip layout

Synack

PATCH FOR **CVE 2015-3715**

what is this 'dylib check'?

```
mov     rsi, cs:selRef_performDylibBundleCheck_
mov     rbx, [rbp+WorkerThreadClass]
mov     rdi, rbx
mov     rdx, r14 ;path to app
call    cs:_objc_msgSend_ptr
test    al, al
jz      checkFailed

checkFailed:
lea     rdi, "Fails dylib check"
xor     eax, eax
call    _NSLog
```

error msg in
XProtectFramework

```
if(![WorkerThreadClass performDylibBundleCheck:app])
{
    NSLog(@"Fails dylib check");
}
```

translated to C

1. boot into recovery mode via cmd+r
2. csrutil disable (from Terminal.app)
3. reboot

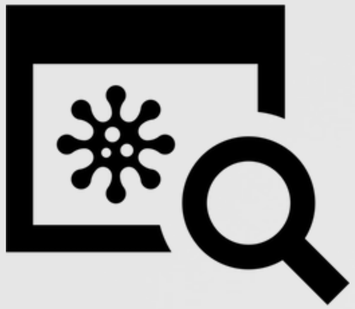
'enable' debugging OS X 10.11

```
(lldb) br s -a 0x00007FFF9A12AA22
Breakpoint 1: where = XprotectFramework`+[WorkerThreadClass
threadEntry:] + 4845, address = 0x00007fff9a12aa22

Process 381 stopped
XprotectFramework`+[WorkerThreadClass threadEntry:] + 4845:
-> 0x7fff9a12aa22:  callq  *%r13

(lldb) po $rdi
WorkerThreadClass

(lldb) x/s $rsi
0x7fff9a12cb84: "performDylibBundleCheck:"

(lldb) po $rdx
file:///Volumes/unsafe/Applications/Instruments.app/
```

debugging with LLDB

PATCH FOR CVE 2015-3715

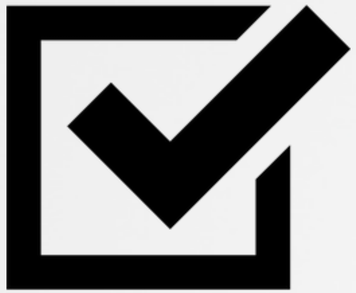
overview of `performDylibBundleCheck:`



`+[WorkerThreadClass performDylibBundleCheck:]`

`+[XProtectDylibCheck alloc]`

`-[XProtectDylibCheck parseMacho]`



`-[XProtectDylibCheck checkCommandsWithBundleURL:]`

```
$ classdump XprotectFramework

@interface XProtectDylibCheck : NSObject
{
    NSString *_absolutePath;
    NSMutableArray *_rPaths;
    NSMutableArray *_loadCommands;
    unsigned long long _numCommands;
    NSURL *_executablePath;
    NSURL *_loaderPath;
    BOOL _isExecutable;
    NSMutableDictionary *_scannedLibraries;
    ....
}

+ (BOOL)path:(id)arg1 isInsideBundle:(id)arg2;
+ (BOOL)path:(id)arg1 isSafeWithBundle:(id)arg2;
+ (id)allowedLibraryPaths;
- (BOOL)parseMacho;
- (id)parseExececutableAndLoaderPaths:(id)arg1;
- (BOOL)parseLoadCommands;
- (id)substituteRpath:(id)arg1;
- (BOOL)checkCommandsWithBundleURL:(id)arg1;
....
```

XProtectDylibCheck class

PATCH FOR **CVE 2015-3715**

dylib location verification(s)

scans all statically linked dylibs

```
-[XProtectDylibCheck checkCommandsWithBundleURL:]

mov     rdi, r12
mov     rsi, cs:selRef_path_isSafeWithBundle_
mov     rdx, r15
mov     rcx, rax
call    rbx
test    al, al
jz      unsafeDylib
```

invoking **`path:isSafeWithBundle:`**

1. allows if dylib falls in an '**`allowLibraryPath`**'

```
(lldb) po $rax
<__NSArrayI 0x7f89ca4ed960>(
 /usr/,
 /opt,
 /System/,
 /Library/,
 /Network/,
 /AppleInternal/,
 /Developer,
 /build
)
```

allowed library paths

2. allows if dylib falls within the (verified) application bundle

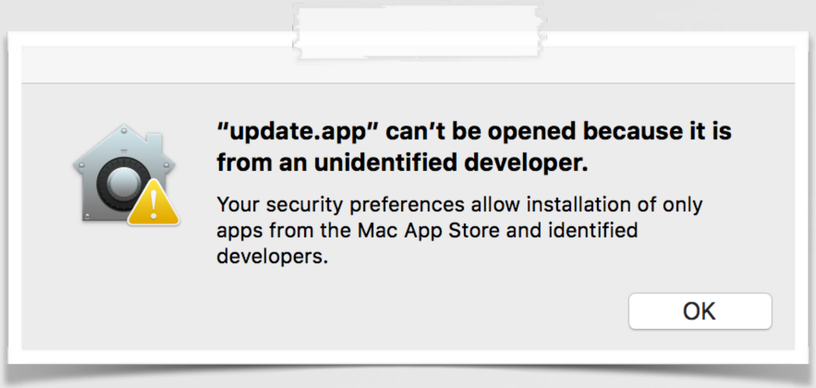
```
if(YES != [dylib hasPrefix:appBundle])
{
    //NOT SAFE!
}
```

dylib inside app bundle?

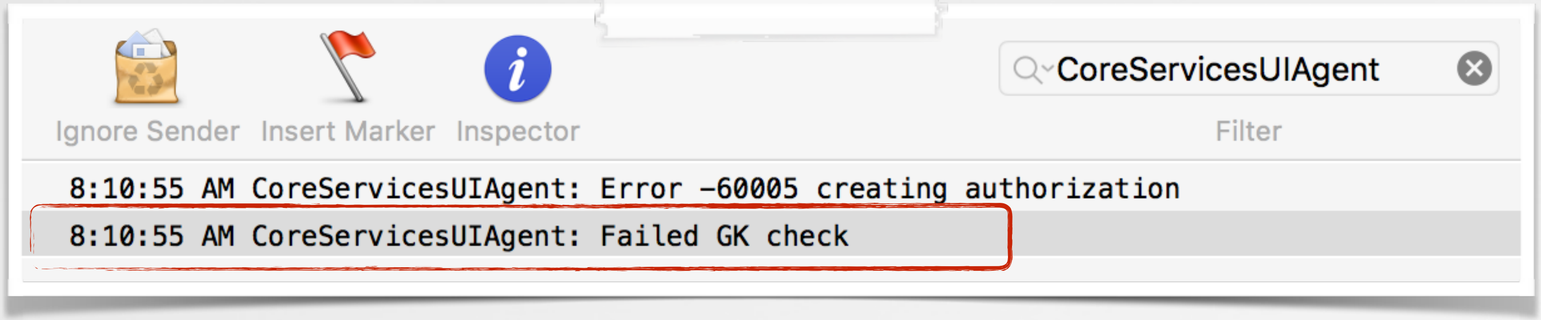
Synack

PATCHING CVE 2015-7024

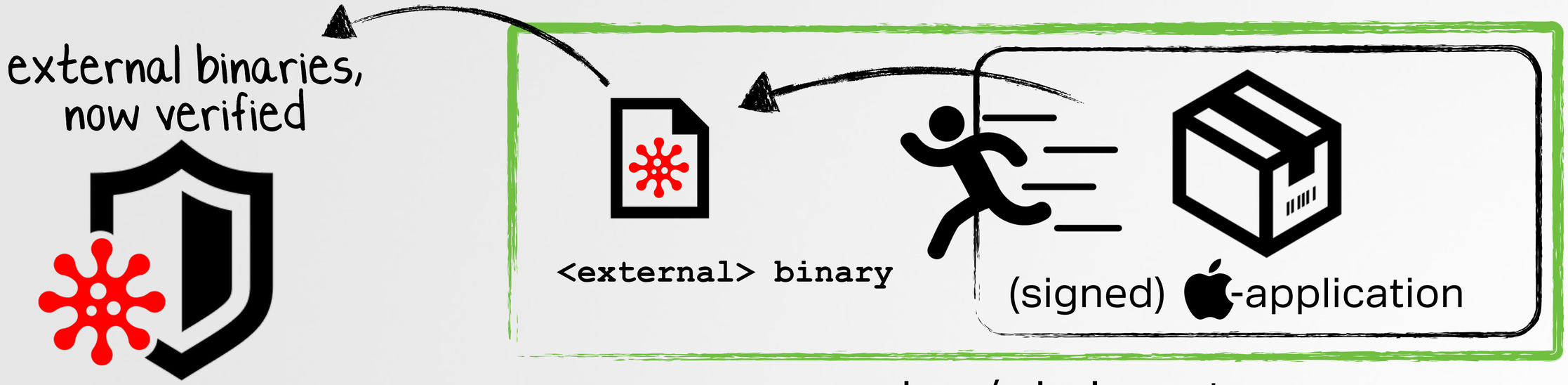
external binaries; verified?



gatekeeper in action



debug messages in syslog



.dmg/.zip layout

Synack

PATCH FOR CVE 2015-7024

what is this 'Failed GK check'?

```
mov     rsi, cs:selRef_performBlockListCheck_blockDict_
mov     rbx, [rbp+WorkerThreadClass]
mov     rdi, rbx
mov     rdx, r14
mov     rcx, rax
call    cs:_objc_msgSend_ptr
test    al, al
jz      short checkFailed

checkFailed:
lea     rdi, "Failed GK check"
xor     eax, eax
call    _NSLog
```

error msg in
XProtectFramework

```
if(![WorkerThreadClass
     performBlockListCheck:app blockDict:blockedSigs])
{
    NSLog(@"Failed GK check");
}
```

translated to C

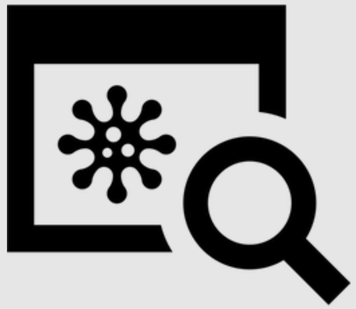
```
(lldb) br s -a 00007FFF9A12A9FE
Breakpoint 1: where = XprotectFramework`+[WorkerThreadClass
threadEntry:] + 4809, address = 0x00007fff9a12a9fe

Process 381 stopped
XprotectFramework`+[WorkerThreadClass threadEntry:] + 4809:
-> 0x00007fff9a12a9fe:  callq  *%r13

(lldb) po $rdi
WorkerThreadClass

(lldb) x/s $rsi
0x7fff9a12cb63: "performBlockListCheck:blockDict:"

(lldb) po $rdx
file:///Volumes/unsafe/update.app

(lldb) po $rcx
{
    CodeSignatureIDs =     (
        "com.apple.a2p",
        "com.apple.ibtool",
        "com.apple.pythonw",
        "com.apple.python",
        "com.apple.ictool"
    );
}
```

debugging with LLDB

Synack

Patch for CVE 2015-7024

performBlockListCheck: blockDict:



takes app & black-listed IDs

`+[WorkerThreadClass performBlockListCheck: blockDict:]`

`-[blackListedID isEqualToString:appID]`

`SecStaticCodeCreateWithPath() &`
`SecCodeCopySigningInformation()`

```
(lldb) po <app's code signing info>
{
   "digest-algorithm" = 1;
    identifier = "com.apple.ictool";
    "main-executable" =
                "file:///Volumes/unsafe/update.app";
     ...
}
(lldb) po <block dictionary>
CodeSignatureIDs = (
            "com.apple.a2p",
            "com.apple.ibtool",
            "com.apple.pythonw",
            "com.apple.python",
            "com.apple.ictool"
);
```

debugging

```
appID = //get app's id from code signing blob

//check if app's ID matches any black listed ones
for(blackListedID in blockDict)
{
    if(YES == [blackListedID isEqualToString:appID])
        //black-listed! GTFO
}
```

in pseudo code

Synack

Patch(s) Summary

are OS X users' now protected? hint: **NO**

		
2015-3715 (external dylib hijack)	scan for external dylibs	☑ effective patch ☒ only blocks specific vector
2015-7024 (run-time exec's)	blacklist binaries	☒ ineffective patch

neither generically blocks the execution of unsigned internet code

Synack

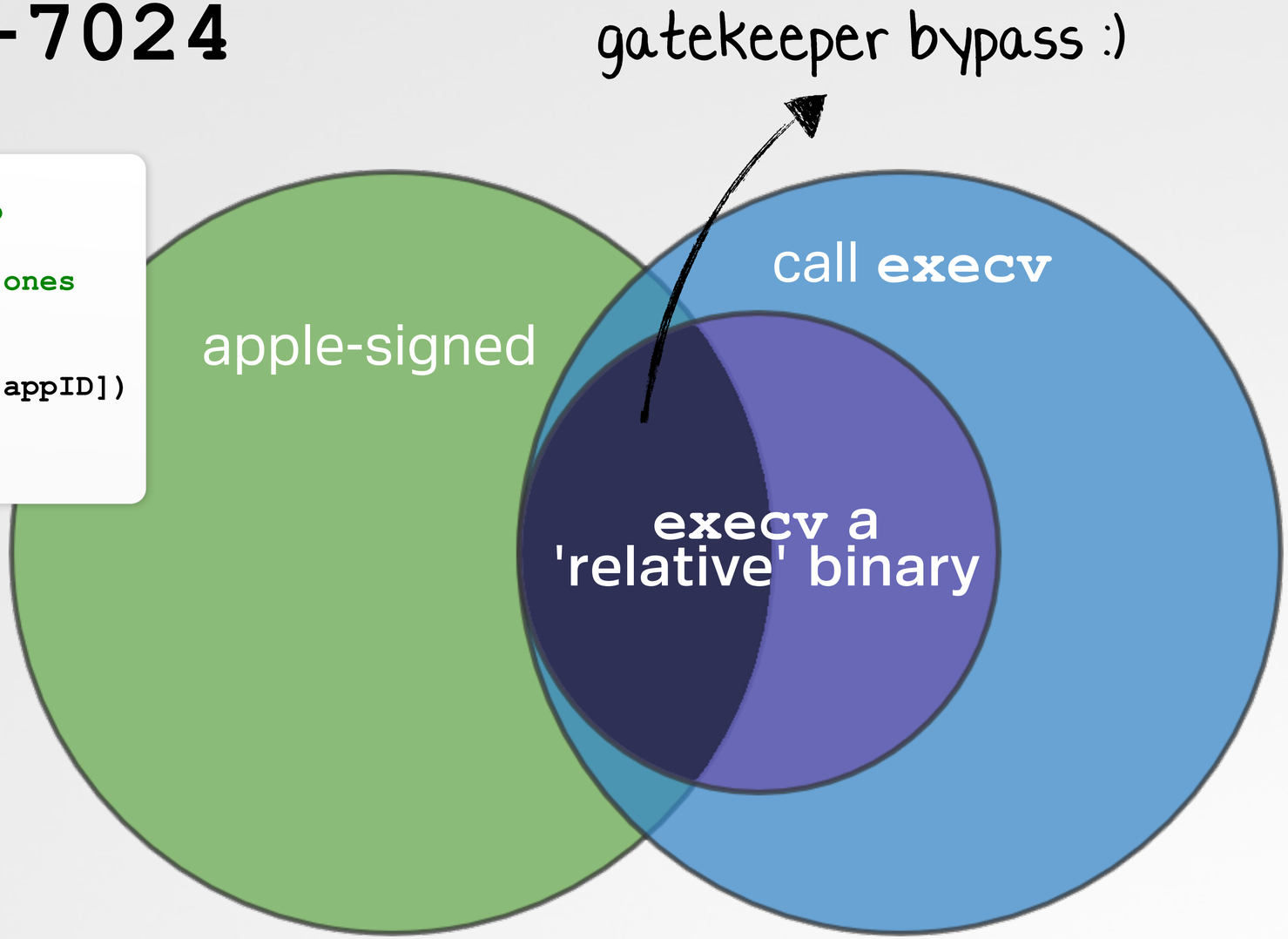
BYPASSING **CVE 2015-7024**

...with ease

```
appID = //get app's id from code signing blob

//check if app's ID matches any black listed ones
for(blackListedID in blockDict)
{
    if(YES == [blackListedID isEqualToString:appID])
        //black-listed! GTFO
}
```

"patch"

1. apple-signed binaries
2. + that calls **execv**
3. + on a 'relative' binary



"Wardle said he suspects there are other Apple-trusted binaries ...that will also allow attackers to bypass Gatekeeper." (summer 2015)

BYPASSING CVE 2015-7024

finding moar binaries to abuse

```
def scan(rootDir):

    #dbg msg
    print 'scanning %s' % rootDir

    #enum bins
    #->signed by apple proper
    appleBins = enumBinaries(rootDir)

    #check imports
    #->looking for execv/etc
    candidateBins = checkImports(appleBins)

    #dbg
    print candidateBins
```

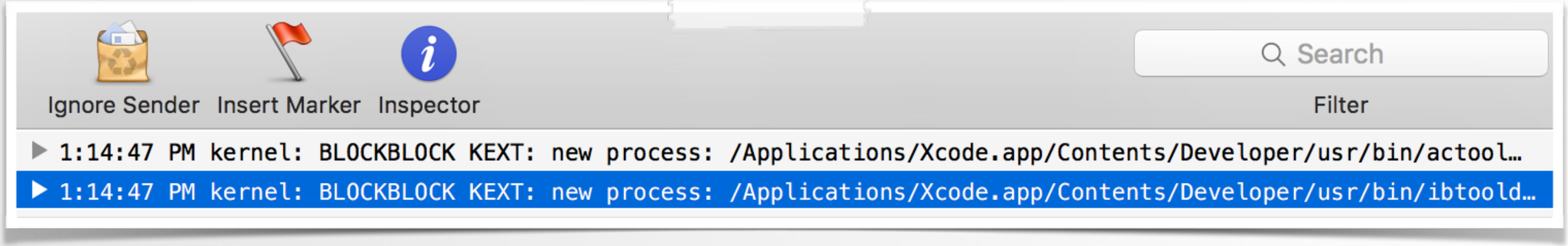
scan.py

```
$ python scan.py /Applications/Xcode.app/Contents/Developer/usr/bin/

scanning /Applications/Xcode.app/Contents/Developer/usr/bin/

['/Applications/Xcode.app/Contents/Developer/usr/bin/actool',
 '/Applications/Xcode.app/Contents/Developer/usr/bin/atos', ... ]
```

scanner output

```
$ sudo fs_usage -w -f filesystem | grep -i actool

getattrlist   /Applications/Xcode.app/Contents/Developer/usr/bin/actool
stat64        /Applications/Xcode.app/Contents/Developer/usr/bin/ibtoold
```

file i/o for **actool**

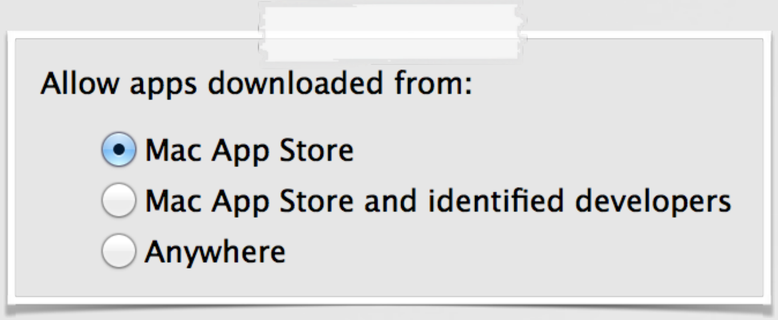


'process monitoring' **actool**

Synack

CVE 2015-7024 BYPASS

replace `ictool` with `actool`



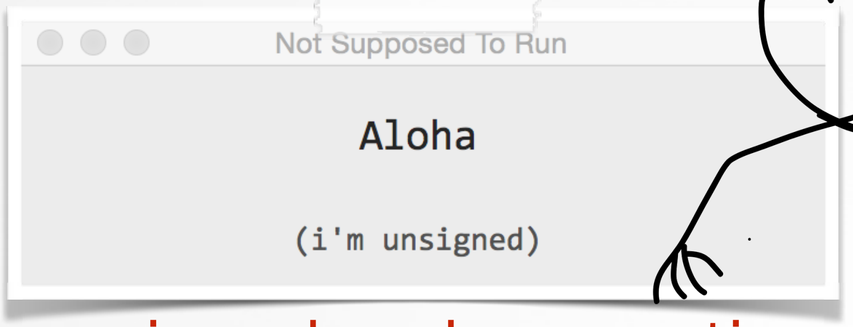
gatekeeper setting's (max.)

.dmg setup

1. alias to **`'update.app'`** (~~**`ictool`**~~) — **`actool`**
   ...name & icon attacker controlled
2. apple-signed **`'update.app'`** (~~**`ictool`**~~) — **`actool`**
   .app extension prevents **`Terminal.app`** popup

hide:

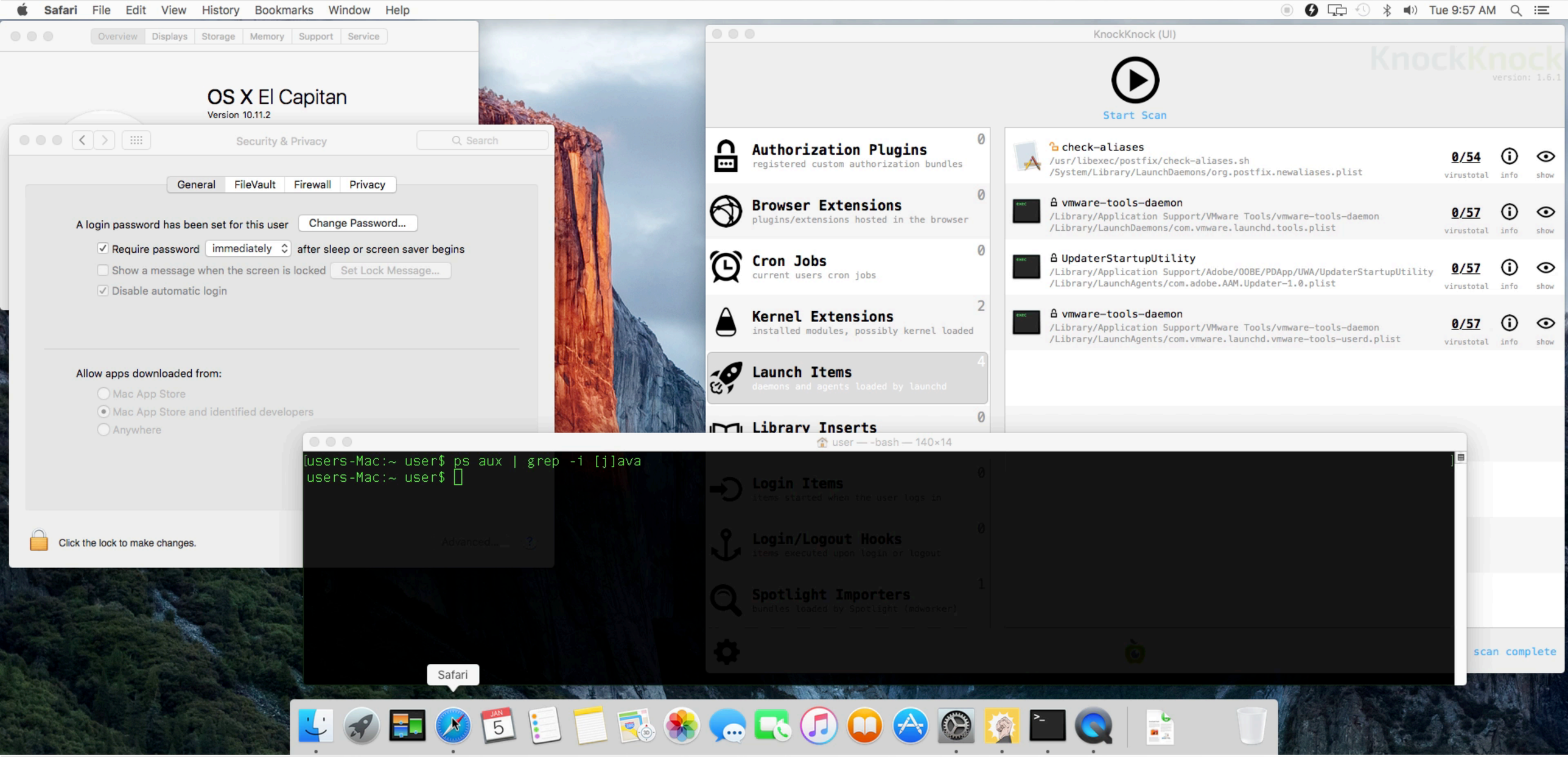
3. unsigned **`ibtoold`**
   command-line executable
4. unsigned application



unsigned code execution

Synack

CVE 2015-7024 Bypass



Synack

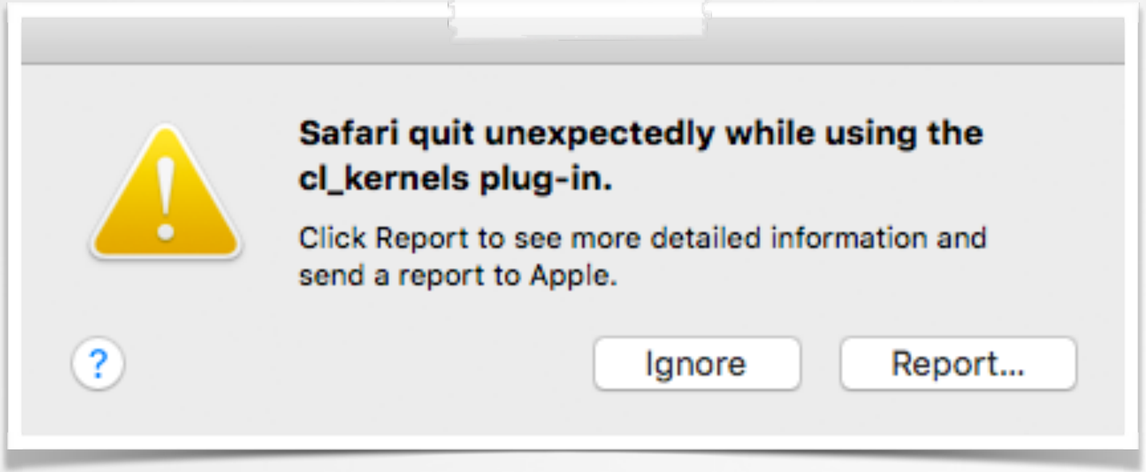
Leveraging OS-Level Mitigations?

"A kernel extension to mitigate Gatekeeper bypasses"

> *"[**sysctl**] allows processes with appropriate privilege to set kernel state" -apple.com*

```
$ sysctl vm | grep cs_

vm.cs_force_kill:  0
vm.cs_force_hard:  0
vm.cs_all_vnodes:  0
vm.cs_enforcement: 0
....
```

code-signing **sysctl** variables



lots of OS issues

"require enforcement"

```
$ sudo sysctl -w vm.cs_enforcement=1
vm.cs_enforcement:0 -> 1
```

enabling code-signing enforcement

"Code Signing–Hashed Out"
J. Levin

Synack

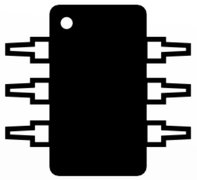
GATEKEERPER

"A kernel extension to mitigate Gatekeeper bypasses"

Pedro Vilaça

@osxreverser

MAC hook (on **mmap**)



```
int mpo_file_check_mmap_t(...)
{
   ...

   //determine if main binary is signed
   is_main_signed = ((csproc_get_teamid_t*)(void*)(cloned_csproc_get_teamid))(p);

   //determine if mapped section is signed
   is_mapped_signed = ((csfg_get_platform_binary_t*)cloned_csfg_get_platform_binary)(fg);

   //block unsigned dylibs in signed app/binary
   if(is_mapped_signed == 0 && is_main_signed == 1)
   {
      //GTFO!
   }
```

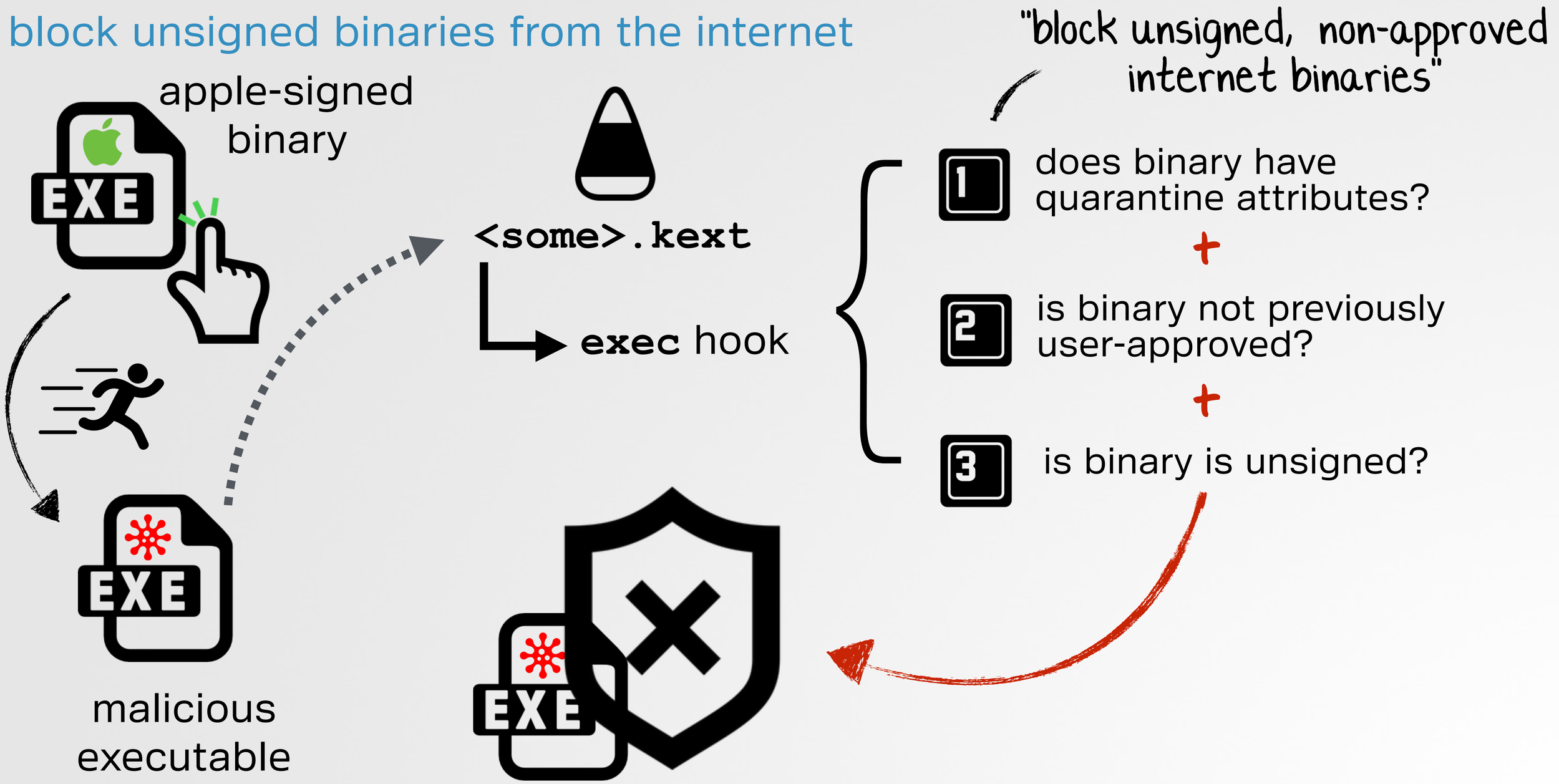
gatekeerper kext
(**github.com/gdbinit/Gatekeerper**)

blocks unsigned dylibs, but not unsigned (stand-alone) binaries from the internet

Synack

Validate All Binaries At Runtime

block unsigned binaries from the internet

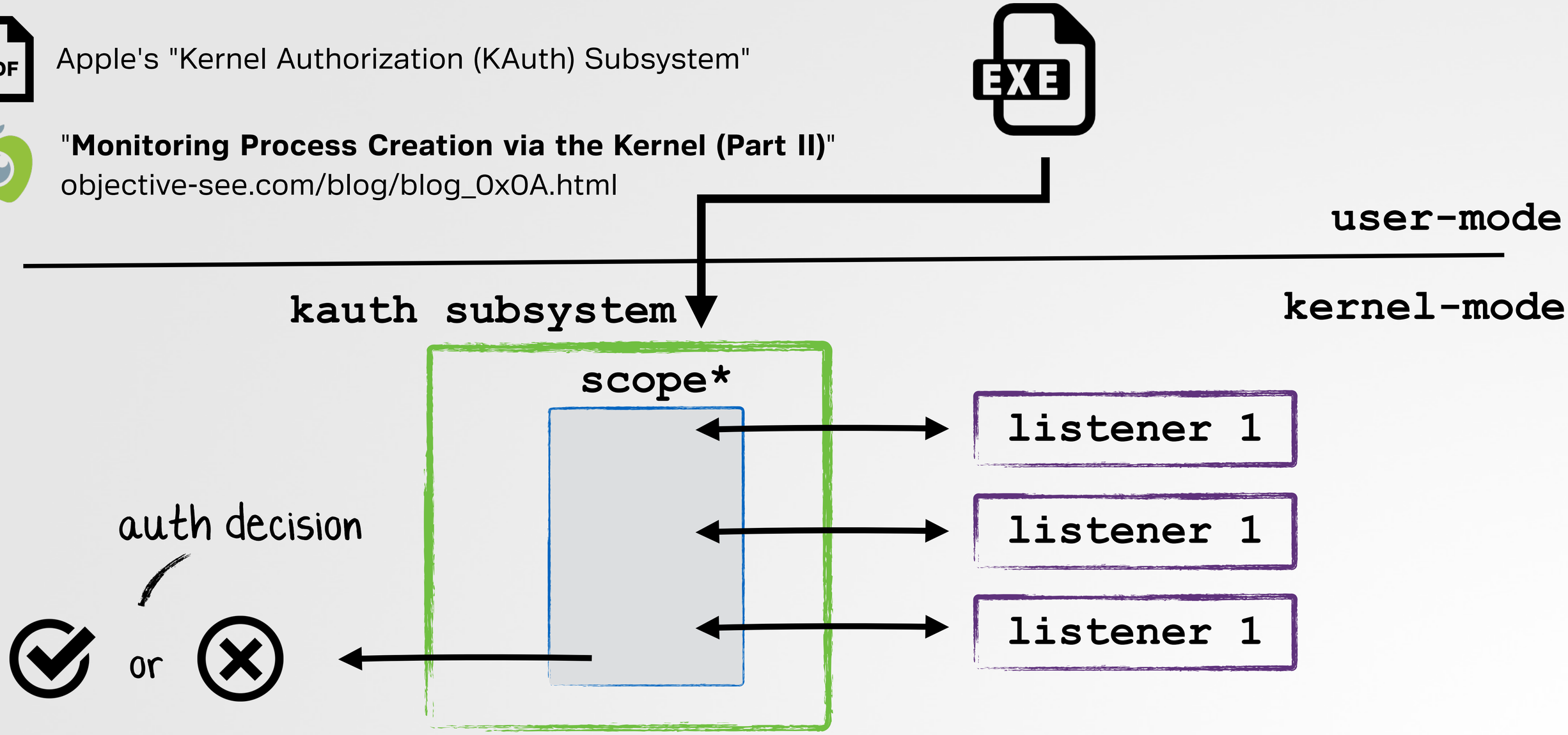


VALIDATE ALL BINARIES AT RUNTIME

step 0: register exec hook

Apple's "Kernel Authorization (KAuth) Subsystem"

"**Monitoring Process Creation via the Kernel (Part II)**"
objective-see.com/blog/blog_0x0A.html



Validate All Binaries At Runtime

step 0: register exec hook

```
//kauth listener
kauth_listener_t kauthListener = NULL;

//register listener ('KAUTH_SCOPE_FILEOP')
kauthListener = kauth_listen_scope(KAUTH_SCOPE_FILEOP, &processExec, NULL);
```

register **KAUTH_SCOPE_FILEOP** listener

```
//kauth callback
static int processExec(kauth_cred_t credential, void* idata, kauth_action_t action, uintptr_t arg0, uintptr_t arg1,
uintptr_t arg2, uintptr_t arg3)
{
     //return var, default to defer
    int kauthResult = KAUTH_RESULT_DEFER;

     //ignore all non exec events
    if(KAUTH_FILEOP_EXEC != action)
    {
        //bail
        goto bail;
    }

    //get path
    vn_getpath((vnode_t)arg0, path, &pathLength);

    //dbg msg
    DEBUG_PRINT(("OSTIARIUS: new process: %s %d\n", path, proc_selfpid()));
```

kauth listener

Synack

Validate All Binaries At Runtime

step 1: ignore 'non-internet' binaries (NULL quarantine attributes)

```
hook_vnode_check_exec
    call     _quarantine_get_flags
```

```
_quarantine_get_flags
    call   quarantine_getinfo
```

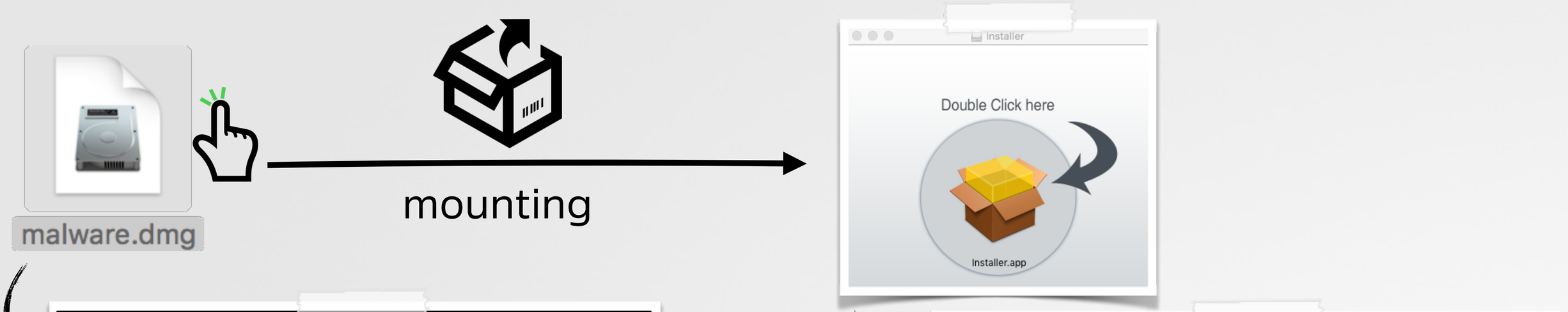
```
quarantine_getinfo

    lea      rsi, "com.apple.quarantine"
    lea      r8, [rbp+qAttrsSize]
    mov      rdi, r14
    mov      rdx, [rbp+qAttrs]
    mov      rcx, r15
    call     _mac_vnop_getxattr
```

Quarantine.kext

```
//get quarantine attributes
// ->if this 'fails', simply means binary doesn't have quarantine attributes (i.e. not from the internet)
if(0 != mac_vnop_getxattr((vnode_t)arg0, QFLAGS_STRING_ID, qAttr, QATTR_SIZE-1, &qAttrLength))
{
    //dbg msg
    DEBUG_PRINT(("binary has NO quarantine attributes (not from the internet), so allowing\n"));

    //bail
    // ->process is allowed
    goto bail;
}
```

Validate All Binaries At Runtime

step 1: ignore 'non-internet' binaries (NULL quarantine attributes)



disk image: quarantine attributes

files: no quarantine attributes

how to tell such files are from the internet?

while disk images have quarantine attributes,
their mounted (executable) files don't...

Validate All Binaries At Runtime

step 1: ignore 'non-internet' binaries (NULL quarantine attributes)

1. is binary path is within **`/Volumes?`**
2. get **`mount`** struct for binary's **`vnode`**
3. get **`vfsstatfs`** struct for **`mount`** and extract **`f_mntfromname`** value (e.g. '**`/dev/disk1s2`**')
4. iterate over the **`IORegistry`** to find a parent ('**`IOHDIXHDDriveOutKernel`**') that has a child ('**`IOMedia`**') with matching mount point (e.g. '**`/dev/disk1s2`**')

   parent will have the original dmg path, in '**`image_path`**' — what we need!
5. with a dmg path, can access quarantine attributes
   if dmg is from the internet & binary is unsigned: **BLOCK**

Synack

Validate All Binaries At Runtime

step 2: ignore 'user-approved' binaries

```
-[GKQuarantineResolver
approveUpdatingQuarantineTarget:recursively:volume:]

call      __qtn_file_get_flags
or        eax, 40h
mov       rdi, [rbp+var_B8]
mov       esi, eax
call      __qtn_file_set_flags
```

updating quarantine attributes

```
call      _quarantine_get_flags
test      eax, eax
jnz       leaveFunction

mov       edx, [rbp+qFlag]
mov       eax, edx
and       eax, 40h
jnz       leaveFunction
```

Quarantine.kext

```
$ xattr -l ~/Downloads/KnockKnock.app
com.apple.quarantine: 0001;55f3313d;...

$ xattr -l ~/Downloads/KnockKnock.app
com.apple.quarantine: 0041;55f3313d;...
```

before & after

```
//CoreServicesUIAgent sets flags to 0x40 when user allows
// ->so just allow such binaries
if(0 != (qFlags & 0x40))
{
    //dbg msg
    DEBUG_PRINT(("previously approved, so allowing\n"));

   //bail
   goto bail;
}
```

allowing previous approved binaries

Synack

Validate All Binaries At Runtime

step 3: ignore signed binaries

@osxreverser; how apple does it

```
int csfg_get_platform_binary(struct fileglob *fg)
{
    int platform_binary = 0;
    struct ubc_info *uip;
    vnode_t vp;

    if (FILEGLOB_DTYPE(fg) != DTYPE_VNODE)
        return 0;

    vp = (struct vnode *)fg->fg_data;
    if (vp == NULL)
        return 0;

    vnode_lock(vp);
    if (!UBCINFOEXISTS(vp))
        goto out;

    uip = vp->v_ubcinfo;
    if (uip == NULL)
        goto out;

    if (uip->cs_blobs == NULL)
        goto out;
```

1. lock **vnode**
2. grab **v_ubcinfo** (**ubc_info** structure)
3. check if **cs_blobs** are NULL (i.e. binary is unsigned)

vnode_lock() is non-exported function

vnode, **ubc_info**, etc all private/ undocumented structures

Validate All Binaries At Runtime

step 3: ignore signed binaries

```
//lock vnode
lck_mtx_lock((lck_mtx_t *)arg0);

//init offset pointer
offsetPointer = (unsigned char*)(vnode_t)arg0;

//get pointer to struct ubc_info in vnode struct
// ->disasm from kernel:  mov rax, [vnode+70h]
offsetPointer += 0x70;

//dref pointer to get addr of struct ubc_info
offsetPointer = (unsigned char*)*(unsigned long*)(offsetPointer);

//get pointer to cs_blob struct from struct ubc_info
// ->disasm from kernel: mov rax, [ubc_info+50h]
offsetPointer += 0x50;

//null csBlogs means process is NOT SIGNED
// ->so block it
if(0 == *(unsigned long*)(offsetPointer))
{

  //kill the process
  proc_signal(pid, SIGKILL);

}

//unlock vnode
lck_mtx_unlock((lck_mtx_t *)arg0);
```

`lck_mtx_lock`, exported :)

```
_vnode_lock proc
  push    rbp
  mov     rbp, rsp
  pop     rbp
  jmp     _lck_mtx_lock
_vnode_lock     endp
```

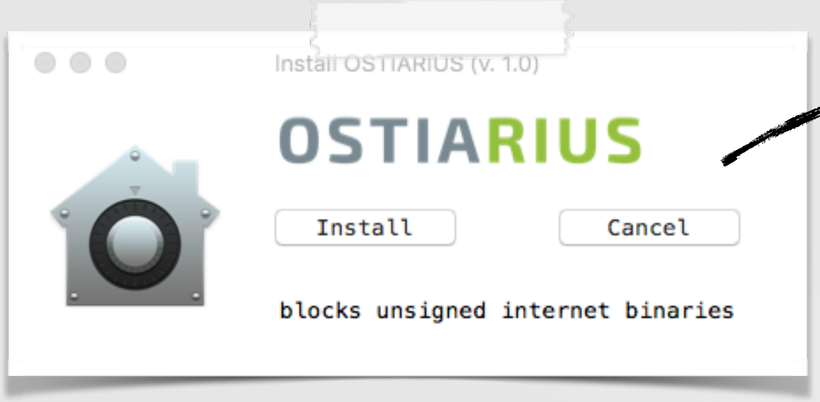
static offsets for OS X 10.11.*

block unsigned internet binary

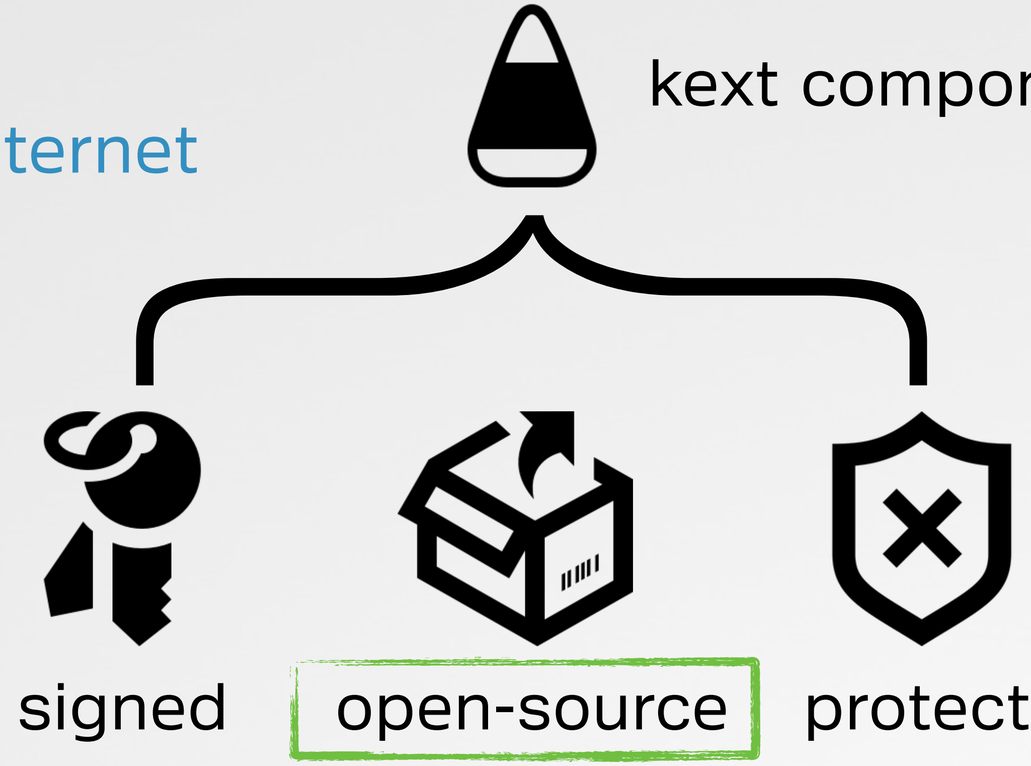
Synack

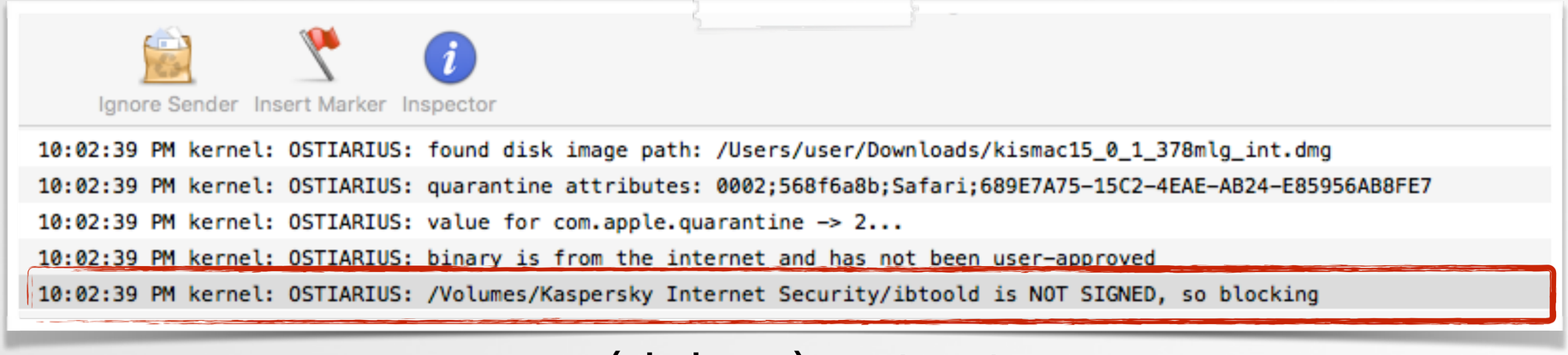
OSTIARIUS

blocking unsigned binaries from the internet



installer



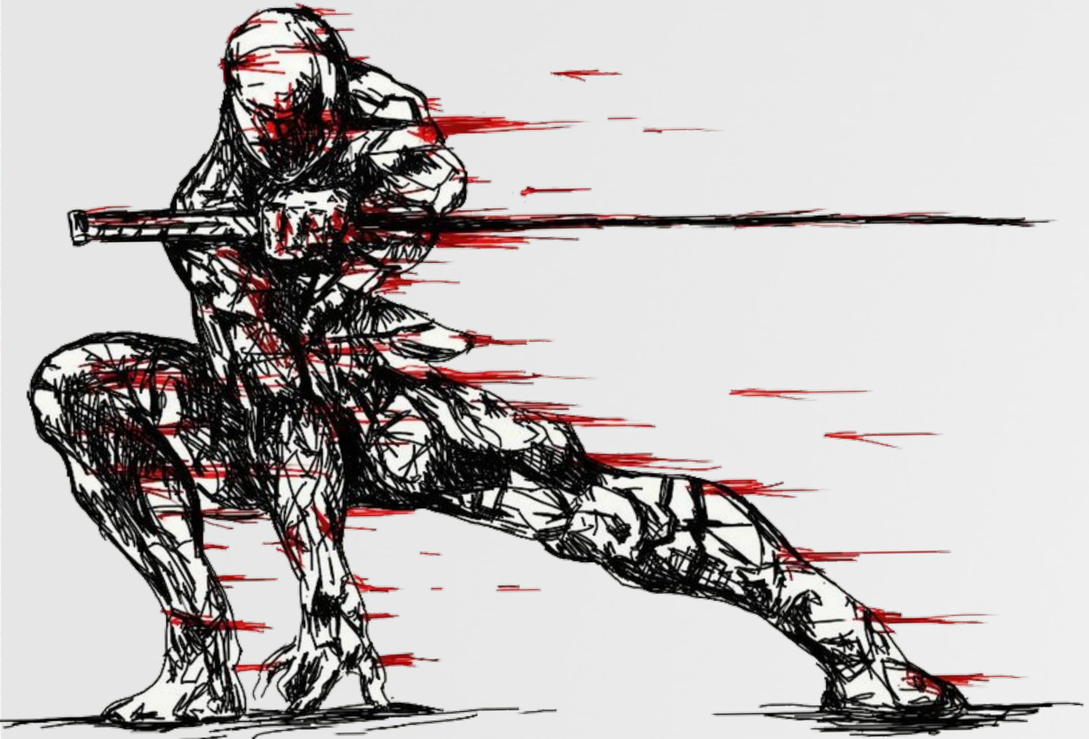


(debug) output

objective-see.com/products/ostiarius.html

CONCLUSIONS

wrapping it up



Synack

GATEKEEPER

theory (or, apple marketing)

protects naive OS X users
from attackers

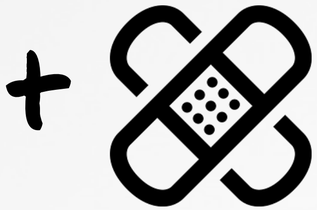
"*omg, my mac is so secure, no need for AV*" -mac users

GATEKEEPER

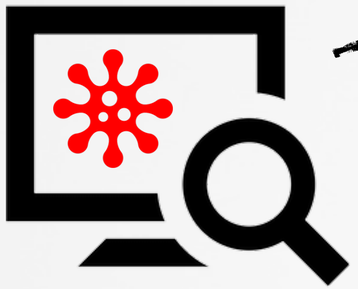
the unfortunate reality

protects naive OS X users
from lame attackers

& false sense
of security?



patch fails

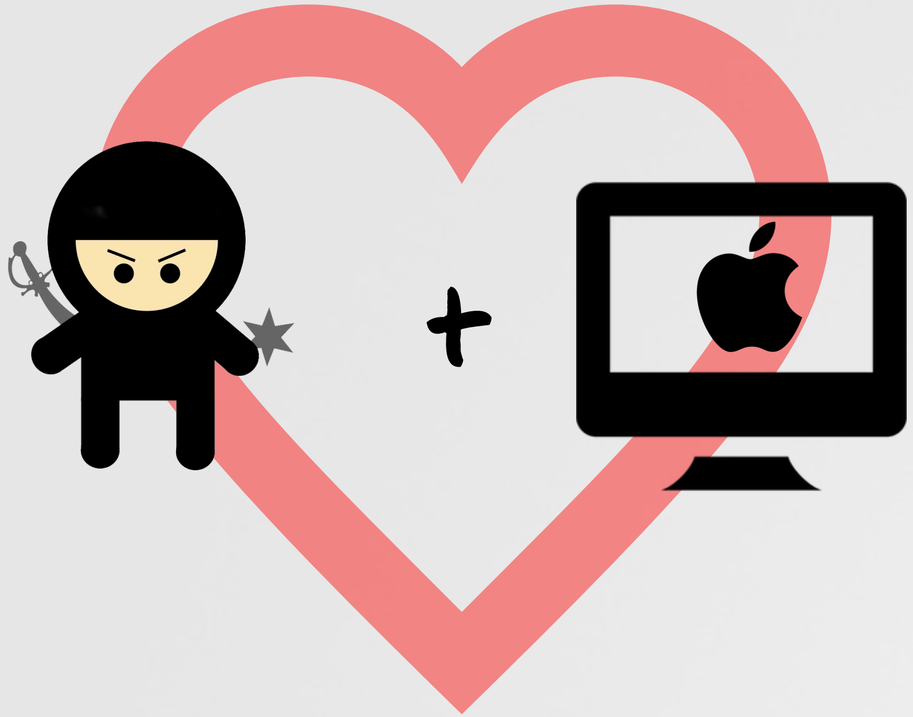


highly recommend;
3rd-party security tools

Synack

My Conundrum

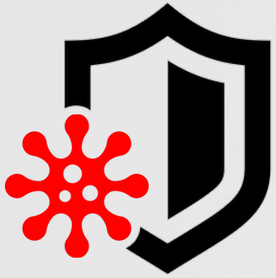
...I love my mac, but it's so easy to hack



I should write some OS X security tools to protect my Mac

....and share 'em freely :)

i humbly disagree

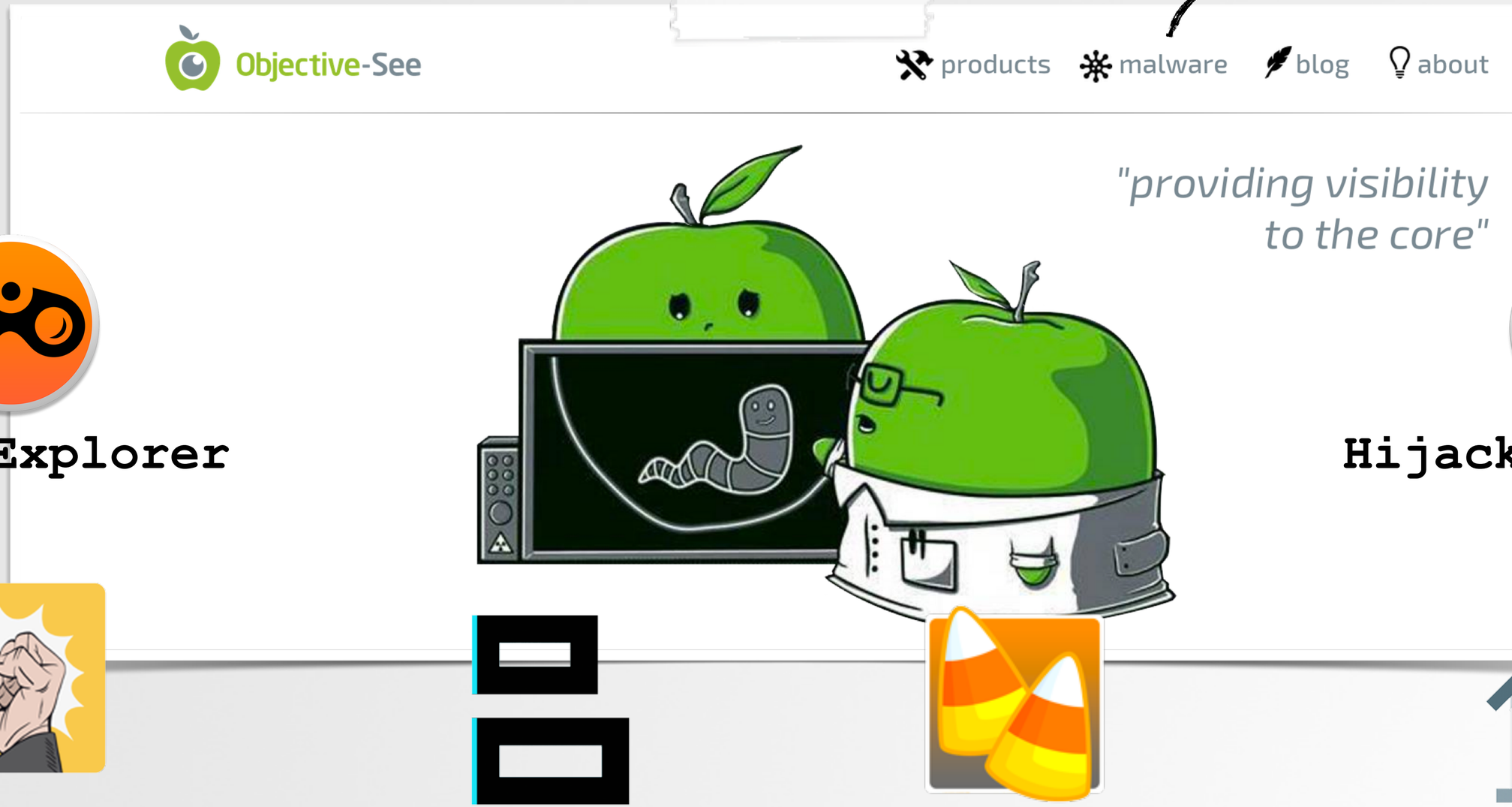


"No one is going to provide you a quality service for nothing. If you're not paying, you're the product." -unnamed AV company

Objective-See(.com)

free security tools & malware samples

os x malware samples



TaskExplorer

Hijack Scanner

KnockKnock

BlockBlock

KextViewr

Ostiarius

Synack

Questions & Answers

feel free to contact me any time!

patrick@synack.com

@patrickwardle

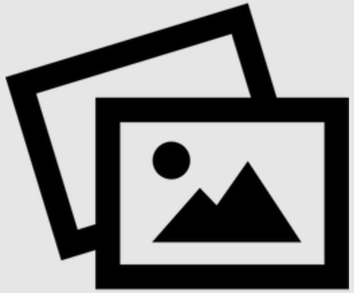
Objective-See

Synack

final thought ;)

"*Is it crazy how saying sentences backwards creates backwards sentences saying how crazy it is?*" -Have_One, reddit.com

credits



images

- flaticon.com
- thezooom.com
- iconmonstr.com
- http://wirdou.com/2012/02/04/is-that-bad-doctor/
- http://th07.deviantart.net/fs70/PRE/f/2010/206/4/4/441488bcc359b59be409ca02f863e843.jpg

resources

- thesafemac.com
- "Mac OS X & iOS Internals", Jonathan Levin
- http://newosxbook.com/articles/CodeSigning.pdf
- https://securosis.com/blog/os-x-10.8-gatekeeper-in-depth
- https://reverse.put.as/2015/11/09/gatekeerper-a-kernel-extension-to-mitigate-gatekeeper-bypasses/