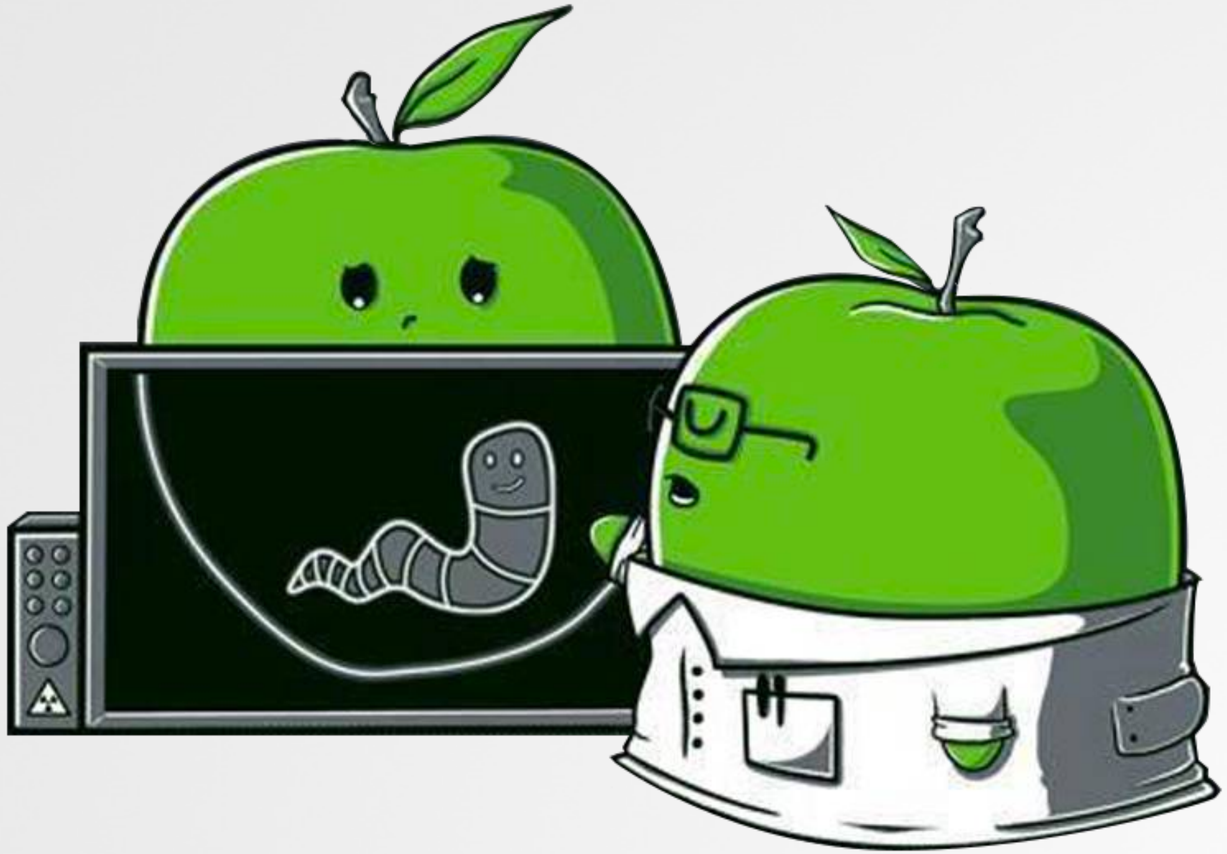
Demystifying & breaking macOS's

Background Task Management



WHOAMI

Patrick Wardle

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

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

What You Will Learn

Although the talk is largely focused on macOS's Background Task Management (BTM) - we'll also cover topics of reversing, malware detection, macOS internals, and more!

1

Internals of macOS's BTM

2

Leveraging BTM to detect malware

3

Bypassing BTM (alerts & ES messaging)

4

Detecting these bypasses

How We're Going To Get There

Overview

Internals

Tools

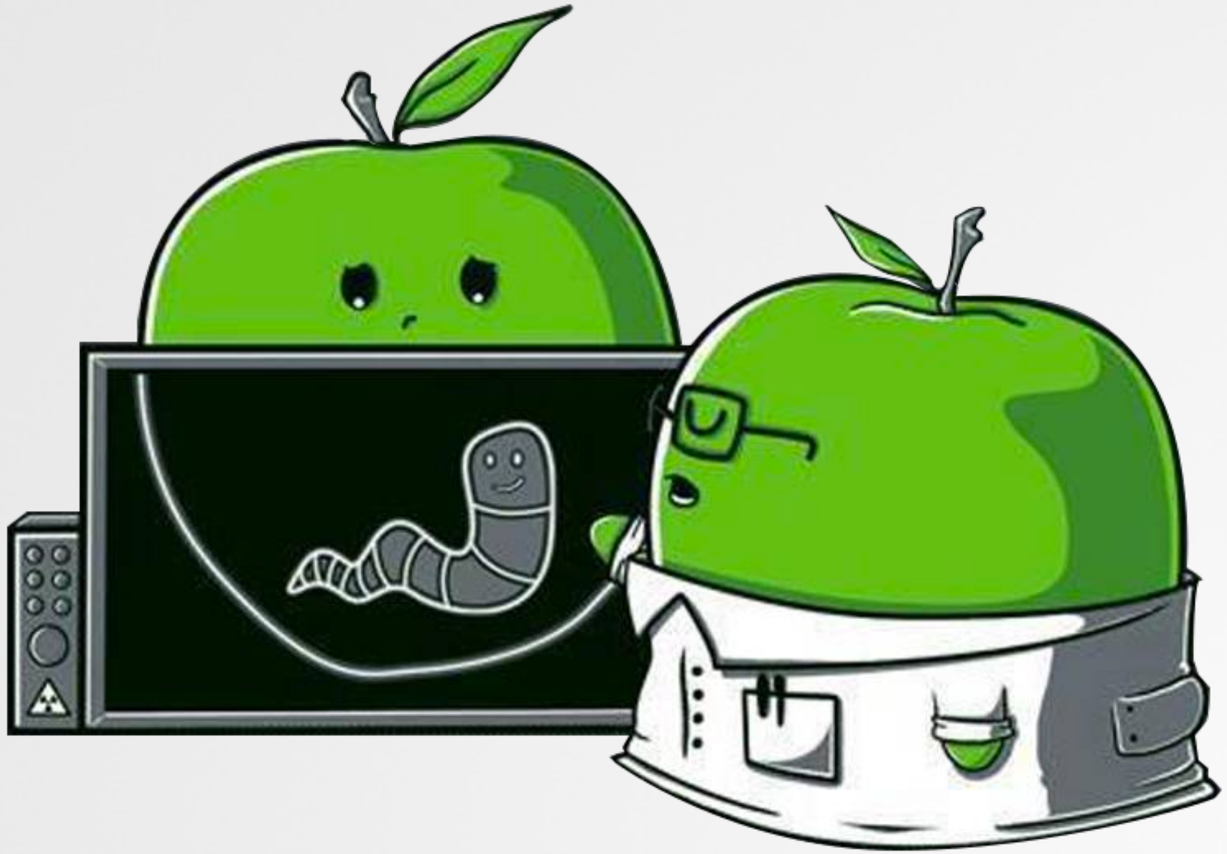
Bypasses

Detection

Dumper

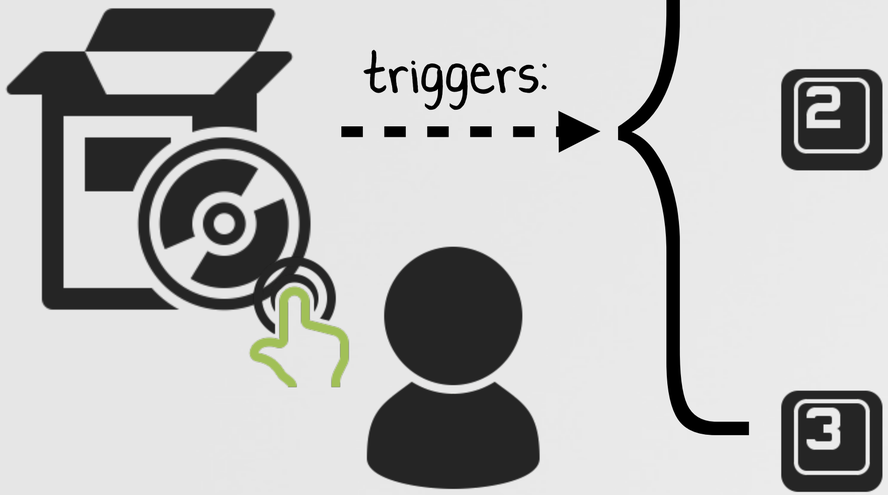
Monitor

Overview

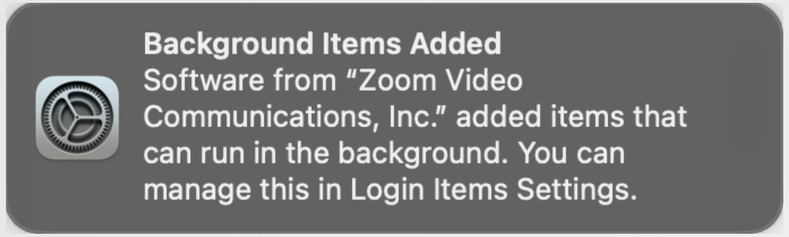


So What is Background Task Management (BTM)?

governance of persistent items ("background tasks")



1



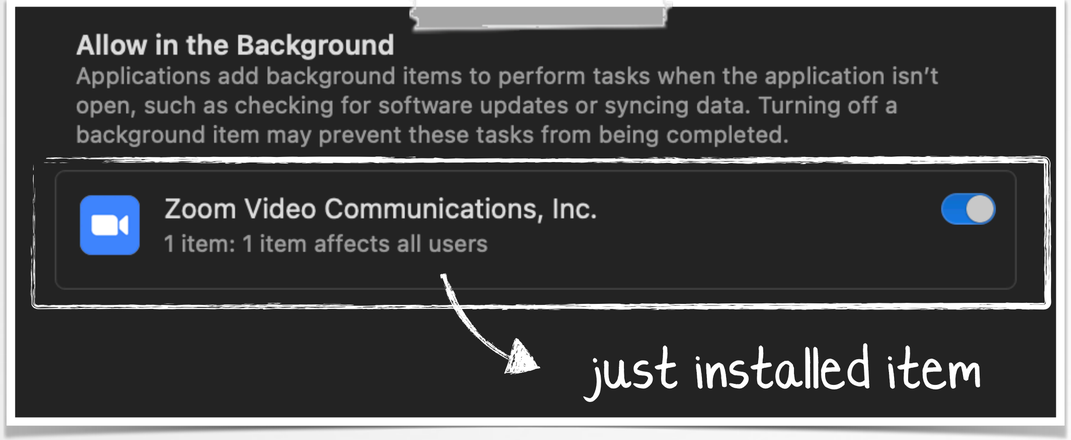
System Alert shown to user

2

```
# ./btm_monitor
New Event: 'ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD'

Item: /Library/PrivilegedHelperTools/us.zoom.ZoomDaemon
```

Endpoint Security event

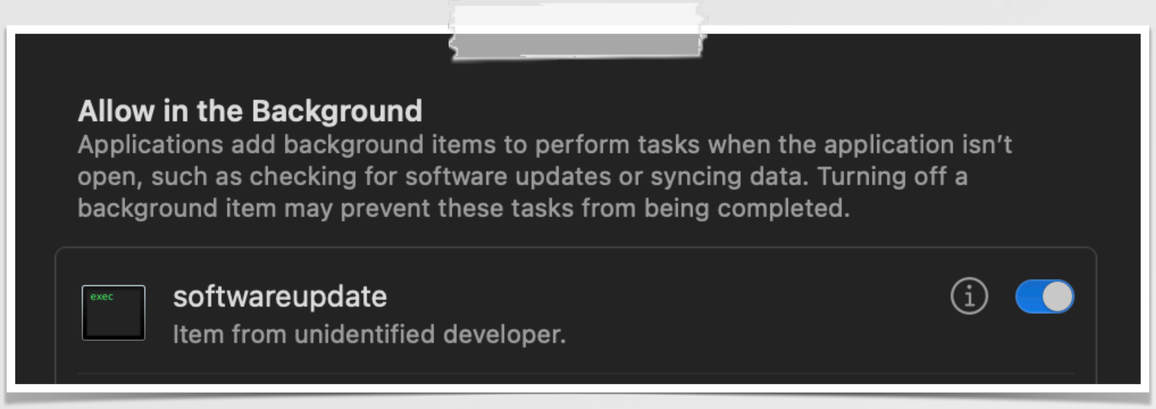
3

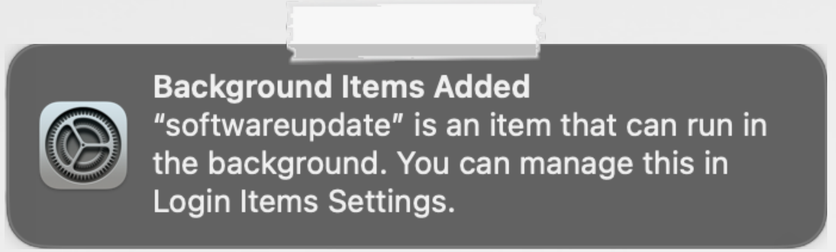


System "registration"

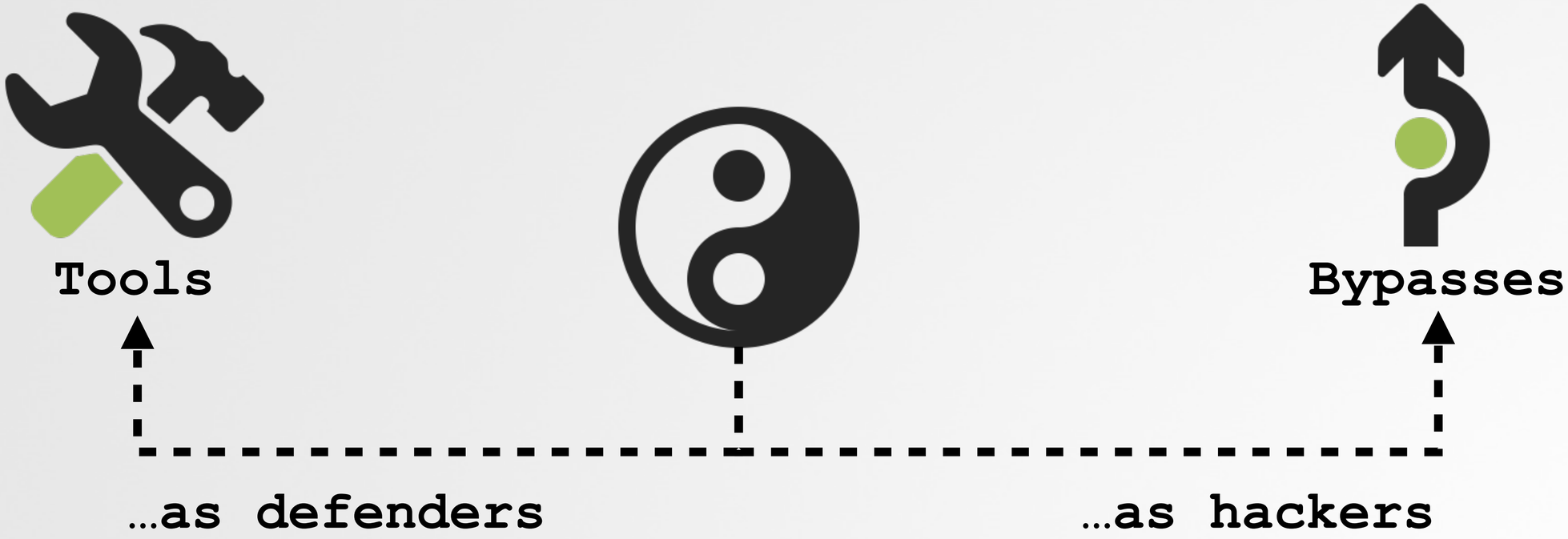
Who Cares?

well, we do!





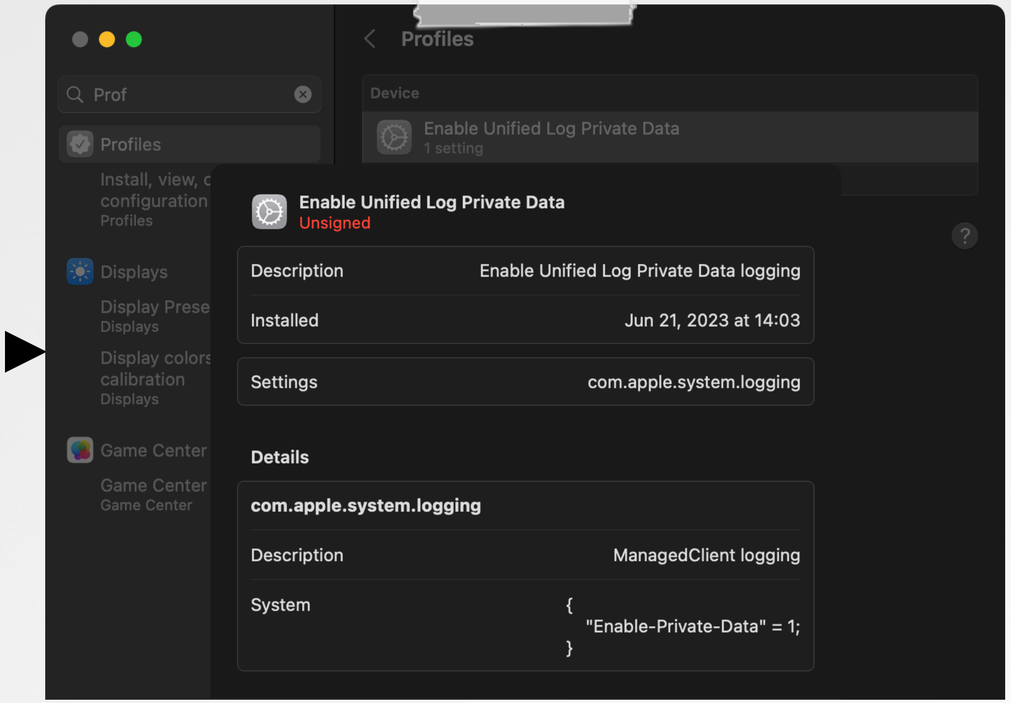

```
# ./btm_monitor
'ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD'
~/Library/LaunchAgents/com.apple.softwareupdate.plist
```



Where to Begin?

...to the logs (after enabling 'private' data)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" ...>
<plist version="1.0">
<dict>
  <key>PayloadContent</key>
  <array>
    <dict>
      <key>PayloadDisplayName</key>
      <string>ManagedClient logging</string>
      <key>PayloadEnabled</key>
      <true/>
      <key>PayloadIdentifier</key>
      <string>com.apple.logging.ManagedClient.1</string>
      <key>PayloadType</key>
      <string>com.apple.system.logging</string>
      <key>PayloadUUID</key>
      <string>ED5DE307-A5FC-434F-AD88-187677F02222</string>
      <key>PayloadVersion</key>
      <integer>1</integer>
      <key>System</key>
      <dict>
        <key>Enable-Private-Data</key>
        <true/>
      </dict>
    </dict>
  </array>
  …
```



Private Data Logging
(installed profile)

"Unified Logs:
How to Enable Private Data"
(www.cmdsec.com)

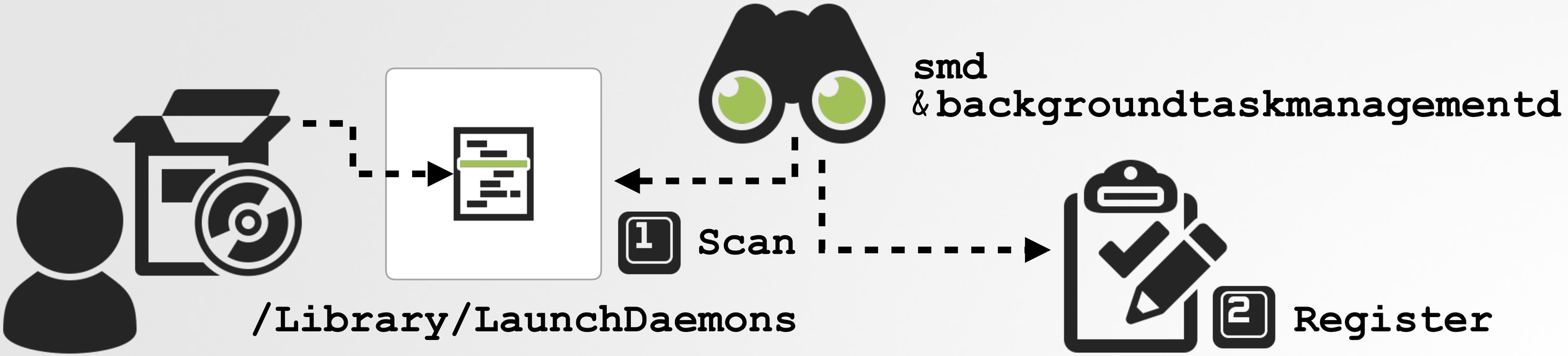
LOGGING

...while installing a launch daemon

```
% log stream --debug --info --predicate "subsystem = 'com.apple.backgroundtaskmanagement'"

smd: [com.apple.xpc.smd:all] Got a fsevent. Doing re-register
smd: [com.apple.xpc.smd:all] Bootstrapping legacy plists

backgroundtaskmanagementd: -[BTMService launchdWillScanPath:reply:]: /Library/LaunchDaemons

smd: (BackgroundTaskManagement) BTMManager.registerLaunchItemWithBundleURL

backgroundtaskmanagementd: registerLaunchItem: pid=0, uid=0, type=legacy daemon, parentURL=(null),
url=/Library/LaunchDaemons/us.zoom.ZoomDaemon.plist, config=
    BTMConfigArguments =
        "/Library/PrivilegedHelperTools/us.zoom.ZoomDaemon"
```

method: "launchd will scan path"

method: "register launch item"



"SMD": System Management Daemon

file system event notification, for new item

```
01  sub_100005a28
02  ...
03
04      adr   x3, #0x1027d7243 ; argument "format"
05                               ("Got a fsevent. Doing re-register")
06      nop
07      mov   x4, sp    ; argument "buf"
08      mov   x1, x19   ; argument "log"
09      mov   w2, #0x0  ; argument "type"
10      mov   w5, #0x2  ; argument "size"
11      bl    os_log()
```

log message: "Got a fsevent"

```
Process 300 stopped
smd`___lldb_unnamed_symbol:
->  0x100005a28 <+0>:  pacibsp

(lldb) po $x1
 <OS_xpc_dictionary: dictionary[0x13290d290]: { refcnt = 1, xrefcnt = 1, subtype = 1, count = 1, transport = 0,
 dest port = 0x0, dest msg id = 0x0, transaction = 1, voucher = 0x13290bff0 } <dictionary: 0x13290d290>
 { count = 1, transaction: 1, voucher = 0x13290bff0,
   contents = "XPCEventName" => <string: 0x131f06eb0> { length = 28, contents = "com.apple.xpc.smd.WatchPaths" }
 }>
```

"XPCEventName: com.apple.xpc.smd.WatchPaths"

debugging SMD

"SMD": System Management Daemon

file system event notification, for new item

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" ... >
03  <plist version="1.0">
04  <dict>
05      ...
06      <key>Program</key>
07      <string>/usr/libexec/smd</string>
08
09      <key>LaunchEvents</key>
10      <dict>
11         <key>com.apple.fsevents.matching</key>
12         <dict>
13            <key>com.apple.xpc.smd.WatchPaths</key>
14         <dict>
15         <key>Path</key>
16         <string>/Library/LaunchDaemons/</string>
17      </dict>
18      ...
```

file system watch events:
/Library/LaunchDaemons/

/Library/LaunchDaemons

SMD

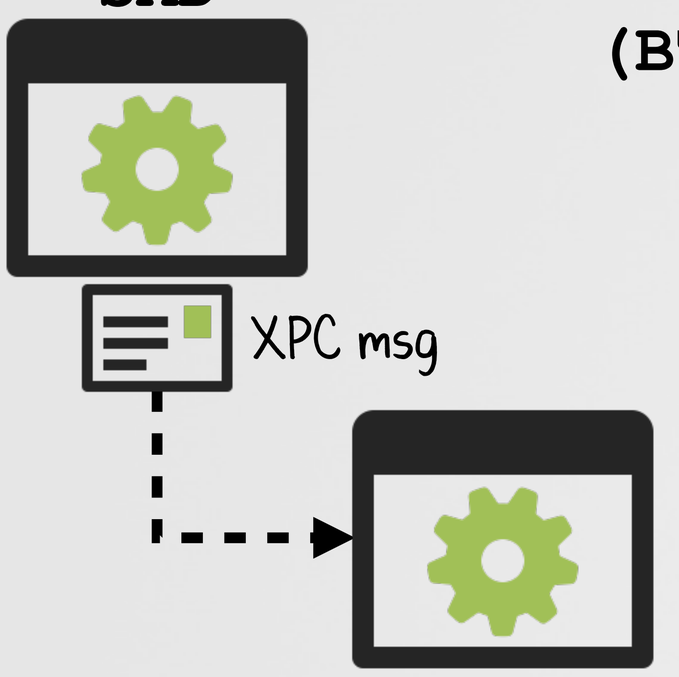
"SMD": System Management Daemon

...and its "remote" XPC interactions

```
01  * @class BTMManager */
02  -(int)launchdWillScanPath:(int)arg2 {
03      x1 = arg1;
04      x2 = [arg2 retain];
05      x0 = [arg0 connection];
06
07      x21 = [x0 synchronousRemoteObjectProxyWithErrorHandler:...];
08      [x21 launchdWillScanPath:x2 reply:...];
```

<NSXPCConnection> connection to service named com.apple.backgroundtaskmanagement

launchdWillScanPath:
(BTM framework, linked into SMD)




```
Process 300 stopped
BackgroundTaskManagement`-[BTMManager launchdWillScanPath:]:

(lldb) po $x0
<__NSXPCInterfaceProxy_V2ServiceInterface: 0x131e0b690>

(lldb) x/s $x1
0x1d4037471: "launchdWillScanPath:reply:"

(lldb) po $x2
/Library/LaunchDaemons
```

"scan path": /Library/LaunchDaemons

debugging SMD

XPC Messages

...between SMD and BTM daemon

```
# ps aux | grep backgroundtaskmanagementd
root  4643  /System/Library/PrivateFrameworks/BackgroundTaskManagement.framework/Resources/backgroundtaskmanagementd -daemon

# lsmp -p 4643
Process (4643) : backgroundtaskmanagementd
  name       ipc-object    rights      identifier   type
---------    ----------    ------
0x0000310b   0x66b39d3f    send        0x00001807   (300) smd

0x00003b0b   0x677147df    recv        0x0000000000000000
                   +      send        0x000010af   (300) smd
```

XPC messages
...to and from SMD

XPC messages (via lsmp)

```
Process 4643 stopped
-[BTMService launchdWillScanPath:reply:]
->  0x102da9634 <+0>:  pacibsp

(lldb) bt
  * frame #0: 0x0000000102da9634 [BTMService launchdWillScanPath:reply:]
    frame #1: 0x000000018547eca0 Foundation`__NSXPCCONNECTION_IS_CALLING_OUT_TO_EXPORTED_OBJECT_S2__ + 16
    frame #2: 0x0000000185a9e594 Foundation`-[NSXPCConnection _decodeAndInvokeMessageWithEvent:reply:flags:] + 1592
    frame #3: 0x0000000185a9fd88 Foundation`message_handler_message + 88
    frame #4: 0x0000000185a9f7f4 Foundation`message_handler + 152
    frame #5: 0x0000000184163e74 libxpc.dylib`_xpc_connection_call_event_handler + 152
```

call stack w/ XPC functions

call stack (BTM daemon)

"BACKGROUNDTASKMANAGEMENTD": BTM DAEMON

com.apple.backgroundtaskmanagementd.plist

```
01  <key>Label</key>
02  <string>com.apple.backgroundtaskmanagementd</string>
03
04  <key>MachServices</key>
05  <dict>
06    <key>com.apple.backgroundtaskmanagement</key>
07    <true/>
08    <key>com.apple.backgroundtaskmanagement.sfl</key>
09    <true/>
10    <key>com.apple.backgroundtaskmanagement.responses</key>
11    <true/>
12  </dict>
13
14  <key>RunAtLoad</key>
15  <false/>
16
17  <key>Program</key>
18  <string>/System/Library/PrivateFrameworks/BackgroundTaskManagement.framework/Resources/backgroundtaskmanagementd
19  </string>
20
21  <key>ProgramArguments</key>
22  <array>
23  <string>/System/Library/PrivateFrameworks/BackgroundTaskManagement.framework/Resources/backgroundtaskmanagementd
24  </string>
25  <string>-daemon</string>
26  </array>
```

Mach interfaces
(for entitled clients)

BTM daemon binary

"BACKGROUNDTASKMANAGEMENTD": BTM DAEMON

binary & entitlements

(always) running as root

```
% ps aux | grep backgroundtaskmanagementd
root  /System/Library/PrivateFrameworks/BackgroundTaskManagement.framework/Resources/backgroundtaskmanagementd -daemon
```

backgroundtaskmanagementd

(/System/Library/PrivateFrameworks/BackgroundTaskManagement.framework)

Entitlements:

```
01  {
02      "com.apple.private.backgroundtaskmanagement.notifications" = 1;
03      "com.apple.private.security.storage.backgroundtaskmanagement" = 1;
04
05      "com.apple.private.endpoint-security.submit.btm" = 1;
06
07      "com.apple.private.tcc.allow" =     (
08          kTCCServiceSystemPolicyAllFiles
09      );
10      ...
11  }
```

BTM specific

Endpoint Security

"BACKGROUNDTASKMANAGEMENTD": BTM DAEMON

launch item registration

```
01  -[BTMService registerLaunchItemWithAuditToken:parentURL:type:relativeURL:configuration:uid:reply:]
02  adr    x3, #0x102dcf8cc   ; argument "format"
03                                "registerLaunchItem: pid=%d, uid=%d, ... url=%@, config=%@"
04  nop
05  add    x4, sp, #0x20      ; argument "buf"
06  mov    x1, x27            ; argument "log"
07  mov    w2, #0x0           ; argument "type"
08  mov    w5, #0x36          ; argument "size"
09  bl     os_log()
```

registerLaunchItemWithAuditToken: (BTM daemon)

```
Process 4643 stopped
-[BTMService registerLaunchItemWithAuditToken:parentURL:type:relativeURL:configuration:uid:reply:]
->  0000000102da9898 <+0>:  pacibsp

(lldb) po $x0
<BTMService: 0x129904630>

(lldb) po $x5
file:///Library/LaunchDaemons/us.zoom.ZoomDaemon.plist
```

plist (of launch item) to register

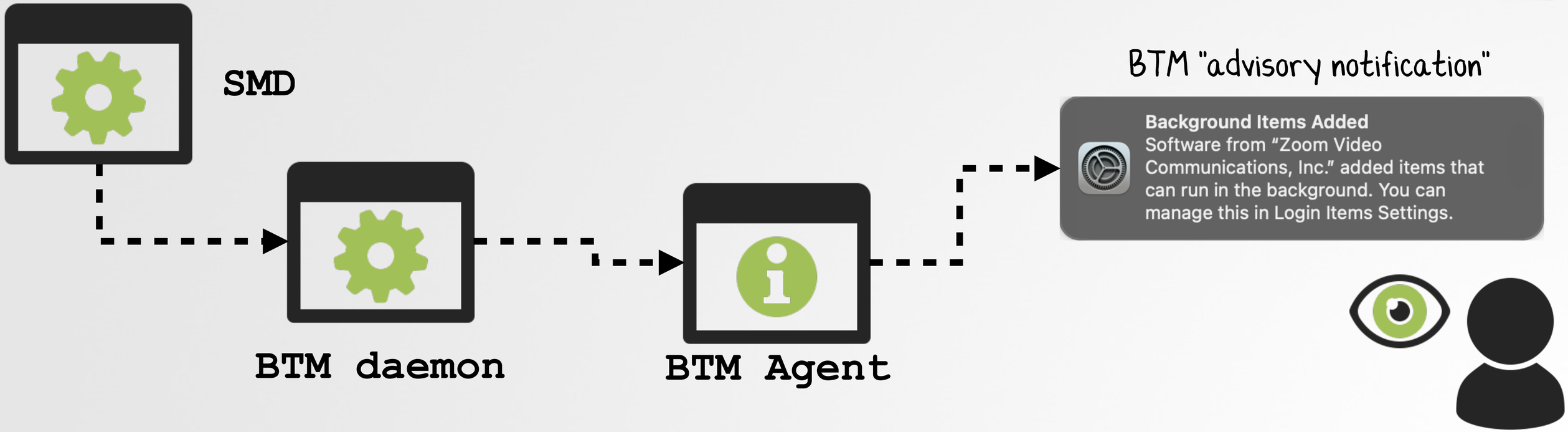
More Logging

BTM daemon & agent: posting "advisory" notification

```
% log stream --debug --info --predicate "subsystem = 'com.apple.backgroundtaskmanagement'"

backgroundtaskmanagementd: [com.apple.backgroundtaskmanagement:main] should post new container advisory=true for uid=501,
id=905C1526-CC7A-4343-9C97-55DED87CC397, item=Zoom Video Communications, Inc.

BackgroundTaskManagementAgent: [com.apple.backgroundtaskmanagement:main] Posting new container advisory notification
request: identifier=905C1526-CC7A-4343-9C97-55DED87CC397, body='Software from "Zoom Video Communications, Inc.…'
```

"should post advisory = true"

"posting new advisory notification"



"BackgroundTaskManagementAgent": BTM Agent

com.apple.backgroundtaskmanagement.agent.plist

```
01  <key>Label</key>
02  <string>com.apple.backgroundtaskmanagement.agent</string>
03
04  <key>MachServices</key>
05  <dict>
06    <key>com.apple.backgroundtaskmanagementagent</key>
07    <true/>
08    <key>com.apple.backgroundtaskmanagement.notifications</key>
09    <true/>
10    <key>com.apple.usernotifications.delegate.com.apple.BTMNotificationAgent</key>
11    <true/>
12  </dict>
13
14  <key>RunAtLoad</key>
15  <false/>
16
17  <key>Program</key>
18  <string>/System/Library/PrivateFrameworks/BackgroundTaskManagement.framework/Support/
19  BackgroundTaskManagementAgent.app/Contents/MacOS/BackgroundTaskManagementAgent</string>
```

Mach interfaces (for by entitled clients)

BTM agent binary

BTM agent's property list

"BackgroundTaskManagementAgent": BTM Agent

binary & entitlements

```
% ps aux | grep -i backgroundtaskmanagementagent
patrick   /System/Library/PrivateFrameworks/BackgroundTaskManagement.framework/Support/BackgroundTaskManagementAgent.app/
Contents/MacOS/BackgroundTaskManagementAgent
```

runs as user

BackgroundTaskManagementAgent

(/System/Library/PrivateFrameworks/BackgroundTaskManagement.framework/Support/BackgroundTaskManagementAgent.app/)

Entitlements:

```
01  {
02    "com.apple.private.backgroundtaskmanagement.manage" = 1;
03    "com.apple.private.backgroundtaskmanagement.responses" = 1;
04    "com.apple.private.coreservices.canaccessanysharedfilelist" = "read-write";
05
06    "com.apple.private.tcc.allow" =      (
07       kTCCServiceSystemPolicyAllFiles
08    );
09
10    "com.apple.private.usernotifications.bundle-identifiers" =      (
11       "com.apple.BTMNotificationAgent"
12    );
13  }
```

BTM specific

Notifications

"Should Post New Container Advisory"?

interactions between SMD & BTM daemon

invoked by: -[DaemonNotificationManager postAdvisoryNotificationForContainer:uid:]

```
01  bl   [DaemonNotificationManager shouldNotifyUpdatingItem:...]
02  mov  x20, x0
03
04  cbz  w20, noNeedToPostNotification
05
06  rax = [rdi synchronousProxyForUID:...]
07  [rax postAdvisoryForContainer:rax reply:&var_B0]
08
09  rax = [NSDictionary dictionaryWithObjects:&var_68 forKeys:&var_70 count:0x1]
10  AnalyticsSendEvent(@"com.apple.BackgroundTaskManager.Notification", [rax retain])
11
12  noNeedToPostNotification:
13  ret
```

if this return true:
then, show notification
(via XPC message to BTM agent)

"shouldNotifyUpdatingItem:" method check:

- If item is managed (MDM)
- Was allowed / was already notified
- "AlwaysPostNotifications" preference

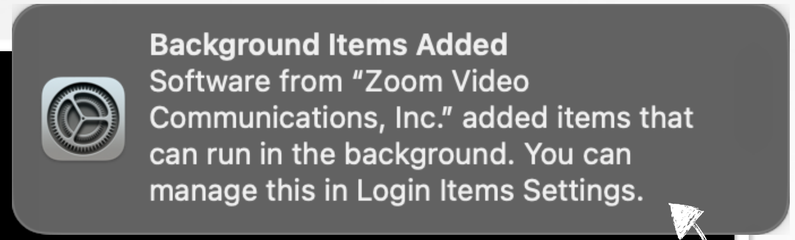
"Post Advisory For Container"

interactions BTM daemon & agent

in BTM Agent,
(invoked by BTM Daemon, via XPC)

```
01  /* @class NotificationManager */
02  -(void)postAdvisoryForContainer:(BTMItem*)item reply:(void *)arg3 {
03
04  content = [NotificationManager
05  copyAdvisoryContentWithBodyKey:@"NOTIFICATION_BODY_BACKGROUND_ITEM_ADVISORY" name:name];
06
07  notification = [UNNotificationRequest requestWithIdentifier:id content:content trigger:0x0];
08
09  notificationCenter = [NotificationManager notificationCenter];
10  [notificationCenter addNotificationRequest:notification withCompletionHandler:...];
```

```
% ps aux | grep BackgroundTaskManagementAgent
patrick   88262  /System/Library/PrivateFrameworks/BackgroundTaskManagement.framework
/Support/BackgroundTaskManagementAgent.app/Contents/MacOS/BackgroundTaskManagementAgent

(lldb) process attach --pid 88262

Target 0: (BackgroundTaskManagementAgent) stopped.
(lldb) po $x0
<UNMutableNotificationContent: 0x12684df10; title: Background Items Added, subtitle: (null), body: Software from "Zoom
Video Communications" added items that can run in the background. You can manage this in Login Items Settings.,
summaryArgument: , summaryArgumentCount: 0, categoryIdentifier: com.apple.BackgroundTaskManagement.advisory,
launchImageName: , threadIdentifier: , attachments: (
), badge: (null), sound: (null), realert: 0, interruptionLevel: 1, relevanceScore: 0.00, filterCriteria: (null)
```



notification (+content)

"BACKGROUNDTASKMANAGEMENTD": BTM DAEMON

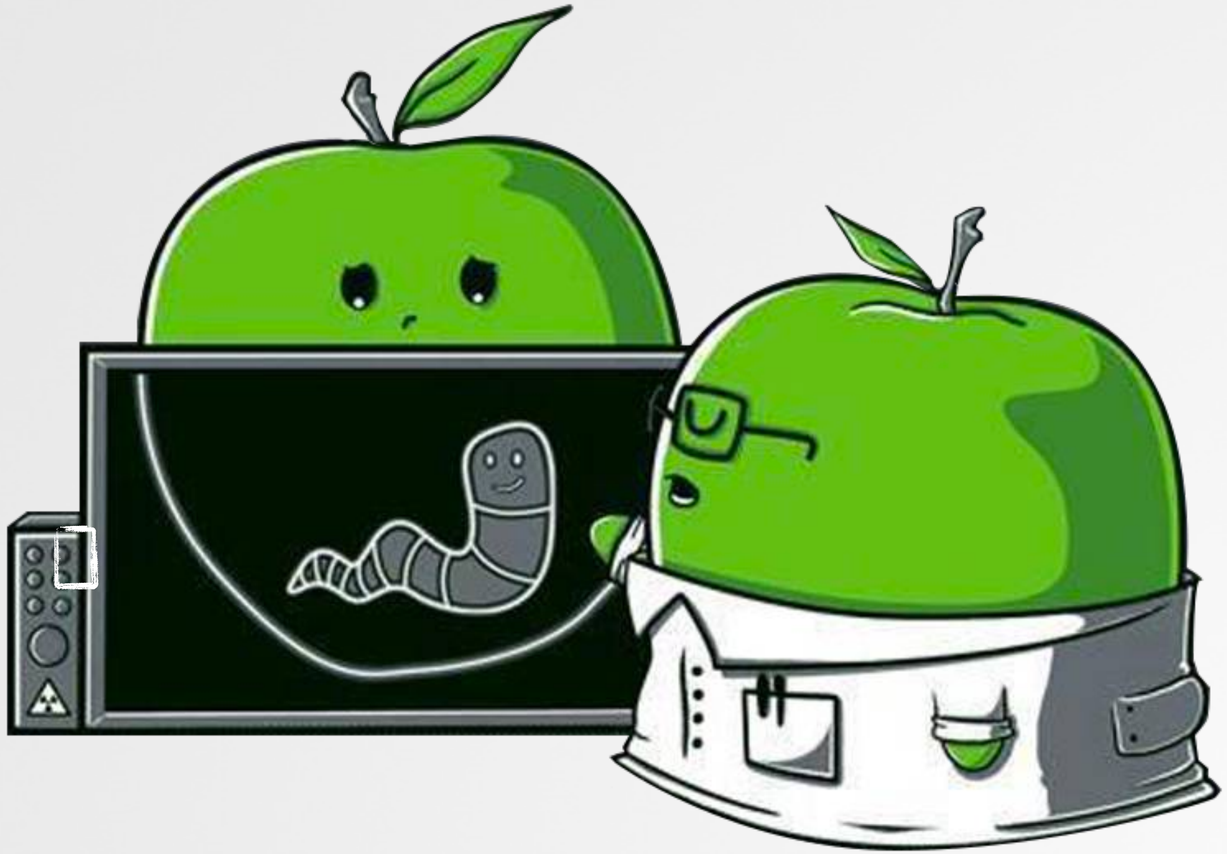
saving registered item to "BackgroundItems-v8.btm"

```
% log stream --debug --info --predicate "subsystem = 'com.apple.backgroundtaskmanagement'"

backgroundtaskmanagementd: [com.apple.backgroundtaskmanagement:main] BTMStore:
 store saved to /var/db/com.apple.backgroundtaskmanagement/BackgroundItems-v8.btm
```

```
# FileMonitor.app/Contents/MacOS/FileMonitor -pretty
{
  "event" : "ES_EVENT_TYPE_NOTIFY_CLOSE",
  "file" : {
    "destination" : "/private/var/db/
                     com.apple.backgroundtaskmanagement/BackgroundItems-v8.btm",
    "process" : {
      "name" : "backgroundtaskmanagementd",
      "path" : "/System/Library/PrivateFrameworks/
                BackgroundTaskManagement.framework/Versions/A/Resources/backgroundtaskmanagementd",
    }
...
```

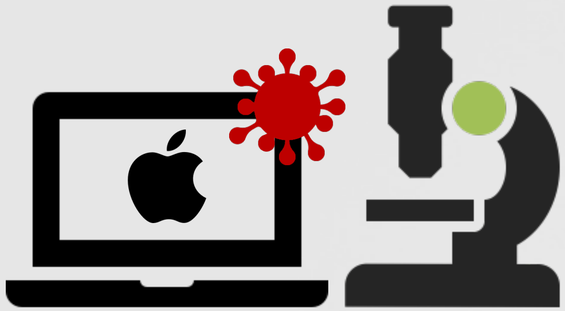
Tools (Part I)

leveraging BTM to detect persistent malware



Mac Malware

...the vast majority persists!



The Mac Malware of 2022 👾

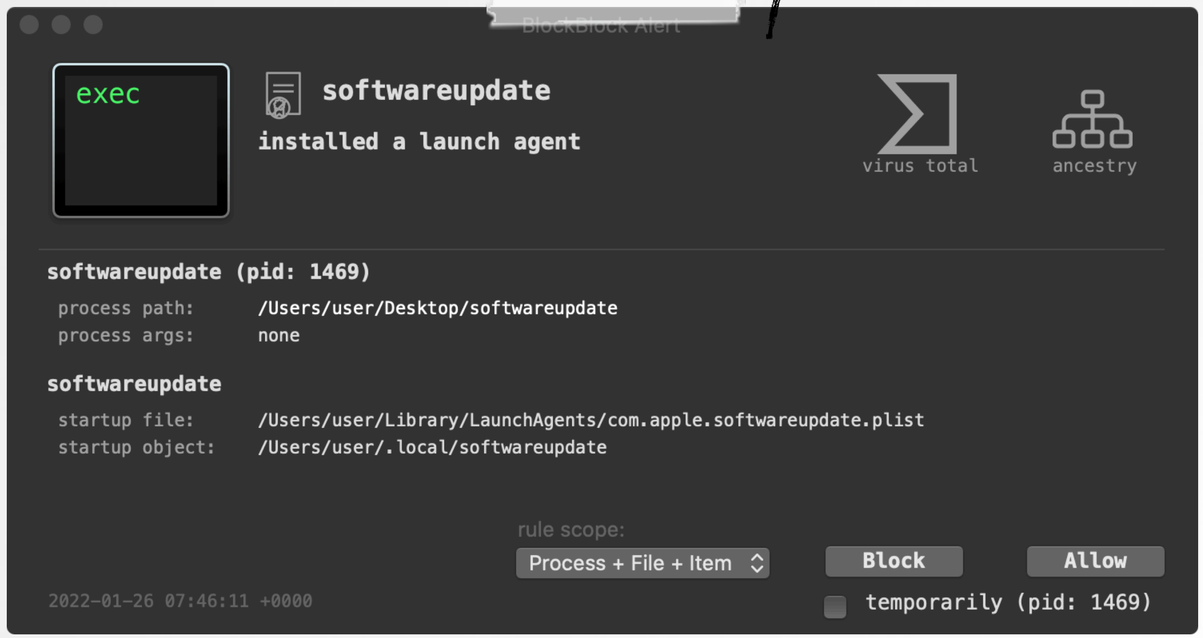
A comprehensive analysis of the year's new malware

by: Patrick Wardle / January 1, 2023

2022: ~80% persisted
(all as launch items)

2021: ~90% persisted
(all as launch items)

alert user when something persists !



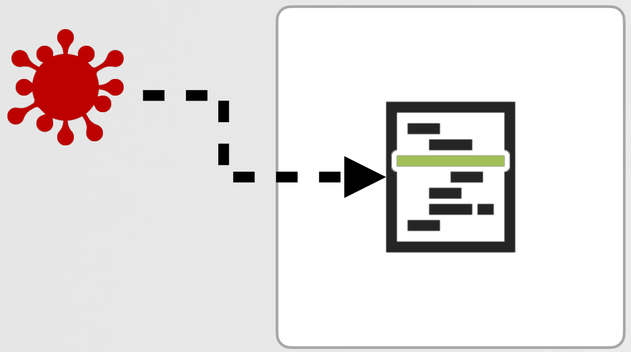
BlockBlock
...block block'ing since 2015!

Example: DazzleSpy

…deployed via Safari 0days

```
% strings - DazzleSpy/softwareupdate
...
%@/Library/LaunchAgents
/com.apple.softwareupdate.plist

launchctl unload %@
RunAtLoad
```

Embedded strings



/Library/LaunchAgents

```
# FileMonitor.app/Contents/MacOS/FileMonitor -pretty
...
{
  "event" : "ES_EVENT_TYPE_NOTIFY_CREATE",
  "file" : {
    "destination" : "~/Library/LaunchAgents/
                     com.apple.softwareupdate.plist",

    "process" : {
       "pid" : 1469
       "name": softwareupdate
       "path" : "/Users/user/Desktop/softwareupdate",
    }
  }
}
```

Launch agent persistence

(~/Library/LaunchAgents/com.apple.softwareupdate.plist)

Dumping the BTM Database

…to enumerate persistently installed software

and malware!

```
# sfltool
Usage: sfltool csinfo|dumpbtm|archive|clear|resetbtm|resetlist|list|list-info [options]
```

```
# sfltool dumpbtm
…

                  UUID: EAE1D8DC-2928-47C1-BC53-6274590FE3D1
                  Name: us.zoom.ZoomDaemon
        Developer Name: Zoom Video Communications, Inc.
       Team Identifier: BJ4HAAB9B3
                  Type: legacy daemon (0x10010)
           Disposition: [enabled, allowed, visible, notified] (11)
            Identifier: us.zoom.ZoomDaemon
                   URL: file:///Library/LaunchDaemons/us.zoom.ZoomDaemon.plist
       Executable Path: /Library/PrivilegedHelperTools/us.zoom.ZoomDaemon
            Generation: 1
     Parent Identifier: Zoom Video Communications, Inc.
```

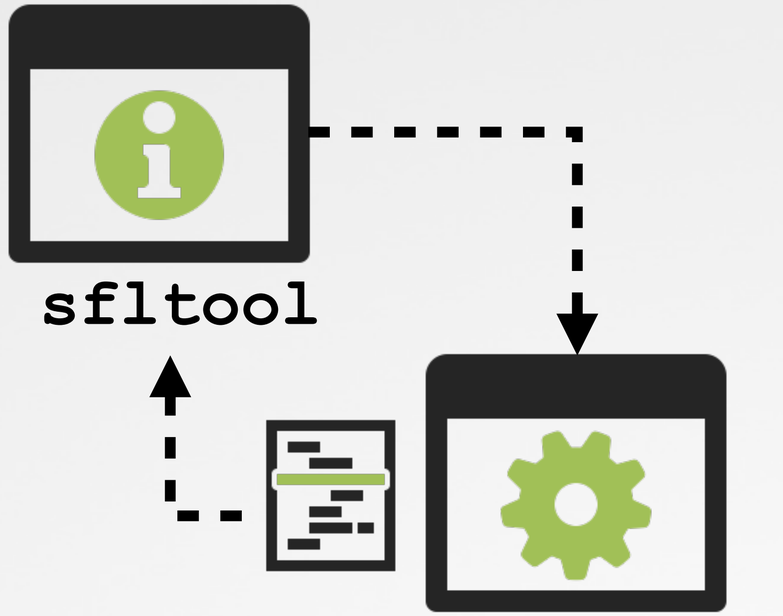
sfltool (option: dumpbtm)

REVERSING SFLTOOL

…how does it enumerate background tasks?

```
01  /* @class DumpBTMCommand */
02  -(int)run {
03      ...
04      rax = [BTMManager shared];
05      r12 = [rax dumpDatabaseWithAuthorization:r14 error:&var_28];
06      if(r12 != 0x0) {
07          r12 = [objc_retainAutorelease(r12) UTF8String];
08          puts(r12);
09      }
10      else {
11          NSLog(@"Error fetching the database dump: %@", r15);
12      }
```

sfltool decompilation
[DumpBTMCommand run];




```
% log stream …
```

mach connection (from sfltool) to: com.apple.backgroundtaskmanagement"

```
backgroundtaskmanagementd: -[BTMService listener:shouldAcceptNewConnection:]:
connection=<NSXPCConnection: 0x152307aa0> connection from pid 52886 on mach service named
com.apple.backgroundtaskmanagement

backgroundtaskmanagementd: dumpDatabaseWithAuthorization: "noErr: Call succeeded with no error"
```

Programmatically Dumping the BTM Database

mimicking: `sfltool dumpbtm`

```
01  @interface BTMManager : NSObject
02
03  +(id)shared;
04  -(id)dumpDatabaseWithAuthorization:(SFAuthorization*)arg1 error:(id*)arg2;
05
06  @end
07
08  void *handle = dlopen("/System/Library/PrivateFrameworks/BackgroundTaskManagement.framework/Versions/
09  A/BackgroundTaskManagement", RTLD_LAZY);
10
11  Class BTMManager = NSClassFromString(@"BTMManager");
12  id sharedInstance = [BTMManager shared];
13
14  SFAuthorization *authorization = [SFAuthorization authorization];
15  [authorization obtainWithRight:"system.privilege.admin", ...];
16
17  id dbContents = [sharedInstance dumpDatabaseWithAuthorization:authorization error:NULL];
```

invoke: `dumpDatabaseWithAuthorization:...`

```
% log stream …

backgroundtaskmanagementd: -[BTMService listener:shouldAcceptNewConnection:]: process with pid=20987
lacks entitlement 'com.apple.private.coreservices.canmanagebackgroundtasks'
```

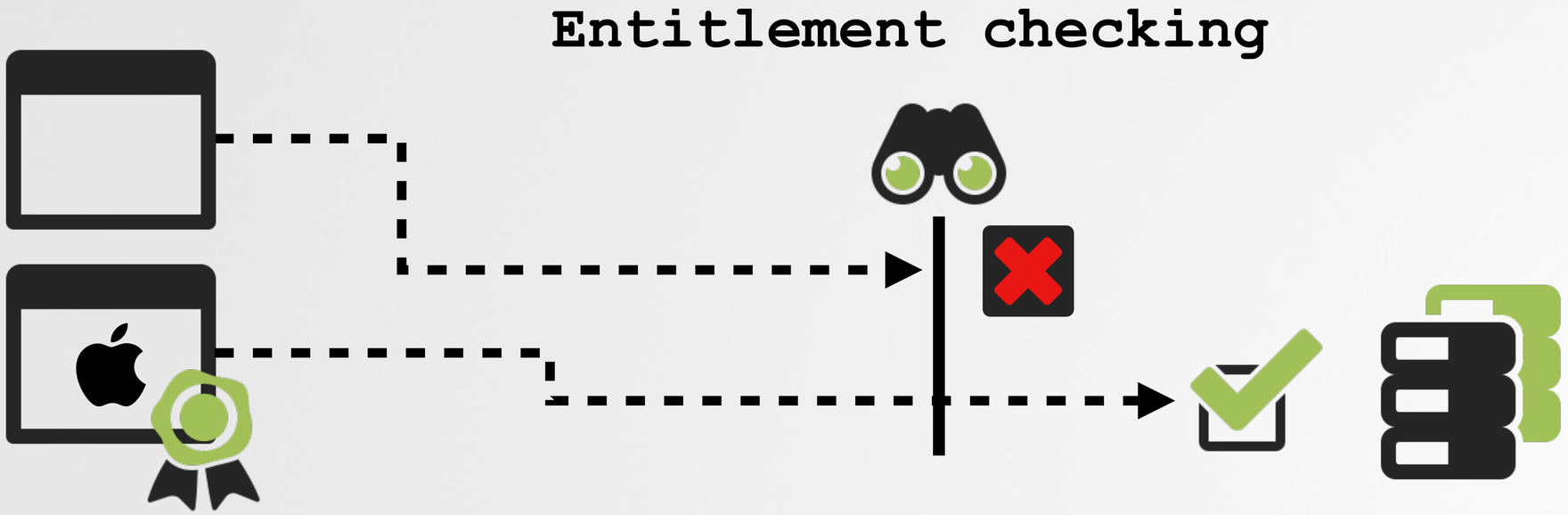
GTFO …you're not entitled

fail :

Dumping the BTM Database

…entitlement check(s) in BTM daemon

```
01  /* @class BTMService */
02  -(BOOL) listener: (NSXPCListener *)listener shouldAcceptNewConnection:(NSXPCConnection *)newConnection {
03  …
04
05  rax = [r15 valueForEntitlement:@"com.apple.private.backgroundtaskmanagement.manage"];
06  if(rax == 0x0) {
07      rax = [r15 valueForEntitlement:@"com.apple.private.coreservices.canmanagebackgroundtasks"];
08  }
09
10  if(objc_opt_isKindOfClass(rax, objc_opt_class(@class(NSNumber))) == 0x0 || [rax boolValue] == 0x0) {
11
12      //leave, don't accept connection
13  }
```



What about the (Raw) BTM Database

…a binary plist, containing serialized objects

```
% file /var/db/com.apple.backgroundtaskmanagement/BackgroundItems-v8.btm

Apple binary property list

% plutil -p /var/db/com.apple.backgroundtaskmanagement/BackgroundItems-v8.btm
{
  "$archiver" => "NSKeyedArchiver"
  "$objects" => [
    0 => "$null"
    1 => {
      "$class" => <CFKeyedArchiverUID 0x600003e57640 [0x1e69087f8]>{value = 142}
      "itemsByUserIdentifier" => <CFKeyedArchiverUID 0x600003e57660 [0x1e69087f8]>{value = 2}
      "mdmPaloadsByIdentifier" => <CFKeyedArchiverUID 0x600003e57680 [0x1e69087f8]>{value = 140}
      "userSettingsByUserIdentifier" => <CFKeyedArchiverUID 0x600003e576a0 [0x1e69087f8]>{value = 134}
   ...
```

BTM database (contains NSKeyedArchiver)

Serialization, saves objects to a persistent archive (e.g. BTM database)

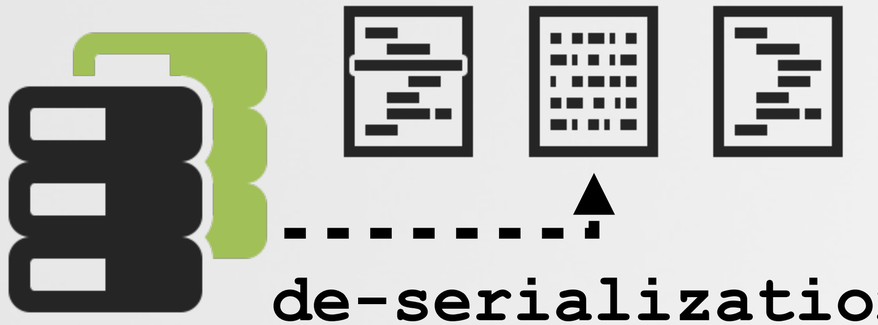
sfltool's Item De-serialization

…implemented in an initWithCoder: method

```
01  /* @class ItemRecord */
02  -(void *)initWithCoder:(NSCoder *)decoder {
03
04      rax = objc_opt_class(@class(NSUUID));
05      rax = [r14 decodeObjectOfClass:rax forKey:@"uuid"];
06
07      rax = objc_opt_class(@class(NSString));
08      rax = [r14 decodeObjectOfClass:rax forKey:@"executablePath"];
09
10      rax = objc_opt_class(@class(NSString));
11      rax = [r14 decodeObjectOfClass:rax forKey:@"teamIdentifier"];
12
13      …
```

method automatically invoked to de-serialize stored objects

BTM Daemon
[ItemRecord initWithCoder];



de-serialization

ItemRecord

Tag Scope

Idx	Name
173	-[ItemRecord uuid]
174	-[ItemRecord setUuid:]
175	-[ItemRecord identifier]
176	-[ItemRecord setIdentifier:]
177	-[ItemRecord disposition]
178	-[ItemRecord setDisposition:]
179	-[ItemRecord generation]
180	-[ItemRecord setGeneration:]
181	-[ItemRecord url]
182	-[ItemRecord setUrl:]
183	-[ItemRecord modificationDate]
184	-[ItemRecord setModificationDate:]
185	-[ItemRecord executablePath]
186	-[ItemRecord setExecutablePath:]
187	-[ItemRecord executableModificationDate]
188	-[ItemRecord setExecutableModificationDate:]
189	-[ItemRecord teamIdentifier]
190	-[ItemRecord setTeamIdentifier:]

'ItemRecord' properties

Item De-serialization

…reimplemented in our own code

```
01  @interface ItemRecord : NSObject <NSSecureCoding>
02  @property NSInteger type;
03  @property(nonatomic, retain)NSURL* url;
04  @property(nonatomic, retain)NSUUID* uuid;
05  …
06  @property(nonatomic, retain)NSString* executablePath;
07  @property(nonatomic, retain)NSString* teamIdentifier;
08  @property(nonatomic, retain)NSString* bundleIdentifier;
09
10  @end
11
12  -(id)initWithCoder:(NSCoder *)decoder
13  {
14      ...
15      self.url = [decoder decodeObjectOfClass:[NSURL class] forKey:@"url"];
16      self.uuid = [decoder decodeObjectOfClass:[NSUUID class] forKey:@"uuid"];
17      self.executablePath = [decoder decodeObjectOfClass:[NSString class] forKey:@"executablePath"];
18      self.teamIdentifier = [decoder decodeObjectOfClass:[NSString class] forKey:@"teamIdentifier"];
19      self.bundleIdentifier = [decoder decodeObjectOfClass:[NSString class] forKey:@"bundleIdentifier"];
20      ...
```

ItemRecord deserialization
(custom initWithCoder: method)

Programmatically Dumping the BTM Database

…reimplemented in our own code

```
01  //load
02  // path set to BTM database
03  NSData* data = [NSData dataWithContentsOfURL:path options:0 error:NULL];
04
05  //init keyed unarchiver with database data
06  NSKeyedUnarchiver* keyedUnarchiver = [[NSKeyedUnarchiver alloc] initForReadingFromData:data error:NULL];
07
08  //de-serialize
09  // will trigger call to "initWithCoder:" and de-serialize all objects
10  Storage* storage = [keyedUnarchiver decodeObjectOfClass:[Storage class] forKey:@"store"];
11
12  //print out all items
13  for(NSString* key in storage.items) {
14
15      NSArray* items = storage.items[key];
16
17      for(ItemRecord* item in items){
18          printf(" #%d\n", ++itemNumber);
19          printf(" %s\n", [[item dumpVerboseDescription] UTF8String]);
20
21      }
22  }
```

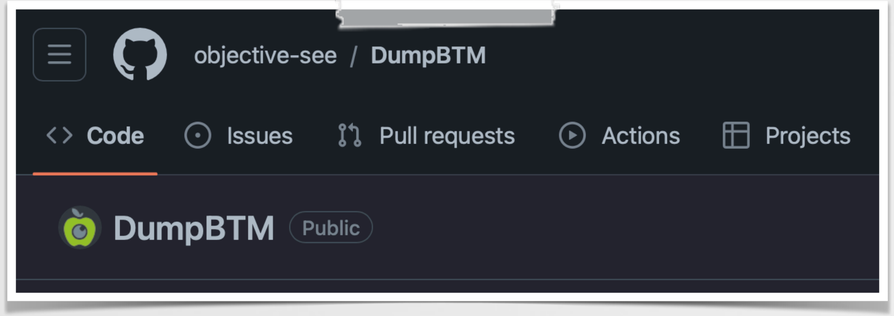
"DUMP BTM" library

Programmatically Dumping the BTM Database

…reimplemented in our own code

link in the 'Dump BTM' library


```
01  #include "dumpBTM.h"
02
03  int main(int argc, const char * argv[]) {
04
05      //set if you want
06      // a custom btm file
07      NSURL* btmPath = nil;
08
09      //dump stdout
10      // (aka sfltool dumpbtm)
11      dumpBTM(btmPath);
12
13      //or parse into a dictionary...
14      NSDictionary* contents = parseBTM(path);
15      ...
16
```



github.com/objective-see/DumpBTM

DUMPBTM OUTPUT

...and no root access required

```
% ./dumpBTM
Opened /private/var/db/com.apple.backgroundtaskmanagement/BackgroundItems-v8.btm

=========================
 Records for UID 0 : FFFFEEEE-DDDD-CCCC-BBBB-AAAA00000000
=========================

...
 UUID:                EAE1D8DC-2928-47C1-BC53-6274590FE3D1
  Name:                us.zoom.ZoomDaemon
  Developer Name:      Zoom Video Communications, Inc.
  Team Identifier:     BJ4HAAB9B3
  Type:                legacy daemon  (0x10010)
  Disposition:         [enabled allowed visible notified] (11)
  Indentifier:         us.zoom.ZoomDaemon
  URL:                 file:///Library/LaunchDaemons/us.zoom.ZoomDaemon.plist
  Executable Path:     /Library/PrivilegedHelperTools/us.zoom.ZoomDaemon
  Generation:          2
  Assoc. Bundle IDs:   [us.zoom.xos]
  Parent Identifier:   Zoom Video Communications, Inc.
```

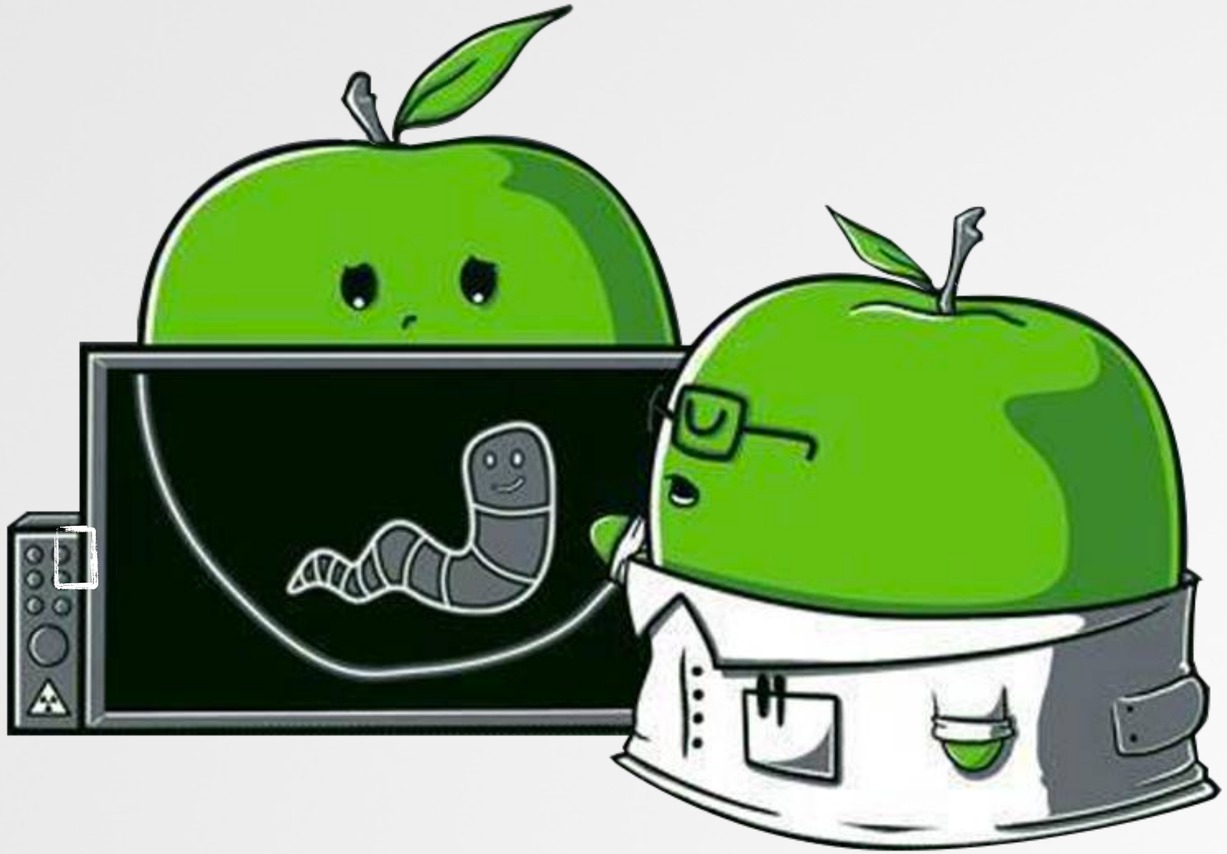
"Dump BTM" output

KnockKnock


KnockKnock
KnockKnock
version: 2.4.0
Start Scan
Categories:
Authorization Plugins
registered authorization bundles
0
Browser Extensions
extensions hosted in the browser
3
Background Managed Tasks
agents, daemons, & login items managed by BTM
8
Cron Jobs
current user's cron jobs
0
Dir. Services Plugins
registered directory services bundles
0
Event Rules
actions executed by emond
0
Extensions and Widgets
plugins that extend/customize the OS
3
Kernel Extensions
installed kexts, likely kernel loaded
0
Items:
CCXProcess
/Applications/Utilities/Adobe Creative Cloud Experience/CCXProcess/CCXProcess.app/Contents/MacOS/CC…
/Library/LaunchAgents/com.adobe.ccxprocess.plist
0/75
virustotal
info
show
ChmodBPF
/Library/Application Support/Wireshark/ChmodBPF/ChmodBPF
/Library/LaunchDaemons/org.wireshark.ChmodBPF.plist
0/75
virustotal
info
show
Creative Cloud
/Applications/Utilities/Adobe Creative Cloud/ACC/Creative Cloud.app/Contents/MacOS/Creative Cloud
/Library/LaunchAgents/com.adobe.AdobeCreativeCloud.plist
?
virustotal
info
show
com.adobe.acc.installer.v2
/Library/PrivilegedHelperTools/com.adobe.acc.installer.v2
/Library/LaunchDaemons/com.adobe.acc.installer.v2.plist
?
virustotal
info
show
magnetLauncher
/Applications/Magnet.app/Contents/Library/LoginItems/magnetLauncher.app/Contents/MacOS/magnetLaunch
virustotal
info
show
OSX.DazzleSpy
? softwareupdate
/Users/Patrick/.local/softwareupdate
/Users/patrick/Library/LaunchAgents/com.apple.softwareupdate.plist
33/75
virustotal
info
show
us.zoom.ZoomDaemon
/Library/PrivilegedHelperTools/us.zoom.ZoomDaemon
/Library/LaunchDaemons/us.zoom.ZoomDaemon.plist
0/74
virustotal
info
show
zoom.us
/Applications/zoom.us.app/Contents/MacOS/zoom.us
?
virustotal
info
show
Scan Complete

Tools (Part II)

leveraging BTM to detect persistent malware



BTM Daemon's References to "Endpoint Security"

```
% strings - backgroundtaskmanagementd | grep ES

submitItemAddToES: No URL provided, cannot submit event to EndpointSecurity
submitItemAddToES: Invalid BTMItemType, cannot submit event to EndpointSecurity
```

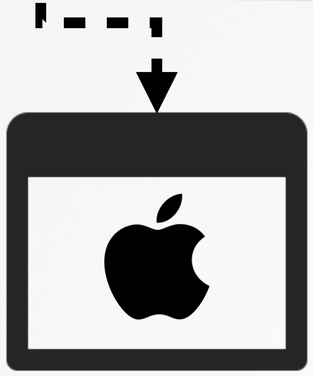
References to 0x100007b84

Search

Address	Value	
0x10000383b (-[SFLClientProcess addURL:managed:reply:] + 0x677)	call	submitItemAddToES
0x1000038ac (-[SFLClientProcess addURL:managed:reply:] + 0x6e8)	call	submitItemAddToES
0x100005053 (-[BTMService registerLaunchItemWithAuditToken:parentURL:type:relativeUR...	call	submitItemAddToES
0x100005084 (-[BTMService registerLaunchItemWithAuditToken:parentURL:type:relativeUR...	call	submitItemAddToES
0x10000519c (-[BTMService registerLaunchItemWithAuditToken:parentURL:type:relativeUR...	call	submitItemAddToES
0x1000051cc (-[BTMService registerLaunchItemWithAuditToken:parentURL:type:relativeUR...	call	submitItemAdd
0x1000065dd (-[BTMService addItem:uid:reply:] + 0x1ef)	call	submitItemAdd
0x100006611 (-[BTMService addItem:uid:reply:] + 0x223)	call	submitItemAdd

Cancel

```
01  submitItemAddToES:
02   ...
03
04   bl          ess_notify_btm_launch_item_add
```



"ess_notify_btm_launch_item_add" Google

Your search - **"ess_notify_btm_launch_item_add"** - did not match any documents.

very undocumented :/

implemented in:

libEndpointSecuritySystem.dylib

Endpoint Security

ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD event

```
01  /* The valid event types recognized by EndpointSecurity */
02  typedef enum {
03  ...
04
05  // The following events are available beginning in macOS 13.0
06  ...
07      ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD,
08      ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_REMOVE
```

REGISTERING FOR ES EVENTS

ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD event

```
01  //client/event of interest
02  es_client_t* esClient = NULL;
03  es_event_type_t events[] = {ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD};   1
04
05  //new client
06  //callback will process 'ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD' events
07  es_new_client(&esClient, ^(es_client_t *client, const es_message_t *message)
08  {
09      //TODO: handle event
10  }                                                                          2  callback for BTM events
11
12  //subscribe
13  es_subscribe(endpointProcessClient, events, sizeof(events)/sizeof(events[0])));   3
14
```

ES BTM Monitor
(ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD)

1. Specify events of interest
2. Create callback for events
3. Subscribe for events

HANDLING ES NOTIFICATIONS

ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD event

```
01  typedef struct {
02    es_process_t * instigator;
03    es_process_t * app;
04    es_btm_launch_item_t * item;
05    es_string_token_t executable_path;
06  } es_event_btm_launch_item_add_t;
```

es_event_btm_launch_item_add_t

our helper function

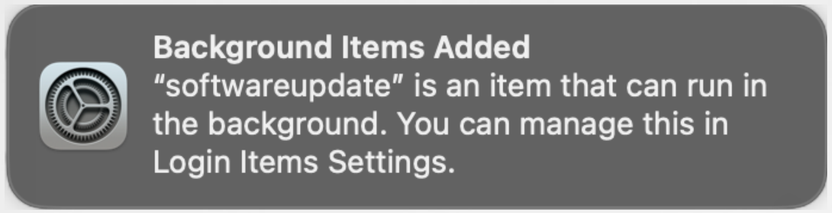
```
01  NSString* toString(es_string_token_t* token) {
02
03        return [[NSString alloc]
04                  initWithBytes:token->data
05                  length:token->length
06                  encoding:NSUTF8StringEncoding];
07  }
```

```
01  //new client
02  // callback will process 'ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD' events
03  es_new_client(&esClient, ^(es_client_t *client, const es_message_t *message) {
04
05     NSLog(@"Item URL: %@", toString(&message->event.btm_launch_item_add->item->item_url);
06     NSLog(@"Item Executable: %@", toString(event btm_launch_item_add->executable_path);
07
08     es_process_t* app = message->event.btm_launch_item_add->app;
09     if(NULL != app) NSLog(@"App: %@", toString(&app->executable->path));
10
11     es_process_t* instigator = message->event.btm_launch_item_add->instigator;
12     if(NULL != instigator) NSLog(@"Instigator: %@", toString(&instigator->executable->path));
13     ...
```

Handler for ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD

Hooray, Persistent Malware Detection

...via ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD




```
# ./btm_monitor
New Event: 'ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD'

Type: ES_BTM_ITEM_TYPE_AGENT

Instigator: /usr/libexec/smd

Agent exe: /Users/User/.local/softwareupdate
Agent url: /Users/User/Library/LaunchAgents/com.apple.softwareupdate.plist
```

Persistence detection
(OSX.DazzleSpy)

Note: No delivery of BTM removal events
(ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_REMOVE) ...broken!

"BTM LAUNCH ITEM ADD" EVENT (WAS) BROKEN

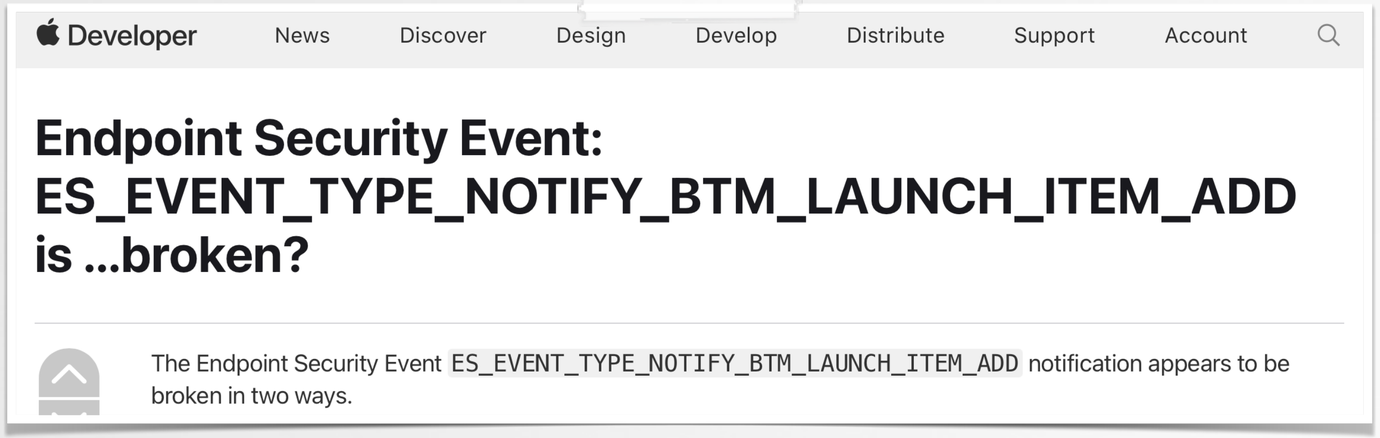
...in two separate ways 🤦

1️⃣ **ES_BTM_ITEM_TYPE_AGENT / ES_BTM_ITEM_TYPE_DAEMON:**

When a new item installed, it would deliver a notification for *every* installed item.

2️⃣ **ES_BTM_ITEM_TYPE_LOGIN_ITEM:**

When a new was item installed, no notification would be delivered.

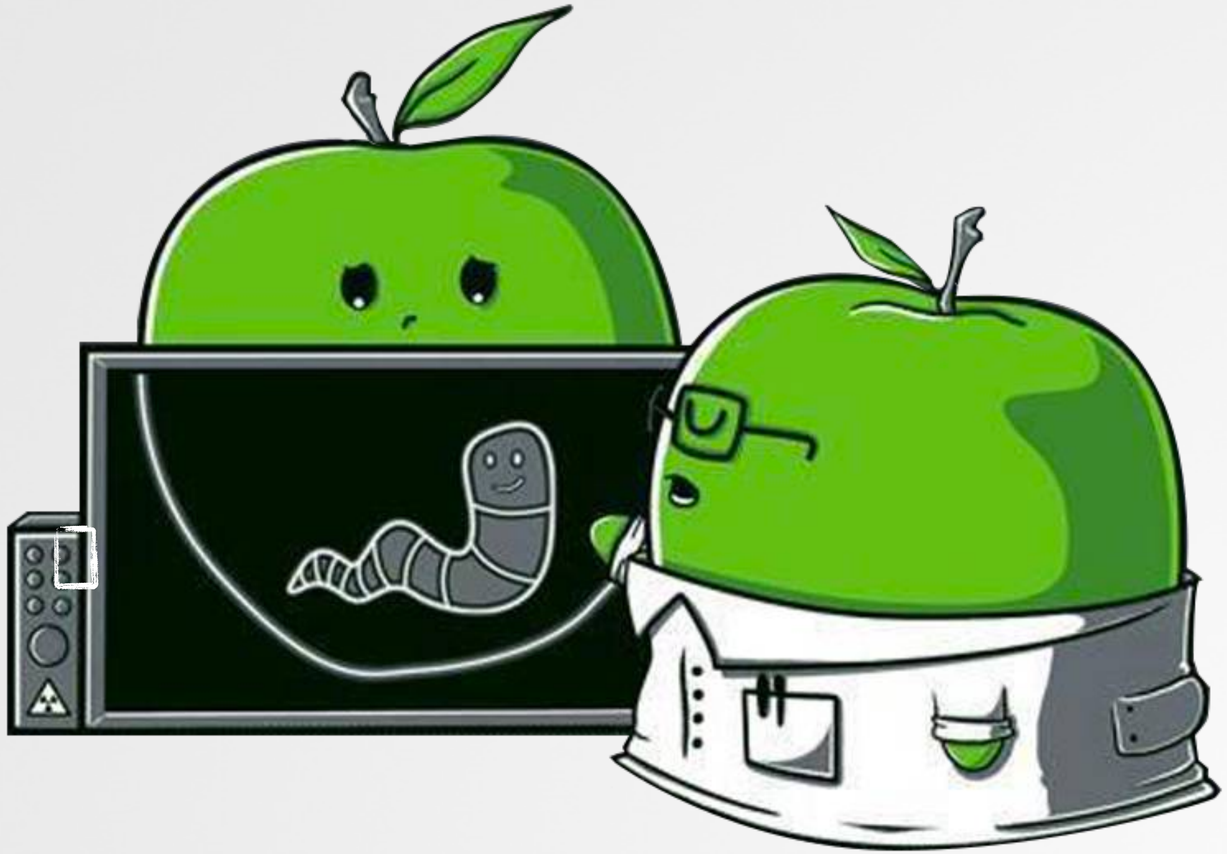


(submitted) bug report

...fixed in macOS 13.3

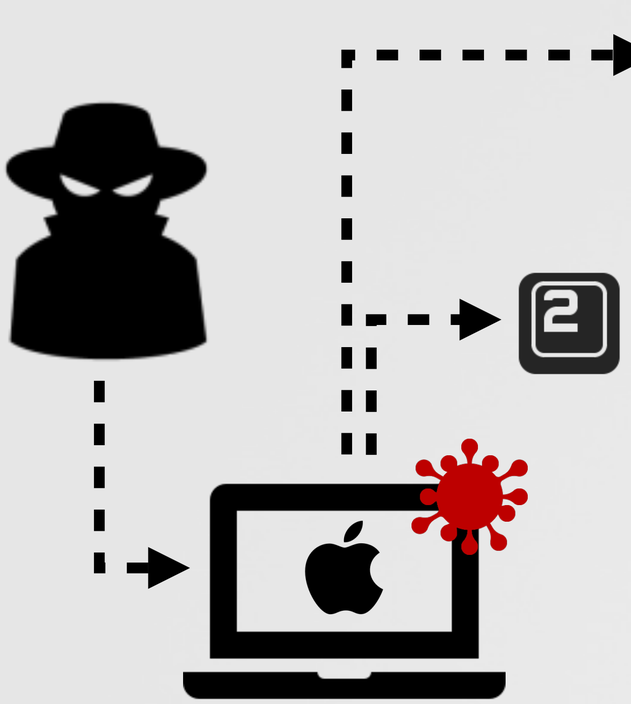
Breaking

...avoiding BTM alerts & ES messages

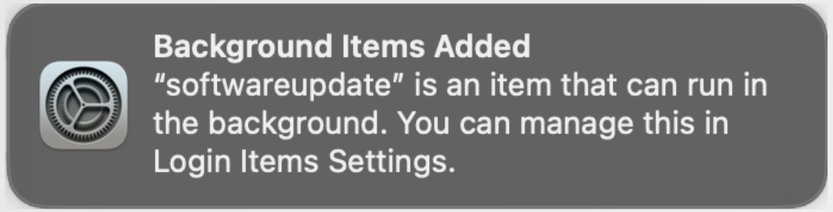


Hackers Hate Alerts & Messages

...that can give away the attack!



OMG, i've been hacked!!



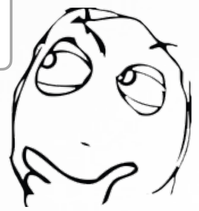

```
# ./btm_monitor
New Event: 'ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD'

Type: ES_BTM_ITEM_TYPE_AGENT
Agent exe: /Users/User/.local/softwareupdate
Agent url: /Users/User/Library/LaunchAgents/com.apple.softwareupdate.plist
```

Can we (as hackers) prevent:

A: The system alert?

B: The Endpoint Security event?

...want to persist silently!



Persist "Outside" BTM

...avoid it all together!

```
01  def format(self, src, des, uc):
02    ...
03    if not os.path.isfile(des):
04      os.system('cp ' + src + ' ' + des)
05
06    if des[-3:] == '.py':
07      os.system('sudo crontab -l 2>/dev/null;
08      echo "*/2 * * * * python ' + des + '" | sudo crontab -')
```

Persistence via cron (OSX.Enigma)

lots of other ways to persist

THEEVILBIT BLOG

Beyond the good ol' LaunchAgents - Introduction

by: Csaba Fitzl

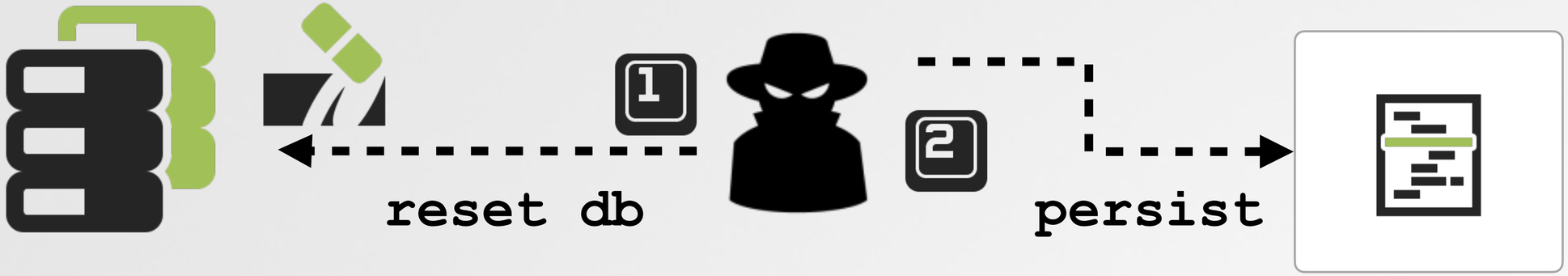
Remember, BTM (currently) only covers launch agents/daemons, and login items.

Bypass System Alert (Method 0x1)

reset the BTM database, via sfltool

note: requires root privileges

```
# sfltool resetbtm
sfltool[1261:13567] Database reset.
```

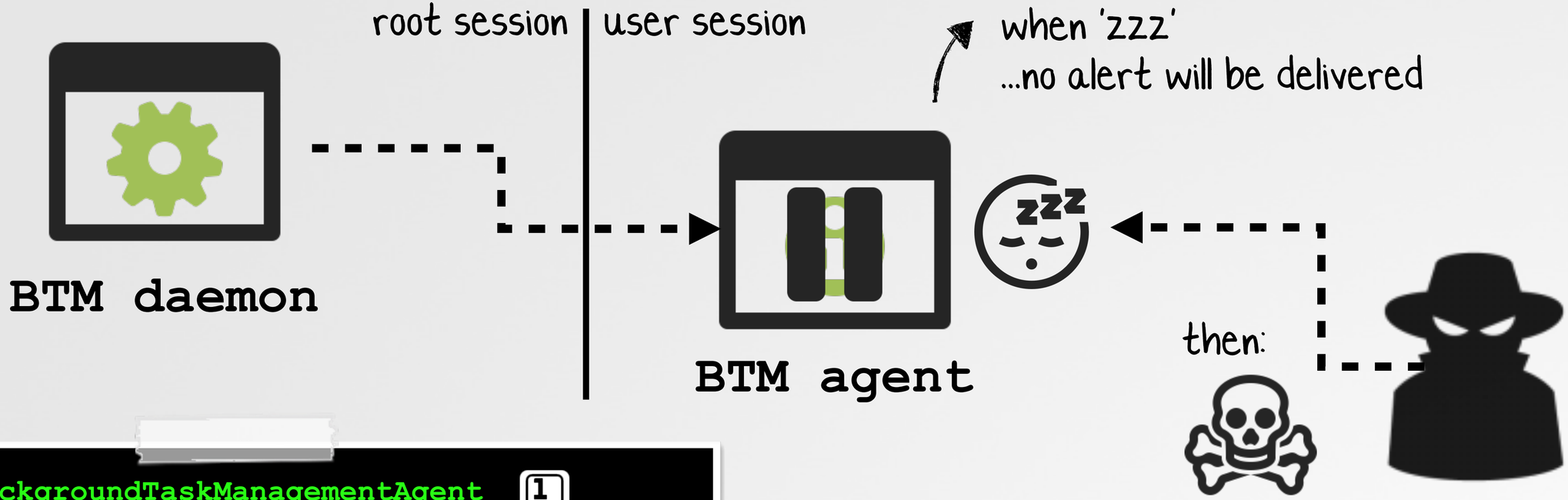



```
% log stream --debug --info --predicate "process = 'backgroundtaskmanagementd'"
backgroundtaskmanagementd: registerLaunchItem: result=no error, new item
                           disposition=[enabled, allowed, visible, not notified],
                           identifier=com.apple.softwareupdate,
                           url=file:///Users/user/Library/LaunchAgents/com.apple.softwareupdate.plist

backgroundtaskmanagementd: should post advisory=false for uid=501,
                           id=6ED3BEBC-8D60-45ED-8BCC-E0163A8AA806, item=softwareupdate
```

"should post advisory=false"

Bypass System Alert (Method 0x2)

pause, then kill the BTM agent



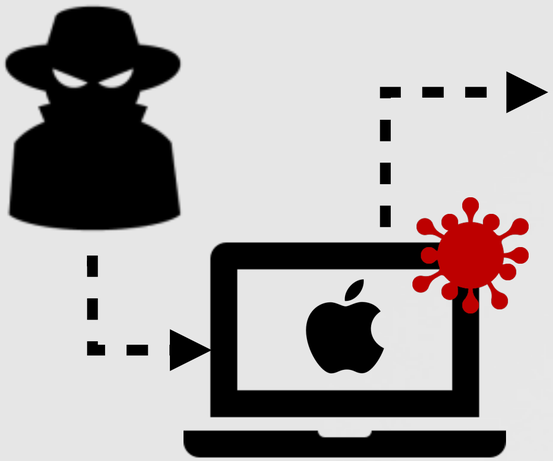

```
% pgrep BackgroundTaskManagementAgent   1
88262

% kill -SIGSTOP 88262   2

% ps -o state 88262
T

% kill -SIGKILL 88262   3
```

'T' means, stopped

1. Get pid of BTM agent
2. Send it a 'STOP' signal
3. Once persisted, kill it (to prevent alert re-delivery)

BYPASS ES EVENTS

prevent ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD



```
# ./btm_monitor
New Event: 'ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD'

item exe: /Users/User/.local/softwareupdate
item url: /Users/User/Library/LaunchAgents/com.apple.softwareupdate.plist
```

want to prevent this !

BTM daemon

```
01  def submitItemAddToES()
02    ...
03    ess_notify_btm_launch_item_add();
```

found in libEndpointSecuritySystem

```
01  def ess_notify_btm_launch_item_add()
02    ...
03    downcallNotifyOnly();
```

```
01  def downcallNotifyOnly()
02    ...
03    if(!mac_syscall("EndpointSecurity", ...)
04       os_log_impl(..., "Failed to submit event using ES syscall %d: %d %s", &var_50, 0x18);
```

mac_syscall: to the kernel

(Responsible?) Process Lookup

...via proc_find & proc_getexecutablevnode

EndpointSecurity.kext

```
% lldb
(lldb) gdb-remote 8084
...
(lldb) bt
EndpointSecurity`pdbReadAuditToken()
...
EndpointSecurity`EndpointSecurityEventManager::sendNotifyOnly()
EndpointSecurity`EndpointSecurityEventManager::sendBtmLaunchItemAdd()
EndpointSecurity`EndpointSecurityEventManager::es_policy_syscall()
kernel`hndl_unix_scall64
```

call stack

```
01  def pdbReadAuditToken()
02    ...
03    PackedDataBufferReader::read
04    ...
05    AutoReleasingProc::createFromPidVer();
06    ...
07    set_common_proc_info();
08
09    //error? no ES message delivered!
```

```
01  def AutoReleasingProc::createFromPidVer();
02    ...
03    proc* process = proc_find(<pid>);
```

```
(lldb) p *(proc*)$rdi
(proc) $1 = {
  p_pid = 866
  p_name = {'OverSight'}
```

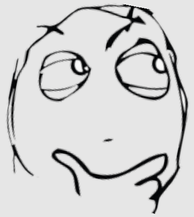
reported event (in user-mode)

```
# eslogger btm_launch_item_add
"event": {
      "btm_launch_item_add": {
       "pid": 866
       "path": "/Applications/OverSight.app
                    /Contents/MacOS/OverSight"
        ...
```

```
01  def set_common_proc_info();
02    ...
03    vnode = proc_getexecutablevnode(process);
04    if(!vnode)
05       //set error to 0x3 (ESRCH / "No Such Process")
```

(Responsible?) Process Lookup

...via proc_find & proc_getexecutablevnode



What happens if the responsible process ...has (quickly) exited already?

```
01  def AutoReleasingProc::createFromPidVer();
02    ...
03    proc* process = proc_find(NULL)
```

proc_find(NULL/0)
...will return the kernel task

```
(lldb) p *(proc*)$rdi
(proc) $1 = {
  p_pid = 0
  p_name = {'kernel_task'}
```

the kernel task: no executable vnode!

```
01  def set_common_proc_info();
02    ...
03    vnode = proc_getexecutablevnode(process)
04    if(!vnode)
05      //return ERROR 0x3: ('ESRCH' / "No Such Process")
06      rax = 0x3;
```

Error
...and no ES message delivered ! 🫣

```
% log stream ...

Error
backgroundtaskmanagementd: (libEndpointSecuritySystem.dylib)
Failed to submit event using ES syscall 20: 3 No such process
```

How to Trigger?

...as easy as installing BlockBlock 😅

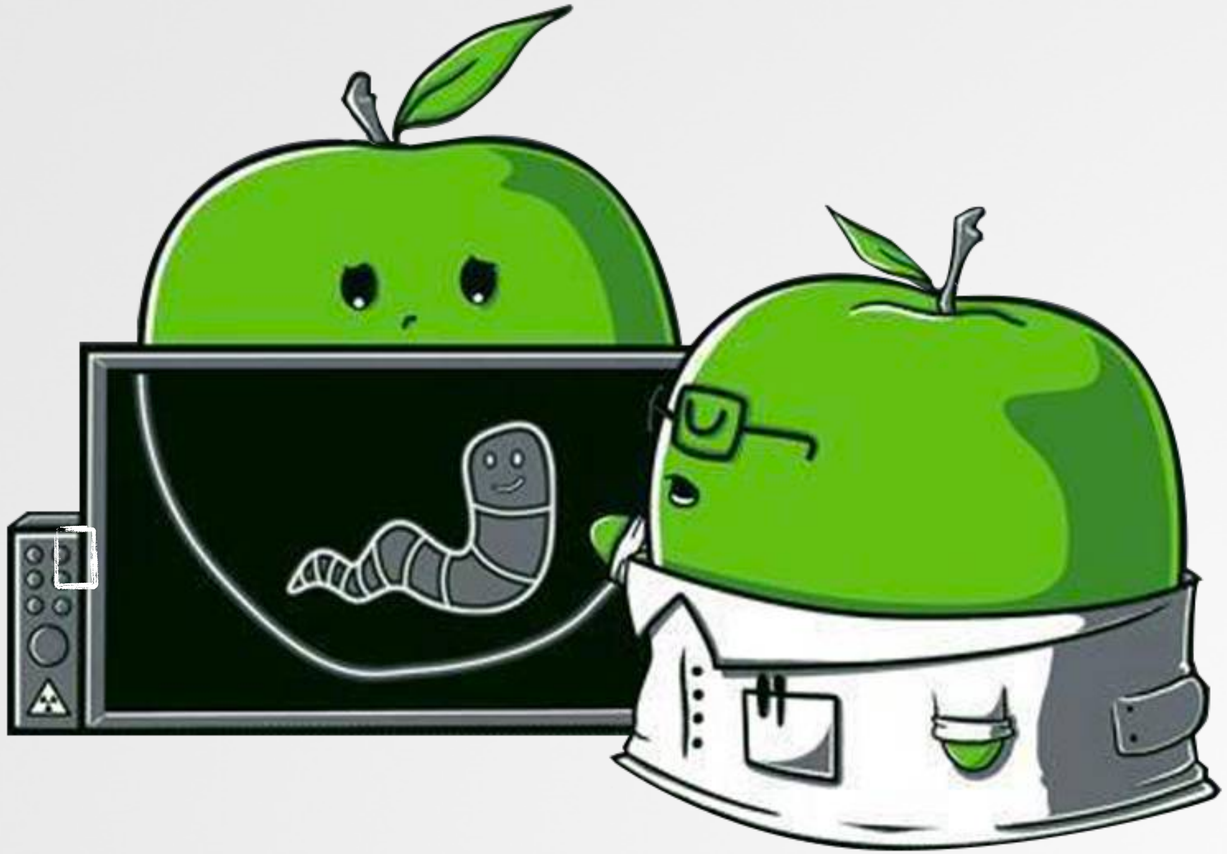
```
01  #install logic
02  if [ "${1}" == "-install" ]; then
03   ...
04
05  cp "com.objective-see.blockblock.plist" /Library/LaunchDaemons/
06
07   echo "launch daemon installed"
```

BlockBlock Installer

DEMO

Detections

...of bypasses



Attack Detection

via Endpoint Security (ES)

Detecting BTM Database Reset

via process monitor (+ arguments)

```
# sfltool resetbtm
sfltool[1261:13567] Database reset.
```

```
{
  "event" : "ES_EVENT_TYPE_NOTIFY_EXEC",
  "process" : {
     "pid" : 44100
     "name" : "sfltool",
     "path" : "/usr/bin/sfltool",

    "arguments" : [
      "sfltool",
      "resetbtm"
    ],
    ...
```

Or, block all together via
ES_EVENT_TYPE_AUTH_EXEC event...

MONITORING SIGNALS
via ES/ES_EVENT_TYPE_NOTIFY_SIGNAL

```
01  es_event_type_t events[] = {ES_EVENT_TYPE_NOTIFY_SIGNAL};
02
03  es_new_client(&endpointClient, ^(es_client_t *client, const es_message_t *message) {
04
05      int signal = message->event.signal.sig;
06      es_process_t* sourceProcess = message->process;
07      es_process_t* targetProcess = message->event.signal.target;
08
09      //have signal, source, & target process
10      // TODO: check if SIGSTOP (17), being sent to BTM agent
11
12  }
13
14  es_subscribe(endpointClient, events, sizeof(events)/sizeof(events[0]));
```

attack detected

ES Monitoring (via: ES_EVENT_TYPE_NOTIFY_SIGNAL)

```
# ./monitorSignals
New Signal (ES_EVENT_TYPE_NOTIFY_SIGNAL): SIGSTOP (17)

Source process (687): /bin/zsh
Target process (590): /System/Library/PrivateFrameworks/BackgroundTaskManagement.framework
      /Support/BackgroundTaskManagementAgent.app/Contents/MacOS/BackgroundTaskManagementAgent
```

Blocking Signals
via ES/ES_EVENT_TYPE_AUTH_SIGNAL

```
01  es_event_type_t events[] = {ES_EVENT_TYPE_AUTH_SIGNAL};
02
03  es_new_client(&endpointClient, ^(es_client_t *client, const es_message_t *message) {
04
05      int signal = message->event.signal.sig;
06      es_process_t* sourceProcess = message->process;
07      es_process_t* targetProcess = message->event.signal.target;
08
09      if( (signal == SIGSTOP) &&
10          (btmAgent == audit_token_to_pid(targetProcess->audit_token))) {
11            es_respond_auth_result(client, message, ES_AUTH_RESULT_DENY, false);
12      }
13  }
14  ...
```

block via
ES_AUTH_RESULT_DENY

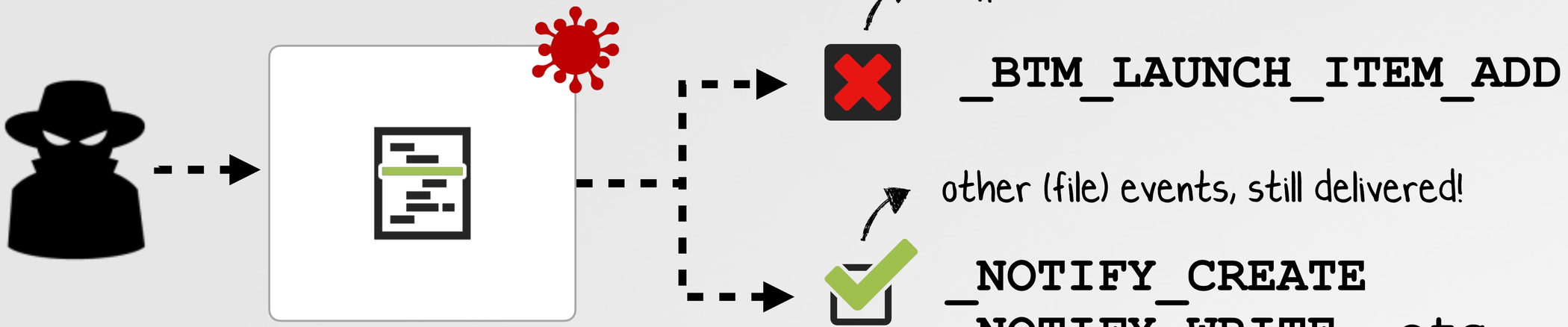
ES blocking (via: ES_EVENT_TYPE_AUTH_SIGNAL)

```
% pgrep BackgroundTaskManagementAgent
590

% kill -SIGSTOP 590
% kill: kill 590 failed: operation not permitted
```

attack, now blocked :)

Monitoring for Persistence

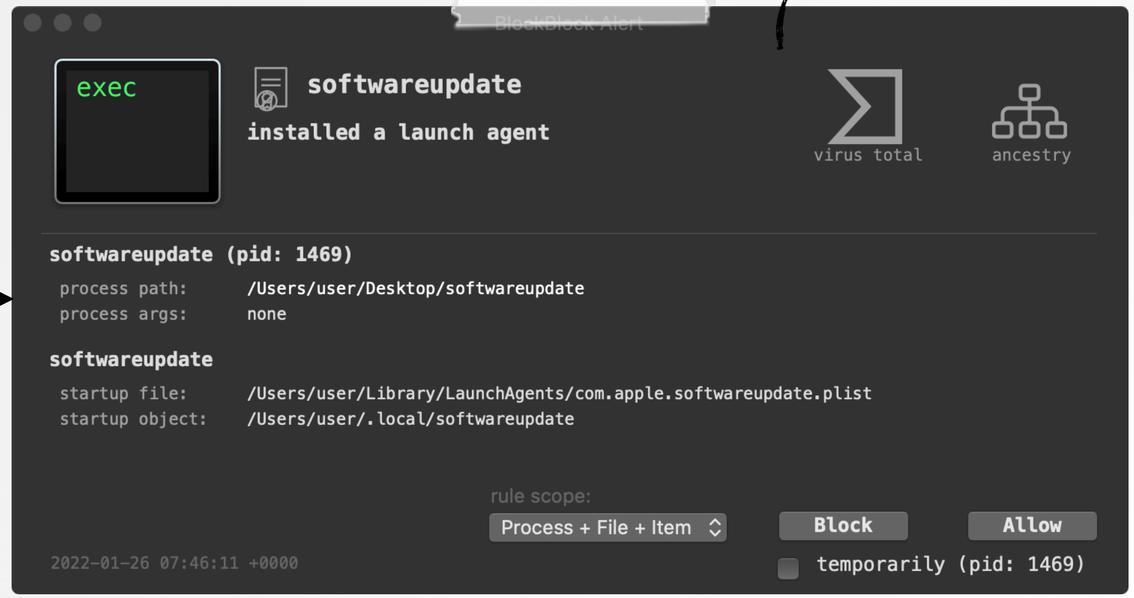
via other ES events



BlockBlock:

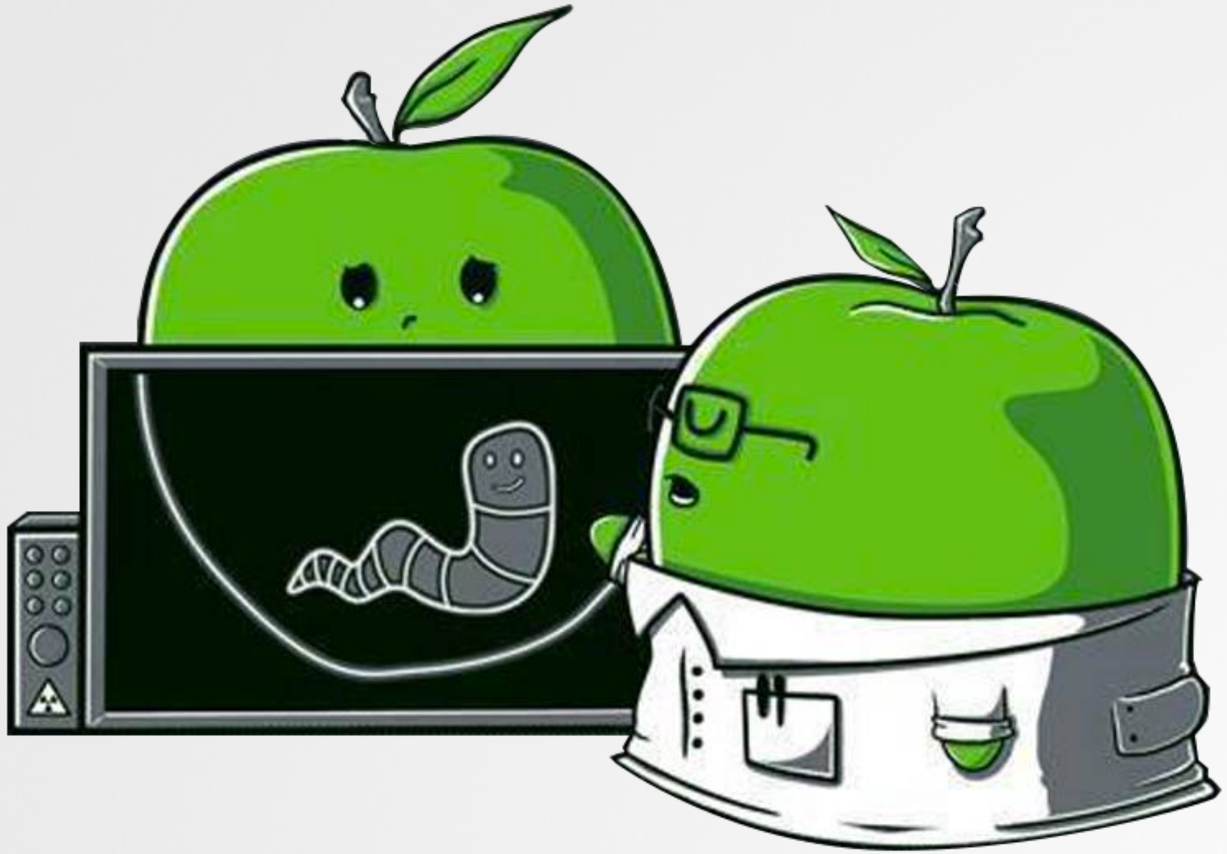
1. Leverage ES file events
2. Launch item location?
3. Alert user, and block

OSX.DazzleSpy



Conclusions

...& take aways



Takeaways

Understanding
macOS's BTM

New malware
detection tools

A myriad of
trivial bypasses

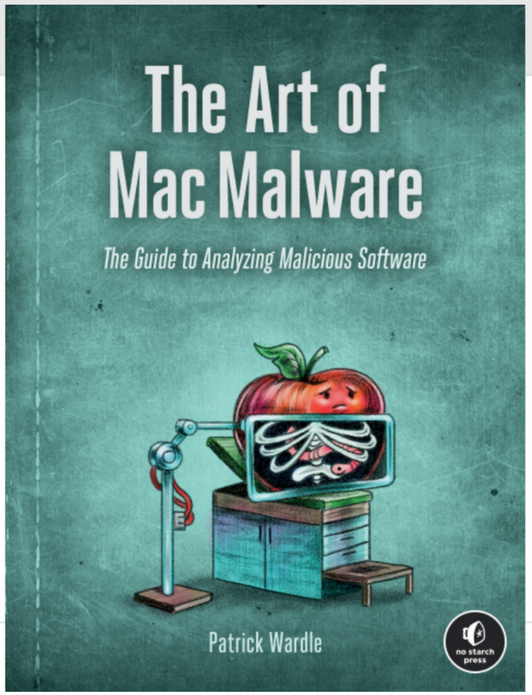
If white hats don't point out flaws in Apple's "security improvements" ...who will? (hint: not Apple!)

Interested in Learning More?

read, "The Art of Mac Malware" book(s)

Books about Mac Malware

by Patrick Wardle



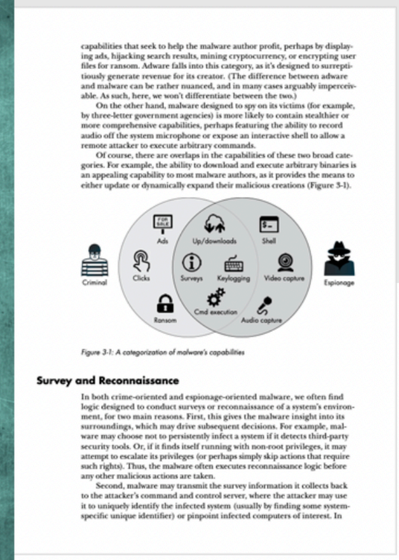
INTRODUCTION

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 imperceivable. 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

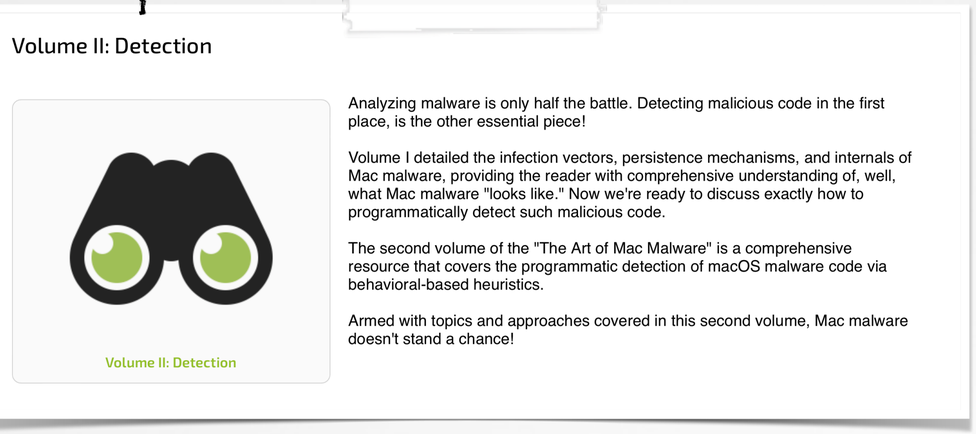
Survey and Reconnaissance

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

Coming soon!

Vol. II: (programmatic) detection



Volume II: Detection

Volume II: Detection

Analyzing malware is only half the battle. Detecting malicious code in the first place, is the other essential piece!

Volume I detailed the infection vectors, persistence mechanisms, and internals of Mac malware, providing the reader with comprehensive understanding of, well, what Mac malware "looks like." Now we're ready to discuss exactly how to programmatically detect such malicious code.

The second volume of the "The Art of Mac Malware" is a comprehensive resource that covers the programmatic detection of macOS malware code via behavioral-based heuristics.

Armed with topics and approaches covered in this second volume, Mac malware doesn't stand a chance!

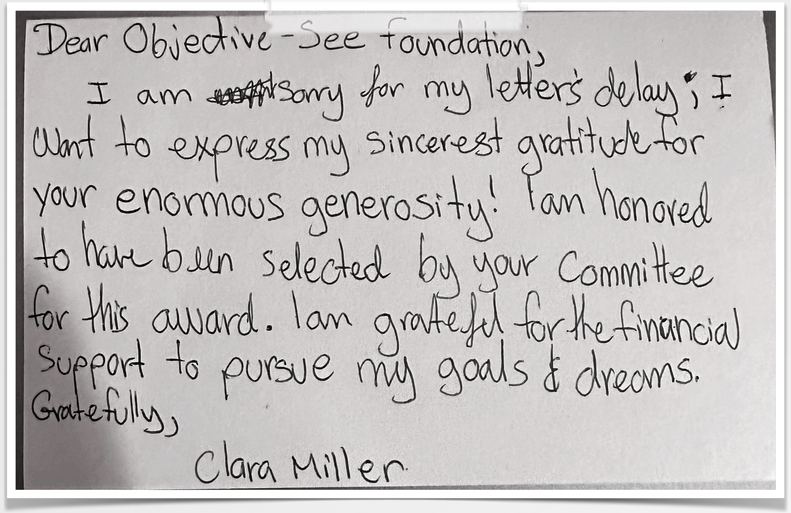
Book Signing: today @ 11:00 am

"The Art of Mac Malware"
free @ https://taomm.org

Objective-See Foundation 501(c)(3)

learn more our community efforts ...& support us! 🥰

#OBTS Conference



College Scholarships

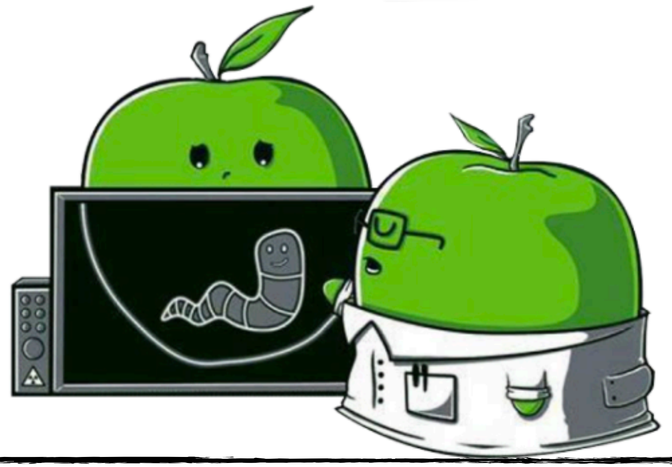
Diversity Programs ("Objective-We")

The Objective-See Foundation
objective-see.org/about.html

Objective-See Foundation Fundraiser

Maui wildfire relief fund

Maui Wildfire Relief Fund



Latest News:

New Video:
Watch **What is #OBTS?**

Tool Update (KnockKnock):
Just released **KnockKnock v2.4.2**

New Blog Post:
Read: **"LockBit ransomware comes for macOS"**

#OBTS v6.0
Just announced **#OBTS v6.0**

Our home, Maui, was recently **devastated by fires**. Many of our friends and neighbors lost everything. We're raising money to help them!

Please consider making a donation via our **fundraiser** 🙏

To help: objective-see.org 🙏

Mahalo to the "Friends of Objective-See"

jamf

kandji

paloalto NETWORKS

SOPHOS

mosyle

KOLIDE

CleanMyMac X

SmugMug

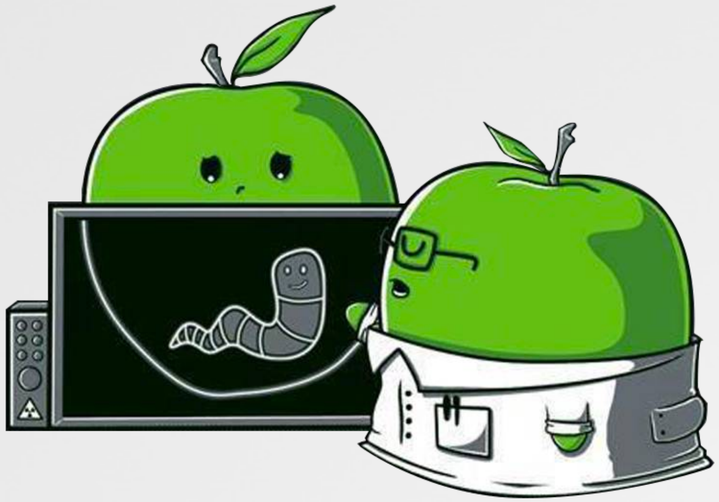
Guardian Mobile Firewall

The Mitten Mac

iVerify

Halo Privacy

Demystifying macOS's BTM



RESOURCES:

"DumpBTM"
https://github.com/objective-see/DumpBTM

"Manage login items and background tasks on Mac"
https://support.apple.com/guide/deployment/manage-login-items-background-tasks-mac-depdca572563/web