Crowd Security Intelligence

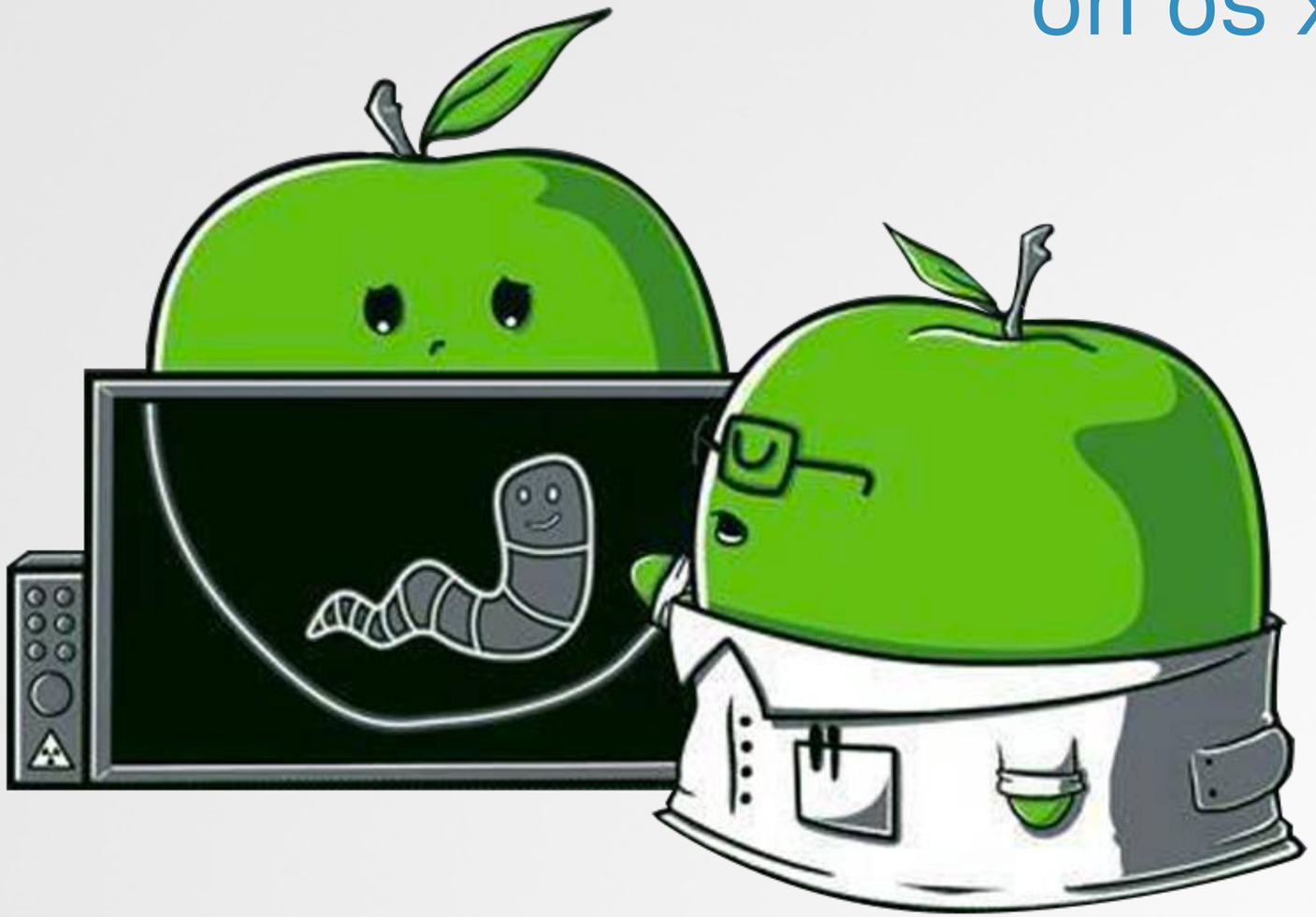
(download slides)

syn.ac/virusb2014

@patrickwardle

METHODS of MALWARE PERSISTENCE

on os x mavericks



About

"[synack] *sources a global contingent of vetted security experts worldwide and pays them on an incentivized basis to discover security vulnerabilities in our customers' web apps, mobile apps, and infrastructure endpoints.*"

`patrick wardle`

`/NASA /NSA /VRL /SYNACK`

AN OUTLINE

background, persistence, malware, & detection

background

methods of persistence

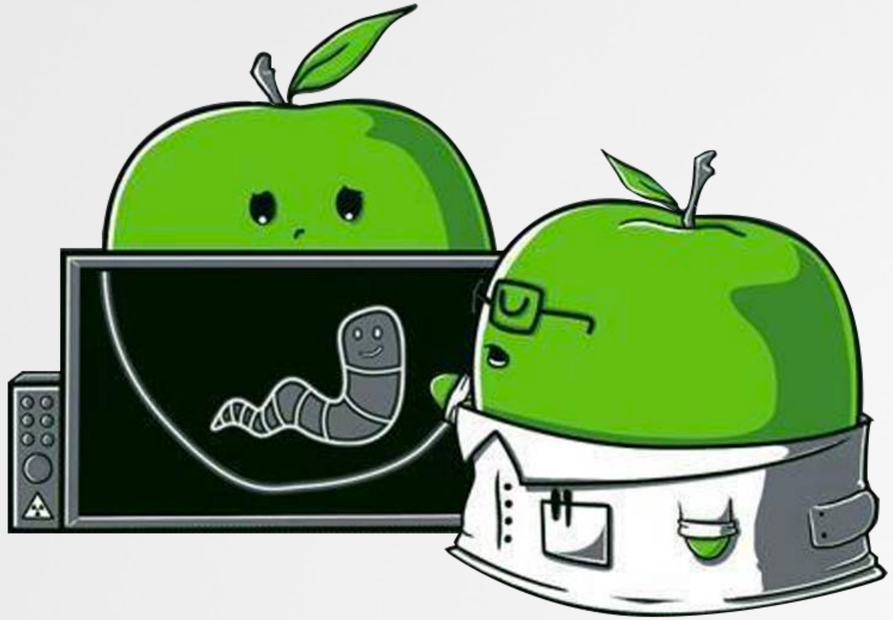
os x malware

‘autoruns’ for os x

Synack

BACKGROUND

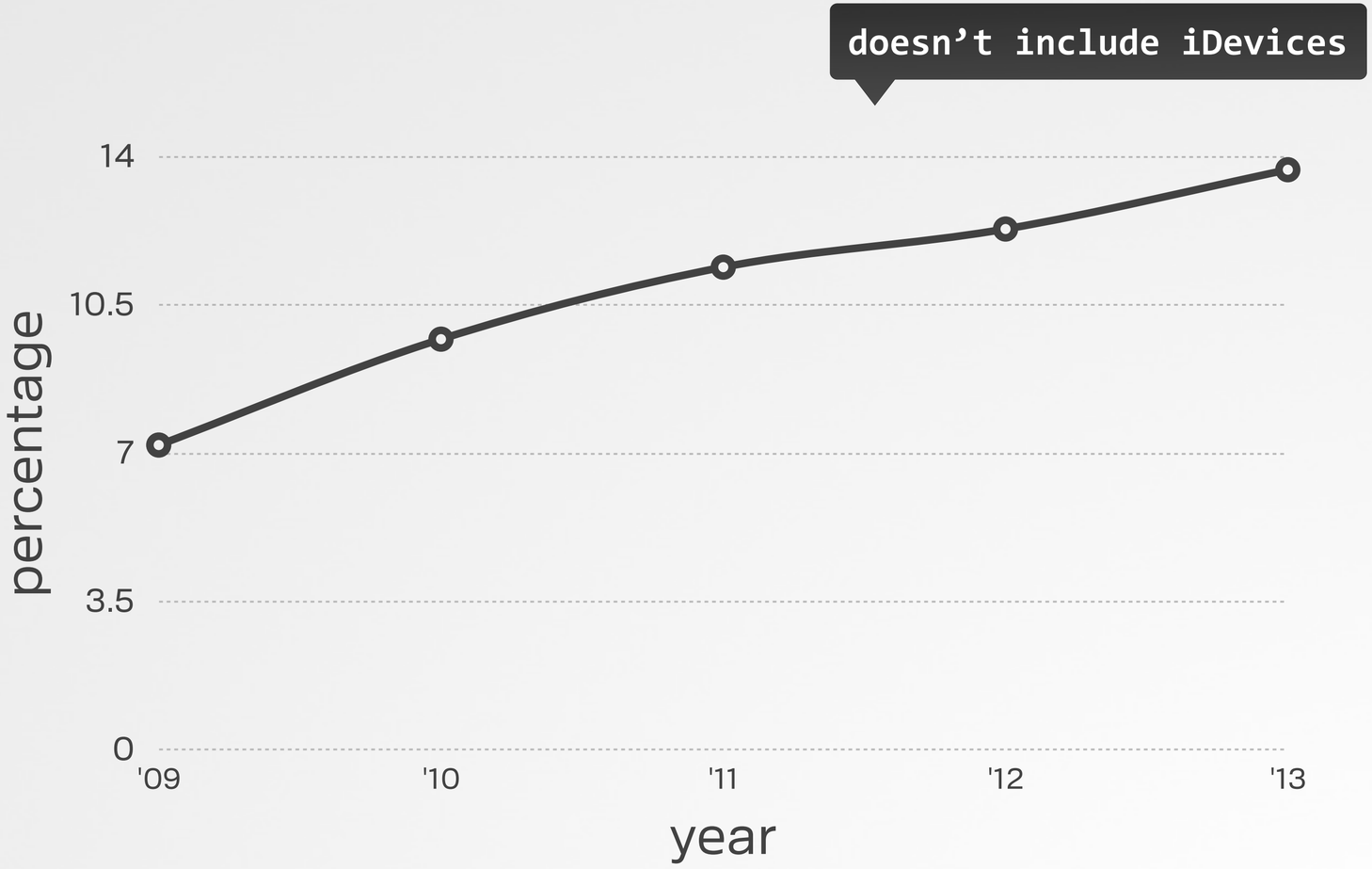
why you should care



The Rise of Macs

macs are everywhere (home & enterprise)

apple is now the #3 vendor
in usa pc shipments

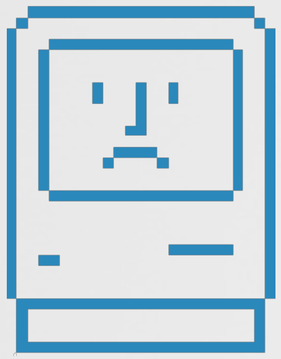


macs as % of total usa pc sales

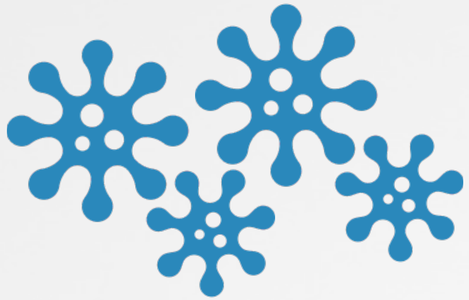
Malware on OS X?

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

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



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



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?!?

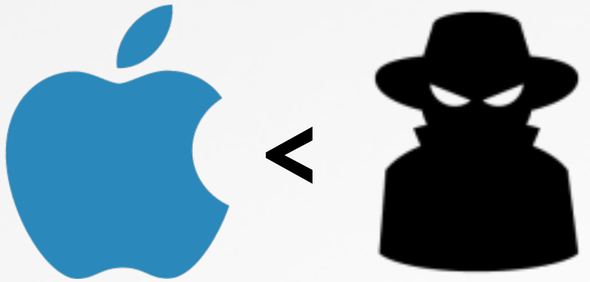
xprotect

gatekeeper

os x sandbox

code-signing

nope!



'wins'

The Current Situation

the apple juice is sour...

+

+

+

lots of macs

feeble anti-malware protections

os x malware

limited os x malware analysis tools

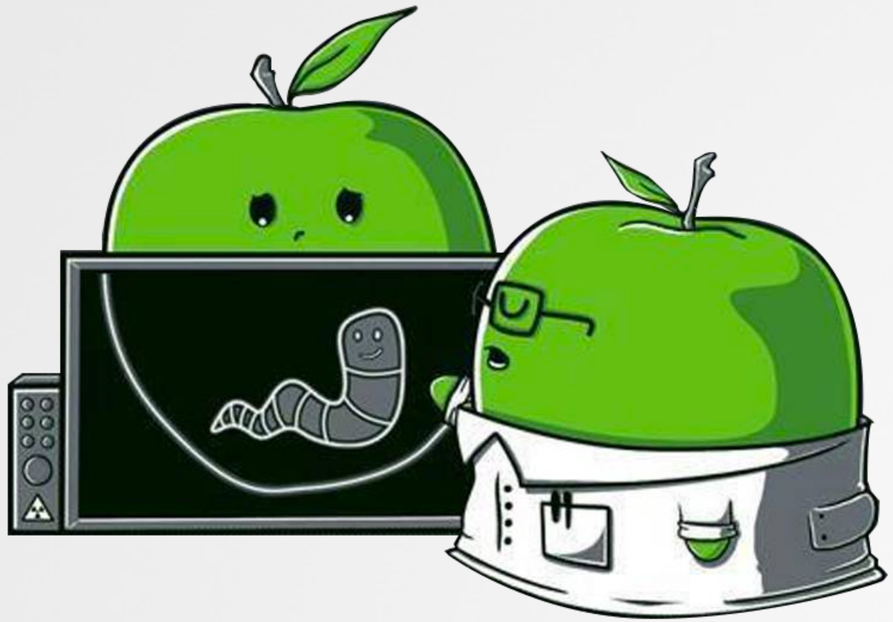
“by identifying persistence mechanisms in os x and studying malware that abuses these, we can (better) protect ourselves”

....and a new analysis tool can help as well

Synack

METHODS OF PERSISTENCE

where malware may live



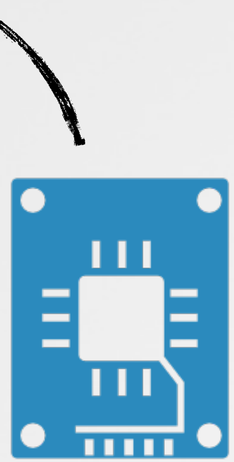
Synack

Low Level

the boot process affords several opportunities for persistence

often highly complex, though very insidious and difficult to detect

install malicious EFI components?



replace/patch the **`boot.efi`**?

re-flash the bootROM?

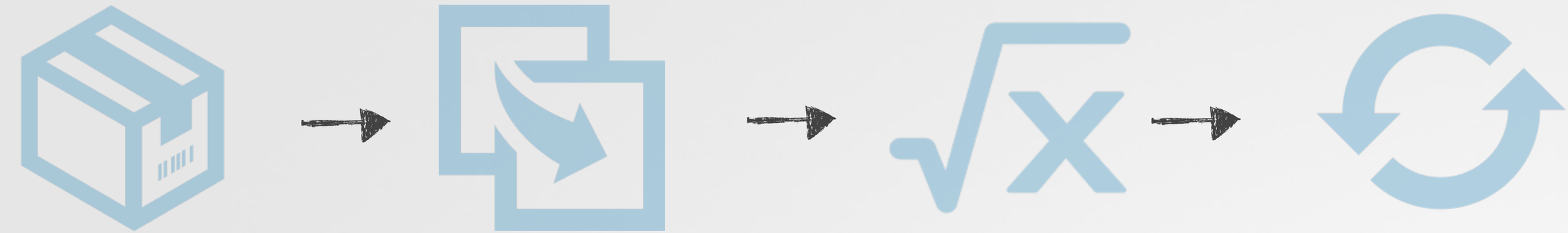
infecting the boot process

‘mac efi rootkits’
by loukas k (snare)

Synack

KERNEL EXTENSIONS

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
```

installing a kext

Launch Daemons & Agents

similar to windows services

daemons and agents are started by **launchD**

daemons

non interactive, launched pre-login

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

agents

interactive, launched post-login

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

Synack

Launch Daemons & Agents

registered ('installed') via a property list

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

label/identifier

binary image

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>
```

daemon/agent plist

Cron Jobs

used to automatically run scrips/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

Synack

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
```

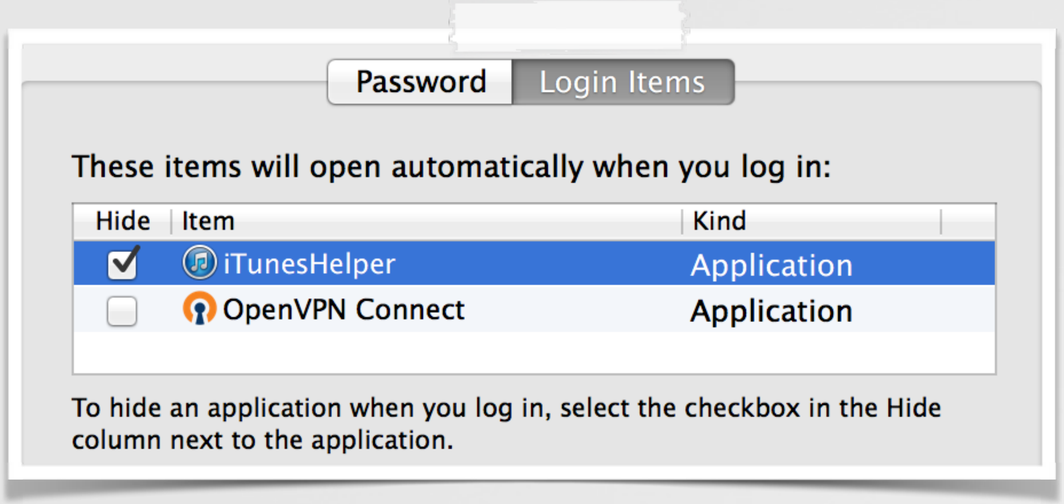
```
# ~/Library/Preferences/com.apple.loginwindow.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist ...>
<plist version="1.0">
<dict>
    <key>LoginHook</key>
    <string>/usr/bin/hook.sh</string>
</dict>
</plist>
```

script

login hook

Login Items

‘legitimate’ method to ensure apps are executed at login



System Preferences -> Users & Groups -> Login Items

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

```
<dict>
 <key>com.apple.LSSharedFileList.Binding</key>
 <data>
 ZG5pYgAAAACAAAAAAAAAAAAAAAAAAAAA...
 </data>
 <key>com.apple.LSSharedFileList.ItemIsHidden</key>
 <true/>
 <key>com.apple.loginitem.HideOnLaunch</key>
 <true/>
 <key>Name</key>
 <string>iTunesHelper</string>
</dict>
```

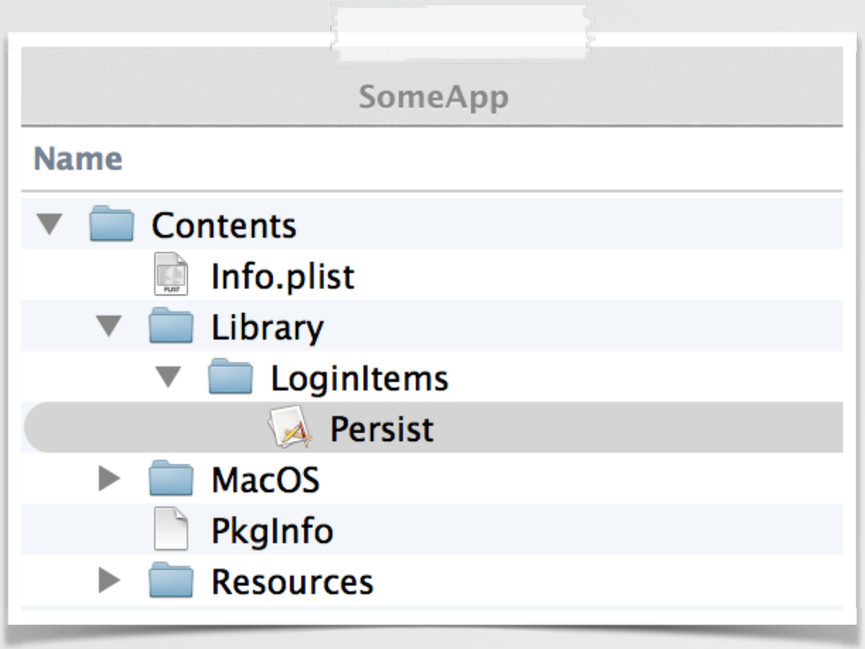
base64 data (path, etc.)

login item

Synack

Login Items (sandboxed)

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



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

does not show up in (any) GUI

invoke **`SMLoginItemSetEnableD()`** 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
SMLoginItemSetEnabled((__bridge CFStringRef) @"com.company.persistMe", YES);
```

sandboxed login item

Synack

Startup Items

allow a script to be automatically executed at each reboot

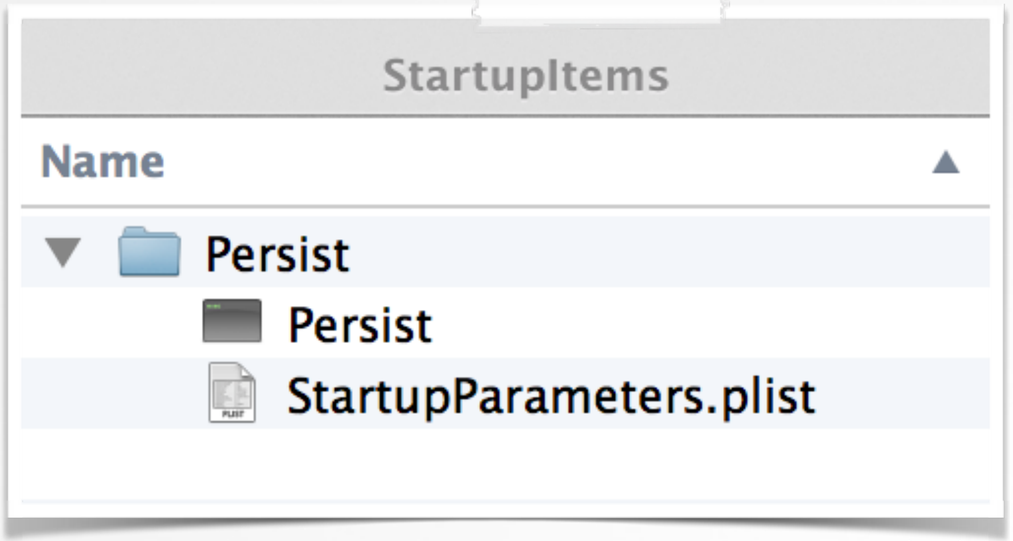
match script's name

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

StartService()
{
  #anything here
}

RunService "$1"
```

persistent script

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

StartupParameters.plist



/System/Library/StartupItems

/Library/StartupItems

RC.COMMON

allows scripts or commands to automatically execute

another linux'y-based technique

how easy is this?

```
# vim /etc/rc.common

...

add any commands (at end)
```

modifying rc.common

LAUNCHD.CONF

allows **`launchCtl`** commands to be automatically executed

file does not exist by default

```
# 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

application

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

unsign the target binary

launch item

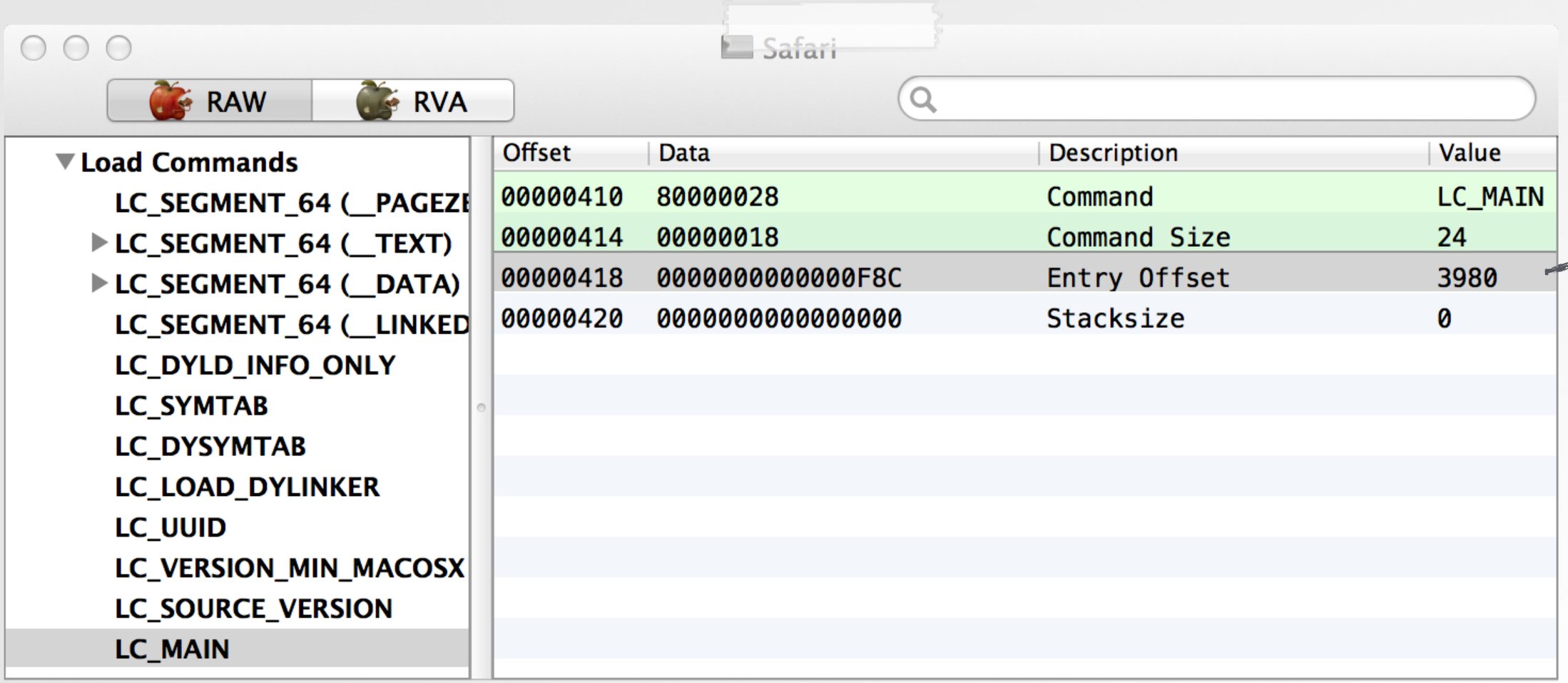
```
$ less /System/Library/LaunchDaemons/com.apple.mDNSResponder.plist
...
<key>EnvironmentVariables</key>
<dict>
  <key>DYLD_INSERT_LIBRARIES</key>
  <string>/usr/bin/evil.dylib</string>
</dict>
```

dyld_insert_libraries (app & launch item)

Mach-O Infection

ensures (injected) code is executed when host is run

read, “infecting mach-o files”
(roy g biv)



entry point

mach-o structure

Synack

Application Specific

plugins or extensions can provide automatic code execution

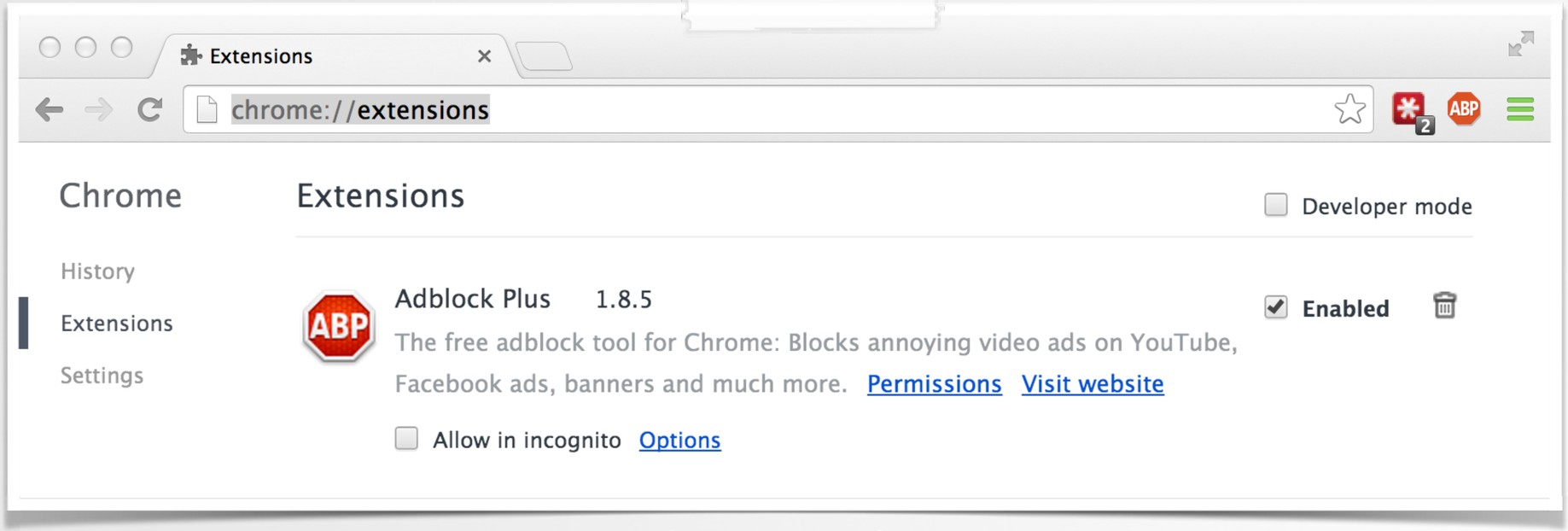
‘evil plugin’ (fG!)

safari

firefox

chrome

iTunes

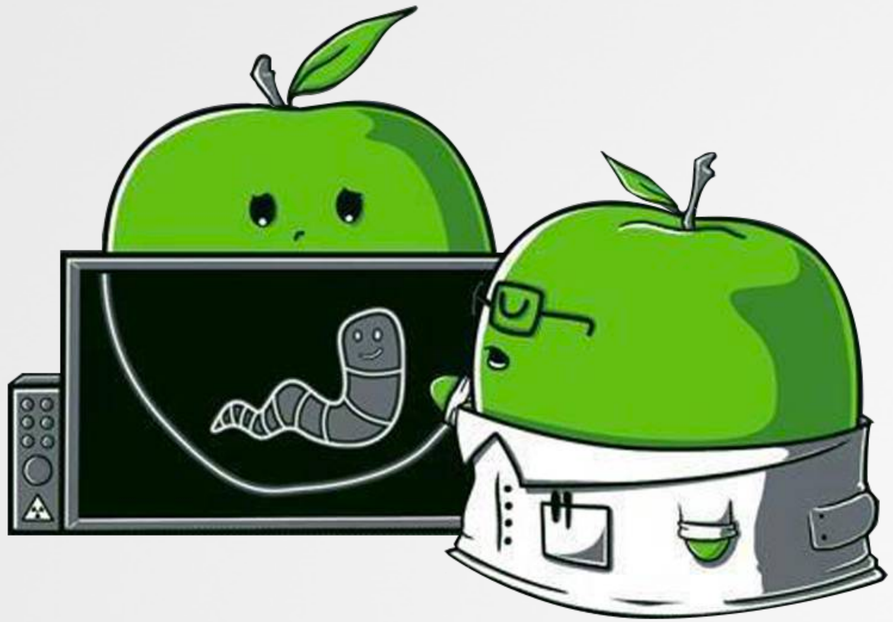


browser (chrome) extensions

Synack

PERSISTENCE OS X MALWARE

how the bad guys do it



Synack

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
```

```
$ ls /Library/LaunchDaemons/real*
realPlayerUpdate.plist

$ ps aux | grep -i real
root 0:00.06 /Library/Application Support/.realPlayerUpdate
```

the malware

launch daemon persistence

Synack

OSX/FlashBack (LaunchAgent)

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/>
```

malware's binary

persist

(user) launch agent persistence

OSX/Crisis (LauchAgent)

collects audio, images, screenshots and

IDA (pseudo) disassembly

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"];
```

Synack

OSX/XSLCmd (launch Agent)

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

persistence

```
__cstring:0000E910
 clipboardd db 'clipboardd',0
 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>
```

launch daemon persistence

Synack

OSX/JANICAB (CRONJOB)

collects audio and 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

Synack

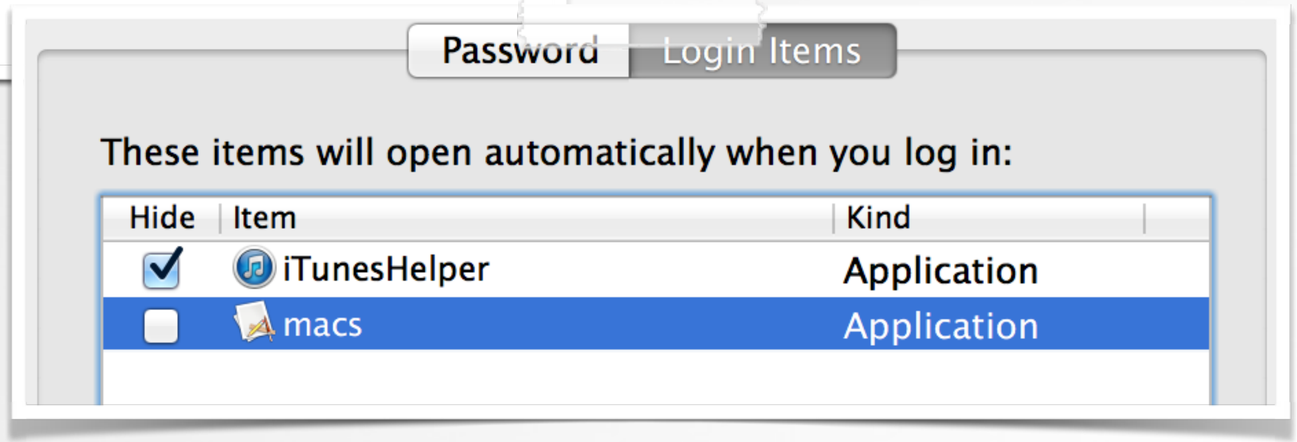
OSX/KITMOS (LOGIN ITEM)

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

user's login items

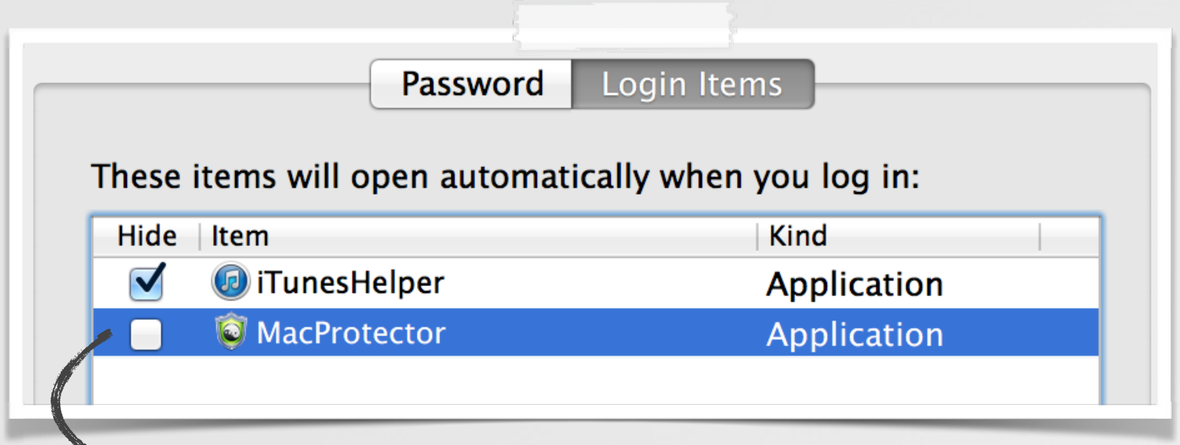


login item persistence

Synack

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>
  ZG5pYgAAAAACAAAAAAAAAAAAAAAAAAAAA...
  </data>
  <key>Name</key>
  <string>MacProtector</string>
</dict>
```

base64 encoded path, etc

login item persistence

Synack

OSX/YONTOO (BROWSER EXTENSION)

injects adds into users' browser sessions

extracted from IDA disassembly

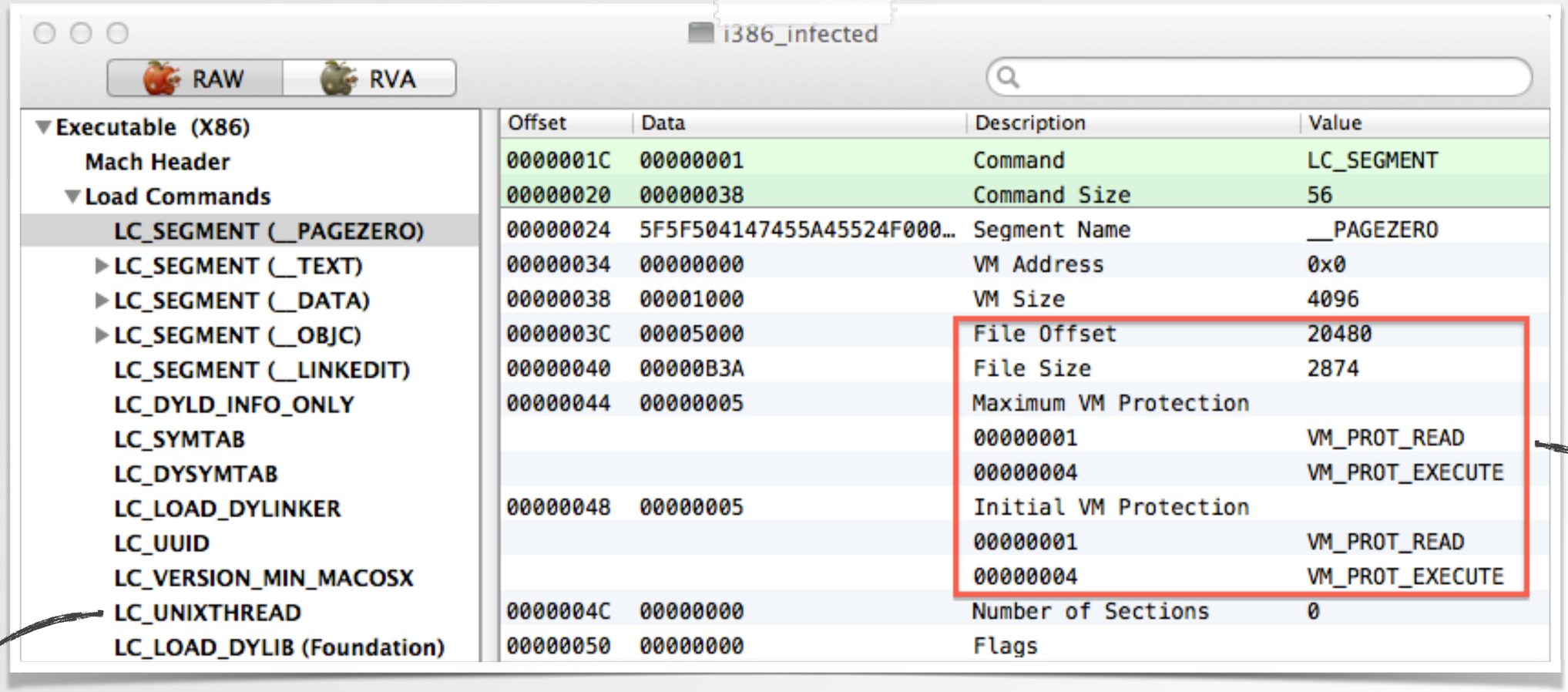
```
;create paths for malicious plugins

lea   edi, cfstr_InstallingExte; "Installing extensions"
lea   ebx, cfstr_Ok            ; "Ok"
...

+[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

OSX/CLAPZOK (VIRAL INFECTION)

researcher released proof-of-concept

image src: http://reverse.put.as/ (fG!)



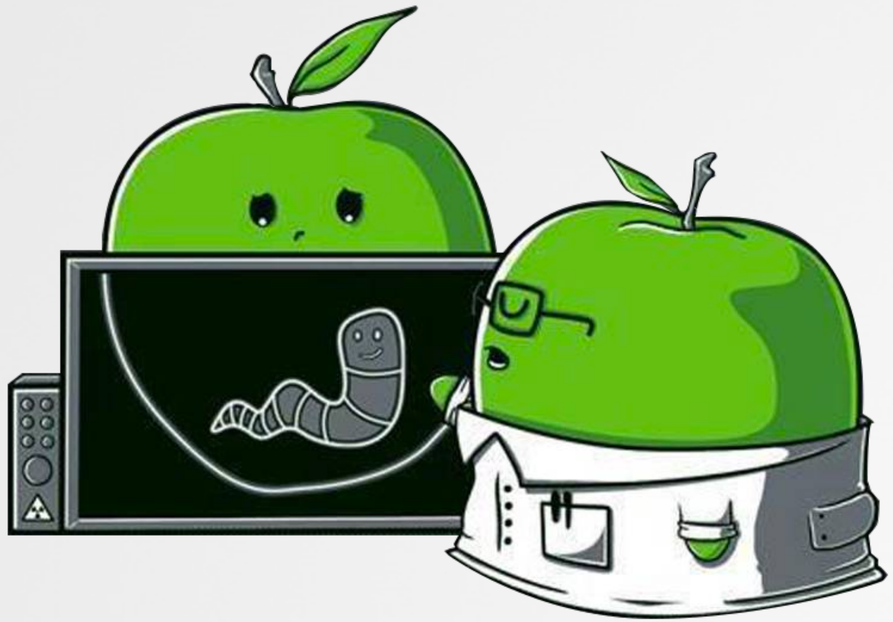
entry point
load command

binary infection persistence

Synack

KNOCK KNOCK

‘autoruns’ for OS X



Synack

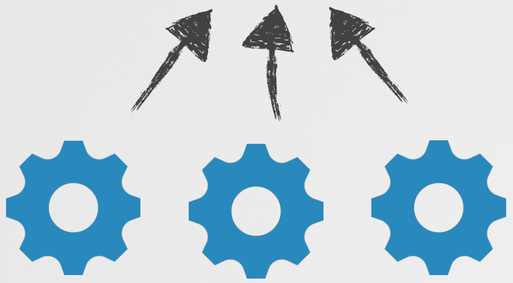
KNOCK KNOCK'S DESIGN & GOALS

finds stuff that will automatically execute during startup

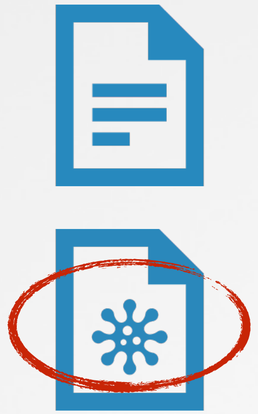
scan

core engine

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



plugins



malware

auto run binaries
& commands

github.com/synack/knockknock

Synack

KnockKnock's Plugins

one for each particular persistence technique

```
#launch daemon directories
LAUNCH_DAEMON_DIRS = ['/System/Library/LaunchDaemons/', '/Library/LaunchDaemons/']

#init results
results['launchAgents'] = []

#get all files in launch daemon directories
for directory in LAUNCH_DAEMON_DIRS:
   launchItems.extend(glob.glob(directory + '*'))

#iterate over launch daemon files (plists)
# ->find/save ones that are set to auto run
for item in launchItems:
   plistData = utils.loadPlist(item)

   if plistData[‘RunAtLoad'] or plistData['KeepAlive']:
      results.append(file.File(plistData['ProgramArguments'][0]))
```

persistence?

(simplified) launch daemon plugin

KnockKnock Output

(currently) command-line

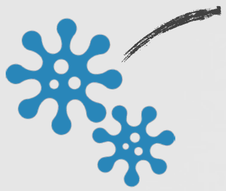
SHA256: 1db30d5b2bb24bcc4b68d647c6a2e96d984a13a28cc5f17596b3bfe316cca342

File name: clipboardd

Detection ratio: 0 / 51

Analysis date: 2014-08-12 16:34:34 UTC (1 week, 2 days ago)


```
$ 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: 60242ad3e1b6c4d417d4dfeb8fb464a1

TOTAL ITEMS FOUND: 1
```



OSX/XSLCmd detection

Synack

Some Conclusions

with tonnes of persistence methods

insecure macs

+

os x malware

=

but knowledge is power
& **knockknock** can help!

Questions & Answers

...feel free to contact me any time!

patrick@synack.com

@patrickwardle

syn.ac/virusb2014

try it out!

github.com/synack/knockknock

Synack

credits

thezooom.com
deviantart.com (FreshFarhan)

iconmonstr.com
flaticon.com